Thèse de doctorat Novembre/ 2024

Université Paris - Panthéon - Assas

École doctorale d'Économie, Gestion, Information et Communication

Thèse de doctorat en Informatique

Préparée à l'École Normale Supérieure et Sorbonne Université

soutenue le 25 Novembre 2024

Towards Unclonable Cryptography in the Plain Model

Paul Hermouet

Sous la direction de Céline Chevalier et Elham Kashefi

Membres du jury :

Céline Chevalier PSL University Université Paris-Panthéon-Assas	*Directrice de thèse*
Elham Kashefi Sorbonne Université University of Ediburgh	*Directrice de thèse*
Prabhanjan Ananth University of California, Santa Barbara	*Rapporteur*
Christian Majenz Technical University of Denmark	*Rapporteur*
Anne Broadbent Université of Ottawa	*Examinatrice*
Stacey Jeffery CWI QuSoft	*Examinatrice*
Damien Vergnaud Sorbonne Université	*Examinateur*

Avertissement

L'université n'entend donner aucune approbation ni improbation aux opinions émises dans cette thèse ; ces opinions doivent être considérées comme propres à leur auteur.

Résumé

La mécanique quantique génère de nouvelles menaces pour la cryptographie, mais elle offre également de nouveaux outils pour construire des primitives cryptographiques qui ne peuvent être réalisées dans le monde classique. Parmi ces primitives figurent les *primitives inclonables*, qui exploitent le théorème de non-clonage des états quantiques : la propriété surprenante qu'ils ne peuvent être copiés. Dans la première partie de cette thèse, nous donnons un aperçu approfondi des primitives inclonables, et en particulier de trois primitives principales : la *quantum money*, fournissant des jetons vérifiables inclonables; l'*unclonable encrytion*, un schéma de chiffrement avec chiffrés inclonables; et enfin la *copy-protection*, dans laquelle un vendeur envoie des programmes inclonables à ses clients. Nous définissons chacune de ces primitives, discutons de leur historique, et donnons des exemples de constructions.

Dans la seconde partie, nous nous concentrons sur la *copy-protection* et l'*unclonable encryption*. Bien que ces primitives aient reçu beaucoup d'attention ces dernières années, elles ne sont pas encore entièrement comprises. En particulier, décider si elles existent dans le *plain model* (dans lequel nous ne pouvons pas utiliser d'oracles aléatoires (quantiques)) reste une question ouverte. Nous progressons vers cet objectif en présentant de nouvelles constructions pour ces primitives, avec une sécurité dans le plain model, en supposant de nouvelles conjectures. Pour prouver la sécurité de nos constructions, nous définissons et prouvons une nouvelle propriété dit de "monogamie" des *coset states* — sans doute les états quantiques les plus utiles lorsqu'il s'agit de construire des primitives inclonables. Nous pensons que cette nouvelle propriété est d'un intérêt indépendant. Comme contributions indépendantes, nous définissons deux nouvelles propriétés de sécurité pour une autre primitive inclonable, les tokenized signatures, et présentons une construction satisfaisant ces nouvelles propriétés. Les preuves de sécurité découlent de notre nouvelle propriété de "monogamie", ainsi que d'une variante d'une des propriétés principales des *coset states*, que nous définissons et prouvons.

Les protocoles cryptographiques inclonables nécessitent souvent une puissance quantique importante pour toutes les parties impliquées. Comme un tel modèle est peu susceptible d'être réalisable dans un avenir proche, nous examinons dans une troisième partie la cryptographie inclonable *semi-quantique*, dans laquelle des utilisateurs classiques interagissent avec un serveur quantique puissant. La plupart des primitives inclonables sont basées sur les *coset states*, nous construisons donc un *protocole de préparation à distance* de ces états, dans lequel un utilisateur classique fournit des instructions au serveur quantique sur la manière de construire coset states. De plus, cette préparation est effectuée de manière aveugle, dans le sens où le serveur n'a pas d'information sur les *coset states* qu'il a préparés. Cela nous permet d'utiliser ce protocole pour construire des primitives inclonables de manière semi-quantique.

Abstract

Although it is well known that quantum mechanics enable new threats to cryptography, it also provides new tools to construct cryptographic primitives that cannot be achieved in the classical world. Among these primitives are the *unclonable primitives*, which harness the non-cloning theorem of quantum states: the surprising fact that they cannot be copied. In the first part of this thesis, we give an extensive overview of unclonable primitives, and in particular of three main primitives: quantum money, providing unclonable verifiable tokens; unclonable encryption, an encryption scheme with unclonable ciphertexts; and finally copy-protection, in which a vendor sends unclonable programs to clients. For each of these primitives, we define them, discuss their history, and give example constructions.

In the second part, we focus on *copy-protection* and *unclonable encryption*. Despite receiving a lot of attention in the past years, these primitives are not yet fully understood. In particular, deciding whether they exist in the plain model (in which we cannot use any (quantum) random oracles) is still an open question. We progress towards this goal, by presenting new constructions for these primitives, with security in the plain model, assuming post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions, and two new conjectures. In order to prove the security of our constructions, we define and prove a new monogamy-of-entanglement property for coset states — arguably the most useful quantum states when it comes to constructing unclonable primitives — which we think is of independent interest. As independent contributions, we define two new security properties for another unclonable primitive, tokenized signature schemes, and present a construction satisfying these new properties. The security proofs follow from our new monogamy-of-entanglement property, as well as a variant of the direct product hardness property for coset states, that we define and prove, assuming post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions.

Unclonable cryptographic protocols often require a significant amount of quantum power for all involved parties. As such a model is unlikely to be realizable in the near future, we investigate in a third part *semi-quantum* unclonable cryptography, in which classical parties interact with a powerful quantum server. As most of the unclonable primitives are based on coset states, we construct a *remote coset states preparation protocol*, that enables a classical party to instruct the quantum server on how to construct coset states. Furthermore, this preparation is done in a blind way, in the sense that the server does not learn which coset states they prepared. This allows us to leverage this protocol to construct unclonable primitives in a semi-quantum way, assuming post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions.

Acknowledgments

In addition to discovering the world of academic research, with all its positive sides and, let's admit it, all its less-positive sides as well, I had the chance to meet many amazing people during this four-year journey.

I would first like to thank my supervisors: Céline for believing in me, giving me this opportunity of doing the thesis, and for all your useful advice; and Elham for welcoming me in this fantastic cross-countries team you've built, and for sharing your vision of research. I would also like to thank the members of my jury: the reviewers Prabhanjan and Christian; and Anne, Stacey, and Damien, for kindly accepting to be part of my jury.

I would then like to thank the people I have collaborated with during these past years, especially Huy who provided precious support throughout this thesis, but also Michael and Lucas — your energy is so communicative that it is a real pleasure to work with you guys.

Cryptomania. I had the chance to be part of two teams, the cryptography "CASCADE" team of ENS, and the quantum information team of LIP6. As my research lies at the intersection of cryptography and quantum information, being able to go back and forth (literally) between these two teams proved highly valuable. I spent most of the first half of my PhD (or at least, when we were not confined) in the classical world, at ENS. I'd like to thank the permanent members of the team, David, Brice, Phong, and Céline, and to extend these thanks to Lise-Marie and Valérie for all the administrative help you provided. I'd also like to thank the PhDs of the team, in particular Lénaïck for all the nice discussions we had, Léonard, Antoine, and Guirec for the coffee breaks' endless debates over pretty much everything; Paola and Henry for your sense of humour; Ky and Robert for welcoming me in your sunny corner in the open space; Michael for your passion of beers, and Hugo for your passion of food; and Nicolas for not giving up on FHE.

Quantumania. I spent most of the second half of my thesis in the quantum world, at Jussieu, where I stopped hearing about reductions, lattices, oil and vinegar - who would believe that cryptographers are so fond of salads ? - and where I started to be surrounded by $|\text{kets}\rangle$ and $\langle\text{bras}|$, superposition, and non-locality. On the scientific side, being in this environment helped me a lot to put a concrete meaning to some blurry notions that I used in my work. And on the personal side, I met amazing people, some of them became close friends, even outside the lab (yes, there is a world out there). I have a thought for the permanent members of the team: Damian, Eleni, Fred, Alex, Marco, and Elham; and I extend these thanks to Gizem. You're doing a wonderful job at making this place a good work environment. I also have a thought, of course, for the many people I've met there and are still taking care of this lab: Adriano, Manon, Kim, Nicolas, Yoann, Verena, Laura, and Paolo (even if the last four of you won't be there for long) among many others. I don't forget the ones who left us — for some it feels that it was aged ago - especially

Pascal, Valentina, and Raja.

The Clique and the 104. I have many thanks to make to the following persons. In addition to being inspiring researchers, they became friends I could rely on during the last months of this thesis.

To Dominik for backing me up when it comes to arguing that cryptography is a real science, Yao for your communicative laugh, Santiago and your surprisingly intense passion for wine, Matilde for giving me a new first name, and Uta for being the best office mate I could hope for in these last months. Also, to the countless games of Hanabi and The Crew we played these last two years, and to everything we shared. I hope we'll stay close in the future.

To my current office mates, Anton for your dry-humour, Vanessa for your passion for philosophy, Naomi for teaching me British slang, Cica (unofficial member of this office) for being such a badass, and again Uta. Thank you all for making this office a warm and welcoming place.

The seaside. Aux potes du Discord, Tom et Tristan évidemment, mais aussi à tous les autres. Pouvoir "brasser" avec vous quand je veux, et me plaindre d'à peu près tous mes problèmes est un fantastique outil thérapeutique, et ces pauses-cafés et soirées improvisées furent particulièrement salvatrices durant le (bien trop long) confinement.

À mes vieux potes de La Rochelle, et à toutes ces soirées durant lesquelles je peux complètement déconnecter du boulot et avoir l'impression de rajeunir de dix ans. Avoir été capable de rester aussi proches malgré la distance et les différentes directions qu'ont prises nos vies est précieux.

Et évidemment, *last but not least*, à mes parents, Philippe et Laurence, et à ma sœur Suzanne, merci d'être toujours présents.

Contents

Chapter

1 Introduction en Français

Faire de la cryptographie, c'est travailler avec l'information. La protéger, bien sûr, mais aussi la certifier, prouver que l'on possède une certaine information, etc. La cryptographie, et en particulier le chiffrement, est utilisé depuis des millénaires, mais on peut retracer la naissance de sa version "moderne" aux travaux de Claude Shannon, le créateur de la théorie de l'information. Cette nouvelle théorie conduisit à la notion de cryptographie prouvable, faite d'hypothèses, de théorèmes, de preuves. Lorsqu'on pense à cette science, la première application qui vient à l'esprit est le chiffrement: deux personnes communiquent de façon privée en utilisant une clef partagée, qui leur permet de chiffrer et de déchiffrer des messages, avec la promesse qu'aucun espion lisant les messages chiffrés ne peut accéder au contenu des messages. Ce modèle simple changea radicalement avec les travaux de Diffie and Hellman (1976), et la création de la cryptographie asymétrique. Basé sur des problèmes supposés difficiles à résoudre efficacement — et par là, nous entendons qu'il est pratiquement impossible de résoudre des grandes instances de ces problèmes — ils conçurent une manière de rendre publique assez d'information sur un système cryptographique pour que chacun puisse avoir accès à certaines fonctionnalités du système, tout en n'ayant aucun accès aux autres fonctionnalités.

Appliquée au chiffrement, cette méthode permet la mise en place de réseaux d'utilisateurs, tous capables de chiffrer des messages, tandis qu'un seul d'entre eux peut les déchiffrer. Elle permit également les premières signatures numériques, où une seule partie peut signer des messages, mais tout le monde peut vérifier les signatures. Finalement, ces nouvelles possibilités conduisirent à la création des protocoles de sécurité que nous utilisons maintenant chaque jour dans nos communications, en particulier sur internet.

L'informatique quantique. La mécanique quantique vise à décrire le comportement de particules infiniment petites. Ces particules présentent des propriétés physiques surprenantes, qui ne se rapportent à rien de ce que nous connaissons dans notre monde macroscopique. Au lieu d'être dans un état *classique* bien défini, comme les objets autour de nous, les particules quantiques (ou *états quantiques*) peuvent être dans une superposition de deux états classiques (ou plus !). Dans les années 80, certains scientifiques (Feynman (2018) and Manin (1980)) commencèrent à imaginer ce que la mécanique quantique pourrait apporter à l'informatique. Ce fut le début de l'information quantique et l'apparition de la notion d'ordinateur quantique. Une série de travaux montra que l'exploitation des propriétés des états quantiques pourrait apporter un avantage significatif pour certaines tâches spécifiques. Prenons par exemple le travail de Deutsch and Jozsa (1992). Les auteurs conçurent un algorithme permettant de décider si une fonction binaire inconnue $f : \{0,1\}^n \to \{0,1\}$ est soit constante, soit équilibrée. Pour résoudre ce problème avec une probabilité de 1, un algorithme classique nécessite au moins $2^{n-1}+1$ requêtes à la fonction f dans le pire des cas. En effet, pour être totalement sûr que la fonction est équilibrée, un

algorithme classique n'a pas d'autre choix que d'examiner différentes images de f, jusqu'à en trouver deux différentes, ce qui se produit dans le pire des cas à la $2^{n-1}+1$ requête (si au contraire, les $2^{n-1}+1$ images sont identiques, l'algorithme conclut que la fonction est constante). Cependant, l'algorithme de Deutsch and Jozsa le résout avec une seule requête. De manière similaire, Simon (1994) conçut un algorithme quantique de recherche de période — dont l'objectif est de trouver la période d'une fonction f — nécessitant un nombre de requêtes à f linéaire, tandis que le nombre de requêtes qu'effectue un algorithme classique pour la même tâche est exponentiel.

Les menaces des ordinateurs quantiques. Ces deux algorithmes quantiques offrent tous deux un *avantage quantique* exponentiel par rapport à leurs meilleurs homologues classiques, et on pourrait craindre que cet avantage puisse être exploité par des utilisateurs malveillants pour briser la sécurité des systèmes actuels. Ce n'est en fait pas le cas pour ces deux algorithmes (du moins pas directement), mais c'est le cas pour un troisième : le désormais célèbre algorithme de Shor (1999). Comme mentionné plus haut, la plupart de la sécurité des communications sur internet, y compris les transactions ou le transfert d'informations sensibles, repose sur un problème conjecturé difficile à résoudre efficacement. Ce *problème du logarithme discret* peut se formuler de la manière suivante. Étant donné un élément x d'un groupe G (par exemple l'ensemble des entiers modulo un certain nombre premier p), et un autre élément y de G généré par x, trouver l'entier s tel que $y = x^s$. Bien qu'il soit conjecturé qu'aucun algorithme classique ne soit capable de résoudre ce problème efficacement pour certains groupes, l'algorithme (quantique) de Shor fournit un moyen de le faire en temps polynomial. La menace que cet algorithme représente n'est pas encore concrète, car la construction d'un ordinateur quantique suffisamment puissant pour l'exécuter sur des instances de problèmes utilisés en pratique est une tâche extraordinairement difficile, et il semble improbable que cela soit réalisé dans un avenir proche. Cependant, la communauté cryptographique prend cette menace au sérieux, et envisage de remplacer les algorithmes utilisés dans les protocoles internet par de nouveaux, basés sur des problèmes supposés difficiles même pour les ordinateurs quantiques (par exemple, le problème d'apprentissage avec erreurs, proposé pour la première fois par Regev (2005)).

Exploiter l'avantage quantique. Comme nous venons de le voir, les ordinateurs quantiques peuvent poser un problème, mais ils nous fournissent également de nouveaux outils pour construire des primitives cryptographiques avec un niveau de sécurité inatteignable dans le monde classique. C'est le cas du célèbre protocole de distribution quantique de clés de Bennett and Brassard (1984), permettant à deux utilisateurs d'échanger une clé, qu'ils pourront utiliser plus tard pour d'autres applications cryptographiques. Ce protocole de distribution de clés possède une sécurité dite statistique : contrairement à ses homologues classiques, sa sécurité provient directement de la théorie de l'information quantique et ne repose sur aucune hypothèse calculatoire (comme la difficulté de certains problèmes).

Peut-être encore plus surprenant, la mécanique quantique nous permet de construire des systèmes cryptographiques qui n'existent tout simplement pas dans le monde classique. Dans cette thèse, nous étudions de manière approfondie les *primitives non clonables*, dont la sécurité repose sur le principe de non-clonage : aucun état quantique général ne peut être copié. Ce principe est la pierre angulaire de la *quantum money*, définie par Wiesner (1983). La quantum money permet à une banque de générer des pièces quantiques : des états quantiques vérifiables par tous, mais impossibles à cloner. Ce domaine n'a pas reçu

beaucoup d'attention depuis ce premier travail de Wiesner, du moins jusqu'à récemment. Au cours de la dernière décennie, de nouvelles primitives ont été imaginées, puis construites, et nous pouvons maintenant les catégoriser en trois grandes familles, suivant une idée de Broadbent, Jeffery, Lord, Podder, and Sundaram (2021) :

- la classe d'authenticité, où l'on retrouve la quantum money mentionnée plus haut, mais aussi les *tokenized signatures* (Ben-David and Sattath (2023)) : des jetons quantiques qui peuvent être consommés pour produire des signatures (dans le sens où l'état quantique est détruit durant le processus de signature) ;

- la classe d'information, où l'on trouve en particulier des schémas de chiffrement non clonables (Broadbent and Lord (2020)), qui peuvent être considérés comme des schémas de chiffrement réguliers avec des chiffrés quantiques non clonables ;

- et enfin la classe de fonctionnalité, un ensemble de primitives diverses qui peuvent toutes être englobées dans la large primitive de *copy-protection* (Aaronson (2009)): un moyen de produire des encodages quantiques de programmes classiques, qui peuvent être évalués sur n'importe quelle entrée, autant de fois que l'on veut, tout en étant protégés contre la copie.

Copy-protection de point functions dans le modèle standard. Nous nous concentrons sur la dernière catégorie, et plus spécifiquement sur la copy-protection des *point functions* : des fonctions f_y, indexées par une chaîne de bits — ou *point* — y, qui prennent en entrée une chaîne de bits x, et renvoient 1 si et seulement si $x = y$. Un tel schéma est plus précisément défini comme une procédure qui transforme un point y en un état quantique (que l'on appelle parfois encodage quantique), que l'on peut utiliser comme une sorte de clé dans une procédure d'évaluation pour calculer f_y sur n'importe quelle entrée, autant de fois qu'on le souhaite. La sécurité de cette primitive exige qu'aucun client malveillant, ayant reçu un encodage quantique d'un point y, ne puisse produire deux états qui pourraient tous deux être utilisés pour calculer f_y. De manière informelle, pour tester si les deux états produits par un tel adversaire sont utiles pour évaluer f_y, on sélectionne au hasard deux chaînes de bits x_1 et x_2, puis, en considérant chaque état comme un programme quantique, on exécute le premier état avec l'entrée x_1 le second avec x_2. Enfin, on vérifie si les résultats obtenus sont respectivement $f_y(x_1)$ et $f_y(x_2)$.

Les travaux antérieurs à cette thèse ont réalisé la copy-protection de point functions dans différents modèles. En particulier, les constructions de Aaronson, Liu, Liu, Zhandry, and Zhang (2021), Coladangelo, Majenz, and Poremba (2024), Ananth, Kaleoglu, Li, Liu, and Zhandry (2022), and Ananth, Kaleoglu, and Liu (2023) reposent sur des oracles puissants. D'autre part, peu de progrès ont été réalisés dans la construction de copy-protection de point functions dans le *modèle standard*, c'est-à-dire sans oracles.[1] Nous posons donc la question suivante :

Existe-t-il des schémas de copy-protection de point functions avec une sécurité négligeable[2] *dans le modèle standard ?*

[1] Dans des travaux parallèles aux nôtres, Ananth and Behera (2024) ont proposé une construction de copy-protection de point functions dans le modèle standard, en supposant deux conjectures.

[2] Par sécurité négligeable, nous entendons que la probabilité qu'un client malveillant réussisse la tâche décrite plus-haut est négligeable.

Cryptographie semi-quantique. La plupart des protocoles cryptographiques quantiques existants impliquent un réseau de clients, chacun possédant son propre ordinateur quantique, et communiquant entre eux via des canaux de communication quantiques. Bien que des progrès significatifs aient été réalisés dans la construction d'ordinateurs quantiques ces dernières années, la construction d'un ordinateur quantique, même pour des tâches simples, reste extrêmement difficile et coûteuse. Pour cette raison, il est intéressant de considérer des protocoles cryptographiques impliquant un réseau de clients classiques, interagissant avec un serveur quantique uniquement via des canaux de communication classiques. Une série de travaux a montré que, même dans ce modèle restreint, il est toujours possible de constuire des primitives inatteignables classiquement. Parmi ceux-ci, nous pouvons citer les travaux de Mahadev (2018) and Brakerski, Christiano, Mahadev, Vazirani, and Vidick (2018), permettant à un utilisateur classique de s'assurer que le serveur agit de manière honête, et les travaux de Gheorghiu and Vidick (2019) and Gheorghiu, Metger, and Poremba (2022) qui proposent tous deux une *préparation d'état à distance* pour certaines familles d'états quantiques: un protocole dans lequel un utilisateur classique donne des instructions à un serveur quantique sur la manière de préparer un certain état, d'une manière qui empêche le serveur d'apprendre quel état il a préparé.

Cette idée de "déquantification" des protocoles a finalement été appliquée aux primitives non clonables, avec les travaux de Radian and Sattath (2019) and Shmueli (2022a) concernant la quantum monney avec une banque classique, et le travail de Shmueli (2022b) concernant les tokenized signatures. Dans cette thèse, notre objectif est de déquantifier un ensemble plus large de primitives non clonables, y compris la copy-protection. La plupart de ces primitives utilisent des *coset states*: des états quantiques structurés qui possèdent des propriétés de non-clonabilité très fortes. Plus précisément, un coset state $|A_{s,s'}\rangle$ peut être vu comme la superposition des vecteurs d'un espace affine $A + s$, et des vecteurs de son *espace dual* $A^{\perp} + s'$. Les coset states présentent deux principales propriétés de non-clonabilité. La première, *direct product hardness*, stipule qu'il est difficile d'extraire un vecteur de l'espace régulier $A + s$, et un de l'espace dual $A^{\perp} + s'$. La seconde, *monogamy-of-entanglement*, stipule qu'étant donné un coset state $|A_{s,s'}\rangle$, il est difficile de produire deux états quantiques à partir desquels on pourrait extraire, en connaissant la description du sous-espace A, un vecteur de l'espace régulier (à partir du premier état) et un vecteur de l'espace dual (à partir du second état). Notre espoir est donc de construire un protocole de préparation d'état à distance pour les coset states, et de l'utiliser pour déquantifier les primitives non clonables. Concrètement, nous posons les questions suivantes:

Existe-t-il un protocole de préparation d'état à distance pour les coset states ?

Existe-t-il des schémas de copy-protection avec un vendeur classique et un client quantique ?

1.0.1 Plan et contributions

Cette thèse a donné lieu à trois articles :

- "Security Models and Cryptographic Protocols in a Quantum World" (à paraître dans *Foundations and Trends in Theoretical Computer Science*)
- "Towards Unclonable Cryptography in the Plain Model" (eprint:2023/1825)
- "Semi-quantum copy-protection and more" (publié dans *Theory of Cryptography Conference (2023)*, eprint:2023/244)

Comment lire cette thèse. Dans cette thèse, nous avons fait le choix de fournir une description informelle de nos résultats et des techniques utilisées, en omettant certains détails et certaines preuves techniques. Notre objectif est de donner aux lecteurs et aux lectrices une intuition sur la sécurité de nos constructions et sur les problèmes que nous avons rencontrés. Nous fournissons néanmoins des preuves importantes détaillées dans les annexes de cette thèse, et nous renvoyons les lecteurs et les lectrices aux articles Chevalier, Hermouet, and Vu (2023) and Chevalier, Hermouet, and Vu (2024b) pour toutes les preuves manquantes.

Les chapitres 5 et 6 sont indépendants. Bien que nous pensions qu'il apporte des éclairages utiles sur les primitives non clonables et les outils pour les construire, nous n'attendons pas des lecteurs et des lectrices qu'ils ou elles aient lu le chapitre 4 avant les chapitres 5 et 6, et nous fournissons de brèves explications sur les concepts que nous utilisons dans chaque chapitre.

Primitives non clonables. Le premier article ci-dessus (Chevalier, Hermouet, and Vu (2024a)) est une étude sur de nouvelles notions de sécurité pour la cryptographie dans un monde quantique, ainsi que des protocoles cryptographiques quantiques, y compris la distribution quantique de clés et les primitives non clonables. Dans le chapitre 4, nous approfondissons l'étude sur les primitives non clonables et fournissons une présentation poussée de ces primitives et de leur histoire.

Copy-protection et chiffrement non clonable dans le modèle standard. Dans le chapitre 5, nous répondons par l'affirmative à la première question mentionnée plus-haut. Nous présentons une construction pour la copy-protection des fonctions de point dans le modèle standard, en supposant deux conjectures que nous définissons et discutons dans le chapitre.

Nouvelles propriétés des coset states. Prouver la sécurité de notre schéma de copy-protection nécessite de définir et de prouver une nouvelle propriété de monogamy-of-entanglement pour les coset states. De manière similaire à la propriété de monogamy-of-entanglement originale, nous montrons qu'il est difficile de produire, étant donné un coset state, deux états quantiques satisfaisant une certaine propriété, avec une probabilité supérieure à $1/2$. La différence réside dans la propriété que nous considérons: nous montrons qu'il est difficile d'extraire deux vecteurs (un à partir de chaque état) appartenant au même espace — soit l'espace régulier, soit l'espace dual — à condition que ce coset soit inconnu au moment où les deux états sont générés, et qu'il soit seulement décidé et révélé publiquement après.

Nous prouvons également une généralisation de la propriété de direct product hardness. À partir d'un coset state $|A_{s,s'}\rangle$, il est difficile d'extraire un vecteur dans l'espace régulier $A+s$, et un vecteur dans l'espace dual $A^\perp+s'$, et il est également difficile d'extraire deux vecteurs *différents* dans le même espace (soit dans $A+s$, soit dans $A^\perp+s'$).

Sécurité des tokenized signatures. Nous utilisons ces nouvelles propriétés pour définir et prouver deux nouvelles propriétés de sécurité pour les tokenized signatures :

- *inforgeabilité non clonable*, qui demande qu'aucun adversaire ne puisse copier (même imparfaitement) un jeton de manière à ce que les deux copies puissent être utilisées pour signer un message arbitraire (inconnu au moment de la copie);

- *inforgeabilité forte*, qui demande qu'aucun adversaire, ayant reçu un jeton, ne puisse générer deux paires (message, signature) valides et différentes.

Cryptographie non clonable semi-quantique. Dans le chapitre 6, nous répondons par l'affirmative aux deuxième et troisième questions mentionnées plus-haut. En nous basant sur les protocoles de Gheorghiu and Vidick (2019), Gheorghiu, Metger, and Poremba (2022), and Shmueli (2022a), nous construisons un protocole de préparation d'état à distance pour les coset states. Dans le contexte de la copy-protection, ce protocole permet à un vendeur classique de donner des instructions à un client quantique pour construire des coset states aléatoires. À la fin du protocole, le vendeur reçoit la description des coset states, et un client honnête possède cet état en mémoire. De plus, la sécurité de ce protocole stipule que la direct product hardness et la monogamy-of-entanglement des coset states sont préservées. Pour la direct product hardness, cela signifie qu'aucun client malveillant, après avoir exécuté le protocole de préparation d'état à distance, ne peut produire des vecteurs dans les deux espaces, régulier et dual; cela se définit de manière analogue pour la monogamy-of-entanglement. En insérant ce protocole dans des constructions de copy-protection existantes, nous parvenons à obtenir une copy-protection semi-quantique pour diverses familles de fonctions.

Chapter

2 Introduction

Cryptography is about working with information. Protecting information, of course, but also certifying information, committing on information, proving statements, etc. Cryptography, and in particular encryption, has literally been used for thousands of years, but we can trace back modern cryptography to the works of Claude Shannon, the inventor of information theory. This led to the notion of provable cryptography, made of formal assumptions, theorems and proofs. When thinking of this science, the first application that comes to the mind is encryption: two persons communicate privately using a shared key, that allows them to encrypt and decrypt messages, with the promise that no spy, eavesdropping on them, is able to learn the content of these messages. This simple model radically changed with the works of Diffie and Hellman (1976), and Merkle (1978), and the creation of asymmetric cryptography.[1] Based on problems conjectured to be hard to solve efficiently — and by that, we mean that it is nearly impossible to solve large problems' instances — they designed a way to make public enough information on a cryptographic system such that anyone can perform some particular task, while being unable to perform others. Applied to encryption, this allows a network of users, all able to encrypt messages, while only one of them can decrypt them. This also permitted the first digital signatures, where only one party can sign messages, but everyone else has the ability to verify the signatures. This led to the creation of the security protocols we now use every day in our communications, in particular on the internet.

Quantum computing. Quantum mechanics aims to describe the behavior of infinitesimally small particles. These particles, indeed, exhibit surprising physical properties, that do not relate to anything we know in our macroscopic world. Instead of being in a well-defined *classical* state, as objects around us, quantum particles (or *quantum states*) can be in a superposition over two (or more !) classical states. In the 80s, some scientists (Feynman (2018) and Manin (1980)) started to imagine what quantum mechanics could bring to computer science. This was the beginning of quantum information, and the apparition of the notion of quantum computer. A series of works showed that harnessing the properties of quantum states could bring significant advantage regarding some specific tasks. Take the work of Deutsch and Jozsa (1992) for instance. The authors designed an algorithm deciding whether an unknown binary function $f : \{0,1\}^n \to \{0,1\}$ is either constant or balanced. In order to solve this problem with probability 1, a classical algorithm requires at least $2^{n-1}+1$ queries to the function f in the worst case. Indeed, to be entirely certain that the function is balanced, a classical algorithm has no choice but to look at

[1] Classified research by James Ellis, Clifford Cocks, and Matthew Williamson at the UK Government Communications Headquarters in the late 1960s and early 1970s achieved what is now known as RSA cryptosystem and Diffie-Hellman key-exchange. This work was made public by the UK government in 1997.

different images of f, until it finds two different ones, which happens in the worst case at the $2^{n-1}+1$ query (the algorithm learns that the function is constant if all the $2^{n-1}+1$ images are the same). The Deutsch and Jozsa algorithm however, solves it with only one query. Similarly, Simon (1994) designed quantum period finding algorithm — whose purpose is to find the period of a function f — requiring a linear amount of queries to f, while a classical algorithm for the same task provably requires an exponential number of queries.

The threats of quantum computers. Both these quantum algorithms provide an exponential *quantum advantage* over their best classical counterparts, and one might fear that such power could be leveraged by malicious users to break security of current systems. This is actually not the case for these two algorithms (at least not directly), but it is for a third one: the well-known Shor's algorithm (Shor (1999)). As mentioned above, most of the security of internet communications, including payment transactions or transfer of sensitive information, comes from a problem conjectured hard to solve efficiently. This *discrete logarithm problem* can be stated as follows. Given an element x from a group G (for instance the set of integers modulo some prime p), and some other element y of G, generated by x, find the integer s such that $y = x^s$. While no classical algorithm is able to solve this problem efficiently, the Shor's algorithm provides a way to do it in polynomial time. The threat caused by this algorithm is not concrete yet, as building a quantum computer powerful enough to run it on problem instances used in practice is an extraordinary difficult task, and it does not seem that it will be achieved in a near future. Notice though, that the cryptographic community takes this threat seriously, and consider replacing the algorithms used in internet protocols by new ones, based on problems believed to be hard even for quantum computers (e.g. the learning with errors problem, first proposed by Regev (2005)).

Harnessing quantum power. Quantum computers can thus pose a problem, but they also provide us new tools in order to construct cryptographic primitives with a security level unachievable in the classical world. This is the case of the famous quantum key distribution protocol of Bennett and Brassard (1984), allowing two parties to agree on a common key, that they can use later for other cryptographic applications. This key distribution (or key agreement) protocol achieves statistical security: contrary to its classical counterparts, its security comes directly from quantum information theory, and does not rely on any computational assumptions (the hardness of some problems).

Perhaps even more surprisingly, quantum mechanics allows us to construct cryptographic systems that simply do not exist in the classical world. In this thesis, we consider extensively the *unclonable primitives*, whose security relies on the no-cloning principle: no general quantum state can be copied. This principle is the corner stone of *quantum money*, defined by Wiesner (1983). Quantum money allows a bank to generate quantum coins: quantum states that are verifiable by everyone, but impossible to clone. This field did not receive a lot of attention since this first work of Wiesner, at least until recently. In the last decade, new primitives were imagined, and then constructed, and we can now categorize them in three main families, following an idea of Broadbent, Jeffery, Lord, Podder, and Sundaram (2021):

- the authenticity class, in which we find the aforementioned quantum money, but also the *tokenized signatures* (Ben-David and Sattath (2023)): quantum tokens that

can be consumed to produce signatures;

- the information class, in which we find in particular unclonable encryption schemes (Broadbent and Lord (2020)), that can be seen as regular encryption schemes with quantum unclonable ciphertexts;

- and finally the functionality class, a set of different primitives that can all be captured by the broad copy-protection primitive (Aaronson (2009)): a way of producing quantum encoding of classical programs, that can be evaluated on any input, as many times as one wants, while being protected against copy.

Copy-protection of point functions in the plain model. We focus on the last category, and more specifically on copy-protection of point functions: functions f_y, indexed by a bitstring — or *point* — y, that take as input a bitstring x, and return 1 if and only if $x = y$. Such a scheme is more precisely defined as a procedure that turns a point y into a quantum state (we sometimes write quantum encoding) that one can use as a sort of key in an evaluation procedure to compute f_y on any input, as many times as they want. The security of this primitive asks that no malicious client, given a quantum encoding of a point y, can produce two states that both can be used to compute f_y. Informally, testing whether the two states produced by such an adversary are helpful in evaluating f_y is done by sampling two bitstrings x_1 and x_2, using each state as a quantum program and running it on one input (x_1 for the first state, x_2 for the second one), and finally checking whether the outcome is $f_y(x_1)$ or $f_y(x_2)$ depending on the input.

Works prior to this thesis achieved copy-protection of point functions in different models. In particular, Aaronson, Liu, Liu, Zhandry, and Zhang (2021), Coladangelo, Majenz, and Poremba (2024), Ananth, Kaleoglu, Li, Liu, and Zhandry (2022), and Ananth, Kaleoglu, and Liu (2023) constructions rely on powerful oracles. On the other hand, little progress has been made in constructing copy-protection of point functions in the *plain model*, that is without oracles.[2] [3] We therefore ask the following question:

Do copy-protection schemes for point functions with negligible security against fully malicious adversaries exist in the plain model?

Semi-quantum cryptography. Most of existing quantum cryptographic protocols involve a network of clients, all owning their own quantum computer, and communicating among themselves through quantum channels. Although significant progress was made in building quantum computers in the last years, constructing a quantum computer, even for simple tasks, is still extremely difficult and expensive. For this reason, it might be interesting to consider cryptographic protocols involving a network of classical clients, interacting with a quantum server through classical channels only. Interestingly, a series of work showed that it is still possible to have constructions that are classically impossible, even in this restricted model. We can cite among them the seminal works of Mahadev (2018) and Brakerski, Christiano, Mahadev, Vazirani, and Vidick (2018) that allow a classical user to enforce the server to behave in a certain way, and the works of Gheorghiu

[2] In a concurrent work, Ananth and Behera (2024) provided a construction copy-protection of point functions in the plain model, assuming two conjectures.

[3] Broadbent, Jeffery, Lord, Podder, and Sundaram (2021) presented a construction in the plain model, without assumptions, secure against a weaker form of adversaries (so-called honest-malicious adversaries). In this thesis, we only consider fully malicious adversaries.

and Vidick (2019) and Gheorghiu, Metger, and Poremba (2022) that both propose a *remote state preparation* for some specific families of quantum states: a protocol in which a classical user instructs a quantum server on how to prepare a certain state, in a way that the server does not learn the state they prepared.

This idea of "dequantizing" protocols eventually reached unclonable primitives, with the works of Radian and Sattath (2019) and Shmueli (2022a) for quantum money with a classical bank, and the work of Shmueli (2022b) for tokenized signatures.[4] In this thesis, our goal is to dequantize a larger set of unclonable primitives, including copy-protection. Most of these primitives use *coset states*: structured quantum states that satisfy some strong unclonability properties. More precisely, a coset state $|A_{s,s'}\rangle$ can be seen as the superposition of the vectors of one coset $A+s$, and the vectors of its *dual coset* $A^{\perp}+s'$. The coset states feature two main unclonability properties. The first one, *direct product hardness*, states that it is hard to extract a vector in the regular coset $A+s$, and one in the dual one $A^{\perp}+s'$. The second one, *monogamy-of-entanglement*, states that given a coset state $|A_{s,s'}\rangle$, it is hard to produce two quantum states from which one can extract, given the description of the subspace A, a vector in the regular coset (from the first state), and one in the dual coset (from the second state). Our hope is then to construct a remote state preparation protocol for coset states, and to use it to dequantize unclonable primitives. Concretely, we ask the following questions:

Does remote state preparation for coset states exits ?

Does copy-protection schemes with a classical vendor and a quantum client exist ?

2.0.1 Roadmap and Contributions

This thesis led to three papers:

- "Security Models and Cryptographic Protocols in a Quantum World" (to appear in *Foundations and Trends in Theoretical Computer Science*). This paper provides a survey on cryptographic notions of security in the context of quantum adversaries; and on unclonable cryptographic primitives. The content of Chapter 4 is an extended version of the unclonable primitives part of this paper. The content of this part is largely the work of the author of this thesis.

- "Towards Unclonable Cryptography in the Plain Model" (eprint:2023/1825). This paper contains a part on copy-protection of point functions in the plain model, and a second part on semi-quantum copy-protection. It is a joint work with Quoc-Huy Vu, who also was a PhD student at the time of writing: the main contributor of the former part is the author of this thesis, and the main contributor of the second one is Quoc-Huy Vu.

- "Semi-quantum copy-protection and more" (published in *Theory of Cryptography Conference (2023)*, eprint:2023/244). This work improves the copy-protection of point functions construction presented in the previous paper, and is largely the work of the author of this thesis.

[4] The construction of Radian and Sattath (2019) is secure assuming the existence of noisy trapdoor claw free functions, and the ones of Shmueli (2022a) and Shmueli (2022b) are secure assuming the existence of indistinguishability obfuscation, and the sub-exponential hardness of the learning with errors problem.

How to read this thesis. In this thesis, we made the choice of providing high-level description of our results and techniques used, avoiding some details and technical proofs. Our goal is to give the reader an intuition on the security of our constructions, and on the problems we encountered. We provide detailed proofs in the appendices of this thesis, and refer the curious reader to the papers Chevalier, Hermouet, and Vu (2023) and Chevalier, Hermouet, and Vu (2024b) for all the missing proofs.

Chapters 5 and 6 are independent. Although we think it provides useful insights on unclonable primitives and tools to construct them, we do not expect the reader to have read Chapter 4 before Chapters 5 and 6, and provide brief explanations on the concepts we use in each chapter.

Unclonable primitives. The first paper above (Chevalier, Hermouet, and Vu (2024a)) is a survey on new security notions for cryptography in a quantum world, as well as quantum cryptographic protocols, including quantum key distribution and unclonable primitives. In Chapter 4, we extend the survey on unclonable primitives, and provide an extensive presentation of these primitives and their history.

Copy-protection and unclonable encryption in the plain model. In Chapter 5, we answer the first question above by the affirmative. We present a construction for copy-protection of point functions in the plain model, assuming post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions, and two new conjectures that we define and discuss in the chapter.

New properties for coset states. Proving the security of our copy-protection scheme requires us to define and prove a new monogamy-of-entanglement property for coset states. Similarly to the monogamy-of-entanglement property, we show that it is hard to produce, given a coset state, two quantum states satisfying some specific property, with probability greater than 1/2. The difference lies in the property we consider: we show that it is hard to extract two vectors (one from each state) that belong to the same coset — either the regular or the dual one — provided that this coset is unknown at the moment when the two states are generated, and only decided and publicly revealed afterward.

We also show a strengthening of the direct product hardness. From a coset state $|A_{s,s'}\rangle$, it is hard to extract a vector in the regular coset $A + s$, and one in the dual one $A^\perp + s'$, and it is also hard to extract two *different* vectors in the same coset (both in $A + s$ or both in $A^\perp + s'$).

Security of tokenized signatures. We use these new properties to define and prove two new security properties for tokenized signatures:

- *unclonable unforgeability*, that asks that no adversary can copy (even imperfectly) a token in such a way that the two copies can be used to sign an arbitrary message (unknown at the moment of the copy);

- *strong unforgeability*, that asks that no adversary given a token can generate two valid and different (message, signature) pairs.

Semi-quantum unclonable cryptography. In Chapter 6, we answer the second and third questions by the affirmative. Based on protocols of Gheorghiu and Vidick (2019), Gheorghiu, Metger, and Poremba (2022), and Shmueli (2022a), we construct a remote state preparation protocol for coset states. In the context of copy-protection, this protocol allows a classical vendor to blindly instruct a quantum client on how to construct random coset states. In the end of the protocol, the vendor receives the description of coset states, and an honest client holds these cosets states in their memory. Furthermore, the security of this protocol states that the direct product hardness and monogamy-of-entanglement properties of coset states are preserved. For the direct product hardness, it means no malicious client, after executing the remote state preparation protocol, can output vectors in both the regular and dual cosets; for the monogamy-of-entanglement, it is defined analogously. Our protocol is secure assuming post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Plugging this protocol in existing copy-protection constructions permits us to achieve semi-quantum copy-protection of various families of functions.

Chapter

3 Preliminaries

We start by providing some classical, quantum, and cryptographic preliminaries and notations.

Chapter content

3.1 Classical Preliminaries

3.1.1 Notations

Throughout this thesis, λ denotes the security parameter: a non-negative integer. The notation $\mathsf{negl}(\lambda)$ denotes any function f such that $f(\lambda) = \lambda^{-\omega(1)}$. We write that such a

function f is negligible. $\mathsf{poly}(\lambda)$ denotes any function f such that $f(\lambda) = \mathcal{O}(\lambda^c)$ for some $c > 0$. We write that such a function f is polynomial. We write that a function f is superpolynomial if $f(\lambda) \neq \mathcal{O}(\lambda^c)$ for any $c \in \mathbb{N}$. We write that a function f is logarithmic if $f(\lambda) = \mathcal{O}(\log(\lambda))$. $\mathsf{subexp}(\lambda)$ denotes any function f such that $f(\lambda) \neq (2^{\mathcal{O}(\lambda^c)})$ for any $0 < c < 1$. We write that such a function f is sub-exponential. We write that a function f is exponential if $f(\lambda) \neq \mathcal{O}(2^{\lambda^c})$ for any $c \in \mathbb{N}$. In this thesis, we often omit the dependency on λ when clear from the context. For instance, we sometimes write n instead of $n(\lambda)$ for a polynomial function n.

Sets and distributions. For $a, b \in \mathbb{R}$, we note $[a, b] = \{ x \in \mathbb{R} \ : \ a \leq x \leq b \}$ and $[\![a, b]\!] = \{ x \in \mathbb{Z} \ : \ a \leq x \leq b \}$. When sampling a value x from any distribution $\mathcal{D}$, we employ the notation $x \leftarrow\!\!\$\ \mathcal{D}$. When sampling uniformly at random a value x from a set S, we employ the notation $x \leftarrow\!\!\$\ S$. When sampling a value a from a probabilistic algorithm $\mathcal{A}$, we employ the notation $a \leftarrow \mathcal{A}$.

For any set $I \subseteq [\![1, n]\!]$, and any bitstring $x \in \{0,1\}^n$, we write $x_{|I}$ the bitstring in $\{0,1\}^n$ such that, for any $i \in [\![1, n]\!]$, the i-th component of $x_{|I}$ is x_i if $i \in I$, or 0 otherwise. We write $|S|$ to denote the cardinal of any set S, and $|x|$ to denote the weight of any bitstring x, that is the cardinal of $\{i \in [\![1, n]\!] \ : \ x_i \neq 0\}$.

3.1.2 Circuits and Algorithms

Throughout this thesis, we indifferently write circuits, programs, algorithms, procedures. By PPT we mean a polynomial-time non-uniform family of probabilistic (classical) circuits. We sometimes write that a (classical) circuit is efficient (or computationally bounded) if it is PPT. [1] For a probabilistic circuit C, we write $\mathsf{C}(x; r)$ to denote the computation of C on input x with fixed randomness r. We simply write $\mathsf{C}(x)$ to denote the computation of C on x with a uniformly random r.

An oracle is a function $\mathcal{O} : S \to T$ that can be queried by an algorithm. We write that an algorithm A queries an oracle $\mathcal{O}$ on input x to denote that A receives $\mathcal{O}(x)$ at the cost of one operation. Whenever an algorithm A has such oracle access to $\mathcal{O}$, we denote it by $\mathsf{A}^{\mathcal{O}}$.

$\top$ and $\bot$ are special symbols respectively denoting success and failure.

3.1.3 Distributions

We say that two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically (resp. computationally) indistinguishable and write $\mathcal{D}_1 \approx \mathcal{D}_2$ (resp. $\mathcal{D}_1 \approx_c \mathcal{D}_2$) if no algorithm (resp. no efficient algorithm) $\mathcal{A}$ can tell them apart:

$$|\Pr[\mathcal{A}(x) = 1 \ : \ x \leftarrow\!\!\$\ \mathcal{D}_1] - \Pr[\mathcal{A}(x) = 1 \ : \ x \leftarrow\!\!\$\ \mathcal{D}_2]| \leq \mathsf{negl}(\lambda)$$

We sometimes call the probability above *distinguishing advantage* of $\mathcal{A}$ with respect to $\mathcal{D}_1$ and $\mathcal{D}_2$.

For any distribution $\mathcal{D}$ over a set $\mathcal{X}$, we call *min-entropy* of $\mathcal{D}$ the following quantity.

$$-\log_2 \left(\max_{x \in \mathcal{X}} \Pr[x' = x \ : \ x' \leftarrow\!\!\$\ \mathcal{D}] \right)$$

[1] We also say that QPT algorithms (defined in the next section) are efficient.

3.2 Quantum Preliminaries

3.2.1 Basic Notations

Throughout this thesis, $\mathcal{H}$ denotes an arbitrary finite-dimensional Hilbert space, and use indices to differentiate between distinct spaces. We use the Dirac notations and write general pure quantum states as $|\psi\rangle$ or $|\phi\rangle$. We remove the $|\cdot\rangle$ and write ψ or ϕ to denote mixed states. Tr denotes the trace operator, and Tr_A the partial trace operator over a subsystem $\mathcal{H}_A$. We use subscript to denote a pure quantum state on multiple registers, e.g., $|\psi\rangle_{AB}$ or $|\psi\rangle_{12}$.

3.2.2 Usual States and Measurement Bases

The canonical basis of a single-qubit space is written as $\{|0\rangle, |1\rangle\}$ and denoted as rectilinear or computational basis. We use the notations $|+\rangle = (|0\rangle+|1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle-|1\rangle)/\sqrt{2}$, and denote the basis $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ as diagonal or Hadamard basis. Following these notations, we denote the single-qubit measurement $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ as a measurement in the rectilinear or computational basis, and the single-qubit measurement $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ as a measurement in the diagonal or Hadamard basis.

We denote by EPR pair, and write $|\phi^+\rangle$ the maximally entangled state $(|00\rangle + |11\rangle)/\sqrt{2}$.

3.2.3 Quantum Gates

We use U to denote a general unitary quantum gate, and the following notations for the usual gates.

- X defined as $\mathsf{X}|0\rangle = |1\rangle$ and $\mathsf{X}|1\rangle = |0\rangle$.
- Z defined as $\mathsf{Z}|0\rangle = |0\rangle$ and $\mathsf{Z}|1\rangle = -|1\rangle$.
- H defined as $\mathsf{H}|0\rangle = |+\rangle$ and $\mathsf{H}|1\rangle = |-\rangle$.

In addition, we define the Y-rotation gates as

$$\mathsf{R}_Y(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

for any $\theta \in [0, 2\pi]$. For any gate U, we define the controlled-U gates $\mathsf{Ctrl\text{-}U}$ as

$$\begin{aligned} \mathsf{Ctrl\text{-}U}|0\rangle|\psi\rangle &= |0\rangle|\psi\rangle \\ \mathsf{Ctrl\text{-}U}|1\rangle|\psi\rangle &= |1\rangle\mathsf{U}|\psi\rangle \end{aligned}$$

We write U^x for any $x \in \{0,1\}^n$ to denote $\bigotimes_{i=1}^n \mathsf{U}_i^x$.

3.2.4 Useful Theorems and Lemmas

We present in this subsection some important theorems and lemmas that we use in this thesis.

Throughout this thesis, for sake of readability, we use pure states as much as possible, using sometimes implicitly the following *purification theorem*.

Theorem 1 (Purification Theorem)**.** Let ψ be a mixed state on a Hilbert space $\mathcal{H}_A$. Then, there exists an auxiliary Hilbert space $\mathcal{H}_B$ and a pure state $|\psi\rangle_{AB} \in \mathcal{H}_A \otimes \mathcal{H}_B$ such that the reduced density matrix of $|\psi\rangle_{AB}$ on $\mathcal{H}_A$ is ψ. In other words,

$$\psi = \mathsf{Tr}_B\left(|\psi\rangle\langle\psi|_{AB}\right)$$

Theorem 2 (Gentle Measurement Lemma)**.** Let ψ be a mixed quantum state, and let E be a positive operator such that $0 \leq E \leq \mathbb{I}$. The probability of obtaining a measurement outcome corresponding to E is $\mathsf{Tr}(E\psi)$.

If $\mathsf{Tr}(E\psi) \geq 1-\epsilon$, then the post-measurement state ψ' after performing the measurement corresponding to E satisfies:

$$\left\|\psi - \sqrt{E}\psi\sqrt{E}\right\|_1 \leq 2\sqrt{\epsilon},$$

where $\|\cdot\|_1$ denotes the trace distance between the two states.

3.2.5 Circuits and Oracles

By QPT we mean a polynomial-time non-uniform family of quantum circuits. We sometimes write that a circuit is efficient (or computationally bounded) if it is QPT.

We refer to quantum implementation of a classical circuit (or program) C, as the unitary U_C defined as follows:

$$\mathsf{U}_\mathsf{C}\,|x\rangle_{\mathcal{X}}\,|y\rangle_{\mathcal{Y}} = |x\rangle_{\mathcal{X}}\,|\mathsf{C}(x)_{\mathcal{Y}} \oplus y\rangle$$

We denote the $\mathcal{X}$ register as preimage register, and the $\mathcal{Z}$ one as image register. We omit the dependency on C and simply write U when clear from the context. When writing that we run a classical circuit (or program) coherently on a quantum state $|\psi\rangle$, or in superposition over a quantum state $|\psi\rangle$, we mean that we execute the unitary operator U_C on $|\psi\rangle$ and measure the image register in the computational basis.

Similarly to the classical case, we denote a quantum circuit (or program) C having access to some oracle $\mathcal{O}$ as $\mathsf{C}^{\mathcal{O}}$.

3.2.6 Distances

Definition 1 (Norms)**.** Let $A \in \mathcal{L}(\mathcal{H})$ with singular values $\lambda_1, \ldots, \lambda_n \geq 0$. Then, the trace norm is defined as

$$\|A\|_1 = \sum_i \lambda_i\,.$$

Definition 2 (Trace distance)**.** For two quantum states $\psi, \sigma \in \mathrm{Pos}(\mathcal{H})$, the trace distance between them is

$$\Delta(\psi, \sigma) \coloneqq \frac{1}{2}\|\psi - \sigma\|_1.$$

Definition 3 (Approximate equality, Metger and Vidick (2021, Definition 2.8 and Definition 2.14))**.** We overload the symbol "$\approx$" in the following ways (leaving the dependence on the security parameter implicit in the quantities on the left):

1. **Complex numbers:** For $a, b \in \mathbb{C}$ we define:

$$a \approx_\epsilon b \iff |a-b| = O(\epsilon) + \mathsf{negl}(\lambda)\,.$$

2. **Operators:** For $A, B \in \mathcal{L}(\mathcal{H})$, we define:
$$A \approx_\epsilon B \iff \|A - B\|_1^2 = O(\epsilon) + \mathsf{negl}(\lambda)\,.$$
(We will most frequently use this for (possibly subnormalised) quantum states $A, B \in \mathrm{Pos}(\mathcal{H})$.)

3. **Operators on a state:** For $A, B \in \mathcal{L}(\mathcal{H})$ and $\psi \in \mathrm{Pos}(\mathcal{H})$, we define:
$$A \approx_{\epsilon,\psi} B \iff \mathrm{Tr}\Big[(A-B)^\dagger (A-B)\psi\Big] = O(\epsilon) + \mathsf{negl}(\lambda)\,.$$

4. **Computationally indistinguishable states:** For two (families of not necessarily normalised) states $\psi, \psi' \in \mathrm{Pos}(\mathcal{H})$ which are computationally indistinguishable up to δ (i.e., no QPT distinguisher has advantage exceeding δ in distinguishing ψ from ψ'[2]), we write:
$$\psi \overset{c}{\approx}_\delta \psi'\,.$$
We can also define computational indistinguishability with respect to *non-uniform* QPT algorithms with *quantum advice*, denoted by $\mathcal{A} \coloneqq \{\mathcal{A}_\lambda, \phi_\lambda\}_{\lambda\in\mathbb{N}}$, where each $\mathcal{A}_\lambda$ is the classical description of a $\mathsf{poly}(\lambda)$-size quantum circuit, and ϕ_λ is some (not necessarily efficiently computable) non-uniform $\mathsf{poly}(\lambda)$-qubit quantum advice. In this work, we implicitly consider computational indistinguishability with respect to non-uniform QPT adversaries with quantum advice, unless stated explicitly otherwise.

If we write $\approx_0$, we mean that the quantities are negligibly close. All asymptotic statements are understood to be in the limits $\epsilon \to 0$ and $\lambda \to \infty$.

3.3 Cryptographic Preliminaries

3.3.1 Cryptographic Security Games

Security of cryptographic primitives is often captured with games: an interactive protocol between an honest challenger, and a potentially dishonest player, or adversary. In the end of the game, depending on the transcript, the challenger decides whether the adversary wins the game or not, and we are interested in particular in their winning probability. These games are parameterized by a security parameter λ and therefore, the adversary's behavior, and the winning probability $p(\lambda)$, depends on this parameter. We typically ask that the winning probability of any adversary is asymptotically upper-bounded by some function, the most notable cases being $p(\lambda) \leq \mathsf{negl}(\lambda)$, and $p(\lambda) \leq 1/2 + \mathsf{negl}(\lambda)$. In this thesis, we abuse the notation and only write p instead of $p(\lambda)$ when mentioning this winning probability.

Uniform and non-uniform adversaries. We distinguish two types of adversaries: A *uniform adversary* $\mathcal{A}$ is an efficient algorithm $\mathcal{A}$ that takes as inputs of any size, while a *non-uniform adversary* is a family of efficient algorithms $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, each one designed for the specific security parameter λ. A non-uniform adversary can also be seen as a uniform one $\mathcal{A}$, augmented with some advice $\{|\psi_\lambda\rangle\}_{\lambda\in\mathbb{N}}$ (quantum if $\mathcal{A}$ is QPT, or classical is $\mathcal{A}$ is PPT): $\mathcal{A}$ receives as input the advice corresponding to the security parameter.

In this thesis, unless specified otherwise, we only consider non-uniform adversaries.

[2]A distinguisher $\mathcal{D}$ is a CPTP map from the input state to a classical single-qubit state (i.e. a distribution over $\{0,1\}$). The distinguishability is the trace distance between $\mathcal{D}(\psi)$ and $\mathcal{D}(\psi')$.

3.3.2 Symmetric Encryption Scheme

Definition 4 (Symmetric Encryption Scheme)**.** A symmetric encryption scheme for a message space $\mathcal{M}$ is composed of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined in the following way:

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter, and returns a classical key k.
- $c \leftarrow \mathsf{Enc}(\mathsf{k}, m)$. The encryption algorithm Enc takes as input a key k and a message $m \in \mathcal{M}$ and returns a ciphertext c.
- $m \leftarrow \mathsf{Dec}(\mathsf{k}, c)$. The decryption algorithm Dec takes as input a key k and a ciphertext c and returns a message m.

In the following, we assume that $\mathcal{M} \subseteq \{0,1\}^n$ for some integer n.
A symmetric encryption scheme must in addition satisfy the following properties.

Correctness. The encryption of a message must always decrypt to this message. More precisely, for all message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{k}, c) = m \ : \ \begin{array}{c} c \leftarrow \mathsf{Enc}(\mathsf{k}, m) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Indistinguishability. We distinguish one-time indistinguishability from many-time indistinguishability. For a private unclonable encryption scheme to have *one-time* indistinguishability, the encryption of two messages must be computationally indistinguishable. More formally, for all $m, m' \in \mathcal{M}$,

$$\begin{array}{rcl} \{\mathsf{Enc}(\mathsf{k}, m) & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \\ & \approx_c & \\ \{\mathsf{Enc}(\mathsf{k}, m') & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \end{array}$$

Many-time indistinguishability is defined analogously, except that the adversary is given this time a polynomial number of ciphertexts. More formally, for all $\kappa = \mathsf{poly}(\lambda)$, and all $m_1, \ldots, m_\kappa, m'_1, \ldots, m'_\kappa \in \mathcal{M}$,

$$\begin{array}{rcl} \{(\mathsf{Enc}(\mathsf{k}, m_1), \ldots, \mathsf{Enc}(\mathsf{k}, m_\kappa)) & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \\ & \approx_c & \\ \{(\mathsf{Enc}(\mathsf{k}, m'_1), \ldots, \mathsf{Enc}(\mathsf{k}, m'_\kappa)) & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \end{array}$$

3.3.3 Asymmetric Encryption Scheme

Definition 5 (Asymmetric Encryption Scheme)**.** An asymmetric encryption scheme for a message space $\mathcal{M}$ is composed of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined in the following way:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter, and returns a pair of secret and public keys $(\mathsf{sk}, \mathsf{pk})$.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. The encryption algorithm Enc takes as input a public key pk and a message $m \in \mathcal{M}$ and returns a ciphertext c.

- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. The decryption algorithm Dec takes as input a secret key sk and a ciphertext c and returns a message m.

In the following, we assume that $\mathcal{M} \subseteq \{0,1\}^n$ for some integer n.
An asymmetric encryption scheme must in addition satisfy the following properties.

Correctness. The encryption of a message must always decrypt to this message. More precisely, for all message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c) = m \ : \ \begin{array}{l} c \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Indistinguishability against chosen plaintext attacks. An asymmetric encryption scheme has indistinguishability against chosen plaintext attacks (IND-CPA) security if no efficient adversary $\mathcal{A}$ wins the following game with non-negligible advantage over $1/2$.

- A challenger samples a pair of keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- The challenger sends pk to $\mathcal{A}$.
- $\mathcal{A}$ sends a pair of messages (m_0, m_1) to the challenger.
- The challenger samples a bit $b \in \{0,1\}$.
- The challenger computes $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$.
- The challenger sends c to $\mathcal{A}$.

Let b^* denotes the outcome of $\mathcal{A}$. $\mathcal{A}$ wins if and only if $b^* = b$.
We sometimes denote this property as *semantic security*.

3.3.4 Digital Signature Scheme

Definition 6 (Digital Signature Scheme)**.** A digital signature scheme for a message space $\mathcal{M}$ is composed of three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ defined in the following way:

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter, and returns a pair of signing and verification keys $(\mathsf{sk}, \mathsf{vk})$.
- $sig \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$. The signature algorithm Sign takes as input a signing key sk and a message $m \in \mathcal{M}$ and returns a signature s.
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m, s)$. The verification algorithm Verify takes as input a verification key vk, a message m, and a signature s and returns a bit b, indicating whether s is a valid signature for m $(b = 1)$, or not $(b = 0)$.

A digital signature scheme must in addition satisfy the following properties.

Correctness. The signature of a message produced by the signing procedure must be valid. More precisely, for all message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m, s) = 1 \ : \ \begin{array}{l} s \leftarrow \mathsf{Sign}(\mathsf{sk}, m) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Existential (weak) unforgeability. A digital signature scheme has existential (weak) unforgeability if no efficient adversary $\mathcal{A}$ wins the following game with non-negligible probability.

- A challenger samples a pair of keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- The challenger sends pk to $\mathcal{A}$
- The challenger and $\mathcal{A}$ perform a number of rounds ($\mathcal{A}$ decides when to stop) as follows:
  - $\mathcal{A}$ sends a message m to the challenger.
  - The challenger runs $\mathsf{Sign}(\mathsf{sk}, m)$, and sends the outcome to $\mathcal{A}$.

Let (m, s) denotes the (message, signature) pair output by $\mathcal{A}$, and Q the set of messages sent by $\mathcal{A}$ to the challenger. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, m, s) = 1$, and $m \notin Q$.

Existential strong unforgeability. A digital signature scheme has existential strong unforgeability if no efficient adversary $\mathcal{A}$ wins the following game with non-negligible probability.

- A challenger samples a pair of keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- The challenger sends pk to $\mathcal{A}$
- The challenger and $\mathcal{A}$ perform a number of rounds ($\mathcal{A}$ decides when to stop) as follows:
  - $\mathcal{A}$ sends a message m to the challenger.
  - The challenger runs $\mathsf{Sign}(\mathsf{sk}, m)$, and sends the outcome to $\mathcal{A}$.

Let (m, s) denotes the (message, signature) pair output by $\mathcal{A}$, and Q the set of pairs (message, signature), where the messages are the ones sent by $\mathcal{A}$ to the challenger, and the signatures the challenger's answers. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, m, s) = 1$, and $(m, s) \notin Q$.

3.3.5 One-Way Functions and Pseudorandom Functions

Definition 7 (One-Way Function)**.** A function $f : \mathcal{X} \to \mathcal{Z}$ is a one-way function if there exists an efficient implementation of it and if it is hard to invert. That is, for any efficient adversary $\mathcal{A}$.

$$\Pr[\mathcal{A}(f(x)) = x \ : \ x \leftarrow\!\!\$\ \mathcal{X}] \leq \mathsf{negl}(\lambda)$$

Pseudorandom functions. A pseudorandom function (first defined by Goldreich, Goldwasser, and Micali (1984)) consists of a keyed function PRF_k and a set of keys $\mathcal{K}$ such that for a randomly chosen key $\mathsf{k} \leftarrow\!\!\$\ \mathcal{K}$, the output of the function $\mathsf{PRF}_\mathsf{k}(x)$ for any input x in the input space $\mathcal{X}$ "looks" random to an efficient adversary, even when given a polynomially many evaluations of $\mathsf{PRF}_\mathsf{k}(\cdot)$. Puncturable pseudorandom functions have an additional property that some keys can be generated *punctured* at some point, so that they allow to evaluate the pseudorandom function at all points except for the punctured

points. Furthermore, even with the punctured key, the pseudorandom function evaluation at a punctured point still looks random.

Punctured pseudorandom functions are originally introduced by Boneh and Waters (2013), Boyle, Goldwasser, and Ivan (2014), and Kiayias, Papadopoulos, Triandopoulos, and Zacharias (2013), who observed that it is possible to construct such puncturable pseudorandom functions for the construction from Goldreich, Goldwasser, and Micali (1984), which can be based on any one-way function (as shown by Håstad, Impagliazzo, Levin, and Luby (1999)). We provide formal definitions for pseudorandom function, and its puncturable variant, in the following.

Definition 8 (Pseudorandom Function)**.** A family of keyed functions $\{\mathsf{PRF}_{\mathcal{K}}\}_{\mathsf{k}\in\mathcal{K}}$ with domain $\mathcal{X}$ and codomain $\mathcal{Z} \subset \{0,1\}^{n_{\mathcal{Z}}}$ for some $n_{\mathcal{Z}} \in \mathbb{N}$ is a pseudorandom functions family if there is an efficient key generation procedure KeyGen, taking as input a security parameter, and outputting a key in $\mathcal{K}$, and if no efficient adversary $\mathcal{A}$ can win the following game with non-negligible advantage over $1/2$.

- A challenger samples a key $\mathsf{k} \leftarrow \mathsf{KeyGen}(\lambda)$.
- The challenger sets $\mathcal{O}_0 = \mathsf{PRF}_{\mathsf{k}}$, and samples $\mathcal{O}_1$ uniformly at random from the set of functions with domain $\mathcal{X}$ and codomain $\mathcal{Z}$.
- The challenger samples a bit $b \leftarrow\!\!\$\ \{0,1\}$.
- The challenger sends $\mathcal{O}_b$ to $\mathcal{A}$.

$\mathcal{A}$ outputs b^* and wins the game if $b^* = b$.

Definition 9 (Puncturable Pseudorandom Function)**.** A family of pseudorandom functions $\mathsf{PRF}_{\mathsf{k}} : \mathcal{X} \to \mathcal{Z}$ is *puncturable* if there is an addition key space $\mathcal{K}_p$ and three efficient algorithms $\mathsf{pPRF} = \langle \mathsf{KeyGen}, \mathsf{Puncture}, \mathsf{Eval} \rangle$ such that:

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes the security parameter 1^λ as input and outputs a key $\mathsf{k} \in \mathcal{K}$.
- $\mathsf{k}\{x\} \leftarrow \mathsf{Puncture}(\mathsf{k}, x)$. The puncturing algorithm $\mathsf{Puncture}$ takes as input a pseudorandom function key $\mathsf{k} \in \mathcal{K}$ and $x \in \mathcal{X}$, and outputs a key $\mathsf{k}\{x\} \in \mathcal{K}_p$.
- $y \leftarrow \mathsf{Eval}(\mathsf{k}\{x\}, x')$. The evaluation algorithm takes as input a punctured key $\mathsf{k}\{x\} \in \mathcal{K}_p$ and $x' \in \mathcal{X}$, and outputs a classical string $y \in \mathcal{Z}$.

We require the following properties.

- **Functionality preserved under puncturing.** For all $\lambda \in \mathbb{N}$, for all $x \in \mathcal{X}$,

$$\Pr\left[\forall x' \in \mathcal{X} \setminus \{x\} : \mathsf{Eval}(\mathsf{k}\{x\}, x') = \mathsf{Eval}(\mathsf{k}, x') \;\middle|\; \begin{array}{r} \mathsf{k} \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda) \\ \mathsf{k}\{x\} \leftarrow\!\!\$\ \mathsf{Puncture}(\mathsf{k}, x) \end{array}\right] = 1.$$

- **Pseudorandom at punctured points.** For every efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

and every $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr\left[\, 1 \leftarrow \mathcal{A}_2(\mathsf{k}\{x^*\}, z, \tau) \;\middle|\; \begin{array}{r} (x^*, \tau) \leftarrow \mathcal{A}_1(1^\lambda, \tau) \\ \mathsf{k} \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda) \\ \mathsf{k}\{x^*\} \leftarrow\!\!\$\ \mathsf{Puncture}(\mathsf{k}, x^*) \\ z \leftarrow \mathsf{Eval}(\mathsf{k}, x^*) \end{array} \right] - \Pr\left[\, 1 \leftarrow \mathcal{A}_2(\mathsf{k}\{x^*\}, z, \tau) \;\middle|\; \begin{array}{r} (x^*, \tau) \leftarrow \mathcal{A}_1(1^\lambda, \tau) \\ \mathsf{k} \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda) \\ \mathsf{k}\{x^*\} \leftarrow\!\!\$\ \mathsf{Puncture}(\mathsf{k}, x^*) \\ z \leftarrow\!\!\$\ \mathcal{Z} \end{array} \right]\right| \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness of KeyGen, $\mathsf{Puncture}$, and $\mathcal{A}_1$.

Denote the above probability as $\mathsf{Adv}^{\mathsf{pPRF}}(\lambda, \mathcal{A})$. We further say that a puncturable family of pseudorandom functions is δ-secure, for some concrete negligible function $\delta(\lambda)$, if for all efficient adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{pPRF}}(\lambda, \mathcal{A})$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

The following definitions are taken from Coladangelo, Liu, Liu, and Zhandry (2021).

Definition 10 (Statistically injective pseudorandom function)**.** A family of statistically injective (puncturable) pseudorandom functions with (negligible) failure probability $\varepsilon(\cdot)$ is a (puncturable) pseudorandom functions family $\{\mathsf{PRF}_\mathsf{k}\}_\mathsf{k}$ such that with probability $1 - \varepsilon(\lambda)$ over the random choice of key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$, we have that $\mathsf{PRF}_\mathsf{k}(\cdot)$ is injective.

Definition 11 (Extracting pseudorandom function)**.** A family of extracting (puncturable) pseudorandom functions with error $\varepsilon(\cdot)$ for min-entropy $k(\cdot)$ is a (puncturable) pseudorandom functions family $\{\mathsf{PRF}_\mathsf{k}\}_\mathsf{k}$ mapping $n_\mathcal{X}(\lambda)$ bits to $n_\mathcal{Z}(\lambda)$ bits such that for all $\lambda \in \mathbb{N}$, if X is any distribution over $n_\mathcal{X}(\lambda)$ bits with min-entropy greater than $k(\lambda)$, then the statistical distance between $(\mathsf{k}, \mathsf{PRF}_\mathsf{k}(X))$ and $(\mathsf{k}, r \leftarrow \{0,1\}^{m(\lambda)})$ is at most $\varepsilon(\cdot)$, where $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$.

3.3.6 Indistinguishable Obfuscation

Definition 12 (Indistinguishability Obfuscator Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang (2001))**.** A uniform PPT algorithm iO is called an indistinguishability obfuscator for a classical circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied:

- For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all input x, we have that

$$\Pr\Big[C'(x) = C(x) \;:\; C' \leftarrow \mathsf{iO}(1^\lambda, C)\Big] = 1.$$

- For any (not necessarily uniform) distinguisher $\mathcal{D}$, for all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then

$$\mathsf{Adv}^{\mathsf{io}}(1^\lambda, \mathcal{A}) = |\Pr\Big[\mathcal{D}(\mathsf{iO}(1^\lambda, C_0)) = 1\Big] - \Pr\Big[\mathcal{D}(\mathsf{iO}(1^\lambda, C_1)) = 1\Big]| \leq \mathsf{negl}(\lambda).$$

We further say that iO is δ-secure, for some concrete negligible function $\delta(\lambda)$, if for all QPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{io}}(\lambda, \mathcal{A})$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

3.3.7 Subspace Hiding Obfuscation

In Zhandry (2019) and Shmueli (2022a), it is shown that indistinguishability obfuscation schemes have the property of *subspace hiding*.

Lemma 1 (Zhandry (2019) and Shmueli (2022a))**.** Let iO an indistinguishability obfuscation scheme, and assume that injective one-way functions exist. Let $S = \{S_\lambda\}_{\lambda\in\mathbb{N}}$ a subspace $S \subseteq \mathbb{F}_2^\lambda$. For a subspace S', denote by $C_{S'}$ a classical circuit that checks membership in S'. Then, for every constant $\delta \in (0, 1]$ we have the following indistinguishability:

$$\{\mathsf{iO}(C_{S_\lambda})\}_{\lambda\in\mathbb{N}} \overset{c}{\approx}_0 \{\mathsf{iO}(C_T) \mid T \leftarrow\!\$\, \mathcal{S}_{S_\lambda}\}_{\lambda\in\mathbb{N}},$$

where $\mathcal{S}_{S_\lambda}$ is the set of all subspaces of dimension $\lambda - \lambda^\delta$ that contain S_λ.

3.3.8 Compute-and-Compare Obfuscation

Definition 13 (Point Functions)**.** Let $n \in \mathbb{N}$. A family of point functions $\{\mathsf{PF}_y\}_{y\in\{0,1\}^n}$, parameterized by points y is defined as follows.

$$\mathsf{PF}_y(x) = \begin{cases} 1 \text{ if } y = x \\ 0 \text{ otherwise.} \end{cases}$$

for any $x \in \{0, 1\}^n$.

Definition 14 (Compute-and-Compare Programs)**.** Let $\mathcal{X}, \mathcal{Z}, \mathcal{M}$ three sets. Given a function $f : \mathcal{X} \to \mathcal{Z}$ along with a lock-value $\ell \in \mathcal{Z}$ and a message $m \in \mathcal{M}$, we define the compute-and-compare program:

$$\mathsf{CC}[f, \ell, m](x) = \begin{cases} m & \text{if } f(x) = \ell, \\ \bot & \text{otherwise .} \end{cases}$$

When the function, lock-value, and message of a compute-and-compare program are not useful in the context, we will sometimes simply write CC in lieu of $\mathsf{CC}[f, \ell, m]$.

Definition 15 (Unpredictable Distribution)**.** Let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of distributions over pairs of the form $(\mathsf{CC}[f, \ell, m], \mathsf{aux})$ where $\mathsf{CC}[f, \ell, m]$ is a compute-and-compare program and aux is some (possibly quantum) auxiliary information. We say that $\mathcal{D}$ is an *unpredictable distribution* if for all efficient algorithm $\mathcal{A}$, we have that

$$\Pr\Big[\mathcal{A}(1^\lambda, f, \mathsf{aux}) = y \ : \ (\mathsf{CC}[f, \ell, m], \mathsf{aux}) \leftarrow \mathcal{D}_\lambda\Big] \leq \mathsf{negl}(\lambda).$$

Note that, in this thesis, we abuse the notation and write f to denote indifferently the function f or an efficient description of f.

Definition 16 (Sub-Exponentially Unpredictable Distribution)**.** Let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of distributions over pairs of the form $(\mathsf{CC}[f, \ell, m], \mathsf{aux})$ where $\mathsf{CC}[f, \ell, m]$ is a compute-and-compare program and aux is some (possibly quantum) auxiliary information. We say that $\mathcal{D}$ is a *sub-exponentially unpredictable distribution* if for all efficient algorithm $\mathcal{A}$, we have that

$$\Pr\Big[\mathcal{A}(1^\lambda, f, \mathsf{aux}) = y \ : \ (\mathsf{CC}[f, \ell, m], \mathsf{aux}) \leftarrow \mathcal{D}_\lambda\Big] \leq \frac{1}{\mathsf{subexp}(\lambda)}.$$

Definition 17 (Compute-and-Compare Obfuscator)**.** A PPT algorithm CC-Obf is said to be a compute-and-compare obfuscator for a family of unpredictable distributions $\mathcal{D} = \{\mathcal{D}_\lambda\}$ if:

- CC-Obf is functionality preserving: for all x,

$$\Pr\Big[\mathsf{CC\text{-}Obf}(1^\lambda, \mathsf{CC})(x) = \mathsf{CC}(x)\Big] \geq 1 - \mathsf{negl}(\lambda)$$

- CC-Obf has distributional indistinguishability: there exists an efficient simulator Sim such that

$$\Big\{\mathsf{CC\text{-}Obf}(1^\lambda, \mathsf{CC}), \mathsf{aux}\Big\} \approx_c \Big\{\mathsf{Sim}(1^\lambda, \mathsf{CC.param}), \mathsf{aux}\Big\},$$

  where $(\mathsf{CC}, \mathsf{aux}) \leftarrow \mathcal{D}_\lambda$, and CC.param denotes the input size, output size, and circuit size of CC, that are not required to be obfuscated.

Theorem 3 (Coladangelo, Liu, Liu, and Zhandry (2021))**.** Assuming post-quantum indistinguishable obfuscation, and the hardness of the learning with errors (LWE) problem, there exist compute-and-compare obfuscators for sub-exponentially unpredictable distributions.

3.3.9 Leveled Hybrid Quantum Fully Homomorphic Encryption

We rely on quantum fully homomorphic encryption of a specific structure, which was defined in Shmueli (2022a).

Definition 18 (Leveled Hybrid Quantum Fully Homomorphic Encryption)**.** A hybrid leveled quantum fully homomorphic encryption scheme is given by $\mathsf{QFHE} := \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{QOTP}, \mathsf{Eval}, \mathsf{Dec}\rangle$ with the following syntax:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell)$. A PPT algorithm that given a security parameter $\lambda \in \mathbb{N}$ and target circuit bound $\ell \in \mathbb{N}$, outputs a classical key pair $(\mathsf{sk}, \mathsf{pk})$.

- $|\psi\rangle^{(x,z)} \leftarrow \mathsf{QOTP}((x,z), |\psi\rangle)$. A QPT algorithm that takes as input an n-qubit quantum state $|\psi\rangle$ and classical strings as quantum one-time pad keys $x, z \in \{0,1\}^n$ and outputs its quantum one-time padded transformation $|\psi\rangle^{(x,z)} := \mathsf{Z}^z\mathsf{X}^x |\psi\rangle$. We sometimes call these one-time pad keys (x, z) the Pauli keys. Furthermore, if $|\psi\rangle$ is a *classical* string m, we ignore the Pauli key z and write $\mathsf{QOTP}(x, m)$ whose output is $x \oplus m$.

- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$. A PPT algorithm that takes as input a classical string $x \in \{0,1\}^*$ and the public key pk and outputs a classical ciphertext ct.

- $x \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$. A PPT algorithm that takes as input a classical ciphertext ct and the secret key sk and outputs a classical string x.

- $(|\phi\rangle^{(x',z')}, \mathsf{ct}_{x',z'}) \leftarrow \mathsf{Eval}(\mathsf{pk}, (|\psi\rangle^{(x,z)}, \mathsf{ct}_{x,z}), C)$. A QPT algorithm that takes as input a general quantum circuit C, a quantum one-time pad encrypted state $|\psi\rangle^{(x,z)}$ and a classical ciphertext $\mathsf{ct}_{x,z}$ of the pads. The evaluation outputs a quantum one-time padded encryption of some quantum state $|\phi\rangle$ encrypted under new keys (x', z') and a classical ciphertext $\mathsf{ct}_{x',z'}$.

The scheme satisfies the following.

- **Semantic Security.** For every polynomial $m(\cdot), \ell(\cdot)$, and QPT algorithm $\mathcal{A} := \{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left|\Pr\left[\begin{array}{l} 1 \leftarrow \mathcal{A}_2(m_0 \oplus x, \mathsf{ct}_x) \end{array} \middle| \begin{array}{r} (m_0, m_1) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda, 1^{\ell(\lambda)}) \\ x \leftarrow\!\!\$\ \{0,1\}^{m(\lambda)} \\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{pk}, x) \end{array}\right]\right.$$
$$\left.-\Pr\left[\begin{array}{l} 1 \leftarrow \mathcal{A}_2(m_1 \oplus x, \mathsf{ct}_x) \end{array} \middle| \begin{array}{r} (m_0, m_1) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda, 1^{\ell(\lambda)}) \\ x \leftarrow\!\!\$\ \{0,1\}^{m(\lambda)} \\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{pk}, x) \end{array}\right]\right| \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

  where $\lambda \in \mathbb{N}$ and $m_0, m_1 \in \{0,1\}^{m(\lambda)}$.

  - Denote the above probability as $\mathsf{Adv}^{\mathsf{QFHE}}(\lambda, \mathcal{A})$. We further say that QFHE is δ-secure, for some concrete negligible function $\delta(\lambda)$, if for all QPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{QFHE}}(\lambda, \mathcal{A})$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

- **Homomorphism.** For every polynomial $\ell := \ell(\lambda)$ there is a negligible function $\mathsf{negl}(\cdot)$ such that the following holds. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell)$, let x, z equal-length strings, let $\mathsf{ct}_{x,z} \leftarrow \mathsf{Enc}(\mathsf{pk}, (x, z))$, let C a quantum circuit of size $\leq \ell$, let $|\psi\rangle$ a $|x|$-qubit state input for C. Then, $\Delta\left(D_0, D_1\right) \leq \mathsf{negl}(\lambda)$, where D_0, D_1 are defined as follows.

  - D_0: The output state is $|\psi'\rangle \leftarrow C(|\psi\rangle)$.
  - D_1: The output state generated by first evaluating

$$\left(|\phi\rangle^{(x',z')}, \mathsf{ct}_{x',z'}\right) \leftarrow \mathsf{Eval}\left(\mathsf{pk}, \left(|\psi\rangle^{(x,z)}, \mathsf{ct}_{x,z}\right), C\right),$$

    and then decrypting $(x', z') \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_{x',z'})$, $|\phi\rangle \leftarrow \mathsf{QOTP}\left((x', z'), |\phi\rangle^{(x',z')}\right)$.

Chapter

4 Unclonable Cryptography

In this chapter, we provide a survey on unclonable primitives. This chapter is based on our following work: Chevalier, Hermouet, and Vu (2024a).

Chapter content

4.1 Introduction

Unclonable cryptography is a field that aims to create cryptographic primitives with unclonable properties. In order to give a comprehensive, yet informal definition of these primitives, we describe how they are defined. We first consider a (classical) cryptographic primitive. Take for instance a symmetric encryption scheme. Such a scheme is composed of what we call “objects”, all with “interesting crypto properties”. These objects are the keys — whose property is to be sufficient to decrypt ciphertext; and ciphertexts — whose

property it is to hide messages that one can recover only if they know the corresponding key.[1]

Defining an unclonable primitive out of a cryptographic primitive consists in defining a similar primitive where one of these objects is now a quantum state, and is unclonable in the sense that it is infeasible to produce two (even imperfect) copies of one of these quantum objects in such a way that the two copies conserve the object properties. Continuing with the example above, assume that we choose to create an unclonable primitive using the ciphertexts of an encryption scheme as unclonable objects. The unclonability property in this context requires that there is no way to generate two copies of a ciphertext (now represented by a quantum state) such that both copies leak information on the encrypted message, even given the secret key.

Of course, these unclonable primitives cannot be constructed using classical cryptography only, as in this case, the objects would be represented as classical data, which are always copyable. It is possible however, to think of these primitives in the context of quantum cryptography, where we can use quantum states to encode objects. The no-cloning theorem, that states that it is impossible to clone an unknown quantum state, gives the hint that it might be possible to construct unclonable primitives this way. This of course requires more involved tools — the no-cloning theorem, in particular, is not really usable as is — and we will present them in the following sections.

Several unclonable objects has been investigated in the literature, and we can establish the following hierarchy inspired by Broadbent, Jeffery, Lord, Podder, and Sundaram (2021). This hierarchy is composed of three classes of primitives. The first one, *the authenticity class*, produces unclonable objects which are not meant to store any data, but are verifiable in some way. The second class, *the information class*, is about unclonable objects storing data. Finally, the third class, *the functionality class*, produces unclonable program-like objects. Among these classes, we will consider different types of primitives. For each of these classes, we will present primitives featuring unclonability properties in the sense we described above, and we will also present primitives that feature a different sort of property: either the "quantum object" can be consumed to produce a verifiable certificate of destruction — we name this property *certified deletion* — or can be returned to some authority, who can then verify that the returned state is indeed the one which has been given in the first place — we name this property *revocability*.

This chapter is articulated as follows. In Sections 4.2.1 and 4.2.2, we first present, the two main families of quantum states used in unclonable cryptography: the BB84 and coset states. Then, we cover primitives in the authenticity class: quantum money in Section 4.3 and tokenized signature in Section 4.3.4. In Sections 4.4 and 4.5, we present unclonable encryption and certified deletion as primitives in the information class. We finally present primitives in the functionality class: copy-protection and secure software leasing in Sections 4.6 and 4.6.2.

4.2 BB84 and Coset States

As we already mentioned, constructing unclonable cryptography makes use of the fact that no unknown quantum state can be copied. Although this no-cloning theorem lies in the core of this field, it is by itself not enough to construct unclonable protocols. For that, we

[1] Remark that we do not consider messages as interesting cryptographic objects as they are merely bitstrings without any "interesting property".

need more specific states, that we can sample efficiently, with nice mathematical structures, and strong unclonable properties that we can leverage in the constructions. Two major families of states are considered in the literature, namely BB84 states — simpler — and coset states — entangled but more powerful quantum states. In this section, we present these two families, their properties, and show their similarities, and their differences.

4.2.1 BB84 States

In their seminal paper, Bennett and Brassard (1984) introduced a scheme in which two users agree on a key in an information secure way, that is, no malicious eavesdropper can get any relevant information on the key. This result shows a clear separation between classical and quantum cryptography, as such key-agreement protocol cannot exist with information theoretic security in the classical world. On top of it, this protocol strikes by its simplicity, and in particular, it requires manipulating only the following four single-qubit states: $|0\rangle$, $|1\rangle$, $|+\rangle$, and $|-\rangle$. Although this protocol was not the first one to harness the power of these four states[2], they were later called *the BB84 states*.

In this section, we present these states and some of their most interesting properties, and give intuition on why they can be so useful for quantum — and more particularly unclonable — cryptography.

Conjugate Coding

Conjugate Coding has been introduced by Wiesner in the 60s as a technique to protect classical information. However, as the field of quantum information was at this time not as developed as it is now, this technique did not get a lot of attention, and we had to wait until Wiesner (1983)'s paper to be published. The idea of conjugate coding is to encode classical information into quantum states, using two conjugate bases. We will denote these bases as the *computational basis* $\{|0\rangle, |1\rangle\}$ and the *Hadamard basis* $\{|+\rangle, |-\rangle\}$, where

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \text{ and } |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Definition 19 (BB84 State)**.** We denote by BB84 state any quantum state composed of $n \in \mathbb{N}^*$ registers, where each register is one of the following states: $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. We sometimes write n-long BB84 state to specify how many registers compose the state.

Remark that $|+\rangle = \mathsf{H}|0\rangle$, and $|-\rangle = \mathsf{H}|1\rangle$ — where we recall that H denotes the Hadamard gate. This allows us to describe any n-long BB84 state by two n-long bitstrings x and θ — that we sometimes call *the value* and *the basis* of the state respectively. We then denote a state described by (x, θ) as $|x^\theta\rangle$ and define it as

$$|x^\theta\rangle = \bigotimes_{i=1}^{n} \mathsf{H}^{\theta_i} |x_i\rangle$$

Sampling a BB84 state. In the rest of this section, and in the other parts of this thesis, we sometimes need to prepare random BB84 states as part of a cryptographic protocol. It turns out that it is more convenient for the security analysis of such protocols when the basis θ of a BB84 state is balanced, in the sense that it has the same number

[2]As we will see later, they were indeed used in Wiesner (1983).

Basis θ	0	0	1	1
Bit x	0	1	0	1
Encoding $\|x^\theta\rangle = \mathsf{H}^\theta \|x\rangle$	$\|0\rangle$	$\|1\rangle$	$\|+\rangle$	$\|-\rangle$

Table 4.1: Conjugate Coding of a 1-Long BB84 State.

of 0 and 1. Then, we say that sampling a BB84 state's description consists in sampling a uniformly random n-long bitstring x as the value of the state, and a balanced n-long bitstring θ as its basis More precisely, we sample the basis uniformly randomly in the set $\{\theta \in \{0,1\}^n \ : \ |\theta| = n/2\}$, which can be done efficiently by applying a random permutation to the bitstring $11\ldots100\ldots0$ starting with $n/2$ 'ones', and ending with $n/2$ 'zeros'.

Construction 1: Sampling a BB84 State's Description
Let $n \in \mathbb{N}$.

- Sample $x \leftarrow\!\!\$\ \{0,1\}^n$.
- Sample $\theta \leftarrow\!\!\$\ \{\theta \in \{0,1\}^n \ : \ |\theta| = n/2\}$.
- Return (x, θ).

Properties of BB84 States

We describe in the following some of the most useful properties of BB84 states in the context of unclonable cryptography. Except for the first one, all the properties we describe below aim to capture the unclonable nature of BB84 states.

Mutually unbiased basis. A first interesting property is that the two basis — computational and Hadamard — are as distant as possible as each others, as we can see on the Y plane of the Bloch sphere (whose illustration is given in Figure 4.1). This results in the following property. Consider a 1-long BB84 state, prepared in one of this basis. Measuring this state in the other basis, results in a uniformly random outcome, that is 0 or 1 with probability 1/2 each.

Direct product hardness. Consider a player Alice, playing a game in which she is given a 1-long random BB84 state $|x^\theta\rangle$, and must guess the value x. Crucially, she does not have any information on the basis θ. One trivial strategy would be then to measure the state in a random basis, computational or Hadamard, and to return the outcome. A simple analysis shows us that such a strategy yields the correct value with probability 3/4 (with probability 1 if the basis choice is the correct one, and 1/2 otherwise). This strategy is actually not the best one, as we can show that Alice can make a correct guess with probability close to 0.85 by measuring in a carefully chosen basis.[3]

Importantly, when the game is parallelized, that is when Alice is given an n-long random BB84 state and asked to guess its value, her winning probability becomes negligible.[4] In

[3]She precisely makes a correct guess with probability $\cos^2(\pi/8)$ by measuring the qubit in the basis $\{\cos(\pi/8)\,|0\rangle + \sin(\pi/8)\,|1\rangle\,, -\sin(\pi/8)\,|0\rangle + \cos(\pi/8)\,|1\rangle\}$.

[4]This is a consequence of a theorem by Bartusek and Khurana (2023), informally stating that learning the bits x_i such that $\theta_i = 0$ from a given random BB84 state $|x^\theta\rangle$ destroys all information on the XOR of

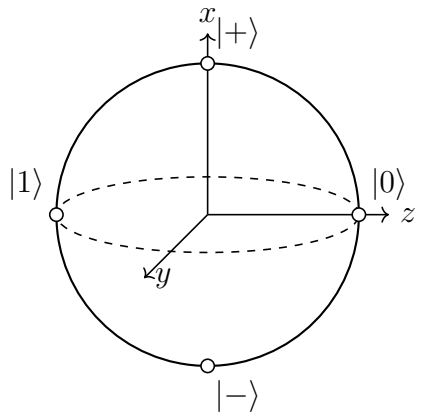

Figure 4.1: The Bloch sphere. The surface of the sphere represents the set of all qubits, in particular the four BB84 states. We denote as Y plan (or sometimes XZ plan) of the Bloch sphere the plan orthogonal to the y axis.

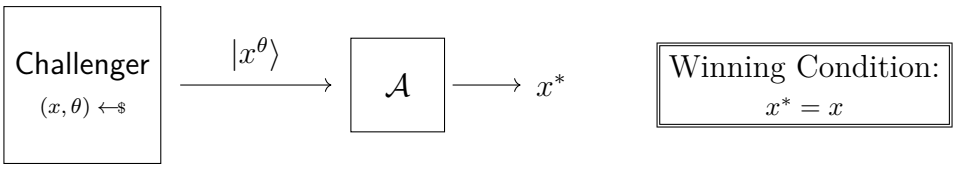

Figure 4.2: Direct product hardness game for BB84 states. (x, θ) is a random n-long BB84 state's description. No adversary wins this game with non-negligible probability in n.

the rest of this paper, we refer to this result on the upper bound of the winning probability of this game as *direct product hardness*.

Theorem 4 (Direct Product Hardness of BB84 States)**.** Let $n = \mathsf{poly}(\lambda)$. For any adversary $\mathcal{A}$,

$$\Pr\left[x^* = x \ : \ \begin{array}{l} x^* \leftarrow \mathcal{A}(|x^\theta\rangle) \\ \theta \leftarrow\$ \{\theta \in \{0,1\}^n \ : \ |\theta| = n/2\} \\ x \leftarrow\$ \{0,1\}^n \end{array}\right] \leq \mathsf{negl}(\lambda)$$

We provide an illustration of this theorem in Figure 4.2.

Monogamy-of-entanglement. This direct product hardness property already gives us some intuition on the unclonability of BB84 states. In the following, we present a stronger property highlighting the fact that BB84 states cannot be efficiently split between two parties. By that, we mean that it is impossible to share a BB84 state between two parties, in a way that both shares contain the value of the state.

We capture this property through a game — played by three players, Alice, Bob, and Charlie — that follows a template that we extensively use in the context of unclonable cryptography. A player Alice is given a random BB84 state, and is asked to split it. That is, she performs an arbitrary quantum operation on it, and returns a two-registers quantum state (intuitively, think that Alice tries to produce a copy of the original state, or something as close as possible). She then sends the first register to Bob, and the second

the bits x_i such that $\theta_i = 1$.

one to Charlie. Bob and Charlie are then given the basis of this BB84 state, and are both asked to return the value of the state. Tomamichel, Fehr, Kaniewski, and Wehner (2013) prove that Bob and Charlie cannot both answer correctly. Later, Culf and Vidick (2022) prove a stronger result, that is this game is still hard to win if Bob only needs to correctly guess the values of qubits that have been encoded in the rectilinear basis, and the same goes for Charlie for the qubits encoded in the diagonal basis. As this strong version of monogamy-of-entanglement implies the regular one, we simply denote the strong version as monogamy-of-entanglement in the following of this thesis.

Theorem 5 (Monogamy-Of-Entanglement of BB84 States)**.** Define the following game, between a challenger and a triple of adversaries $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and parametrized by a security parameter λ. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples an n-long BB84 state description: $r \leftarrow\!\$\ \{0,1\}^n$, and $\theta \leftarrow\!\$\ \{0,1\}^n$ such that $|\theta| = n/2$.
  - The challenger sends $|x^\theta\rangle$ to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite quantum state $|\psi^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\psi^*\rangle_1$ to $\mathcal{B}$ and $|\psi^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:** The challenger sends θ to both $\mathcal{B}$ and $\mathcal{C}$.

Let x_1^* denotes the output of $\mathcal{B}$, and x_2^* denotes the output of $\mathcal{C}$. Let I be the set of indices at which $\theta_i = 0$: $I = \{i \in [\![1,n]\!] \ :\ \theta_i = 0\}$. They win if and only if $x_1^* = x_{|I}$ and $x_2^* = x_{|\bar{I}}$.[5]

The monogamy-of-entanglement property states that no triple of adversaries can win this game with non-negligible probability. In other words, for any triple of adversaries $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$,

$$\Pr\left[\begin{array}{l|l} x_1^* = x_{|I} & x_1^* \leftarrow \mathcal{B}(|\psi^*\rangle_1, \theta), x_2^* \leftarrow \mathcal{C}(|\psi^*\rangle_2, \theta) \\ \quad \wedge & |\psi^*\rangle_{12} \leftarrow \mathcal{A}(|x^\theta\rangle) \\ x_2^* = x_{|\bar{I}} & \theta \leftarrow\!\$\ \{\theta \in \{0,1\}^n \ :\ |\theta| = n/2\} \\ & x \leftarrow\!\$\ \{0,1\}^n \end{array}\right] \leq \mathsf{negl}(\lambda)$$

We provide an illustration of this theorem in Figure 4.3.

Variants of monogamy-of-entanglement. Different variants of this monogamy-of-entanglement property have been investigated in the literature. Broadbent and Culf (2023) investigate for instance the effects of letting Charlie know the answer before answering himself, and Chevalier, Hermouet, and Vu (2024b) investigate the effects of instructing Bob and Charlie to return a vector corresponding to the same basis value (0 or 1). As we use it in the next chapter, we give a brief description of the latter variant in the following.

[5] Recall that, for any bitstring $y \in \{0,1\}^n$ — or equivalently any vector $y \in \mathbb{F}_2^n$ — and any subset $I \subseteq [\![1,n]\!]$, we write $y_{|I}$ to denote the bitstring in $\{0,1\}^n$ — or vector in $\mathbb{F}_2^n$ — whose components are y_i at indices $i \in I$, and 0 elsewhere.

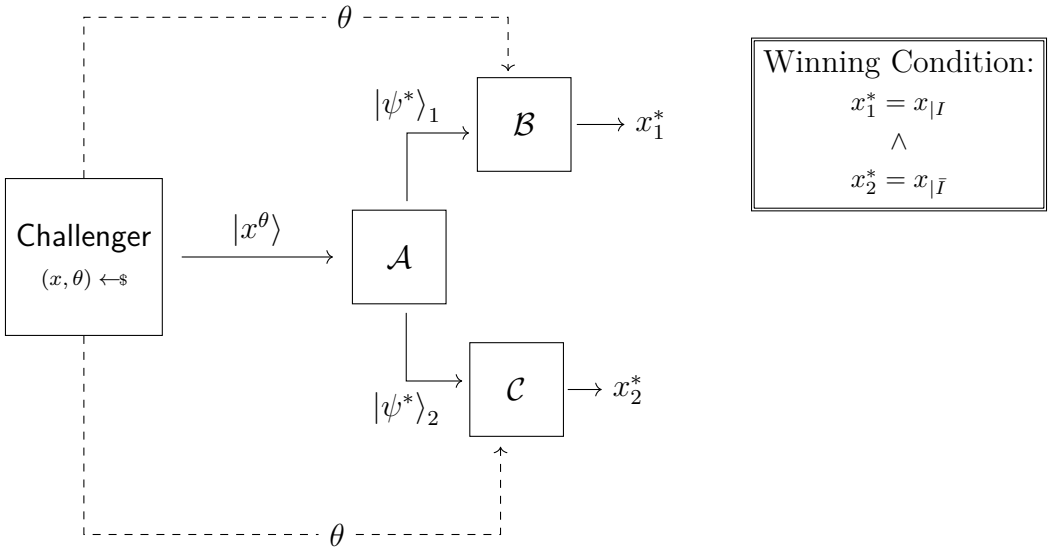

Figure 4.3: Monogamy-of-entanglement game for BB84 states. (x, θ) is a random n-long BB84 state's description. No triple of adversaries can win this game with non-negligible.

Consider the following game. Alice is given a random BB84 state $|x^\theta\rangle$, and a bit b. As before, she needs to split the state, and share it between Bob and Charlie. The latter are then asked to return the values of the qubits that have been encoded in the computational basis if $b = 0$, or in the Hadamard one if $b = 1$. As Alice receives b, she can measure the state in the corresponding basis, and forwards the outcome to Bob and Charlie, who can then both answer correctly with probability 1. Chevalier, Hermouet, and Vu (2024b) asks the following question.

What happens when b is not given to Alice, but instead to Bob and Charlie after they receive their part of the split state ?

Of course, a trivial strategy to win this game is for Alice to make a guess about b, and play the trivial game above. The winning probability is then the probability that she guesses correctly, that is $1/2$. We ask whether the entanglement shared between Bob and Charlie can help them go beyond this probability. It turns out that the answer is no: their winning probability is upper bounded by $1/2$.

4.2.2 Coset States

In this section, we present the coset states introduced by Coladangelo, Liu, Liu, and Zhandry (2021), and discuss how they compare to BB84 states, and how we can leverage them to achieve public unclonable cryptography. Throughout all this section, we sometimes omit the normalization factor of quantum states for sake of readability.

Definitions

We start by defining subspace and coset states.

Definition 20 (Subspace State)**.** Let A a linear subspace of $\mathbb{F}_2^n$. The corresponding subspace state, denoted by $|A\rangle$ is the following state.

$$|A\rangle = \sum_{a \in A} |a\rangle$$

Note that we often use a matrix of elements in $\mathbb{F}_2^n$ to represent the subspace A, and also denote this matrix as A. In this case, the columns of this matrix represent the basis of the subspace.

Definition 21 (Coset State)**.** Let A a linear subspace of $\mathbb{F}_2^n$, and s, s' two vectors of $\mathbb{F}_2^n$. The corresponding coset state, denoted by $|A_{s,s'}\rangle$, is the subspace state $|A\rangle$, quantum-one-time-padded by s and s'.

$$\begin{aligned} |A_{s,s'}\rangle &= \mathsf{X}^s\mathsf{Z}^{s'} \sum_{a\in A} |a\rangle \\ &= \sum_{a\in A} (-1)^{a\cdot s'} |a+s\rangle \end{aligned}$$

In the following, we denote the subspaces $A + s$ and $A^\perp + s'$ by regular and dual cosets respectively, where $A^\perp := \{x \in \mathbb{F}_2^n \mid x \cdot a = 0 \forall a \in A\}$.

An important property of a coset state is that applying a Hadamard gate to it yields its *dual coset state*.

Theorem 6 (Dual Coset State)**.** Let $|A_{s,s'}\rangle$ a coset state, and $A^\perp$ be the dual subspace of A in $\mathbb{F}_2^n$. Then, applying Hadamard gates to all the qubits of $|A_{s,s'}\rangle$ yields the dual coset state $|A^\perp_{s',s}\rangle$.

Proof. We give a brief proof of why this is true for a subspace state in the following. The proof extends easily to the coset states. Applying Hadamard gates to all the qubits of $|A\rangle$ yields

$$\begin{aligned} \mathsf{H}^{\otimes n} |A\rangle &= \mathsf{H}^{\otimes n} \sum_{a\in A} |a\rangle \\ &= \sum_{a\in A} \sum_{y\in\{0,1\}^n} (-1)^{y\cdot a} |y\rangle \\ &= \sum_{y\in\{0,1\}^n} \left(\sum_{a\in A} (-1)^{y\cdot a} \right) |y\rangle \end{aligned}$$

Then, fix $y \in \{0,1\}^n$, and let $\rho_y : A \to \{0,1\}$ be the function that maps $a \in A$ to $y \cdot a$. As ρ_y is linear, either $\rho_y(a) = 0$ for all $a \in A$, or the set of a that map to 0 and the set of a that map to 1 have the same cardinality. By definition of a dual subspace, the first case happens only when $y \in A^\perp$, and the second when $y \notin A^\perp$. Thus, the only vectors in the resulting state with non-zero weight are the ones belonging to $A^\perp$ — and they all have the same weight. □

Sampling coset states. As for BB84 states, we define what we mean by sampling a coset state (description) randomly. In particular, we will sample coset states whose subspace dimension is half the dimension of the whole space. When the coset state we sample is n-qubits long, we write that we sample an n-long coset state.

Construction 2: Sampling a Coset State's Description
Let $n = \mathbb{N}$.

- Sample a matrix $A \leftarrow\!\!\$\ \{A \in \mathbb{F}_2^{n\times n/2} \ : \ A \text{ is full-rank}\}$.

- Sample two vectors $s, s' \leftarrow\!\!\$\, \mathbb{F}_2^n$.
- Return (A, s, s').

BB84 States as Coset States

In this subsection, we give some intuition regarding how BB84 states and coset states compare. In particular, we show that BB84 states are actually coset states, with some restrictions.

When describing n-long BB84 states, we considered them as tensor products of n qubits. In this subsection, we rather look at them as superposition of classical states, and show that these classical states form a specific structure, namely they are vectors in an affine subspace. To see that, let us construct an n-long BB84 state $|x^\theta\rangle$ step by step. At first, prepare an initial state $|0\dots0\rangle$. Let i be the first index at which $\theta_i = 1$. The first step is to apply a Hadamard gate to the i-th qubit. The i-th qubit becomes $|0\rangle + |1\rangle$ — or $|0\rangle - |1\rangle$ depending on the value of x_i — and the others are left unchanged. Applying this gate then results in the following superposition of two classical states[6].

$$\begin{aligned} &|x_1 \dots x_{i-1}\rangle\, |0\rangle\, |x_{i+1} \dots x_n\rangle \\ +(-1)^{x_i}\, &|x_1 \dots x_{i-1}\rangle\, |1\rangle\, |x_{i+1} \dots x_n\rangle \end{aligned}$$

The second step is to apply another Hadamard gate, this time to the j-th qubit, where j denotes the second index at which $\theta_j = 1$. Similarly, the j-th qubit becomes $|0\rangle + (-1)^{x_j} |1\rangle$, and the rest of the qubits are left unchanged, resulting in the following superposition of four classical states

$$\begin{aligned} &|x_1 \dots x_{i-1}\rangle\, |0\rangle\, |x_{i+1} \dots x_{j-1}\rangle\, |0\rangle\, |x_{j+1} \dots x_n\rangle \\ +(-1)^{x_i}\, &|x_1 \dots x_{i-1}\rangle\, |1\rangle\, |x_{i+1} \dots x_{j-1}\rangle\, |0\rangle\, |x_{j+1} \dots x_n\rangle \\ +(-1)^{x_j}\, &|x_1 \dots x_{i-1}\rangle\, |0\rangle\, |x_{i+1} \dots x_{j-1}\rangle\, |1\rangle\, |x_{j+1} \dots x_n\rangle \\ +(-1)^{x_i \oplus x_j}\, &|x_1 \dots x_{i-1}\rangle\, |1\rangle\, |x_{i+1} \dots x_{j-1}\rangle\, |1\rangle\, |x_{j+1} \dots x_n\rangle \end{aligned}$$

Note in particular that, if we look at qubits at positions i and j in the classical states above, we have all possible pairs of classical qubits, that is $|00\rangle, |10\rangle, |01\rangle, |11\rangle$. This will be important in the following. The next steps consist in applying again Hadamard gates to the remaining qubits whose index correspond to the 1 in θ (and are not i or j). In the same way as described above, each application of Hadamard gate doubles the number of classical states in the superposition, and, after all Hadamard gates are applied, the qubits on which a Hadamard gate has been applied form all possible $n/2$-uple of classical qubits. The other half of the qubits are not affected by these gates, and remain unchanged. Remark that, to know whether the phase of a classical state $|u\rangle$ in the superposition is -1 or 1, we count the number of indices $k \in \{1, \dots, n\}$ such that (1) $u_k = 1$, and (2) we applied a Hadamard gate at index k (that is, $x_k = u_k = 1$). The parity of this number of indices gives us the phase: 1 if it is even, 0 if it is odd. In other words, the phase of a classical state $|u\rangle$ is $(-1)^{u_{|I} \cdot x_{|I}}$.[7]

These observations give another, equivalent, definition for BB84 states.

[6] In all this subsection, we remove the global phase for sake of clarity.

[7] Recall that, for any bitstring $y \in \{0,1\}^n$ — or equivalently any vector $y \in \mathbb{F}_2^n$ — and any subset $I \subseteq [\![1, n]\!]$, we write $y_{|I}$ to denote the bitstring in $\{0,1\}^n$ — or vector in $\mathbb{F}_2^n$ — whose components are y_i at indices $i \in I$, and 0 elsewhere.

Definition 22 (BB84 State)**.** Let $n \in \mathbb{N}$, and $x, \theta \in \{0,1\}^n$. Let $I = \{i \in [\![1, n]\!] \ : \ \theta_i = 1\}$. The BB84 state, described by (x, θ), is the state

$$\sum_u (-1)^{u_{|I} \cdot x_{|I}} |u\rangle$$

where we sum over the set $\{u \in \{0,1\}^n \ : \ u_i = x_i \ \forall i \notin I\}$.

Remark that this set is actually an affine space. By treating binary strings as vectors in $\mathbb{F}_2^n$, and denoting the linear space $\mathsf{span}\{e_i \ : \ i \in I\}$[8] by A,this set is $A + x_{|\bar{I}}$. This allows us to equivalently define the BB84 state described by (x, θ) as the following coset state:

$$\begin{aligned} &\sum_{a \in A} (-1)^{a_{|I} \cdot x_{|I}} |a + x\rangle \\ &\qquad = \mathsf{X}^{x_{|\bar{I}}} \mathsf{Z}^{x_{|I}} |A\rangle \\ &\qquad = |A_{x_{|\bar{I}}, x_{|I}}\rangle \end{aligned}$$

Thus, BB84 states are subfamily of coset states, where the subspace A is spanned by $n/2$ canonical vectors depending on the basis θ, and the two vectors s, s' are defined by both the value x, and the basis θ of the BB84 state.

Example 4.2.1. Consider the following BB84 state, with value $x = 10001110$ and basis $\theta = 00111100$. The "BB84 way" of writing this state is $|x^\theta\rangle = |10 + + - -10\rangle$. However, as we saw above, we can write it in a "coset way". Let $A \in (\mathbb{F}_2^n)^{n/2}$ the following matrix representing a subspace of $\mathbb{F}_2^n$, and define s and s' as $x_{|\bar{I}}$ and $x_{|I}$ respectively.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad s = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad s' = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Then we have $|x^\theta\rangle = |A_{s,s'}\rangle$.

Advantage of Coset States

We just saw that coset states are more complex versions of BB84 states. In particular, as mentioned by Coladangelo, Liu, Liu, and Zhandry (2021), they are highly entangled, while BB84 states are not. One natural question to ask then is why using coset states instead of BB84 states. The answer lies in the fact that one can give the ability to others to verify a coset state without revealing its description, while this is not possible with BB84 states. To see that, we define two games, where a player Alice is given a quantum state (a random BB84 state in the first game, and a random coset state in the second one) and must guess its description. To help her, she is also given access to a quantum oracle that verifies if the state given as input is the state she was given. Of course, a trivial strategy would be to simply go through all possible BB84 (or coset) states, until one passes the verification, but this would require an exponentially large number of queries. We rather wonder what is

[8] e_i denotes the i-th canonical vector of $\mathbb{F}_2^n$, that is $e_{i,i} = 1$, and $e_{i,j} = 0$ for all $j \neq i$.

the best strategy Alice can apply if she can only perform a polynomial number of queries. It turns out that, even with this restriction, Alice can completely learn the description of a BB84 state, while Coladangelo, Liu, Liu, and Zhandry (2021), based on the same kind of analysis for subspace state by Ben-David and Sattath (2023), showed that this is impossible with coset states (they actually prove a stronger result, namely *direct product hardness* that we present in the next subsection).

We describe in the following a strategy that allows Alice to win the first game with probability 1. As we will see in the next section, this strategy has been used by Lutomirski (2010) in the context of attacking a quantum money protocol.

Winning strategy for the BB84 game. At the beginning of the game, Alice is given a random n-long BB84 state $|x^\theta\rangle$. She is also given access to a verification oracle $\mathcal{O}_{x,\theta}$ that she can query a polynomial number of times only. We consider that this oracle implements the binary-outcome projective measurement $\{|x^\theta\rangle\langle x^\theta|, \mathbb{I} - |x^\theta\rangle\langle x^\theta|\}$. To learn the i-th bit of θ, Alice applies a X gate to the i-th qubit of $|x^\theta\rangle$, then queries the oracle with the resulting state. If $\theta_i = 0$, then the resulting state is orthogonal to the original one, hence the verification fails with probability 1. Otherwise, applying the X does not change the state (up to a global phase), hence the verification accepts with probability 1. Thus, depending on the oracle's answer, Alice knows with certainty the value of θ_i. She can thus undo the X, and measure the i-th qubit in the basis θ_i to get the value x_i. Now, she simply has to restore the i-th qubit (that is replacing it by $|x_i^{\theta_i}\rangle$) and apply this procedure to all the qubits.

Properties of Coset States

The coset states feature the same kind of unclonability as BB84 states, in the sense that they also have direct product hardness, and monogamy-of-entanglement properties. These properties are actually even stronger for coset states, as the winning probability in the game does not increase when the adversaries are allowed to query (a limited amount of time) membership oracles for the regular coset and its dual. Such a membership oracle, defined for an affine subspace $A + s$, takes as input a vector $x \in \mathbb{F}_2^n$, and returns 1 if $x \in A + s$, and 0 otherwise. In the rest of this thesis, we denote the pair of oracles $(\mathcal{O}_{A+s}, \mathcal{O}_{A^\perp+s'})$ by $\mathcal{O}_{A,s,s'}$.

Direct product hardness. Consider a game in which a player Alice, given a random coset state, and a limited membership oracle access (polynomial number of queries) to the regular and dual cosets, is asked to return one vector in the regular coset, and one in the dual coset. Asking for only one of these vectors would make the task easy, even without the oracles, as Alice would just have to measure the state in the computational basis to get a vector in the regular coset. However, such a measurement makes the coset state collapse to this vector state, and Alice would not be able to get a vector in the dual coset. The direct product hardness, proven by Coladangelo, Liu, Liu, and Zhandry (2021), property states that Alice cannot actually find a significantly better strategy in this game. In other words, winning this game is hard.

Theorem 7 (Direct Product Hardness of Coset States)**.** Let $n \in \mathbb{N}$. For any adversary $\mathcal{A}$,

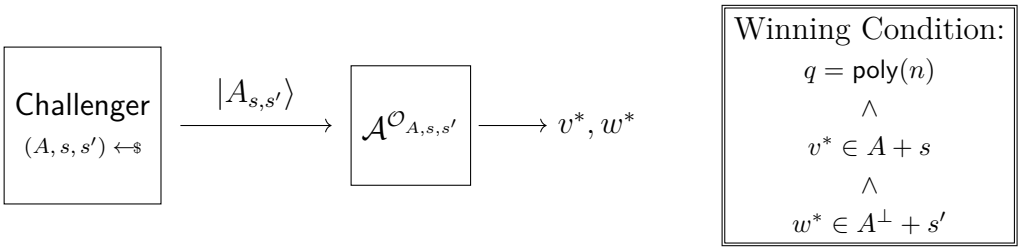

Figure 4.4: Direct product hardness game for coset states. (A, s, s') is a random n-long coset state's description, and q denotes the number of queries to the oracle $\mathcal{O}_{A,s,s'}$. No adversary wins this game with non-negligible probability in n.

let q denotes the number of queries $\mathcal{A}$ makes to the oracle $\mathcal{O}_{A,s,s'}$. Then,

$$\Pr\left[\begin{array}{ccl} q = \mathsf{poly}(n) & & \\ \wedge & & (v^*, w^*) \leftarrow \mathcal{A}^{\mathcal{O}_{A,s,s'}}(|x^\theta\rangle) \\ v^* \in A + s & : & s, s' \leftarrow\$ \mathbb{F}_2^n \\ \wedge & & A \leftarrow\$ \{A \in \mathbb{F}_2^{n \times n/2} \ : \ A \text{ is full-rank}\} \\ w^* \in A^\perp + s' & & \end{array}\right] \leq \mathsf{negl}(n)$$

We provide an illustration of this theorem in Figure 4.4.

Coladangelo, Liu, Liu, and Zhandry (2021) proved that the membership oracle $\mathcal{O}_{A,s,s'}$ can be instantiated using a pair of iO-obfuscated programs $\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'}$ defined as follows.

$$\mathsf{P}_{A+s}(u) = \begin{cases} 1 & \text{if } u \in A + s \\ 0 & \text{otherwise.} \end{cases} \qquad \mathsf{P}_{A^\perp+s'}(u) = \begin{cases} 1 & \text{if } u \in A^\perp + s' \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 8 ((Computational) Direct Product Hardness of Coset States)**.** Let $n \in \mathbb{N}$. For any QPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{ccl} v^* \in A + s & & (v^*, w^*) \leftarrow \mathcal{A}(|x^\theta\rangle, (\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'})) \\ \wedge & : & s, s' \leftarrow\$ \mathbb{F}_2^n \\ w^* \in A^\perp + s' & & A \leftarrow\$ \{A \in \mathbb{F}_2^{n \times n/2} \ : \ A \text{ is full-rank}\} \end{array}\right] \leq \mathsf{negl}(n)$$

Monogamy-of-entanglement of coset states. Consider the following game, played by Alice, Bob, and Charlie, all given limited oracle access (polynomial number of queries) to regular and dual coset's membership oracles. Alice, given a random coset state, is asked to split it, and share the outcome between Bob and Charlie. The latter are then given a description of the linear subspace corresponding to the coset state. Bob is asked to return a vector in the regular coset, and Charlie in the dual coset. Coladangelo, Liu, Liu, and Zhandry (2021) prove that a weaker version of the game — in which Bob and Charlie are both asked a pair of vectors in the regular and dual coset — was hard to win, and conjectured the hardness of the strong version. This was proven shortly after, by Culf and Vidick (2022).

Theorem 9 (Strong Monogamy-of-Entanglement)**.** Define the following game, between a challenger and a triple of adversaries $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and parametrized by a security parameter λ. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples an n-long coset state's description (A, s, s').
  - The challenger sends $|A_{s,s'}\rangle$ to $\mathcal{A}$.
  - The challenger provides membership oracle access $\mathcal{O}_{A,s,s'}$ to $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite quantum state $|\psi^*\rangle_{12}$.[9]
  - $\mathcal{A}$ sends $|\psi^*\rangle_1$ to $\mathcal{B}$ and $|\psi^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:** The challenger sends A to both $\mathcal{B}$ and $\mathcal{C}$.

Let v^* denotes the output of $\mathcal{B}$, and w^* denotes the output of $\mathcal{C}$. They win if and only if $v^* \in A + s$, $w^* \in A^\perp + s'$, and the number of queries they make to the oracle is polynomial in n.

The monogamy-of-entanglement property states that no triple of adversaries can win this game with non-negligible probability. In other words, for any triple of adversaries $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, let q denotes the number of queries $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ make to the oracle $\mathcal{O}_{A,s,s'}$.

$$\Pr\left[\begin{array}{c} q = \mathsf{poly}(n) \\ \wedge \\ u^* \in A + s \\ \wedge \\ w^* \in A^\perp + s' \end{array} \;:\; \begin{array}{l} u^* \leftarrow \mathcal{B}^{\mathcal{O}_{A,s,s'}}(|\psi^*\rangle_1, A), w^* \leftarrow \mathcal{C}^{\mathcal{O}_{A,s,s'}}(|\psi^*\rangle_2, A) \\ |\psi^*\rangle_{12} \leftarrow \mathcal{A}^{\mathcal{O}_{A,s,s'}}(|A_{s,s'}\rangle) \\ s, s' \leftarrow\$ \mathbb{F}_2^n \\ A \leftarrow\$ \{A \in \mathbb{F}_2^{n\times n/2} \;:\; A \text{ is full-rank}\} \end{array}\right] \leq \mathsf{negl}(n)$$

We provide an illustration of this theorem in Figure 4.5.

Similarly to direct product hardness, Coladangelo, Liu, Liu, and Zhandry (2021) proved that the oracle $\mathcal{O}_{A,s,s'}$ can be instantiated using a pair of iO-obfuscated programs $\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'}$.

Theorem 10 ((Computational) Strong Monogamy-of-Entanglement)**.** For any triple of QPT adversaries $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$,

$$\Pr\left[\begin{array}{c} \\ u^* \in A + s \\ \wedge \\ w^* \in A^\perp + s' \\ \\ \end{array} \;:\; \begin{array}{l} u^* \leftarrow \mathcal{B}(|\psi^*\rangle_1, A, (\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'})) \\ w^* \leftarrow \mathcal{C}(|\psi^*\rangle_2, A, (\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'})) \\ |\psi^*\rangle_{12} \leftarrow \mathcal{A}(|A_{s,s'}\rangle, (\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'})) \\ s, s' \leftarrow\$ \mathbb{F}_2^n \\ A \leftarrow\$ \{A \in \mathbb{F}_2^{n\times n/2} \;:\; A \text{ is full-rank}\} \end{array}\right] \leq \mathsf{negl}(n)$$

Variants of direct product hardness and monogamy-of-entanglement. In the rest of this thesis, we will see one variant of the direct product hardness property of coset states, and one of the monogamy-of-entanglement property. Chevalier, Hermouet, and Vu (2023) proposed a version of direct product hardness in which Alice can return two vectors in the same coset (regular or dual), but they need to be different. Analogously to

[9] We stress that the quantum state prepared by the adversary can be a mixed state, contrarily to what the notation could suggest. In the following of this thesis, for sake of simplicity, we abuse the notations and write quantum states prepared by adversaries with pure state notations (e.g. $|\psi^*\rangle_{12}$).

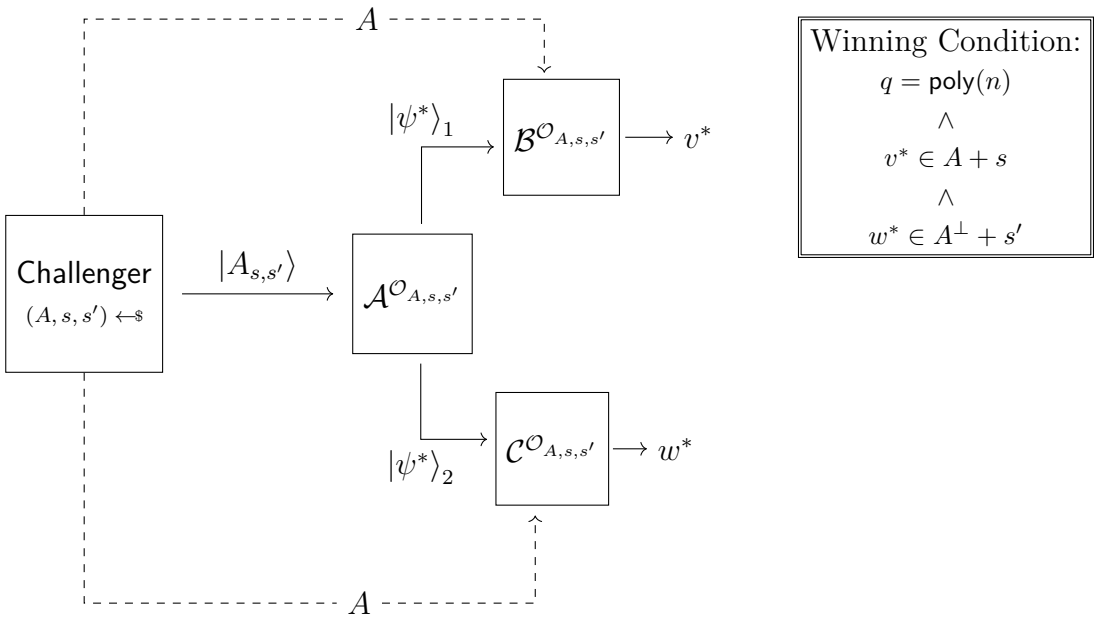

Figure 4.5: Strong monogamy-of-entanglement game for coset states. (A, s, s') is a random n-long coset state's description, and q denotes the number of queries to the oracle $\mathcal{O}_{A,s,s'}$. No adversary wins this game with non-negligible probability in n.

the variant of monogamy-of-entanglement for BB84 states mentioned in the last section, Chevalier, Hermouet, and Vu (2024b) proposed a version of monogamy-of-entanglement where Bob and Charlie must return a vector in the same coset, but this coset is decided during the challenge phase, after the splitting is done by Alice.

4.3 Quantum Money

We now have the tools to present the first class of unclonable primitives, the *authenticity class*. We present this class through two primitives, namely quantum money, whose purpose is to generate unclonable verifiable tokens; and tokenized-signature, a slightly different primitive that provides quantum tokens which can be consumed to generate signatures. In a first time, we present quantum money.

Quantum money can be seen as the digital equivalent of the physical coins and banknotes, assuming these objects are unclonable. When considering a payment infrastructure as a digital network, one can think of a network composed of multiple users, and a central, trusted authority — the *bank* — which maintains a database that maps every user with their balance. Now, when a user — the *client* — wants to transfer some money to another one — the *vendor* — the client simply informs the bank of this transaction. The latter then checks whether the client has enough money to do the transaction, and if yes, performs the transaction — that is, updates the database accordingly. One downside of such a network is that each transaction involves communications between the client, the vendor, and the bank. The bank, in particular, is involved in all the transactions in the network and has to make sure it can handle this huge amount of communications, in addition to maintaining its giant database.

Another way of implementing such a payment infrastructure is through decentralized currencies, like Bitcoin (Nakamoto (2008)). In these blockchain-based networks, users do

not have to trust the bank anymore, but the counterpart is that they all have to maintain the transactions' history of all the network. In addition, every user has to verify all the transactions themselves, implying a huge increase in the amount of communications in the network.

All this communication burden is something that can easily be avoided using physical money. Indeed, when Ava buys a croissant at the nearest bakery, she can simply pay with coins or with a banknote, and the merchant accepts them if they "look real enough". This verification that the merchant performs can be abstracted as a *local verification* procedure, in the sense that the merchant does not need to communicate with the bank. The verifier has some public information that allow them to verify the authenticity of a given coin or banknote. In this example, the verifier is the merchant and the public information is simply their knowledge on the look of a coin or a banknote, but we can also think of a machine to verify coins and banknotes in case of more expensive goods. The reason behind the security of such payment methods is that the coins and banknotes are created with some precise physical properties that make them hard to counterfeit. However, having these security systems *with decentralized digital payment systems* seems to be an impossible task. Indeed, in such an infrastructure, the coins and banknotes are represented with classical data, and thus are in theory easy to copy.

This argument, on the other hand, does not hold if we use quantum states to represent coins and banknotes. Indeed, as we saw earlier, according to the laws of quantum mechanics, it is impossible to copy (or clone) an unknown quantum state. The idea of quantum money is to take advantage of this property to build such "quantum coins" that would be impossible to counterfeit. This primitive has first been introduced by Wiesner (1983) under the of private quantum money — where the coins are verified by the bank — and has led to numerous versions, including public ones — where the users can verify the coins themselves.

In the following, we present private and public quantum money by defining them both and giving some examples of constructions, as well as a history of important results in the field. We also discuss difficulties of constructing public quantum money, through an attack proposed by Brodutch, Nagaj, Sattath, and Unruh (2014).

4.3.1 Private Quantum Money

Private quantum money can be seen as a middle-step between a digital payment system with a central bank, and a digital equivalent of the coins and banknotes with local verification. A vendor and a client using such a system would proceed in the following way. After the client sends a coin to the vendor, the vendor has to send the coin to the bank, and wait for the latter to tell them whether it is valid or not. While this does not seem extremely practical at first glance — indeed, we still need to go through the hassle of asking the bank for verification — it gives good insight about how unclonability of quantum states can be leveraged. Furthermore, we describe below how to simplify the work of the bank, allowing it to get rid of the aforementioned ledger: the giant database that maps every user to their balance.

Definitions

Definition 23 (Private Quantum Money Scheme)**.** A private quantum money protocol is made of a triple of algorithms $\langle \mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Verify} \rangle$ with the following properties:

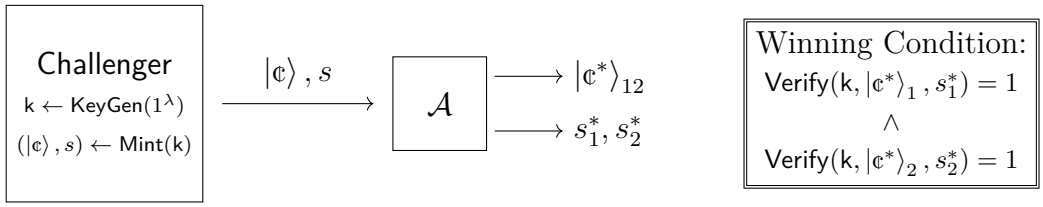

Figure 4.6: Unforgeability of a quantum money scheme. Unforgeability property states that no adversary $\mathcal{A}$ must be able to win this game with non-negligible probability.

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter 1^λ and returns a classical key k.
- $(|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{k})$. The minting algorithm Mint takes as input a key k and returns a quantum state $|\mathbb{c}\rangle$ as a "quantum coin", and a corresponding classical identifier s.
- $b \leftarrow \mathsf{Verify}(\mathsf{k}, |\mathbb{c}\rangle, s)$. The verification algorithm Verify takes as input a key k and an alleged (coin, identifier) pair $(|\mathbb{c}\rangle, s)$ and returns a bit b, indicating whether the coin is accepted ($b = 1$) or not ($b = 0$).

A private quantum money protocol must in addition satisfy the following properties.

Correctness. A coin that has been generated by the bank and that has not been modified must be accepted by the verification protocol, except with negligible probability. More precisely,

$$\Pr\left[\mathsf{Verify}(\mathsf{k}, |\mathbb{c}\rangle, s) = 1 \ : \ \begin{array}{c}(|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{k}) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Unforgeability. No malicious user, given a valid (coin, identifier) pair, must be able to produce two valid (coin, identifier) pairs. Formally, for any quantum adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{k}, |\mathbb{c}_1^*\rangle, s_1^*) = 1 \\ \wedge \\ \mathsf{Verify}(\mathsf{k}, |\mathbb{c}_2^*\rangle, s_2^*) = 1\end{array} \ : \ \begin{array}{c}(|\mathbb{c}_{12}^*\rangle, s_1^*, s_2^*) \leftarrow \mathcal{A}(|\mathbb{c}\rangle, s) \\ (|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{k}) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

We provide an illustration of this unforgeability property in Figure 4.6.

Wiesner's Quantum Money

In this part, we provide a high level description of the quantum money protocol introduced by Wiesner (1983). This protocol is the first quantum money protocol, and is a private one. In this protocol, all parties can communicate through a classical authenticated channel and a quantum channel. The key is a database, initially empty, mapping identifiers to BB84 state descriptions. In order to mint a coin, the bank first samples a random identifier, and an n-long BB84 state description, and stores them in the database. The resulting coin is the corresponding BB84 state. Verifying a (coin, identifier) pair simply consists in retrieving the BB84 state description corresponding to the identifier in the database, then measuring the coin in the description's basis, and verifying that the outcome matches the description's value. We provide a formal description below.

Construction 3: Wiesner's Private Quantum Money Scheme

$\mathsf{KeyGen}(1^\lambda)$:

- Set $n, m = \mathsf{poly}(\lambda)$.
- Initialize and return an empty database db, mapping elements in $\{0,1\}^m$ to elements in $\{0,1\}^n \times \{0,1\}^n$.

$\mathsf{Mint}(\mathsf{db})$:

- Sample $s \leftarrow\!\!\$\ \{0,1\}^m$.
- Sample an n-long BB84 state description, that is $x, \theta \leftarrow\!\!\$\ \{0,1\}^n$ such that $|\theta| = n/2$.
- Store (x, θ) at index s in db.
- Prepare $|\mathfrak{c}\rangle = |x^\theta\rangle$.
- Return $(s, |\mathfrak{c}\rangle)$.

$\mathsf{Verify}(\mathsf{db}, (s, |\mathfrak{c}\rangle))$:

- Retrieve (x, θ) at index s in db.
- Measure $|\mathfrak{c}\rangle$ in basis θ, let x' denote the outcome.
- Return 1 if $x' = x$, otherwise return 0.

This quantum money protocol was the first one being presented and, while being information theoretically sound, it had some issues. Firstly, this is a private scheme and thus, requires the vendor to send the coin to the bank, implying quantum costly communications. Also, this scheme requires the bank to maintain a huge database in order to store the BB84 description of all coins.

Solving the database issue. Bennett, Brassard, Breidbard, and Wiesner (1982) show how avoid the use of such a huge database at the cost of losing the information theoretic security, their scheme being only computationally secure. Roughly, they use a pseudorandom function F[10]— known only by the bank — and encode a banknote with identifier s according to $F(s)$. More precisely, let $\{F_k\}_k$ be a keyed family of pseudorandom functions. The key generation procedure then simply returns a random key k. In order to mint a new coin, the bank samples a random identifier s, then computes $(x\|\theta) := F_k(s)$ — that is, x and θ are respectively the first and second halves of $F_k(s)$. The quantum coin is produced in the same way as in the original protocol, that is, preparing a BB84 state whose description is (x, θ). Finally, the verifying procedure is also similar to the original one, except that, instead of looking for the BB84 description of the coin in a database, the bank computes it using $F_k(s)$. We give an example of a coin in this protocol in Table 4.2.

Although it removes the huge database problem, this scheme does not solve the first issue, namely the vendor still needs to ask the bank to verify banknotes. Note also that revealing the pseudorandom function F to the clients would allow them to perform this

[10]We recall that the security of a pseudorandom function states that its outcome is computationally indistinguishable from the one of a truly random function.

verification on their own but if one of them is malicious, they could use this information to forge money. Solving this issue implies constructing a public quantum money scheme.

s	01011001
$F(s)$	00111011
x	0011
θ	1011
$\lvert \mathfrak{c} \rangle$	$\lvert +0--\rangle$

Table 4.2: Example of a coin $|\mathfrak{c}\rangle$ minted using the quantum money protocol described above.

Attacks on the Wiesner Scheme

At the time Wiesner introduced his money scheme, quantum cryptography was at its very beginning, and no security proof was provided in the paper. Furthermore, there was no formal definition for quantum money, and for the security properties it should feature. Thus, it is not a surprise that several attacks on the Wiesner protocol have been presented afterward, assuming different settings in the protocol. In the following, we present two such attacks. What is crucial in these attacks is when and how the bank sends back a quantum coin after the verification. More precisely, whether it sends back the coin when the verification fails, and whether the coin sent back is the one sent for verification, or a fresh new one.

When the bank always returns the verified state. The first attack, by Lutomirski (2010), allows a malicious user to completely learn the classical description of a quantum coin, and therefore to produce as many as they want, assuming that the bank always returns the quantum coin to the vendor after the verification procedure is complete. Crucially, it must return it *even if the coin is not accepted*. The malicious user Alice proceeds in the following way. Given a quantum coin $|\mathfrak{c}\rangle = \bigotimes_{i=1}^{n} |x_i^{\theta_i}\rangle$, she applies an X gate to the first qubit of $|\mathfrak{c}\rangle$, and then sends the resulting state $|\mathfrak{c}^*\rangle$ to the bank for verification. If the qubit was $|0\rangle$ or $|1\rangle$, then, after the X gate is applied, the value of the qubit is flipped, hence it is rejected by the verification procedure with probability 1. If, on the other hand, the qubit was $|+\rangle$ or $|-\rangle$, the X gate has no effect on it (up to a change in the phase), hence it is accepted by the verification algorithm with probability 1. Thus, depending on the answer of the bank regarding the validity of $|\mathfrak{c}^*\rangle$, Alice learns in which basis the first qubit was prepared. Furthermore, as $|\mathfrak{c}^*\rangle$ is either accepted or rejected *with probability* 1, $|\mathfrak{c}^*\rangle$ is not modified by the verification procedure, and she receives the same state $|\mathfrak{c}^*\rangle$ back from the bank. Then, she simply undoes the X gate operation, and repeats the same procedure with the other qubits, one by one, until she completely learns the basis θ. Finally, once it is done, she measures $|\mathfrak{c}\rangle$ in basis θ to learn the value of the coin, allowing her to produce as many copies of it as she wants.

When the bank returns the verified state only on acceptance. The second attack was given by Brodutch, Nagaj, Sattath, and Unruh (2014). Similarly, it allows an adversary to completely learn the description of a quantum coin, but here, we do not even need to assume that the bank returns the state when the procedure rejects it, only that it returns the state if the procedure accepts. The attack is based on the bomb testing procedure of

Kwiat, Weinfurter, Herzog, Zeilinger, and Kasevich (1995) and exploits the fact that the verification procedure in the Wiesner quantum money scheme boils down to applying the projective measurement $\{|x_i^{\theta_i}\rangle\langle x_i^{\theta_i}|, \mathbb{I} - |x_i^{\theta_i}\rangle\langle x_i^{\theta_i}|\}$ on each qubit $|\mathfrak{c}\rangle_i$ of the coin. The attack is the following. Let ε be a small angle, and $N := \pi/(2\varepsilon)$. A malicious user Alice who wants to learn the description of $|\mathfrak{c}\rangle_i$ — that is (x_i, θ_i) — first creates a "probe" quantum state $|\psi\rangle$, initially $|0\rangle$. Then, she applies a Y-rotation of angle ε (that is the gate $\mathsf{R}_Y(\varepsilon)$) to the probe, performs a CNOT to $|\mathfrak{c}\rangle_i$, controlled by $|\psi\rangle$, and sends the resulting quantum coin to the bank for validation. She repeats these steps N times, and finally measures the probe state in the rectilinear basis.

We distinguish two cases, depending on whether the basis θ_i is the rectilinear or diagonal one. The simplest case is when θ_i is the diagonal basis, meaning that $|\mathfrak{c}\rangle_i$ is either $|+\rangle$ or $|-\rangle$. In this case, applying the CNOT results in $(\alpha\,|0\rangle + \beta\,|1\rangle) \otimes |+\rangle$ or $(\alpha\,|0\rangle - \beta\,|1\rangle) \otimes |-\rangle$ — where α and β are respectively $\cos(\varepsilon)$ and $\sin(\varepsilon)$. In both cases, the probe and the coin are not entangled, and the quantum coin is not modified, hence the bank accepts the coin with probability 1, and the probe is left unchanged by the verification procedure. Thus, repeating the steps above increases little by little β, while decreasing α. After N repetitions, the probe ends up in state $|1\rangle$.

In the other case, when θ_i is the rectilinear basis, $|\mathfrak{c}\rangle_i$ is $|x_i\rangle$. Then, applying the CNOT results in the state $\alpha\,|0\rangle\,|x_i\rangle + \beta\,|1\rangle\,|\bar{x_i}\rangle$ — where again, α and β are respectively $\cos(\varepsilon)$ and $\sin(\varepsilon)$. In this case, crucially, the probe and the coin are entangled, and the verification procedure accepts with probability α^2. If the procedure accepts, because of the entanglement, the two registers collapse to $|0\rangle\,|x_i\rangle$, which is exactly the initial state before the rotation. Thus, repeating these steps N times results, conditioned on the verification procedure accepting at each repetition, in this same state $|0\rangle\,|x_i\rangle$. Thus, after the N repetitions, when Alice measures the probe, the outcome tells her whether the coin is encoded in the rectilinear or the diagonal basis. In other words, she learns θ_i, and she can then measure $|\mathfrak{c}\rangle_i$ in the corresponding basis to learn x_i.

Of course, this attack relies on the fact that the verification procedure accepts N times. Loosely speaking, the analysis provided in Brodutch, Nagaj, Sattath, and Unruh (2014) shows that, for this to happen with non-negligible probability in n (where n in the number of qubits of the coin), N only needs to be polynomial in n.

4.3.2 Public Quantum Money

Public quantum money is the closest digital equivalent to a system with physical coins that a central bank only can produce, but that everyone can verify locally. Such a scheme features the same triple of algorithms — key generation, minting, and verifying procedure — as its private counterpart, with as a main difference two distinct keys to respectively mint, and verify coins. In this public version, it is also important to mention that the verification procedure must not destroy the state (or at least returns a valid coin). Indeed, consider a system where the coin is destroyed after verification, a vendor who wants to verify a coin given by a client would destroy it in the process, rendering it unusable. The reason it was not necessary in the private version is that the bank makes the verification *and mint coins*. So, even if the coin was destroyed, the bank would still be able to produce a new (coin, identifier) pair and send it back the whoever asked for the verification.

Public quantum money is arguably one of the most interesting unclonable primitive, as it has an immediate use-case. However, what perhaps makes it even more interesting is that we still do not now any provably secure public quantum money scheme (relying on

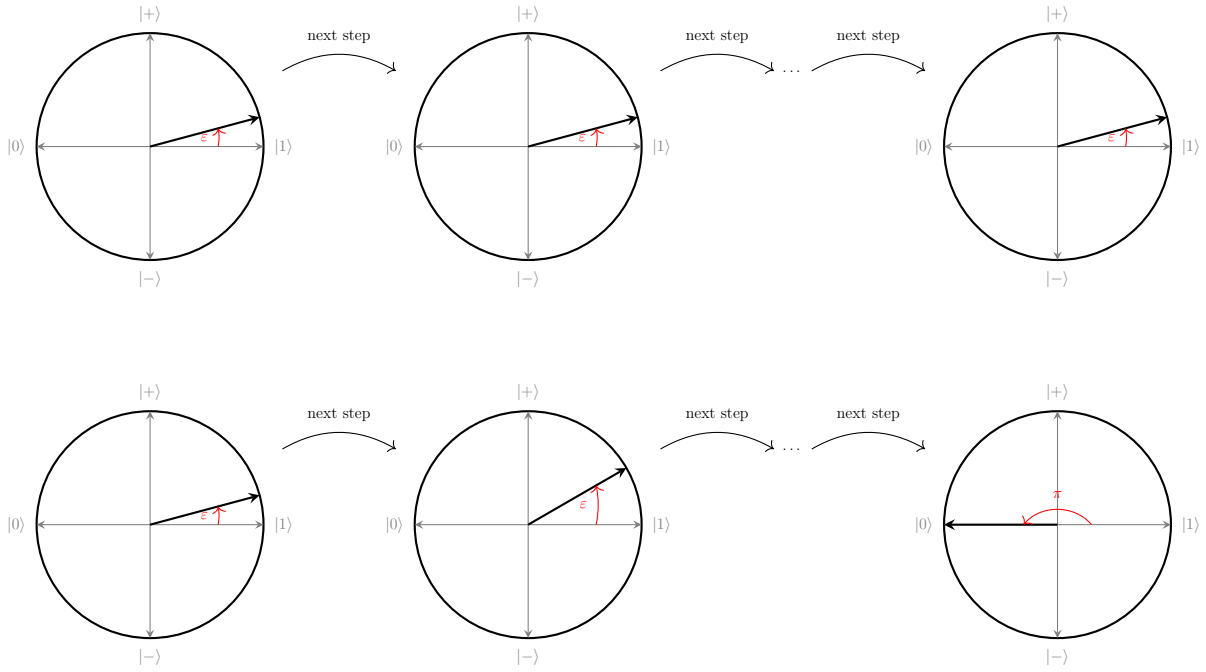

Figure 4.7: Probe state during the bomb testing attack on Wiesner protocol. Each circle represents the XZ plan of the Bloch sphere, and the thick arrow represents the probe state. The top row represents the N repetitions on a qubit encoded in the rectilinear basis: the angle of the probe state remains the same at each repetition. The bottom row represents the N repetitions on a qubit encoded in the diagonal basis: the angle of the probe state increases after each repetition, until it reaches $\pi/2$ (or π in the figure, as the angle are doubled on the Bloch sphere).

common assumptions) as far as the thesis is written.

In the following, we start by presenting a short history of attempts to construct this primitive, then we formally define it, and we finally present a candidate construction, based on oracles, by Aaronson and Christiano (2012), through the so-called *mini-schemes* template.

History

Several attempts have been made in order to construct such a public scheme since the definition of quantum money. Although significant progress has been done these last years, we still do not have a candidate whose security is provably based on standard assumptions. As a consequence, public quantum money is still considered as an open problem. We make some of these candidates in the following, and discuss where they failed or what they lack to be considered as a completely satisfying scheme.

In the very beginning of the quantum cryptography era, Bennett, Brassard, Breidbard, and Wiesner (1982) proposed a candidate scheme whose security relies on the hardness of factoring Blum integers. However, as it later turned out that this task is actually easy for a quantum computer (Shor (1994)), the security of this scheme does not hold anymore. Almost thirty years later, several new candidates emerged: Farhi, Gosset, Hassidim, Lutomirski, and Shor (2012), Kane (2018) and Kane, Sharif, and Silverberg (2022) proposed candidates based on knot theory and quaternion algebras, but the problems based on these fields have not been extensively studied by cryptographers, hence their

hardness is not considered as standard assumptions. Two works by Aaronson (2009), and Aaronson and Christiano (2012) defined a secure public quantum money scheme based on a quantum oracle for the former, and on a classical one for the latter. However, it is unclear how to build such oracles; in fact, both papers propose a construction that was subsequently broken (Lutomirski, Aaronson, Farhi, Gosset, Kelner, Hassidim, and Shor (2010), Conde Pena, Duran Diaz, Faugere, Hernandez Encinas, and Perret (2019)). Zhandry (2021) proposed a scheme from *quantum lightning*[11], a powerful primitive whose candidate construction given in the same paper was later broken (Roberts (2021)). In the same paper, Zhandry also described a public quantum money scheme based on indistinguishable obfuscation (iO). Yet, while iO is proven to exist under standard assumptions if we only consider classical adversaries, it is unclear whether this primitive exists in a post-quantum world. More recently, Liu, Montgomery, and Zhandry (2023b) proposed a candidate public quantum money construction based on random lattices, which was later broken by Liu, Montgomery, and Zhandry (2023a). Finally, Zhandry (2023) presented a new construction for quantum lightning — hence yielding directly a quantum money protocol — on a new (and thus non-standard) assumption on group actions and isogenies.

Definitions

We give in this subsection definition of a public quantum money scheme. Although this definition is merely an adaptation of the one for a private scheme, we present it formally for the sake of completeness.

Definition 24 (Public Quantum Money Scheme)**.** A public quantum money protocol is made of a triple of algorithms $\langle \mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Verify} \rangle$ with the following properties:

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter 1^λ and returns a pair of keys: a private key sk, and a verification key vk.

- $(|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{sk})$. The minting algorithm Mint takes as input a secret key sk and returns a quantum state $|\mathbb{c}\rangle$ as a "quantum coin", and a corresponding classical identifier s.

- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, |\mathbb{c}\rangle, s)$. The verification algorithm Verify takes as input a verification key vk, and an alleged (coin, identifier) pair $(|\mathbb{c}\rangle, s)$ and returns a bit b, indicating whether the coin is accepted ($b = 1$) or not ($b = 0$).

A public quantum money protocol must in addition satisfy the following properties.

Correctness. A coin that has been generated by the bank and that has not been modified must be accepted by the verification protocol, except with negligible probability. More precisely,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, |\mathbb{c}\rangle, s) = 1 \ : \ \begin{array}{l}(|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{sk}) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

[11] We give a description on quantum lightning in Section 4.3.3.

Unforgeability. No malicious (computationally bounded) user, given the verification key and a valid (coin, identifier) pair, must be able to produce two valid (coin, identifier) pairs. Formally, for any QPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|c} \mathsf{Verify}(\mathsf{vk}, |\mathbb{c}_1^*\rangle, s_1^*) = 1 & (|\mathbb{c}_{12}^*\rangle, s_1^*, s_2^*) \leftarrow \mathcal{A}(\mathsf{vk}, |\mathbb{c}\rangle, s) \\ \wedge & (|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{sk}) \\ \mathsf{Verify}(\mathsf{vk}, |\mathbb{c}_2^*\rangle, s_2^*) = 1 & (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Importantly, the key generation protocol, in such a public scheme returns a verification key in addition to the private key. This verification key can be distributed publicly, as the unforgeability property ensures that the coins remain unforgeable, even for an adversary possessing the verification key.

Aaronson and Christiano Mini-Schemes

Aaronson and Christiano (2012) introduced the notion of quantum money *mini-schemes*. Loosely speaking, a mini-scheme can be seen as an unkeyed quantum money scheme, only minting and verifying quantum coins, and can be upgraded to a full quantum money scheme using digital signatures. The rest of this section is dedicated to presenting these mini-schemes, together with an informal description of the mini-scheme Aaronson and Christiano proposed in the same paper.

Definition 25 (Quantum Money Mini-Scheme)**.** A quantum money mini-scheme is composed of a pair of algorithms $\langle \mathsf{Mint}, \mathsf{Verify} \rangle$ with the following properties:

- $(|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(1^\lambda)$. The coin generation algorithm Mint takes as input a security parameter (written in unary) and returns a quantum coin $|\mathbb{c}\rangle$ and its classical identifier s.
- $b \leftarrow \mathsf{Verify}(|\mathbb{c}\rangle, s)$. The verification algorithm Verify takes as input an alleged (coin, identifier) pair $(|\mathbb{c}\rangle, s)$ and returns a bit b, indicating whether the coin is accepted ($b = 1$) or rejected ($b = 0$).

Correctness and unforgeability. Correctness for a mini-scheme is defined similarly to its full-scheme counterpart. That is, such a scheme has *correctness* if a (coin, identifier) pair, generated by the Mint protocol, is accepted with probability close to 1. Furthermore, a mini-scheme has *unforgeability* if no malicious user, given a valid (coin, identifier) pair, can produce two (possibly entangled) quantum coins, valid *for the given identifier*. More formally, for any quantum adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|c} \mathsf{Verify}(|\mathbb{c}_1^*\rangle) = 1 & |\mathbb{c}_{12}^*\rangle \leftarrow \mathcal{A}(|\mathbb{c}\rangle, s) \\ \wedge & (|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(1^\lambda) \\ \mathsf{Verify}(|\mathbb{c}_2^*\rangle) = 1 & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

From mini-scheme to full-scheme. Aaronson and Christiano (2012) show how to upgrade a quantum money mini-scheme to a quantum money full-scheme, using a digital signature scheme. The idea is to use the public and private keys of the signature scheme as keys in the full-scheme. Then, a (coin, identifier) pair is obtained by minting one coin through the mini-scheme, and signing the identifier. Verifying such a pair simply consists in verifying the signature, and then the quantum coin itself, using the verification

procedures of the digital signature scheme and the mini-scheme respectively. We give a more formal description below.

Construction 4: Quantum Money Scheme From Mini-Scheme

Let $\mathsf{MS}.\langle\mathsf{Mint},\mathsf{Verify}\rangle$ be a quantum money mini-scheme, and $\mathsf{DS}.\langle\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify}\rangle$ be a digital signature scheme.

$\mathsf{KeyGen}(1^\lambda)$:

- Return $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{DS.KeyGen}(1^\lambda)$.

$\mathsf{Mint}(\mathsf{sk})$:

- Compute $(|\mathbb{c}\rangle, s) \leftarrow \mathsf{MS.Mint}(1^\lambda)$.
- Compute $sig \leftarrow \mathsf{DS.Sign}(\mathsf{sk}, s)$.
- Return the pair $(|\mathbb{c}\rangle, (s, sig))$, where $|\mathbb{c}\rangle$ is the quantum coin, and (s, sig) the identifier.

$\mathsf{Verify}(\mathsf{pk}, (|\mathbb{c}\rangle, s, sig))$:

- Compute $b_1 \leftarrow \mathsf{MS.Verify}(|\mathbb{c}\rangle, s)$.
- Compute $b_2 \leftarrow \mathsf{DS.Verify}(\mathsf{vk}, s, sig)$.
- Return $b_1 \wedge b_2$.

We give some intuition on the security of this transformation. Consider an adversary that, given one (coin, identifier) pair $(|\mathbb{c}\rangle, (s, sig))$, produces two pairs $(|\mathbb{c}_1^*\rangle, (s_1, sig_1))$ and $(|\mathbb{c}_2^*\rangle, (s_2, sig_2))$. Note first that the unforgeability of the underlying signature scheme enforces $s_1 = s_2 = s$. Then, invoking the mini-scheme unforgeability tells us that $|\mathbb{c}_1^*\rangle$ and $|\mathbb{c}_2^*\rangle$ cannot both be valid coins for s, proving the unforgeability of the resulting quantum money scheme.

A mini-scheme based on subspace states. Aaronson and Christiano (2012) provided a construction of a quantum money mini-scheme, based on subspace states, and using classical membership oracles. Recall that a subspace state $|A\rangle$ — where $A \subset \mathbb{F}_2^n$ is a subspace of dimension $n/2$ — is the uniform superposition of all vectors in A, that is $1/\sqrt{|A|}\sum_{a\in A}|a\rangle$. Recall also that applying a Hadamard gate $\mathsf{H}^{\otimes n}$ to $|A\rangle$ yields $|A^\perp\rangle$, where $A^\perp := \{x \in \mathbb{F}_2^n \,:\, x \cdot a = 0 \;\; \forall a \in A\}$ is the dual subspace of A.

With these properties in mind, we are now ready to discuss the scheme informally. This scheme first need a sort of random oracle, which, when run on a bitstring, returns an identifier s, and the description of a subspace A_s of dimension $n/2$. Minting a coin then consists in running this oracle on a random bitstring to get a random (s, A_s), preparing the corresponding subspace state $|A_s\rangle$, and returning $|A_s\rangle$ as the coin, and s as its identifier.

In order to verify a coin, we need three additional oracles. A first one that maps an identifier s to its corresponding subspace A_s; a second one that checks membership in the subspace A_s, that is, given an identifier s and a vector x, returns whether $x \in A_s$ or not; and a last one that checks membership in the dual subspace $A_s^\perp$ similarly.[12] Verifying a

[12]Note that we simplified the oracles for purpose of conciseness. We refer the curious reader to the

(coin, identifier) pair $(s, |\psi\rangle)$ then consists in

- running the first membership oracle in superposition over $|s\rangle_1 \otimes |\psi\rangle_2$;
- applying a Hadamard gate $\mathsf{H}^{\otimes n}$ on the second register;
- running the second membership oracle (for the dual subspace) in superposition over the resulting state;
- applying a Hadamard gate again on the second register.

Note also that, when $|\psi\rangle = |A_s\rangle$, applying both membership oracles do not change the state, and the quantum coin in the end of the verification procedure is $|A_s\rangle$.

4.3.3 Quantum Lightning

Quantum money, even in the public settings, always require users to trust a central authority, the bank. In the private settings, the bank plays a central role, as it is the entity that both creates, and verifies the money. Its role is less important, in the public settings, as it only creates the new coins and verification is done by the users themselves, but it is still the only entity that can perform such a task. Quantum lightning, proposed by Zhandry (2021), can be seen as a strengthening of quantum money, that removes the need for such a central bank, by allowing every user to mint their own coins. We can imagine that, in a network using a quantum lightning scheme, the users are not allowed to mint as many new coins as they want, but rather that, for example, there exists some other mechanism that ensures that users can only mint valid new coins according to a certain policy.

Of course, with such a power given to the users, it is necessary to modify the previous notions of unforgeability. Indeed, any malicious user can easily produce two valid (coin, identifier) pairs by simply running the minting procedure twice. The new security notion asked for a quantum lightning scheme is then that no malicious party can produce two quantum coins, valid for the same identifier. Enforcing this security is not an easy task, as, intuitively, it requires a way to enforce the user who wants to mint a coin to prepare a large superposition of (coin, identifier) pairs, and to measure it to yield a valid pair. This security notion, however, rules out a natural attack on what we can call a quantum lightning network. Imagine, as said above, that users have a way to decide whether a user, has the right to produce a new coin. Imagine that a user Alice has the right to produce a new coin. What would typically be expected from Alice, is minting a new (coin, identifier) pair through the public minting algorithm, and then publish the identifier to the network, so that everyone knows that this identifier points to a valid coin. If Alice is malicious, however, she could try to come up with two valid coins for some arbitrary identifier, and this is exactly what the quantum lightning security property rules out.

In the following, we give a formal definition of quantum lightning, discuss the connection with quantum money and other applications, and finally provide a history of this functionality.

Definition 26 (Quantum Lightning)**.** A quantum lightning scheme is composed of three algorithms $\langle \mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Verify} \rangle$ with the following properties:

paper of Aaronson and Christiano (2012) for a more complete description.

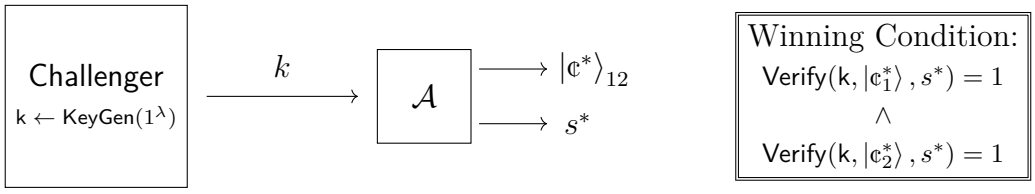

Figure 4.8: Quantum lightning security property. This property states that no efficient adversary must be able to win this game with non-negligible probability.

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter, and returns a classical key k.
- $(|\mathbb{c}\rangle, s) \leftarrow \mathsf{Mint}(\mathsf{k})$. The minting algorithm Mint takes as input a key k and returns a quantum state $|\mathbb{c}\rangle$ as a quantum coin, and a corresponding identifier s.
- $b \leftarrow \mathsf{Verify}(\mathsf{k}, |\mathbb{c}\rangle)$. The verification algorithm Verify takes as input a key k, and a (coin, identifier) pair $(\mathbb{c}, s)$ and returns a bit b, indicating whether the coin is accepted $(b = 1)$ or not $(b = 0)$.

A quantum lightning scheme must in addition satisfy the same correctness property as a quantum money scheme. That is, a (coin, identifier) pair must be accepted by the verification algorithm with probability close to 1. Furthermore, such a scheme must feature the security property defined below.

Security. No malicious (computationally efficient) adversary, must be able to generate two quantum coins, valid for the same identifier. More formally, for any QPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{k}, |\mathbb{c}_1^*\rangle, s^*) = 1 \\ \wedge \\ \mathsf{Verify}(\mathsf{k}, |\mathbb{c}_2^*\rangle, s^*) = 1\end{array} : \begin{array}{c}(|\mathbb{c}_{12}^*\rangle, s^*) \leftarrow \mathcal{A}(\mathsf{k}) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

We provide an illustration of this security property in Figure 4.8.

This definition of quantum lightning, is the one presented in Zhandry (2021). Note that Zhandry (2023) gives a slightly different definition, in which the scheme is defined as a quantum money mini-scheme (Section 4.3.2) — that is a pair of unkeyed algorithms $\langle \mathsf{Mint}, \mathsf{Verify} \rangle$ — with similar correctness and security properties as the ones above, adapted to mini-schemes. The major difference in this mini-scheme-based definition is that it is impossible to achieve security against non-uniform adversaries. Indeed, one can think of a family of non-uniform adversaries $\{\mathcal{A}_\lambda, (|\mathbb{c}_\lambda\rangle |\mathbb{c}_\lambda\rangle, s_\lambda)\}_{\lambda \in \mathbb{N}}$, such that $(\mathbb{c}_\lambda, s_\lambda)$ is generated through $\mathsf{Mint}(1^\lambda)$. An adversary defined this way would break the security property, simply by returning the advice.

Applications of Quantum Lightning

Quantum lightning is a powerful functionality that can be used to build numerous cryptographic primitives such as public quantum money, verifiable randomness and decentralized currency, as mentioned by Zhandry (2021).

Public quantum money. We can construct a public quantum money scheme given a quantum lightning scheme and a public signature scheme similarly to the transformation "quantum money mini-scheme to full-scheme" seen in Section 4.3.2. More precisely, the keys are generated with the signature scheme generation protocol. The minting is then done by generating a (coin, identifier) pair with the quantum lightning scheme, and then returning this pair, as well as a signature of the identifier. And, finally, the verification consists in verifying that both the (coin, identifier) and the signature are valid.

Verifiable randomness. In order to produce verifiable randomness, one can generate a (coin, identifier) pair and sends the identifier as the random string together with the coin as the proof. Any user who wants to verify that the string has been genuinely generated (using a random process) just has to verify that the (coin, identifier) pair is valid.

Decentralized currency. Finally, Zhandry proposed an idea for a decentralized currency using quantum lightning. A key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$ is publicly known by all users and, in order to mint money, a user must provide a coin whose identifier starts with a certain number of zeroes (fixed in advance). The only way for a user to generate such a coin is to repeatedly generate (coin, identifier) pairs until the identifier starts with enough zeroes. Then, users can verify that a (coin, identifier) pair was genuinely minted by first verifying that the identifier starts with enough zeroes and then running the verification procedure on the pair. This minting method reminds the *proof of work* of Bitcoin, ensuring that coins are hard enough to produce. Such a decentralized currency protocol, while being interesting in theory, has several issues that prevents it to be used as it is. However, Coladangelo and Sattath (2020) expanded this idea by proposing a hybrid protocol for a blockchain-based decentralized currency. Roughly, they use a classical blockchain and combine it with quantum lightning. Users can perform transactions classically — as they would do with the classical blockchain — but they can also trade some amount of money for a quantum "banknote" that is worth this amount of money. This banknote is actually composed of a quantum coin and its identifier, and any user who creates the banknote publishes (classically) in the blockchain the identifier and the amount of coins she wants to attach to the banknote. She is now able to perform transactions by sending this banknote to any other user, say Ben, who can verify the banknote validity by searching its identifier on the blockchain (this gives him the amount of money associated to the banknote) and by using the quantum lightning verification procedure to ensure that the (coin, identifier) pair is valid. The major advantage with this protocol is that this new way of doing transactions allows instantaneous verification, while users have to wait some time before a transaction is accepted when using standard blockchain's transactions.

Quantum Lightning Constructions

Quantum lightning is a very powerful cryptographic primitive. Unfortunately however, we still do not know how to construct a protocol for it, based on standard assumptions. Zhandry (2021) proposed a construction based on an assumption on the hardness of a matrix problem, but this assumption was later proven wrong by Roberts (2021). While invalidating Zhandry's construction, Roberts (2021) does not prove that quantum lightning is impossible, so we can still hope to find a protocol for this functionality. In fact, Zhandry (2023) proposed a new template for constructing quantum lightning, and gives a candidate scheme based on a non-standard assumption on isogenies.

4.3.4 Tokenized Signatures

In this section, we discuss a primitive that shares similarities with quantum money, in the sense that the property the quantum states of this primitive have is related to verification.

Consider a network of users, where some user Ava has some authority. As she leaves for some time, she decides to give the ability to her friend Alice to sign arbitrary messages on her behalf. However, her confidence in Alice is limited, so she wants to let her sign up to, say, three messages. A tokenized signature scheme exactly fits this purpose, by allowing Ava to generate and share one-time tokens, that can be used to sign any message, and then self-destroy (here, she would give three tokens to Alice). This notion was first introduced by Ben-David and Sattath (2023), and they constructed a tokenized signature scheme based on oracles.

In this section, we define tokenized signature schemes, and give an example of such a scheme.

Definitions

We start by giving the definition of a tokenized signature scheme, and the properties that such a scheme can (or must) feature.

Definition 27 (Tokenized Signature Scheme)**.** A tokenized signature scheme is a tuple of algorithms $\langle \mathsf{KeyGen}, \mathsf{TokenGen}, \mathsf{Sign}, \mathsf{Verify} \rangle$ with the following properties:

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. On input a security parameter 1^λ, the key generation algorithm outputs a secret key sk and a verification key vk.
- $|♙\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk})$. On input a secret key sk, the quantum token generation algorithm outputs a quantum signing token $|♙\rangle$.
- $sig \leftarrow \mathsf{Sign}(|♙\rangle, m)$. On input a signing token $|♙\rangle$ and a message $m \in \mathcal{M}$ to be signed, the signing algorithm outputs a classical signature sig of m.
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m, sig)$. On input a verification key vk, a message m, and a signature sig, the verification algorithm returns a bit b, indicating whether the verification accepts ($b = 0$) or rejects ($b = 1$) the signature.

We present four properties that a tokenized signature scheme can have. In the following, we consider that such a scheme *must* feature at least the two first properties — namely correctness and weak unforgeability. Whenever a tokenized signature scheme has one of the two other properties — namely strong unforgeability and unclonable unforgeability — we explicitly mention it.

Correctness. A tokenized signature scheme has correctness if a signature of any message m produced by a valid token is accepted with overwhelming probability. More formally, for all $m \in \{0, 1\}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m, sig) = 1 \ : \ \begin{array}{l} sig \leftarrow \mathsf{Sign}(|♙\rangle, m) \\ |♙\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk}) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

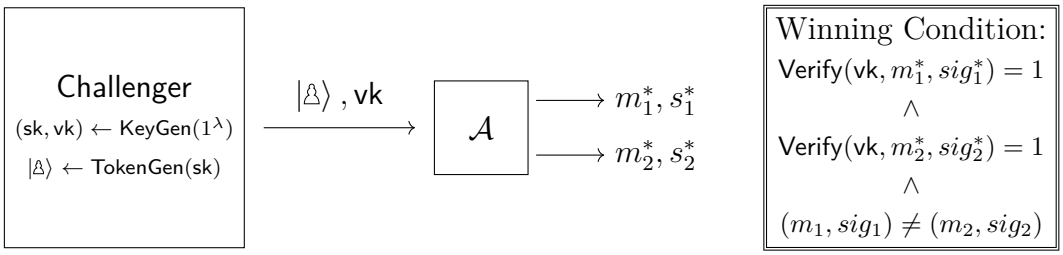

Figure 4.9: Strong unforgeability property of a tokenized signature scheme. This property states that no efficient adversary must be able to win this game with non-negligible probability.

Weak unforgeability. A tokenized signature has weak unforgeability if no computationally efficient malicious user, given one token, can produce two different messages, together with a valid signature for each message. More precisely, for all QPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|l} \mathsf{Verify}(\mathsf{vk}, m_1^*, sig_1^*) = 1 & \\ \wedge & (m_1^*, sig_1^*, m_2^*, sig_2^*) \leftarrow \mathcal{A}(\mathsf{vk}, |♙\rangle) \\ \mathsf{Verify}(\mathsf{vk}, m_2^*, sig_2^*) = 1 & |♙\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk}) \\ \wedge & (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ m_1 \neq m_2 & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Note that, as we consider public verification, the adversary is also given the verification key.

Strong unforgeability. Chevalier, Hermouet, and Vu (2023) defined a stronger unforgeability property, namely strong unforgeability. One can see the weak definition as the equivalent of the existential unforgeability for classical digital signature schemes, adapted to our settings. Then strong unforgeability is defined as the equivalent of strong existential unforgeability of digital signatures. A tokenized signature has strong unforgeability if no computationally efficient adversary can produce two different (message, signature) pairs. Note that, contrary to the weak definition, the messages here can be equal, we just ask the signatures to be different. More precisely, for all QPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|l} \mathsf{Verify}(\mathsf{vk}, m_1^*, sig_1^*) = 1 & \\ \wedge & (m_1^*, sig_1^*, m_2^*, sig_2^*) \leftarrow \mathcal{A}(\mathsf{vk}, |♙\rangle) \\ \mathsf{Verify}(\mathsf{vk}, m_2^*, sig_2^*) = 1 & |♙\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk}) \\ \wedge & (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (m_1, sig_1) \neq (m_2, sig_2) & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

We provide an illustration of this strong unforgeability property in Figure 4.9.

Remark 1. Chevalier, Hermouet, and Vu (2024b) proposed a new security property for tokenized signatures, that they named *unclonable unforgeability*. To understand this last property, consider the following scenario. Alice owns a signing token and wants to send it to both Bob and Charlie, two non-communicating users, such that both of them can sign an arbitrary message. Crucially, they need to be able to choose this message *after the token is split*. We elaborate on this property in Chapter 5.

Construction

We present a simple construction, based on BB84 states, which only features private verification. This scheme was presented by Behera, Sattath, and Shinar (2021), based on

a work of Pastawski, Yao, Jiang, Lukin, and Cirac (2012). In the example we gave at the beginning of the section, it means that the user who sign messages is the only one who can verify the validity of the signatures produced with the tokens, otherwise the scheme is insecure. Note that Coladangelo, Liu, Liu, and Zhandry (2021) constructed a tokenized signature scheme, with public verification, and satisfying weak unforgeability property. As one of the contributions of this thesis is to prove that this scheme satisfies other properties, we present it in details in Chapter 5.

A simple scheme based on BB84 states. This scheme is for single-bit messages. The secret (and verification) key is composed of two random n-long bitstrings, representing respectively the value x and the basis θ of a BB84 state. A token is then the corresponding BB84 state, and signing a bit b simply consists in measuring the state in either the rectilinear, or diagonal basis. Finally, let I be the set of indices at which $\theta_i = 0$. A valid signature for 0 is a bitstring sig such that $sig_{|I} = x_{|I}$ and a valid one for 1 is such that $sig_{|\bar{I}} = x_{|\bar{I}}$. In other terms, sig must match x at indices on which $\theta = 0$ for a valid signature of 0, and the same must hold at the indices on which $\theta = 1$ for a valid signature of 1. We provide a more formal description of this scheme below.

The correctness of this scheme is immediate, and the weak unforgeability property is implied by the monogamy-of-entanglement property of BB84 states (Theorem 5). Indeed, consider an adversary Alice who produces two valid pairs (message, signature) (m_1, sig_1) and (m_2, sig_2), where $m_1 \neq m_2$, with probability p. We can assume without loss of generality that $m_1 = 0$ and $m_2 = 1$, meaning that $s_1 = x_{|I}$ and $s_2 = x_{|\bar{I}}$. Then, we construct a triple of adversaries Alex, Billy, and Clover, for the monogamy-of-entanglement game for BB84 states. Alex, given as input a BB84 state $|x^\theta\rangle$, runs Alice on it to get (m_1, sig_1) and (m_2, sig_2). She then sends s_1 and s_2 to Billy and Clover, and the latter simply return s_1 and s_2 respectively, and win the game with the same probability p.

Construction 5: A Single-Bit Tokenized Signature Scheme

Set $n = \mathsf{poly}(\lambda)$.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample n-long BB84 state's description: $x \leftarrow\!\$\ \{0,1\}^n$ and $\theta \leftarrow\!\$\ \{0,1\}^n$ such that $|\theta| = n/2$.
- Return (x, θ).[13]

$\mathsf{TokenGen}(\mathsf{sk})$:

- Parse sk as (x, θ).
- Return $|x^\theta\rangle$.

$\mathsf{Sign}(m, |\triangle\rangle)$:

- Apply $\mathsf{H}^{\otimes n}$ to ρ only if $m = 1$; otherwise, leave the state unchanged.
- Measure the resulting state in the computational basis. Let sig be the outcome.
- Return sig.

$\mathsf{Verify}(\mathsf{sk}, m, sig)$:

- Parse sk as (x, θ).
- Set $I := \{i \in \{1, \ldots, n\} \mid \theta_i = 0\}$.
- If $m = 0$: return 1 if $sig_{|I} = x_{|I}$; otherwise return 0.
- If $m = 1$: return 1 if $sig_{|\bar{I}} = x_{|\bar{I}}$; otherwise return 0.

4.4 Unclonable Encryption

We now present the second class of unclonable primitives. This class is made of primitives that hold information, in an unclonable way. As a warm-up example, consider a (private) encryption scheme — composed of a key generation, encryption and decryption procedures — with the usual IND-CPA security. Now, say that you want the ciphertexts to be unclonable, meaning they are now quantum states, and cannot be split into two states that both hold information on the plaintext. This is exactly unclonable encryption, first defined by Broadbent and Lord (2020), and exists in private and public settings.

In both settings, the correctness and semantic security must hold, as in their classical counterpart. Furthermore, they must achieve the following unclonability property. Consider a malicious user Alice, who splits the ciphertext of a random message into two states. It must be impossible for two other (non-communicating) users collaborating with Alice, Bob and Charlie, to simultaneously guess which plaintext has been encrypted, even given the decryption key. Note that, provided that the scheme is secure, Alice cannot guess the plaintext. However, she would be able to do it easily if she is provided what is given to Bob and Charlie after the split, namely the decryption key.

As mentioned above, this primitive has first been defined in the private settings by Broadbent and Lord (2020).[14] In this paper, Broadbent and Lord also gave a construction of a one-time private unclonable encryption scheme — meaning that the key can only be used to encrypt one message — based on BB84 states. Ananth and Kaleoglu (2021) later extended the definition to the public settings, presenting in the same paper a generic way to transformation to construct both private and public unclonable encryption schemes from a one-time one. However, it is worth pointing out that none of the constructions above achieve the stronger *unclonable-indistinguishability* security property (Broadbent and Lord (2020)). This property can be seen as a combination of the semantic security for classical encryption schemes, and the unclonability property described above. More precisely, the difference with the unclonable property is that the ciphertext given to Alice is the encryption of one of two messages, chosen by Alice beforehand. The unclonable-indistinguishability property then states that Bob and Charlie must not be able to both guess which message has been encrypted with probability greater than 1/2. Unclonable-indistinguishability is considerably more difficult to achieve than the simple unclonability, as, while constructions with this property has been proposed in the QROM (Ananth, Kaleoglu, Li, Liu, and Zhandry (2022) and Ananth, Kaleoglu, and Liu (2023))[15], and

[13] Remark that, as the verification key here is actually the secret key, we simply write sk for both keys, hence the KeyGen algorithm returns only sk.

[14] In an earlier work, Gottesman (2002) also defines unclonable encryption, in a slightly different way. Throughout this thesis, we use the definition of Broadbent and Lord (2020).

under unproven conjectures (Ananth and Behera (2024) and Chevalier, Hermouet, and Vu (2024b)), achieving a construction in the plain model (that is, without oracle) remains an open problem at the time of writing this thesis.

In the rest of this section, we formally define private and public unclonable encryption schemes, and present the one-time constructions proposed by Broadbent and Lord (2020) and Ananth, Kaleoglu, and Liu (2023), as well as the generic transformation to turn such a one-time scheme into a private key and public key one.

4.4.1 Private Unclonable Encryption

In the following, we define private key and public key unclonable encryption, as well as their correctness and security properties.

Definition 28 (Private Unclonable Encryption)**.** A private unclonable encryption scheme for a message space $\mathcal{M}$[16] is composed of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined in the following way:

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter, and returns a classical key k.
- $|\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m)$. The encryption algorithm Enc takes as input a key k and a classical message $m \in \mathcal{M}$ and returns a quantum ciphertext $|\mathbb{M}\rangle$.
- $m \leftarrow \mathsf{Dec}(\mathsf{k}, |\mathbb{M}\rangle)$. The decryption algorithm Dec takes as input a key k and a quantum ciphertext $|\mathbb{M}\rangle$ and returns a classical message m.

A private unclonable encryption scheme must in addition satisfy the following properties.

Correctness. The encryption of a message must always decrypt to this message. More precisely, for any message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{k}, |\mathbb{M}\rangle) = m \ : \ \begin{matrix} |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Indistinguishability. We distinguish one-time indistinguishability from many-time indistinguishability. For a private unclonable encryption scheme to have *one-time* indistinguishability, the encryption of two messages must be computationally indistinguishable. More formally, for all $m, m' \in \mathcal{M}$,

$$\begin{aligned} \{\mathsf{Enc}(\mathsf{k}, m) \ &: \ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \\ &\approx_c \\ \{\mathsf{Enc}(\mathsf{k}, m') \ &: \ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \end{aligned}$$

Many-time indistinguishability is defined analogously, except that the adversary is given this time a polynomial number of ciphertexts. More formally, for all $\kappa = \mathsf{poly}(\lambda)$, and all $m_1, \ldots, m_\kappa, m'_1, \ldots, m'_\kappa \in \mathcal{M}$,

$$\begin{aligned} \{(\mathsf{Enc}(\mathsf{k}, m_1), \ldots, \mathsf{Enc}(\mathsf{k}, m_\kappa)) \ &: \ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \\ &\approx_c \\ \{(\mathsf{Enc}(\mathsf{k}, m'_1), \ldots, \mathsf{Enc}(\mathsf{k}, m'_\kappa)) \ &: \ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \end{aligned}$$

[15] Recall that the QROM (quantum random oracle model) allows oracle queries (in superposition) to a perfect random function, and cannot be implemented efficiently in practice.

[16] In the following, we set $\mathcal{M} = \{0,1\}^n$ for $n = \mathsf{poly}(\lambda)$.

Unclonability. Consider a triple of collaborating malicious users Alice, Bob, and Charlie. In a first phase, Alice splits a random ciphertext into two (possibly entangled) states; and in a second phase, Bob and Charlie, each one given one of these states and the secret key, make a guess about the plaintext. The unclonability property states that Bob and Charlie must not make both a correct guess.

More formally, we define the following game, parameterized by a security parameter λ, between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples a key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and a message $m \leftarrow\!\!\$\ \mathcal{M}$.
  - The challenger computes $|\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m)$.
  - The challenger sends $|\mathbb{M}\rangle$ to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\mathbb{M}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\mathbb{M}^*\rangle_1$ to $\mathcal{B}$, and $|\mathbb{M}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:** The challenger sends k to both $\mathcal{B}$ and $\mathcal{C}$.

Let m_1^* denotes the output of $\mathcal{B}$, and m_2^* denotes the output of $\mathcal{C}$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if $m_1^* = m$ and $m_2^* = m$. We then say that an unclonable encryption scheme has unclonability if no adversaries can win this game with probability significantly greater than $1/|\mathcal{M}|$. That is, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{lcl} & & m_1^* \leftarrow \mathcal{B}(|\mathbb{M}^*\rangle_1, \mathsf{k}), m_2^* \leftarrow \mathcal{C}(|\mathbb{M}^*\rangle_2, \mathsf{k}) \\ m_1^* = m & & |\mathbb{M}^*\rangle_{12} \leftarrow \mathcal{A}(|\mathbb{M}\rangle) \\ \quad \wedge & : & |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m) \\ m_2^* = m & & m \leftarrow\!\!\$\ \mathcal{M} \\ & & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{|\mathcal{M}|} + \mathsf{negl}(\lambda)$$

Unclonable-indistinguishability. Unclonable-indistinguishability property is defined similarly to unclonability, except that the ciphertext given to Alice is now an encryption of one of two plaintexts, that she has chosen beforehand.

More formally, we define the following game, parameterized by a security parameter λ, and between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - $\mathcal{A}$ sends a pair of messages $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.
  - The challenger samples a key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and a bit $b \leftarrow\!\!\$\ \{0, 1\}$.
  - The challenger computes $|\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m_b)$.
  - The challenger sends $|\mathbb{M}\rangle$ to $\mathcal{A}$.
- **Splitting phase:**

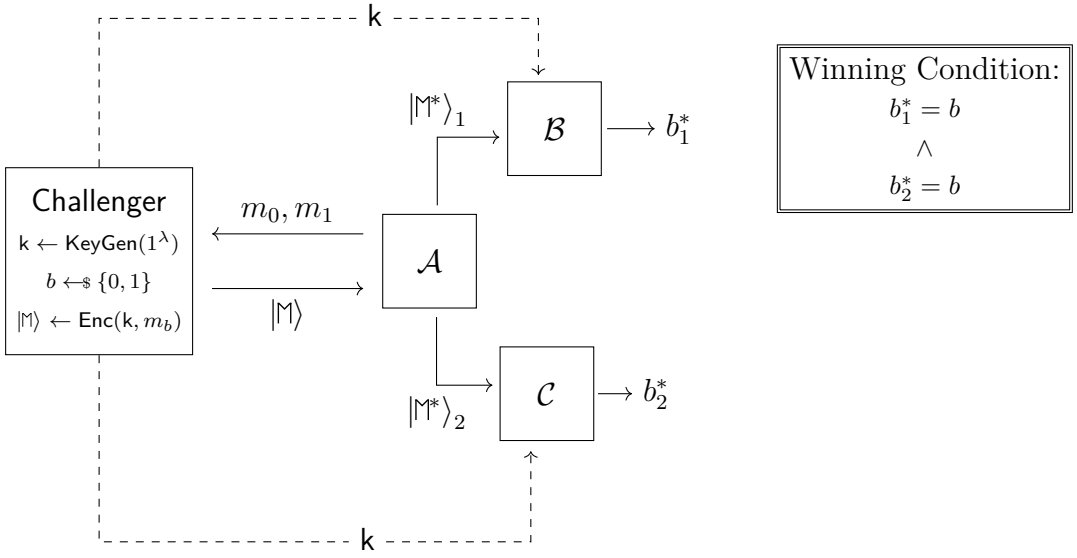

Figure 4.10: Unclonable-indistinguishability of an unclonable encryption scheme. This property states that no efficient adversary must be able to win this game with probability significantly greater than $1/2$.

  - $\mathcal{A}$ prepares a bipartite state $|\mathbb{M}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\mathbb{M}^*\rangle_1$ to $\mathcal{B}$, and $|\mathbb{M}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:** The challenger sends k to both $\mathcal{B}$ and $\mathcal{C}$.

Let b_1^* denotes the output of $\mathcal{B}$, and b_2^* denotes the output of $\mathcal{C}$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if $b_1^* = b$ and $b_2^* = b$. We then say that an unclonable encryption scheme has unclonability if no adversaries can win this game with probability significantly greater than $1/2$. In other words, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{c} b_1^* = b \\ \wedge \\ b_2^* = b \end{array} : \begin{array}{l} b_1^* \leftarrow \mathcal{B}(|\mathbb{M}^*\rangle_1, \mathsf{k}), b_2^* \leftarrow \mathcal{C}(|\mathbb{M}^*\rangle_2, \mathsf{k}) \\ |\mathbb{M}^*\rangle_{12} \leftarrow \mathcal{A}(|\mathbb{M}\rangle) \\ |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m_b) \\ b \leftarrow\$ \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We provide an illustration of this game in Figure 4.10.

4.4.2 Public Unclonable Encryption

We now define unclonable encryption in the public settings. As it is merely the same as its private counterpart, except that the scheme now uses two keys — a private key for decrypting, and a public key for encrypting — and the definitions capture the fact that the public key is available to the adversaries, we highlight the differences between public and private schemes in blue.

Definition 29 (Public Unclonable Encryption)**.** A public unclonable encryption scheme for a message space $\mathcal{M}$ is composed of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined in the following way:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm KeyGen takes as input a security parameter, and returns a secret key sk, and a public key pk.
- $|\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. The encryption algorithm Enc takes as input a public key pk and a classical message $m \in \mathcal{M}$ and returns a quantum ciphertext $|\mathbb{M}\rangle$.
- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, |\mathbb{M}\rangle)$. The decryption algorithm Dec takes as input a secret key sk and a quantum ciphertext $|\mathbb{M}\rangle$ and returns a classical message m.

We present below the correctness, indistinguishability, and unclonability properties that a public unclonable encryption scheme must satisfy, as well as the additional unclonable-indistinguishability property.

Correctness. A public unclonable encryption scheme has correctness if, for any message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, |\mathbb{M}\rangle) = m \ : \ \begin{array}{l} |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Indistinguishability. Indistinguishability for public key unclonable encryption scheme is defined through a game, in which a computationally bounded adversary Alice receives a random public key, then chooses two messages of same length. The encryption of one of these messages (at random) is finally given to her, and she must not be able to guess which message has been encrypted with probability significantly greater than $1/2$.

More formally, a public unclonable encryption scheme has indistinguishability if, for any QPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} |m_0^*| = |m_1^*| \\ \wedge \\ b^* = b \end{array} \ : \ \begin{array}{l} b^* \leftarrow \mathcal{A}(|\phi\rangle, |\mathbb{M}\rangle) \\ |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*) \\ b \leftarrow\!\!\$ \ \{0,1\} \\ ((m_0^*, m_1^*), |\phi\rangle) \leftarrow \mathcal{A}(\mathsf{pk}) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ is stateful, and $|\phi\rangle$ represents its memory state.

Note that, in the public settings, there is no notion of one-time or many-time indistinguishability, as the adversary is given the public key, and hence can generate as many (message, ciphertext) pairs as she wants.

Unclonability. A public unclonable encryption scheme has unclonability if, for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{l} m_1^* = m \\ \wedge \\ m_2^* = m \end{array} \ : \ \begin{array}{l} m_1^* \leftarrow \mathcal{B}(|\mathbb{M}^*\rangle_1, \mathsf{sk}), m_2^* \leftarrow \mathcal{C}(|\mathbb{M}^*\rangle_2, \mathsf{sk}) \\ |\mathbb{M}^*\rangle_{12} \leftarrow \mathcal{A}(\mathsf{pk}, |\mathbb{M}\rangle) \\ |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ m \leftarrow\!\!\$ \ \mathcal{M} \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{|\mathcal{M}|} + \mathsf{negl}(\lambda)$$

Unclonable-indistinguishability. A public unclonable encryption scheme has unclonable-indistinguishability if, for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{c} b_1^* = b \\ \wedge \\ b_2^* = b \end{array} : \begin{array}{l} b_1^* \leftarrow \mathcal{B}(|\mathbb{M}^*\rangle_1, \mathsf{sk}), b_2^* \leftarrow \mathcal{C}(|\mathbb{M}^*\rangle_2, \mathsf{sk}) \\ |\mathbb{M}^*\rangle_{12} \leftarrow \mathcal{A}(\mathsf{pk}, |\mathbb{M}\rangle) \\ |\mathbb{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b \leftarrow\$ \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}, 1^\lambda) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

4.4.3 Construction

In this subsection, we present a construction of a private unclonable encryption scheme, presented in a work of Broadbent and Lord (2020).

Broadbent and Lord's construction. We start with the protocol of Broadbent and Lord (2020). Let $n = \mathsf{poly}(\lambda)$, and set the message space $\mathcal{M} = \{0,1\}^n$. The secret key in this protocol is a random description of an n-qubits BB84 state (r, θ). Encrypting a message bit m is done by preparing and returning the quantum ciphertext $|(m \oplus r)^\theta\rangle$ (the BB84 state whose description is $(m \oplus r, \theta)$). Finally, in order to decrypt a ciphertext $|(m \oplus r)^\theta\rangle$, given the key (r, θ), one simply measures the state in basis θ, then uncompute the mask by XOR-ing the outcome with x.

We provide a more formal description of this construction below.

Construction 6: Broadbent and Lord's Unclonable Encryption Scheme

$\mathsf{KeyGen}(1^\lambda)$:

- Set $n = \mathsf{poly}(\lambda)$.
- Sample an n-long BB84 description, that is $r \leftarrow\$ \{0,1\}^n$, and $\theta \leftarrow\$ \{0,1\}^n$ such that $|\theta| = n/2$.
- Return (r, θ).

$\mathsf{Enc}((r, \theta), m)$:

- Compute $x = m \oplus r$.
- Return $|x^\theta\rangle$.

$\mathsf{Dec}((r, \theta), |\mathbb{M}\rangle)$:

- Measure $|\mathbb{M}\rangle$ in basis θ, let x denotes the outcome.
- Return $x \oplus r$.

The correctness of this protocol is immediate, and the indistinguishability comes from the indistinguishability of two one-time padded bitstrings with an unknown key. Finally, the unclonability property comes directly from the monogamy-of-entanglement property of BB84 states. More precisely, consider Alice, Bob, and Charlie winning the unclonability game of this scheme with some probability p, then another triple of adversaries, Alex, Billy, and Clover, wins the aforementioned monogamy-of-entanglement game with the

same probability p. Alex first simulates Alice on her input $|x^\theta\rangle$, and sample a random n-long bitstring r. Note that $|x^\theta\rangle$ follows the exact same distribution as $|(m \oplus r)^\theta\rangle$ — an encryption of the message $m = x \oplus r$ with a random key (r, θ) in the scheme. Then, Alex sends the two states she received from simulating Alice, as long as the bitstring r, to Billy and Clover. The latter respectively simulate Bob and Charlie on, as input, the state and bitstring received from Alex, and the basis θ received by the challenger. The outcome of Bob and Charlie are both m with probability p, hence Billy and Clover both return these outcomes, XOR-ed with r to both guess x with the same probability p.

This protocol, on the other hand, does not satisfy unclonable-indistinguishability, and is also secure only if a key is used for no more than one message. To see that the latter property does not hold, consider this simple attack, performed by an adversary Alice, provided with some κ copies of the encryption of the message $0 \dots 0$, that is $|r^\theta\rangle \otimes \cdots \otimes |r^\theta\rangle$. She measures the two first copies in the rectilinear basis to get outcomes $r^{(1)}$ and $r^{(2)}$. Now, she knows that, on the indices i where $r_i^{(1)} \neq r_i^{(2)}$ (on average 1/4 of the indices), she measured with the "wrong" basis, hence $\theta_i = 1$. She repeats this operation with the other copies, and keeps comparing the outcomes. Formally, for any index i for which she measured in the wrong basis (that is $\theta_i = 1$), the probability that all the measurement outcomes are the same on this index is negligible in κ, meaning that, if κ is polynomial in the security parameter, she completely learns θ (hence r by measuring a last copy in basis θ), and breaks the scheme.

Broadbent and Lord also presented a simple attack against this construction's unclonable-indistinguishability property in the case where the messages' length n is at least 2. Alice sends $m_0 = 00$ and $m_1 = 11$ as the two candidate messages to the challenger. When receiving the encryption $|\mathsf{M}\rangle$ of one of these messages, she literally splits it in half, sending the first register $|(b \oplus r_1)^{\theta_1}\rangle$ to Bob, and the second one $|(b \oplus r_2)^{\theta_2}\rangle$ to Charlie. Bob and Charlie can then both simply decrypt their half message using the key (r, θ) they receive in the challenge phase, and output b.

While this limitation can be seen as a problem for such a scheme at first glance, Ananth, Kaleoglu, Li, Liu, and Zhandry (2022) show how to generically upgrade a one-time private scheme to a fully-secure private scheme assuming pseudorandom functions, or even to a public scheme, assuming functional encryption. We present the first transformation in the following subsection.

One-Time to Many-Time Transformation

We present in this subsection a transformation to uplift a one-time secure private unclonable encryption scheme to a many-time secure one, presented by Ananth, Kaleoglu, Li, Liu, and Zhandry (2022). In addition to the one-time secure scheme, this transformation uses a symmetric classical encryption scheme, with a special fake-key property, that we describe later. In the final unclonable encryption scheme, a key is simply a key of the classical scheme. To encrypt a message, one samples a key from the one-time scheme, encrypt this using the classical scheme, and the message using the one-time scheme. The final ciphertext is composed of these two (classical and quantum) ciphertexts. Finally, decrypting a ciphertext consists in, first decrypting the classical ciphertext to get the one-time key, and then decrypting the quantum one with it to get the message.

Construction 7: Many-Time Unclonable Encryption Scheme

Let $\mathsf{OT}.\langle\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\rangle$ be a private unclonable encryption scheme with correctness, one-time indistinguishability, and unclonability. Let $\mathsf{SKE}.\langle\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\rangle$ be a (classical) symmetric key encryption scheme with post-quantum indistinguishability security.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample $\mathsf{k}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
- Return $\mathsf{k}_{\mathsf{SKE}}$.

$\mathsf{Enc}(\mathsf{k}_{\mathsf{SKE}}, m)$:

- Sample $\mathsf{k}_{\mathsf{OT}} \leftarrow \mathsf{OT.KeyGen}(1^\lambda)$.
- Compute $c \leftarrow \mathsf{SKE.Enc}(\mathsf{k}_{\mathsf{SKE}}, \mathsf{k}_{\mathsf{OT}})$.
- Compute $|\mathsf{M}\rangle \leftarrow \mathsf{OT.Enc}(\mathsf{k}_{\mathsf{OT}}, m)$.
- Return $(c, |\mathsf{M}\rangle)$.

$\mathsf{Dec}(\mathsf{k}_{\mathsf{SKE}}, (c, |\mathsf{M}\rangle))$:

- Sample $\mathsf{k}_{\mathsf{OT}} \leftarrow \mathsf{SKE.Dec}(\mathsf{k}_{\mathsf{SKE}}, c)$.
- Return $\mathsf{OT.Dec}(\mathsf{k}_{\mathsf{OT}}, |\mathsf{M}\rangle)$.

The correctness follows from the correctness of both underlying encryption schemes, and the many-time indistinguishability from their indistinguishability properties: the indistinguishability of the classical scheme states that all the classical ciphertexts (the encryptions of the one-time key) can be replaced by encryptions of 0, and, now that the quantum ciphertexts are no longer dependent of the classical ones, we can invoke the one-time indistinguishability property of the unclonable encryption scheme to remove all information on the message they contain, finishing the proof.

The unclonability, however, is more complicated to prove. Indeed, in the corresponding game, Bob and Charlie are both given the key (here the classical scheme key), and only indistinguishability security is not enough to argue that the classical ciphertexts can be replaced by encryption of 0, as we did above. That is where the *fake-key* property comes into play. This property provides an efficient way to generate a fake key, from an encryption of a message m, such that the pair (real key, encryption of m) is indistinguishable from the pair (fake key, encryption of 0). Crucially, this property allows to generate the fake key *after* the encryption of m is computed, and then to simulate the (fake) key, allowing to proceed in the proof as above.

We note that the same argument applies to unclonable-indistinguishability, meaning that a one-time private unclonable encryption scheme with unclonable-indistinguishability can be generically transformed into a many-time one with the same property.

Transformation to public key scheme. Ananth, Kaleoglu, Li, Liu, and Zhandry (2022) also present a similar transformation, to uplift a one-time private unclonable encryption scheme to a many-time public one, while preserving the correctness, unclonability, and unclonable-indistinguishability properties. This transformation is more involved, using

this time a functional encryption scheme, but still uses the technique of generating a fresh new one-time key, and producing a pair of ciphertexts (one of the one-time key and one of the message), for every encryption.

Quantum Money From Unclonable Encryption

Broadbent and Lord (2020) proposed a way to construct private quantum money from unclonable encryption. As they just sketch the proof in this paper, we make in the following an attempt to formalizing this transformation. Let us first precise that we need a private one-time secure encryption scheme, with unclonable security, and that the quantum money scheme we obtain is a private one, with a slightly weakened unforgeability definition: instead of asking for the hardness of generating two valid (coin, identifier) pairs out of one, we ask the hardness of generating two coins out of a valid (coin, identifier) pair, such that *both coins are valid for the given identifier*. As the former task is easier than the latter, it results in a quantum money scheme with a slightly weaker unforgeability property.

The idea of the transformation is the following. The bank is a database, initially empty, that maps identifiers to (key, message) pairs. To mint a coin, the bank samples a random identifier, a key, and a message, and stores them in the database The coin is then the encryption of the message with the key. Finally, verifying a coin consists in first, finding in the database the key and message associated to the identifier, then decrypting the coin with the database's key and verifying that the outcome is the database's message. We present the transformation more formally in the following.

Construction 8: Private Quantum Money From Unclonable Encryption

Let $\langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ be a private unclonable encryption scheme with one-time indistinguishability. Let $\mathcal{M}$ and $\{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$ be respectively the message space and key space of this encryption scheme.

$\mathsf{KeyGen}(1^\lambda)$:

- Set $n = \mathsf{poly}(\lambda)$.
- Initialize and return an empty database db, mapping elements in $\{0,1\}^n$ to elements in $\mathcal{M} \times \mathcal{K}_\lambda$.

$\mathsf{Mint}(\mathsf{db})$:

- Sample $s \leftarrow\!\!\$\ \{0,1\}^n$.
- Sample $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- Sample $m \leftarrow\!\!\$\ \mathcal{M}$.
- Store (k, m) at index s in db.
- Compute $|\mathsf{M}\rangle \leftarrow \mathsf{Enc}(\mathsf{k}, m)$.
- Return $(s, |\mathsf{M}\rangle)$.

$\mathsf{Verify}(\mathsf{db}, (s, |\mathsf{M}\rangle))$:

- Retrieve (k, m) at index s in db.

– Compute $m' \leftarrow \mathsf{Dec}(\mathsf{k}, |\mathbb{M}\rangle)$.

– Return 1 if $m' = m$, otherwise return 0.

The correctness of this scheme is immediate, as the space in which we sample identifiers is large enough, ruling out collisions with probability close to 1.

To prove that the scheme has unforgeability — more precisely the aforementioned weaker definition — consider an adversary Alice who breaks the unforgeability property of this scheme. That is, given a valid (coin, identifier) pair $(s, |\mathbb{M}\rangle)$, Alice generates $|\mathbb{c}^*_{12}\rangle$ such that both $(|\mathbb{c}^*_1\rangle, s)$ and $(|\mathbb{c}^*_2\rangle, s)$ are valid. We present a triple of adversaries Alex, Billy, and Clover, who break the unclonability game of the underlying unclonable encryption scheme. Alex, given the encryption $|\mathbb{M}\rangle$ of a random message m, samples a random identifier $s \leftarrow\!\!\$\ \{0,1\}^n$, then simulates Alice on $(s, |\mathbb{M}\rangle)$ to get $|\mathbb{M}^*\rangle_{12}$; she sends $|\mathbb{M}^*\rangle_1$ to Billy, and $|\mathbb{M}^*\rangle_2$ to Clover. Note that, as $|\mathbb{M}^*\rangle_1$ and $|\mathbb{M}^*\rangle_2$ are both valid quantum coins, it means that they are both valid encryption of m. Then, Billy and Clover, given the key k, simultaneously run the decryption algorithm on $|\mathbb{M}^*\rangle_1$ and $|\mathbb{M}^*\rangle_2$, yielding m for both of them, hence allowing them to win the game.

4.5 Encryption With Certified Deletion

Consider the following scenario. Ava wants to store some private information on a remote — *untrusted* — server, so that she can retrieve them later. As Ava does not trust the server, she encrypts the information before sending it. However, when she finally retrieves this information and/or asks the server to delete them, nothing proves her that the server really deleted them. A malicious server could have, indeed, kept the information, waiting for the moment where they will be powerful enough to break the encryption scheme (assuming its security is based on computational assumptions), or even for a possible leak of the key. If the encryption scheme used by Alice, on the other hand, has certified deletion, she can ask the server for a *certificate of deletion* of the ciphertext, ensuring that the server really destroyed the ciphertext.

In this section, we define formally what an encryption scheme with certified deletion is, present the protocol of Bartusek and Khurana (2023) as an example, and finally discuss some recent advances in the area.

Definitions

We start by formally defining what an encryption scheme with certified deletion is, and what are its associated properties. We define the scheme for message space of one bit, but it can be extended naturally for a general message space.

Definition 30 (Encryption Scheme with Certified Deletion)**.** A (public key) encryption scheme with certified deletion for a message space $\{0,1\}$ is composed of five algorithms $\langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Verify} \rangle$ defined in the following way.

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm takes as input a security parameter 1^λ, and returns a secret key sk and a public key pk.
- $(|\mathbb{M}\rangle, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. The encryption algorithm takes as input a public key pk and a classical message $m \in \{0,1\}$, and returns a quantum state ciphertext $|\mathbb{M}\rangle$, and a verification key vk for this ciphertext.

- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, |\mathsf{M}\rangle)$. The decryption algorithm takes as input a secret key sk and a quantum ciphertext $|\mathsf{M}\rangle$, and returns a classical message $m \in \{0,1\}$.
- $\mathsf{crt} \leftarrow \mathsf{Del}(|\mathsf{M}\rangle)$. The deletion algorithm takes as input a quantum ciphertext $|\mathsf{M}\rangle$, and returns a classical certificate of deletion crt.
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, \mathsf{crt})$. The verification algorithm takes as input a verification key vk and a certificate of deletion crt, and returns a bit b, indicating that whether the certificate is accepted ($b = 1$) or rejected ($b = 0$).

An encryption scheme with certified deletion must in addition satisfy the following properties.

Correctness of decryption. The encryption of a message must always decrypt to this message. More precisely, for all message $m \in \{0,1\}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, |\mathsf{M}\rangle) = m \ : \ \begin{array}{c} (|\mathsf{M}\rangle, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Correctness of deletion. The verification algorithm always accepts honestly generated certificates. More formally, for all message $m \in \{0,1\}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{crt}) = 1 \ : \ \begin{array}{c} \mathsf{crt} \leftarrow \mathsf{Del}(|\mathsf{M}\rangle) \\ (|\mathsf{M}\rangle, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Security of deletion. Consider a pair of collaborating adversaries, Alice and Bob, playing a game in which Alice is first given a random ciphertext, and then asked to return a valid certificate for this ciphertext. She then sends a quantum state to Bob, who is also given the secret key of the encryption scheme, and asked to guess which message has been encrypted. The security of deletion states that they cannot both provide a valid certificate and make a correct guess. Importantly, and to capture the scenario of a server who would wait a long time to decrypt the message (see example scenario at the beginning of the section), we consider that Bob is unbounded. This means that the deletion algorithm must actually destroy all information on the message in an information-theoretic way.

More formally, for any QPT adversary $\mathcal{A}$, and any adversary $\mathcal{B}$,

$$\Pr\left[\begin{array}{c} \mathsf{Verify}(\mathsf{vk}, \mathsf{crt}^*) \\ \wedge \\ m^* = m \end{array} \ : \ \begin{array}{c} m^* \leftarrow \mathcal{B}(|\mathsf{M}^*\rangle, \mathsf{sk}) \\ (\mathsf{crt}^*, |\mathsf{M}^*\rangle) \leftarrow \mathcal{A}(\mathsf{pk}, |\mathsf{M}\rangle) \\ (|\mathsf{M}\rangle, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ m \leftarrow\!\!\$ \{0,1\} \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We provide an illustration of this security property in Figure 4.11.

Construction

We briefly present the scheme of Bartusek and Khurana (2023), based on an idea of Broadbent and Islam (2020). This scheme uses a classical public key encryption scheme. A pair of keys in this protocol, is the same as in the underlying encryption scheme.

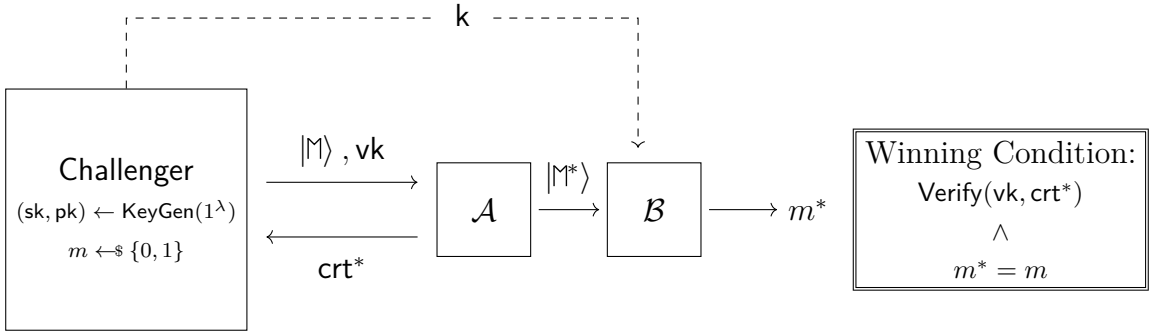

Figure 4.11: Security of deletion of an encryption scheme with certified deletion. This property states that no adversary must win this game with probability significantly greater than $1/2$.

Encrypting a message consists in sampling a random n-long BB84 state's description (x, θ), for some $n = \mathsf{poly}(\lambda)$, and set the corresponding state as the first part of the ciphertext. The second part is the message masked by the XOR of all the bits of x, at indices i such that $\theta_i = 0$. The third and final part of the ciphertext is the encryption of the basis θ using the underlying encryption scheme. Decrypting then consists in decrypting the third part with the secret key, hence recovering the basis; then using this basis to measure the BB84 state and learn x; and XOR-ing the right bits of x to undo the mask and obtain the message. The deletion is done by measuring the BB84 state in the Hadamard basis, the outcome forming the certificate. Finally, verifying such a certificate consists in checking whether the certificate corresponds to x at the indices where $\theta_i = 1$.

Construction 9: Encryption With Certified Deletion

Let $\mathsf{PKE}.\langle\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\rangle$ be a public-key encryption scheme with (post-quantum) IND-CPA security. Let $n = \mathsf{poly}(\lambda)$.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
- Return $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Enc}(\mathsf{pk}, m)$:

- Sample an n-long BB84 description, that is $x \leftarrow\!\$\ \{0,1\}^n$, and $\theta \leftarrow\!\$\ \{0,1\}^n$ such that $|\theta| = n/2$.
- Prepare $|\mathsf{M}\rangle = |x^\theta\rangle$.
- Compute $c_1 = m \oplus \bigoplus_{i\ :\ \theta_i = 0} x_i$.
- Compute $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \theta)$.
- Return $(|\mathsf{M}\rangle, c_1, c_2)$.

$\mathsf{Dec}(\mathsf{sk}, (|\mathsf{M}\rangle, c_1, c_2))$:

- Compute $\theta \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, c_2)$.
- Measure $|\mathsf{M}\rangle$ in basis θ, let x denotes the outcome.

– Return $c_1 \oplus \bigoplus_{i\,:\,\theta_i=0} x_i$.

$\mathsf{Del}(|\mathsf{M}\rangle, c_1, c_2)$:

– Measure $|\mathsf{M}\rangle$ in Hadamard basis; let crt denotes the outcome.

– Return crt

$\mathsf{Verify}((x,\theta), \mathsf{crt})$:

– Let $I = \{i \in [\![1,n]\!] \;:\; \theta_i = 0\}$.

– Return 1 if $\mathsf{crt}_{|\bar{I}} = x_{|\bar{I}}$.

– Otherwise, return 0.

Security of this scheme. The security of the scheme relies on the following observation. When given a random BB84 state $|x^\theta\rangle$, an adversary who does not know θ cannot completely learn x. Actually, by setting $I = \{i \in \{0,1\}^n \;:\; \theta_i = 0\}$, they cannot even learn the values of all x_i for $i \in I$, meaning that the quantity $\bar{x}_I := \bigoplus_{i\in I} x_i$ looks like a random bit for them. As the security of the public-key encryption scheme ensures that c_2 does not reveal any information on θ to an adversary who does not have the secret key, m is completely masked by $\bar{x}_I$. Let us now analyze the deletion part of the security. If one measures $|x^\theta\rangle$ in the diagonal basis, then they learn the values of the bits x_i for $i \in \{0,1\}^n \setminus I$, but they definitely lose information on $x_i \in I$. Because an adversary who does not know θ has no choice but to measure the whole $|x^\theta\rangle$ in the diagonal basis to produce a valid certificate, then they lose all information on the mask $\bar{x}_I$ *permanently*, and thus b remains hidden from them, *even if they become unbounded and are given the secret key.*

History of Certified Deletion

In this subsection, we give a brief history of the recent advances in the certified deletion area. Note that we only consider certified deletion, where the certificate is *classical*, and not revocable keys, where it is *quantum* (as studied by Ananth, Poremba, and Vaikuntanathan (2023)). Certified deletion was introduced by Broadbent and Islam (2020), in a paper in which they provided the first security definition and a one-time secure construction. Bartusek and Khurana (2023), Hiroka, Morimae, Nishimaki, and Yamakawa (2021), and Hiroka, Morimae, Nishimaki, and Yamakawa (2022) extended the idea of certified deletion to other cryptographic primitives, from commitments to more advanced encryption schemes as attribute-based or functional encryption. However, this last work considered private verification only. The following works focused on constructing certified deletion schemes for the same primitives, this time with public verification. Hiroka, Morimae, Nishimaki, and Yamakawa (2021), Bartusek, Garg, Goyal, Khurana, Malavolta, Raizes, and Roberts (2023), Bartusek, Khurana, Malavolta, Poremba, and Walter (2023), and Bartusek, Khurana, and Poremba (2023) showed how to construct such schemes with less and less computational assumptions, ending up with schemes that only require one-way functions.

4.6 Copy-Protection

Last, but not least is the copy-protection primitive. Consider a software company that wishes to sell their program to different users. Provided that the program does not require internet connection, the company might quickly face the piracy problem. Indeed, a malicious group of users, Alice, Bob, and Charlie, could decide to share the expense of the program, by asking Alice to pay for the program, and then copying it and sharing it with Bob and Charlie. We could also imagine Alice paying for the program, and then selling copies of it half-price on a black market.

Several solutions have been imagined by program companies in the last decades, including asking for the user to be connected to the internet when they use the program, providing hardware keys (often under the form of USB keys) that users need to insert in their computers whenever they want to run the program, or simply making it complicated in practice for a malicious user to copy the program, by making use of the underlying operating system's security. None of these solutions is truly satisfying, as they use external means to protect the program, or just make the copying less simple, without completely protecting it. And this is actually unavoidable: as the program is finally a piece of classical information, a malicious user, motivated enough, will always be able to copy it.

However, such a solution might exist in the quantum world. Indeed, the no-cloning principle ensures that it is impossible to copy an arbitrary quantum state. Trying to leverage this property, inherent to quantum computing, to construct copy-protected programs led to the field of *copy-protection*. The idea of this primitive is to have two procedures, the first one to "protect" a program, that is, given a program's description, to produce a quantum encoding of this program. And the second one to evaluate this quantum encoding on any input. Importantly, this evaluation procedure must not destroy the state, as we want the "quantum program" to be reusable. The protection of the program must ensure that no malicious client Alice can produce two functional quantum programs out of a single copy, where, by functional we mean that both programs can be used to compute the correct outcome of the original program on a random input.

As we will see in the following, it turns out that it is impossible to construct a copy-protection scheme for any program, in the sense that there exist families of functions that cannot be copy-protected. Instead of such a generic construction, we can find in the literature attempts to construct this primitive for specific classes of functions, but, even with this restriction, constructing copy-protection schemes remains a difficult task. Indeed, since the introduction of this primitive by Aaronson (2009), constructions have been found for only a few families of functions. In particular, we will present in this section copy-protection for decryption programs (also known as *single-decryptor* schemes), copy-protection for pseudorandom functions, and copy-protection for point functions (functions that return 1 on only one input, and 0 on every other input). Note that we used both the terms of programs and functions in the paragraph above. In the following, we will use the function formalism, meaning that the protection procedure takes as input a function's description, and the evaluation procedure computes this function on a given input.

Therefore, in this section, we present definitions and history of copy-protection for the aforementioned class of functions, and give some constructions examples. In the end of the section, we discuss a primitive close to copy-protection, named *secure software leasing*. This primitive allows a client, previously given a quantum program, to return the program to the vendor who can check whether the returned program is indeed the one they sent.

Definitions

We first give a general definition for copy-protection of an arbitrary family of functions $\mathcal{F}$, including its correctness and security properties. Note that in the following, for ease of understanding, we redefine these notions for each specific family of functions that we consider. Also, we assume that every function f in the family $\mathcal{F}$ can be efficiently described by a — unique — bitstring d_f. For sake of conciseness, as it is always clear from the context, we use f for both the function and its description. For instance, we write $\mathsf{Protect}(f)$ instead of $\mathsf{Protect}(d_f)$.

Definition 31 (Copy-Protection of a Family $\mathcal{F}$)**.** A copy-protection scheme, for a family of functions $\mathcal{F}$ with the same domain $\mathcal{X}$ and codomain $\mathcal{Z}$, is composed of two algorithms $\langle \mathsf{Protect}, \mathsf{Eval} \rangle$ defined in the following way:

- $|\text{✪}\rangle \leftarrow \mathsf{Protect}(1^\lambda, f)$. The protection algorithm takes as input a security parameter, and the classical description of a function $f \in \mathcal{F}$, and outputs a quantum encoding $|\text{✪}\rangle$ of f.
- $z \leftarrow \mathsf{Eval}(|\text{✪}\rangle, x)$. The evaluation algorithm takes as input a quantum encoding $|\text{✪}\rangle$, and an input x in $\mathcal{X}$, and outputs a bitstring z in the codomain of $\mathcal{F}$.

In addition, a copy-protection scheme must satisfy the following properties.

Correctness. The evaluation of the quantum encoding of any function f on any input x, must always return $f(x)$. More precisely, for any $f \in \mathcal{F}$ and any $x \in \mathcal{X}$,

$$\Pr\Big[\mathsf{Eval}(|\text{✪}\rangle, x) = f(x) \ : \ |\text{✪}\rangle \leftarrow \mathsf{Protect}(1^\lambda, f)\Big] \geq 1 - \mathsf{negl}(\lambda)$$

Anti-piracy security. Consider a triple of collaborating malicious users Alice, Bob, and Charlie. In a first phase, Alice splits the quantum encoding of a random function into two (possibly entangled) states; and in a second phase, Bob and Charlie, each one given one of these states and a challenge input, make a guess about the evaluation of the function on this input. The unclonability property states that Bob and Charlie must not make both a correct guess. Note that the challenge inputs given to Bob and Charlie can be different. In particular, the formal definition below is parametrized by a distribution over challenge pairs, to be distributed to Bob and Charlie. We discuss later in the section the need for such challenge distributions, through some specific examples. For the moment, we just claim that, for a given family of functions, some challenge distributions capture the security we expect from this definition, while others do not, and that this can change depending on the family we consider.

More formally, we define the following game, parametrized by a security parameter λ, a distribution $\mathcal{D}$ over $\mathcal{F}$, and a family of challenge distributions $\{\mathcal{D}_f\}_{f\in\mathcal{F}}$, where each $\mathcal{D}_f$ yields pairs over $\mathcal{X} \times \mathcal{X}$. The game is between a challenger and a triple of QPT algorithms $(\mathcal{A}, \mathcal{B}, \mathcal{C})$.

- **Setup phase:**
  - The challenger samples a function $f \leftarrow \mathcal{D}$.
  - The challenger computes $|\text{✪}\rangle \leftarrow \mathsf{Protect}(1^\lambda, f)$.

– The challenger sends $|\text{☼}\rangle$ to $\mathcal{A}$.

- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\text{☼}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\text{☼}^*\rangle_1$ to $\mathcal{B}$, and $|\text{☼}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples $(x_1, x_2) \leftarrow\!\!\$\ \mathcal{D}_f$.
  - The challenger sends x_1 to $\mathcal{B}$, and x_2 to $\mathcal{C}$.

Let z_1^* denotes the output of $\mathcal{B}$, and z_2^* denotes the output of $\mathcal{C}$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if $z_1^* = f(x_1)$ and $z_2^* = f(x_2)$. We say that a copy-protection scheme for a family $\mathcal{F}$ has anti-piracy security with respect to the challenge distribution $\{\mathcal{D}_f\}_{f\in\mathcal{F}}$ if no triple of adversaries wins this game with probability significantly greater than the so-called *trivial winning probability* for this game, defined below.

Definition 32 (Trivial Winning Probability)**.** Consider the following strategy[17]: Alice does not split the quantum encoding, and simply forwards it to Bob, while sending nothing to Charlie. As Bob has the quantum encoding, he makes a correct guess (the correctness property ensures he can always do it). On the other hand, Charlie only receives his challenge, and returns the answer that maximizes his chance of making a correct guess. Denote the winning probability of this strategy as p_1, and define p_2 similarly, except that the roles of Bob and Charlie are swapped. We call trivial winning probability $\max\{p_1, p_2\}$.

More precisely, let $p_1(f|x_2)$ be the probability that $\mathcal{D}$ sampled f knowing that Charlie's input is x_2 (this quantity can be computed for any f and x_2 out of $\mathcal{D}$ and $\{\mathcal{D}_f\}_{f\in\mathcal{F}}$). Then the probability that Charlie makes a correct guess $z_2 \in \mathcal{Z}$, knowing only x_2, is $\sum p(f|x_2)$, where the sum is over the functions f such that $f(x_2) = z_2$. Thus, the best answer for Charlie is the one that maximizes this quantity, that is $\max_{z_2} \sum p(f|x_2)$. We define analogously $p_2(f|x_1)$ as the probability that $\mathcal{D}$ sampled f knowing x_1, hence the best answer for Bob is $\max_{z_1} \sum p(f|x_1)$. Finally, let $p(x_1)$ denotes the probability, averaged over $f \leftarrow\!\!\$\ \mathcal{D}$, that $\mathcal{D}_f$ yields x_1 as the first challenge, and define $p(x_2)$ analogously for the second challenge. The trivial winning probability of the corresponding game is

$$\max\left[\sum_{x_1\in\mathcal{X}} p(x_1)\max_{z_1\in\mathcal{Z}}\left(\sum_{f\in\mathcal{F}:f(x_1)=z_1} p(f|x_1)\right), \sum_{x_2\in\mathcal{X}} p(x_2)\max_{z_2\in\mathcal{Z}}\left(\sum_{f\in\mathcal{F}:f(x_2)=z_2} p(f|x_2)\right)\right]$$

Note that we provided this formula for sake of completeness, but for each family we consider for copy-protection, we always specify the value of this trivial winning probability.

Impossibility Results

It appears that not all families of functions can be copy-protected. In particular, Aaronson (2009) showed that it is impossible to construct a copy-protection scheme for *learnable families* of functions. These families are such that there exists an efficient procedure

[17] Note that the adversaries in the strategy are not necessarily efficient.

to extract the description of a function given only oracle access to it. The correctness of a copy-protection ensures that an adversary Alice, given a copy-protected function from a learnable family, can simulate the oracle access to the function, and hence use the extraction procedure. This gives her the description of the function, thus allowing the adversaries to win the anti-piracy game.

Whether all unlearnable families of functions are copy-protectable or not was later answered by Ananth and La Placa (2021). The authors defined a family of functions that is not learnable, although not copy-protectable. Functions in this family are tailor-made for this purpose. More precisely, a function in this family is such that calling it with its own code yields information on the function, that can later be used to completely recover the function's description, and thus win the anti-piracy game. This can on the other hand not be done with oracle access only, as it does not provide the code of the function, making it unlearnable.

4.6.1 Copy-Protection of Point Functions

We begin with copy-protection of point functions. Such a function is parametrized by an n-long bitstring — *the point* — and returns 0 everywhere, except on this point, where it returns 1.

In this section, we formally define copy-protection of point functions, present a construction introduced by Coladangelo, Majenz, and Poremba — which, unfortunately, does not achieve a fully satisfying security — and give a short history of the different attempts to realize this functionality.

Definitions

We first define point functions.

In all the section, we set $n = \mathsf{poly}(\lambda)$.

Definition 33 (Point Function)**.** A point function PF_y with point $y \in \{0,1\}^n$, is defined in the following way:

$$\mathsf{PF}_y(x) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

We are now ready to define copy-protection of point functions. The definition follows the general definition seen above, with $\mathcal{X} = \{0,1\}^n$, $\mathcal{Z} = \{0,1\}$, and with families of point functions of the form $\{\mathsf{PF}_y\}_{y\in\{0,1\}^n}$, each of them described with its point y.

Definition 34 (Copy-Protection of Point Functions)**.** A copy-protection scheme of a point functions family $\mathcal{F} = \{\mathsf{PF}_y\}_{y\in\{0,1\}^n}$ is composed of two algorithms $\langle \mathsf{Protect}, \mathsf{Eval} \rangle$ defined in the following way:

- $|\text{✡}\rangle \leftarrow \mathsf{Protect}(1^\lambda, y)$. The protection algorithm takes as input a security parameter, and a point $y \in \{0,1\}^n$, and outputs a quantum encoding $|\text{✡}\rangle$ of PF_y.
- $z \leftarrow \mathsf{Eval}(|\text{✡}\rangle, x)$. The evaluation algorithm takes as input a quantum encoding $|\text{✡}\rangle$, and an input x in $\{0,1\}^n$, and outputs a bit z.

In addition, a copy-protection scheme of point functions must satisfy the following properties.

Correctness. The correctness of a copy-protection scheme of point functions is simply defined following the general definition above, that is, for all $y, x \in \{0,1\}^n$,

$$\Pr\left[\mathsf{Eval}(|\text{✿}\rangle, x) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} : |\text{✿}\rangle \leftarrow \mathsf{Protect}(1^\lambda, y)\right] \geq 1 - \mathsf{negl}(\lambda)$$

Challenge distributions. When defining anti-piracy security of point functions, we take as function distribution the uniform distribution over $\{0,1\}^n$. For the challenge distributions, however, considering the uniform distribution would be meaningless, as the trivial winning probability would be almost 1. Indeed, consider that Charlie is given some challenge x_2, and needs to make a guess for $\mathsf{PF}_y(x_2)$. As x_2 is uniformly random, and independent of y, the probability that $x_2 = y$, and hence $\mathsf{PF}(x_2) = 1$, is $1/2^n$. Charlie then only has to answer 0 to make a correct guess with probability $1 - \mathsf{negl}(\lambda)$. This makes the anti-piracy security guaranteed for any scheme, as it requires that no adversaries can win the game with probability significantly greater than this trivial probability, and winning with probability significantly greater than $1 - \mathsf{negl}(\lambda)$ is simply impossible.

We thus need to find more meaningful challenge distributions. Three of them are proposed in the literature, namely the product, identical, and non-colliding distributions. The two first ones are defined according to the following arguments. As any point function PF_y only has two possible outcomes, it seems natural to send, as a challenge, either a preimage of 1, or a preimage of 0. Then, deciding which preimage to take is done by sampling uniformly from these preimages, which yields either y for the preimage of 1, or a uniformly random element in $\{0,1\}^n \setminus \{y\}$ for the preimage of 0. This results in a challenge distribution for Bob, and for Charlie, that yields y with probability $1/2$, and any other element with probability $1/2^{n+1}$. This actually defines only the marginal distribution for these challenges. We finally define the *product distribution* as the challenge distribution that samples a challenge from this marginal one for Bob and Charlie independently, and the *identical distribution* as the one that samples the same challenge — still from the same marginal distribution — for both Bob and Charlie.

As we will see in the next chapter, it turns out that constructing copy-protection of point functions in the plain model remains elusive when considering product or identical challenge distributions. As the main blocking point is the fact that Bob and Charlie can both receive the point y as challenge — in which case, it is unknown how to prove security — another distribution is considered, in which this event does not happen. It results in the *non-colliding distribution*, that is the product distribution if we remove the case when both Bob and Charlie receive the point y.

We define more formally, the product, identical, and non-colliding challenge distributions below.

Definition 35 (Challenge Distributions For Point Functions)**.**

- **Product distribution:** We denote as product distribution the family of distributions $\mathcal{D}^{prod} = \{\mathcal{D}^{prod}_y\}_{y \in \{0,1\}^n}$, defined as the following.
  - Sample $x'_1, x'_2 \leftarrow\!\!\$\ \{0,1\}^n$.
  - Return $\begin{cases} (y,y) & \text{with probability } 1/4 \\ (y,x'_2) & \text{with probability } 1/4 \\ (x'_1,y) & \text{with probability } 1/4 \\ (x'_1,x'_2) & \text{with probability } 1/4 \end{cases}$

- **Identical distribution:** We denote as identical distribution the family of distributions $\mathcal{D}^{id} = \{\mathcal{D}^{id}_y\}_{y\in\{0,1\}^n}$, defined as the following.
  - Sample $x' \leftarrow\!\!\$\ \{0,1\}^n$.
  - Return $\begin{cases} (y,y) & \text{with probability } 1/2 \\ (x',x') & \text{with probability } 1/2 \end{cases}$
- **Non-colliding distribution:** We denote as non-colliding distribution the family of distributions $\mathcal{D}^{nc} = \{\mathcal{D}^{nc}_y\}_{y\in\{0,1\}^n}$, defined as the following.
  - Sample $x'_1, x'_2 \leftarrow\!\!\$\ \{0,1\}^n$.
  - Return $\begin{cases} (y,x'_2) & \text{with probability } 1/3 \\ (x'_1,y) & \text{with probability } 1/3 \\ (x'_1,x'_2) & \text{with probability } 1/3 \end{cases}$

Remark that the distributions we just defined are *not exactly* the ones we described in the paragraph above. Consider for instance the identical definition (the same argument works for the two other distributions). We described it as yielding (y,y) with probability exactly $1/2$, while in the formal definition given above, this probability is slightly above $1/2$, as we do not ensure that x' is different from y. This actually does not make a difference, as the two cases — whether we ensure that x' is different from y or not — are indistinguishable.

Trivial winning probability. The trivial winning probability for both the product and identical challenge distributions is the same, as they both have the same marginals. When Charlie is given a challenge x_2, he learns that the point y can be x_2 with probability $1/2$, in which case the correct guess would be 1, or another random point, also with probability $1/2$, in which case the correct guess is 0. The strategy that maximizes his chances of answering correctly is then to return a random bit, allowing Alice, Bob, and him to win with probability $1/2$, which is thus the trivial winning probability. The same reasoning applies to non-colliding distribution, except that in this case, Charlie has better chances to answer correctly if he returns 0. The trivial winning probability for this distribution is more precisely $2/3$.

Anti-piracy security. We give the definition with respect to the product distribution as an example below, as the two other definitions — with respect to the two other distributions — also follow the general definition described above. This definition is defined through a game, parametrized by a security parameter λ, and between a challenger and a triple of QPT algorithms $(\mathcal{A},\mathcal{B},\mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples a point $y \leftarrow\!\!\$\ \{0,1\}^n$.
  - The challenger computes $|\text{✡}\rangle \leftarrow \mathsf{Protect}(1^\lambda, y)$.
  - The challenger sends $|\text{✡}\rangle$ to $\mathcal{A}$.
- **Splitting phase:**

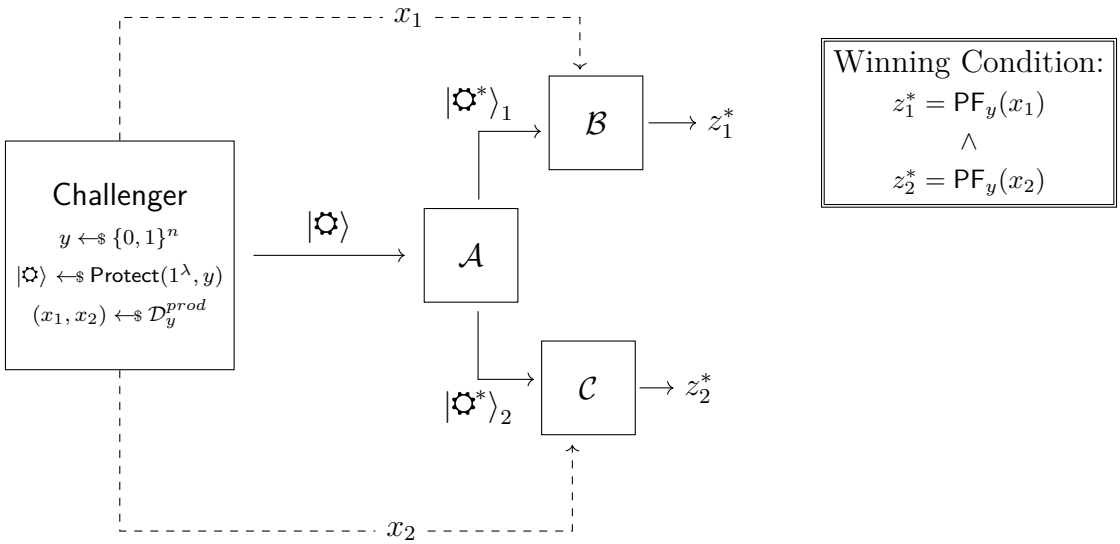

Figure 4.12: Anti-piracy security of a copy-protection scheme of point functions, with respect to the product distribution. This property states that no triple of efficient adversaries must win this game with probability significantly greater than $1/2$.

  - $\mathcal{A}$ prepares a bipartite state $|\text{☼}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\text{☼}^*\rangle_1$ to $\mathcal{B}$, and $|\text{☼}^*\rangle_2$ to $\mathcal{C}$.

- **Challenge phase:**

  - The challenger samples $(x_1, x_2) \leftarrow \mathcal{D}_y^{prod}$. That is, $x_1 = y$ or $x_1 \leftarrow\$ \{0,1\}^n$, with probability $1/2$ for each case, and, independently, $x_2 = y$ or $x_2 \leftarrow\$ \{0,1\}^n$, with probability $1/2$ for each case.
  - The challenger sends x_1 to $\mathcal{B}$, and x_2 to $\mathcal{C}$.

Let z_1^* denotes the output of $\mathcal{B}$, and z_2^* denotes the output of $\mathcal{C}$. $\mathcal{B}$ makes a correct guess if $z_1^* = \mathsf{PF}_y(x_1)$, that is if $x_1 = y$ and $z_1^* = 1$ or if $x_1 \neq y$ and $z_1^* = 0$. Similarly, $\mathcal{C}$ makes a correct guess if $z_2^* = \mathsf{PF}_y(x_2)$, that is if $x_2 = y$ and $z_2^* = 1$ or if $x_2 \neq y$ and $z_2^* = 0$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ make a correct guess. We say that a copy-protection scheme of point functions has anti-piracy security with respect to the product distribution if no triple of QPT adversaries wins this game with probability significantly greater than $1/2$.

In other words, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{cc} & z_1^* \leftarrow \mathcal{B}(|\text{☼}^*\rangle_1, x_1); z_2^* \leftarrow \mathcal{C}(|\text{☼}^*\rangle_2, x_2) \\ z_1^* = \mathsf{PF}_y(x_1) & (x_1, x_2) \leftarrow\$ \mathcal{D}_y^{prod} \\ \wedge & : \quad |\text{☼}^*\rangle_{12} \leftarrow \mathcal{A}(|\text{☼}\rangle) \\ z_2^* = \mathsf{PF}_y(x_2) & |\text{☼}\rangle \leftarrow \mathsf{Protect}(1^\lambda, y) \\ & y \leftarrow\$ \{0,1\}^n \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We provide an illustration of this game in Figure 4.12.

Coladangelo, Majenz, and Poremba's construction

Coladangelo, Majenz, and Poremba (2024) presented the first construction of copy-protection of point functions that does not rely on any "special oracle"[18], only on two quantum random oracles, $\mathcal{O}_g$ and $\mathcal{O}_h$. The protection of a point y in this protocol consists in a pair $(|r^\theta\rangle, s)$, where r is a random bitstring, θ is the image of the point y under the first oracle $\mathcal{O}_g$, and s the image of r under the second oracle $\mathcal{O}_h$. We discuss the importance of these oracles in the following. Then, to evaluate a point x given an encoding $(|r^\theta\rangle, s)$, one measures the BB84 state $|r^\theta\rangle$ in basis $\mathcal{O}_g(x)$, checks whether the image of the outcome under $\mathcal{O}_h$ is s, and returns 1 or 0 accordingly.

Construction 10: Copy-Protection of Point-Functions

Set $n, m = \mathsf{poly}(\lambda)$ such that $m(\lambda) > \lambda$. Let $\mathcal{O}_g : \{0,1\}^n \to \{0,1\}^m$, and $\mathcal{O}_h : \{0,1\}^m \to \{0,1\}^\lambda$ two quantum random oracles.

$\mathsf{Protect}(1^\lambda, y)$:

- Compute $\theta \leftarrow \mathcal{O}_g(y)$.
- Sample $r \leftarrow\!\!\$\ \{0,1\}^n$.
- Compute $s \leftarrow \mathcal{O}_h(r)$.
- Return $\left(|r^\theta\rangle, s\right)$.

$\mathsf{Eval}((|\text{☼}\rangle, s), x)$:

- Compute $\theta \leftarrow \mathcal{O}_g(x)$.
- Apply H^θ to $|\text{☼}\rangle$; store the result in a working register $|\text{☼}'\rangle$.
- Apply $\mathcal{O}_h$ in superposition to $|\text{☼}'\rangle$; store the result in a working register $|\text{☼}''\rangle$.
- Coherently checks whether $|\text{☼}''\rangle$ is equal to s or not: store the result (1 if it does, 0 otherwise) in a working register $|\text{☼}'''\rangle$.
- Measure $|\text{☼}'''\rangle$ and return the outcome.

On the need for random oracles. We briefly discuss why this scheme requires quantum random oracles. Consider first the same scheme, without oracles. That is, a protection is of the form $(|r^y\rangle, r)$. Now, an evaluation of this encoding on a point x, differing from y only on the first input would output the correct result 0 only with probability $1/2$. Using the first oracle $\mathcal{O}_g$ solves this issue as, informally speaking, the difference between the images of two points under the oracle, does not depend on the difference between these points. Concerning the second oracle, its usefulness is that an adversary could potentially use the value r of the BB84 state to break the anti-piracy security. The use of this oracle informally prevents such strategy.

[18] By that, we mean that their construction only uses a quantum random oracle, while the previous constructions proposed by Aaronson and Christiano (2012) and Aaronson, Liu, Liu, Zhandry, and Zhang (2021) are related to tailor made quantum and classical oracles.

Anti-piracy security. The security of this scheme is proven only in the aforementioned non-colliding distribution, and does not achieve super-polynomial security, as it is usually expected.[19] Informally, the reason is that the proof of security relies on what we call *simultaneous-extraction*. The idea is to reduce the task of winning the anti-piracy game, where Bob and Charlie must simultaneously *distinguish* between two different values, to a task where they must both *guess* a certain value. Then, we want to argue that if they can both distinguish with a significant advantage over the trivial winning probability, they both can guess the right value with some non-negligible probability (related to the aforementioned advantage). We do know techniques to prove such a *search-to-decision* reduction in the classical world, and even in the quantum one when we consider only local adversaries. Unfortunately, these techniques do not translate well in the non-local case (essentially because the state shared by Bob and Charlie can be entangled), and induce a loss in the winning probability. As our results suffer the same sort of issues, we give a more in-depth explanation in Chapter 5.

4.6.2 Secure Software Leasing

In this subsection, we describe the "revocable" variant of copy-protection, the so-called secure software leasing, introduced by Ananth and La Placa (2021). This primitive is revocable in the sense that it adds a verification procedure to a regular copy-protection scheme, which verifies whether a given state is a copy-protected function. The typical use-case of this primitive is when a vendor wants to lease a program to a client, and get it back after some time. The vendor then creates the copy-protected program and sends it to the client, who can evaluate it on any number of inputs, as in regular copy-protection. When the time limit is reached, the client sends a state — supposedly the copy-protected program — back to the vendor, who can check whether this is indeed the copy-protected program using the verification procedure. In practice, the protection in such a scheme is done with an additional secret key, which is also used for the verification.

Similarly to copy-protection, the definition of secure software leasing is with respect to a family of functions. In the following, we define formally secure software leasing for point functions, and present a secure protocol for secure software leasing of this family, introduced by Coladangelo, Majenz, and Poremba (2024).

Definitions

Definition 36 (Secure Software Leasing of Point Functions)**.** A secure software leasing scheme of a point functions family $\mathcal{F} = \{\mathsf{PF}_y\}_{y\in\{0,1\}^n}$ is composed of four algorithms $\langle \mathsf{KeyGen}, \mathsf{Protect}, \mathsf{Eval}, \mathsf{Verify}\rangle$ defined in the following way:

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm takes as input a security parameter 1^λ, and returns a secret key k.
- $|\text{☼}\rangle \leftarrow \mathsf{Protect}(\mathsf{k}, y)$. The protection algorithm takes as input a secret key k, and a point $y \in \{0,1\}^n$, and outputs a quantum encoding of PF_y $|\text{☼}\rangle$.
- $z \leftarrow \mathsf{Eval}(|\text{☼}\rangle, x)$. The evaluation algorithm takes as input a quantum encoding $|\text{☼}\rangle$, and an input x in $\{0,1\}^n$, and outputs a bit z.

[19] By superpolynomial security, we mean that no adversary can win the anti-piracy game with probability greater than $1/\mathsf{poly}(\lambda)$.

- $b \leftarrow \mathsf{Verify}(\mathsf{k}, y, |\text{✿}\rangle)$. The verification algorithm takes as input a secret key k, a point y, and a quantum encoding $|\text{✿}\rangle$, and returns a bit b, indicating whether the encoding is valid for the point y ($b = 1$) or not ($b = 0$).

In addition, a secure software leasing scheme of point functions must satisfy the following properties.

Correctness of evaluation. The correctness of evaluation of a secure software leasing scheme of point functions is the same as the one for copy-protection. That is, the evaluation of an honestly generated quantum encoding of a function must return the correct output. Formally, for all $y, x \in \{0,1\}^n$,

$$\Pr\left[\mathsf{Eval}(|\text{✿}\rangle, x) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} : \begin{array}{l} |\text{✿}\rangle \leftarrow \mathsf{Protect}(\mathsf{k}, y) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Correctness of verification. An honestly generated program must always be considered valid by the verification procedure. Formally, for all $y, x \in \{0,1\}^n$,

$$\Pr\left[\mathsf{Verify}(\mathsf{k}, |\text{✿}\rangle) = 1 : \begin{array}{l} |\text{✿}\rangle \leftarrow (\mathsf{k}, y) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Finite-term lessor security. Consider a malicious client Alice, who is given the quantum encoding of a program, and wants to be able to continue using it after the time-limit is over, while convincing the vendor that she has sent them back the valid quantum encoding. The finite-term lessor security states that such task is impossible.

More formally, we define the following game, between a challenger, and a pair of adversaries $\mathcal{A}$, and $\mathcal{B}$, and parametrized by a security parameter λ.

- **Setup phase:**
  - The challenger samples a key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and a point $y \leftarrow\!\!\$\ \{0,1\}^n$.
  - The challenger computes $|\text{✿}\rangle \leftarrow \mathsf{Protect}(\mathsf{k}, y)$.
  - The challenger sends $|\text{✿}\rangle$ to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a state $|\text{✿}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\text{✿}^*\rangle_1$ to the challenger, and $|\text{✿}^*\rangle_2$ to $\mathcal{B}$.
- **Challenge phase:**
  - The challenger computes $b \leftarrow \mathsf{Verify}(\mathsf{sk}, |\text{✿}^*\rangle_1)$.
  - With probability $1/2$, the challenger sets $x = y$, or, also with probability $1/2$, samples $x \leftarrow\!\!\$\ \{0,1\}^n$.
  - The challenger sends x to $\mathcal{B}$.

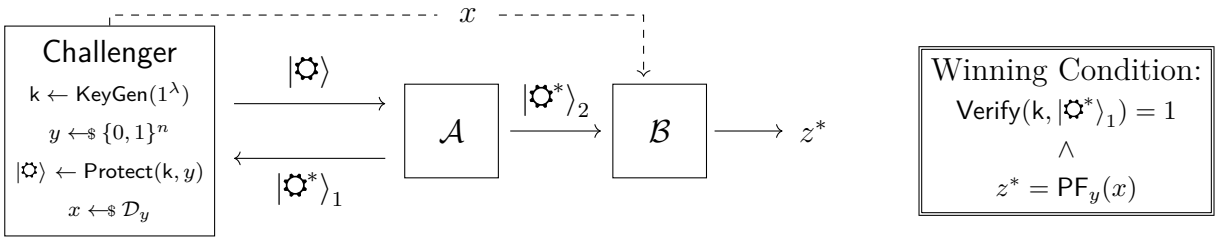

Figure 4.13: Finite-term lessor security of a secure software leasing scheme. $\mathcal{D}_y$ denotes the distribution that yields y with probability $1/2$, and a uniformly random bitstring in $\{0,1\}^n$ with probability $1/2$. This security property states that no efficient adversaries must be able to win this game with probability significantly greater than $1/2$.

Let z^* denotes the output of $\mathcal{B}$. $\mathcal{A}$ and $\mathcal{B}$ win the game if $b = 1$ and $z^* = \mathsf{PF}_y(x)$. We then say that a secure software leasing scheme has finite-term lessor security if no efficient adversaries can win this game with probability significantly greater than $1/2$. In other words, if for all QPT adversaries $\mathcal{A}$, and $\mathcal{B}$:

$$\Pr\left[\begin{array}{c|l} & z^* \leftarrow \mathcal{B}(|\text{☼}^*\rangle_2, x) \\ & x = y \text{ w.p. } 1/2 \text{ or } x \leftarrow\$ \{0,1\}^n \text{ w.p. } 1/2 \\ b = 1 & b \leftarrow \mathsf{Verify}(\mathsf{k}, |\text{☼}^*\rangle_1) \\ \wedge & (|\text{☼}^*\rangle_{12}) \leftarrow \mathcal{A}(|\text{☼}\rangle) \\ z^* = \mathsf{PF}_y(x) & |\text{☼}\rangle \leftarrow \mathsf{Protect}(\mathsf{k}, y) \\ & y \leftarrow\$ \{0,1\}^n \\ & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We provide an illustration of this finite-term lessor security in Figure 4.13.

Remark 2. We presented the definition of Coladangelo, Majenz, and Poremba (2024). Note that, in the original definition by Ananth and La Placa (2021), the verification algorithm does not take the description of the function (here, the point y) as input. We note that this difference is actually insignificant, as we can generically turn any secure software leasing scheme following the definition we presented above to a secure software leasing scheme with the same correctness and security properties following the original definition. This transformation only assumes a symmetric encryption scheme with (post-quantum) semantic security.

Let us denote a scheme following the definition above as the CMP scheme, and the scheme following the original definition obtained by applying the transformation as the AL scheme. A key in the AL scheme is composed of a key in the CMP scheme, and an encryption key. Protecting a function in the AL scheme consists in protecting it using the CMP scheme, and adding to it the encryption of the function's description. Finally, to verify a quantum software in the AL scheme, decrypt the ciphertext part to get the description of the function, and use the verification procedure of the CMP scheme.

The semantic security of the underlying encryption scheme ensures that we can replace the encryption of the function's description by an encryption of 0, resulting in no advantage for the adversaries in the finite-term lessor security game.

Construction

We present below the construction of a secure software leasing scheme of point functions introduced by Coladangelo, Majenz, and Poremba (2024). This construction is based on their copy-protection scheme (construction 10), more precisely the protection and evaluation algorithms are the same. Verifying an alleged quantum encoding of the point function PF_y consists in running the evaluation algorithm on the point y, and accepting the encoding only if the outcome is 1.

We present this modified construction formally below. As the construction does not use keys, we omit the key generation procedure, and the keys as inputs.

Construction 11: Secure Software Leasing of Point Functions

Set $n, m = \mathsf{poly}(\lambda)$ such that $m(\lambda) > \lambda$. Let $\mathcal{O}_g : \{0,1\}^n \to \{0,1\}^m$, and $\mathcal{O}_h : \{0,1\}^m \to \{0,1\}^\lambda$ two quantum random oracles.

$\mathsf{Protect}(1^\lambda, y)$:

- Compute $\theta \leftarrow \mathcal{O}_g(y)$.
- Sample $r \leftarrow\!\!\$\ \{0,1\}^n$.
- Compute $s \leftarrow \mathcal{O}_h(r)$.
- Return $\left(|r^\theta\rangle, s\right)$.

$\mathsf{Eval}((|\text{☼}\rangle, s), x)$:

- Compute $\theta \leftarrow \mathcal{O}_g(x)$.
- Apply H^θ to $|\text{☼}\rangle$; store the result in a working register $|\text{☼}'\rangle$.
- Apply $\mathcal{O}_h$ in superposition to $|\text{☼}'\rangle$; store the result in a working register $|\text{☼}''\rangle$.
- Coherently check whether $|\text{☼}''\rangle$ is equal to s or not: store the result (1 if it does, 0 otherwise) in a working register $|\text{☼}'''\rangle$.
- Measure $|\text{☼}'''\rangle$ and return the outcome.

$\mathsf{Verify}(y, (|\text{☼}\rangle, s))$:

- Compute $z \leftarrow \mathsf{Eval}((|\text{☼}\rangle, s), y)$.
- Return z.

Correctness and security. The correctness of evaluation and verification both follow immediately from the correctness of the underlying copy-protection protocol. Contrarily to its copy-protection version, this construction achieves standard security. We try to give some intuition on why this is the case, we refer the interested reader to the paper of Coladangelo, Majenz, and Poremba (2024) for more detailed information and a complete proof. Notice that the finite-term lessor security game can be seen as a game between three adversaries, Alex, Billy, and Clover, and a challenger, analogously to the copy-protection game. In this version of the game, Alex and Billy respectively represent Alice and Bob

(the adversaries defined above), and Clover is given the "fake" quantum encoding $|\text{☼}^*\rangle_1$, and a challenge y, and *must run the evaluation algorithm on it*, and return the outcome. This version of the game is equivalent to the one we presented, as Clover actually plays the role of the challenger, in the verification part. This important constraint on Clover (recall that in copy-protection games, Charlie can run any (efficient) quantum computation on his state and challenge) is actually the reason why the proof completely follows, and this construction achieves standard security.

4.6.3 History of Copy-Protection and Secure Software Leasing

In this section, we present a history of different results and constructions regarding copy-protection and secure software leasing.

Aaronson (2009) defined copy-protection and presented a construction based on a quantum oracle. In this paper, he also proved that no *learnable functions* (roughly, functions whose behavior can be learned from a polynomial number of input-output pairs) can be copy-protected. This field did not get a lot of attention until recently, where Ananth and La Placa (2021) defined secure software leasing and proved two impossibility results: one on the impossibility of secure software leasing for learnable functions and the other one on the impossibility of copy-protection for the so-called *de-quantumizable functions*, a family of functions that are not learnable, thus extending the impossibility result of Aaronson. In the same paper, they constructed a secure software leasing protocol for a subclass of evasive functions, namely *searchable compute-and-compare functions*. The same year, Coladangelo, Majenz, and Poremba[20] presented a first concrete copy-protection protocol for point functions in the quantum oracle model. However, this protocol does not achieve standard security (an adversary can win the anti-piracy game with a constant — yet small — probability). Nevertheless, they extended this protocol to a secure software leasing one, with standard security. Aaronson, Liu, Liu, Zhandry, and Zhang (2021) provided a new construction for copy-protection, this time based only on a classical oracle, improving his previous construction. This paper also provides a *copy-detection* construction for any watermarked functions. We will not detail the notions of *copy-detection* (a task close to copy-protection but weaker than it) and *watermarkable functions*, we refer the reader to the papers of Aaronson, Liu, Liu, Zhandry, and Zhang (2021) and of Cohen, Holmgren, Nishimaki, Vaikuntanathan, and Wichs (2016) for more information on these notions. Soon after, Kitagawa, Nishimaki, and Yamakawa (2021) constructed a secure software leasing scheme for pseudorandom functions and searchable compute-and-compare functions based on a relaxed version of watermarking. Broadbent, Jeffery, Lord, Podder, and Sundaram (2021) showed that copy-protection can be achieved without any assumptions. To achieve it, the authors used a weaker definition for correctness, namely correctness with respect to a distribution, and a new adversary model: the *honest-malicious adversary*. In this model, in the anti-piracy security game, Bob can perform any computation on σ_B to return y_B as in the original definition, but Charlie has to use the Eval algorithm on σ_C and x_C and returns the output as y_C. They also proved that copy-protection with honest-malicious adversary implies secure software leasing. Coladangelo, Liu, Liu, and Zhandry (2021) presented the first construction of a copy-protection scheme in the plain model by using coset states to copy-protect a subclass of pseudorandom functions. Their construction is based in particular on the computational hardness of LWE and on the existence of

[20]This paper was published a few years later: Coladangelo, Majenz, and Poremba (2024)

post-quantum indistinguishable obfuscation. Finally, two papers (Ananth, Kaleoglu, Li, Liu, and Zhandry (2022) and Ananth, Kaleoglu, and Liu (2023)) improved the result of Coladangelo, Majenz, and Poremba (2024) by constructing a copy-protection scheme for point functions in the quantum oracle model (the former paper leverages coset states, while the latter only BB84 states), with standard security.

Remark that, apart from the copy-protection of pseudorandom functions of Coladangelo, Liu, Liu, and Zhandry (2021), all the other schemes considering standard malicious-malicious adversaries — as opposed to aforementioned honest-malicious adversaries — are constructed in the quantum random oracle model. In particular, copy-protection for point functions in the plain model (that is without using the random oracle model) remains elusive. With this in mind, Kitagawa and Nishimaki (2023), Chevalier, Hermouet, and Vu (2023), Ananth and Behera (2024), and Chevalier, Hermouet, and Vu (2024b) proposed constructions for such a scheme. Although the security of these constructions are indeed in the plain model, the first one only achieves security based on a relaxed security notion (named one-out-of-many security), the second one achieves security for the arguably "less natural" non-colliding challenge distribution, and the two last constructions rely on unproven conjectures.

In the next section, we present a special copy-protection primitive, single-decryptor, that provides unclonable decryption keys.

4.6.4 Single-Decryptor

Consider a public key infrastructure — that is a network of users all owning a pair of encryption and decryption keys. Some user, Ava, wants to give her friend Alice the ability to decrypt the messages she (Ava) encrypts. With a regular encryption scheme, Ava would need to give Alice the decryption key, but she is afraid that Alice might be malicious and try to share the key with other users. A single-decryptor scheme (or simply single-decryptor) exactly gives a solution to this issue, by allowing Ava to generate quantum unclonable decryption keys. With such a scheme, she can give Alice a quantum decryption key, ensuring her that Alice will not be able to send the key to anyone else, without losing the ability to decrypt her messages.

This primitive actually exists in the private and in the public settings. In both settings, a secret key is used to generate decryption keys; in the private one, this secret key also serves for encrypting; while in the public one, there is another (public) key for this purpose.

As mentioned above, single-decryptor can be seen as a special case of copy-protection, where each function to protect is defined by the decryption algorithm of a (public or private) encryption scheme, hardwired with a decryption key. The distribution over the functions for the anti-piracy security is then defined according to the key generation procedure of the underlying encryption scheme, and the challenge distributions simply yields pairs of random ciphertext. Remark that we mentioned that single-decryptor is not exactly captured by copy-protection, as the correctness property is slightly different, as we will see in the following.

In the rest of this subsection, we formally define single-decryptors, and give an example construction, presented by Georgiou and Zhandry (2020).

Definitions

We define single-decryptor in the private settings only, as the public version is defined similarly in a natural way. That is, the key generation also returns a public key, that is used as input in the encryption algorithm, and the adversaries in the security definitions are given this encryption key.

Definition 37 (Single-Decryptor)**.** A single-decryptor for a message space $\mathcal{M}$[21] is composed of four algorithms $\langle \mathsf{KeyGen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ defined in the following way:

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$. The key generation algorithm takes as input a security parameter 1^λ, and returns a secret key k.
- $|\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{k})$. The protection algorithm takes as input a secret key k, and outputs a quantum decryption key $|\mathfrak{k}\rangle$.
- $c \leftarrow \mathsf{Enc}(\mathsf{k}, m)$. The evaluation algorithm takes as input a key k, and a message $m \in \mathcal{M}$, and outputs a classical ciphertext c.
- $m \leftarrow \mathsf{Dec}(|\mathfrak{k}\rangle, c)$. The verification algorithm takes as input a quantum key $|\mathfrak{k}\rangle$, and a ciphertext c, and returns a message m.

In addition, a single-decryptor must satisfy the following properties.

Correctness. The encryption of a message must always decrypt to this message. More precisely, for all message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(|\mathfrak{k}\rangle, c) = m \ : \ \begin{array}{l} c \leftarrow \mathsf{Enc}(\mathsf{k}, m) \\ |\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{k}) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Indistinguishability. We distinguish one-time indistinguishability from many-time indistinguishability. For a private unclonable encryption scheme to have *one-time* indistinguishability, no malicious (computationally bounded) adversary must be able to distinguish between the ciphertexts of two messages of same length. More formally, for all $m, m' \in \mathcal{M}$ such that $|m| = |m'|$,

$$\begin{array}{rcl} \{\mathsf{Enc}(\mathsf{k}, m) & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \\ & \approx_c & \\ \{\mathsf{Enc}(\mathsf{k}, m') & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \end{array}$$

Many-time indistinguishability is defined analogously, except that the adversary is given this time a polynomial number of ciphertexts. More formally, for all $\kappa = \mathsf{poly}(\lambda)$, and all $m_1, \ldots, m_\kappa, m'_1, \ldots, m'_\kappa \in \mathcal{M}$ such that $|m_i| = |m'_i|$ for every i,

$$\begin{array}{rcl} \{(\mathsf{Enc}(\mathsf{k}, m_0), \ldots, \mathsf{Enc}(\mathsf{k}, m_\kappa)) & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \\ & \approx_c & \\ \{(\mathsf{Enc}(\mathsf{k}, m'_0), \ldots, \mathsf{Enc}(\mathsf{k}, m'_\kappa)) & : & \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)\} \end{array}$$

[21] Similarly to unclonable encryption, we set in the following $\mathcal{M} = \{0,1\}^n$ for $n = \mathsf{poly}(\lambda)$.

Anti-piracy security. Consider a triple of collaborating malicious users Alice, Bob, and Charlie. In a first phase, Alice splits a random quantum key into two (possibly entangled) states; in a second phase, Bob and Charlie, each one given one of these states and a random ciphertext, make a guess about the plaintext. The unclonability property states that Bob and Charlie must not make both a correct guess, with probability greater than $1/|\mathcal{M}|$.

More formally, we define the following game, parametrized by a security parameter λ, and between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples a key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  - The challenger prepares $|\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{k})$.
  - The challenger sends $|\mathfrak{k}\rangle$ to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\mathfrak{k}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\mathfrak{k}^*\rangle_1$ to $\mathcal{B}$, and $|\mathfrak{k}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples a message $m \leftarrow\!\!\$\ \mathcal{M}$.
  - The challenger computes $c \leftarrow \mathsf{Enc}(\mathsf{k}, m)$.
  - The challenger sends c to both $\mathcal{B}$ and $\mathcal{C}$.

Let m_1^* denotes the output of $\mathcal{B}$, and m_2^* denotes the output of $\mathcal{C}$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if $m_1^* = m$ and $m_2^* = m$. We then say that an unclonable encryption scheme has unclonability if no adversaries can win this game with probability significantly greater than $1/|\mathcal{M}|$. That is, if for all triples of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{c} m_1^* = m \\ \wedge \\ m_2^* = m \end{array} : \begin{array}{l} m_1^* \leftarrow \mathcal{B}(|\mathfrak{k}^*\rangle_1, c), m_2^* \leftarrow \mathcal{C}(|\mathfrak{k}^*\rangle_2, c) \\ c \leftarrow \mathsf{Enc}(\mathsf{k}, m) \\ m \leftarrow\!\!\$\ \mathcal{M} \\ |\mathfrak{k}^*\rangle_{12} \leftarrow \mathcal{A}(|\mathfrak{k}\rangle) \\ |\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{k}) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{|\mathcal{M}|} + \mathsf{negl}(\lambda)$$

Anti-piracy security (CPA-style). We now define the CPA-style variant of this definition. Define the following game, parametrized by a security parameter λ, and between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples a key $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  - The challenger prepares $|\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{k})$.

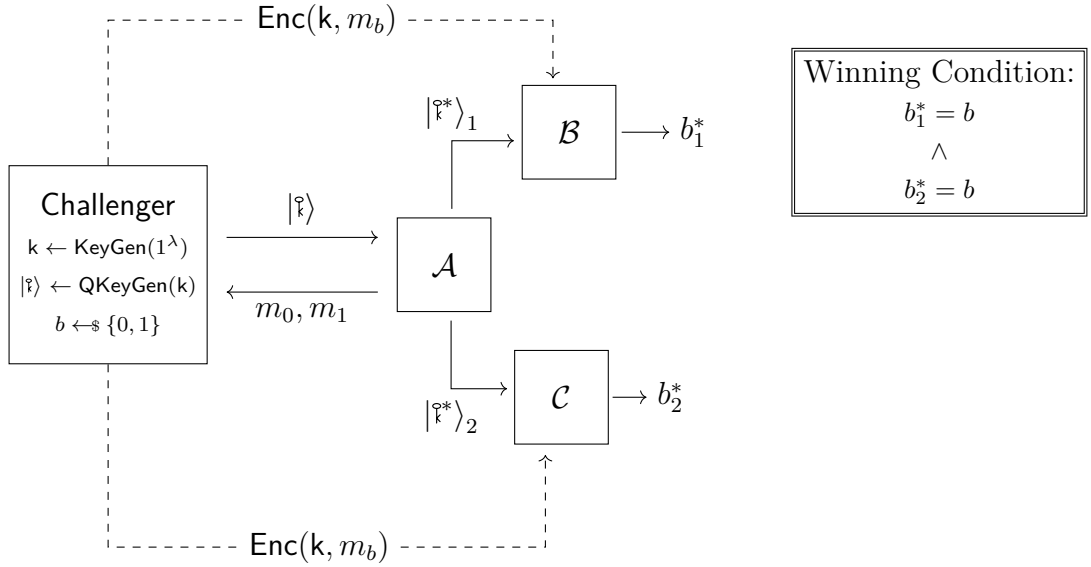

Figure 4.14: Anti-piracy security (CPA-style) of a single-decryptor scheme. This property states that no efficient adversaries must be able to win this game with probability significantly greater than $1/2$.

  - The challenger sends $|\mathfrak{k}\rangle$ to $\mathcal{A}$.

- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\mathfrak{k}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\mathfrak{k}^*\rangle_1$ to $\mathcal{B}$, and $|\mathfrak{k}^*\rangle_2$ to $\mathcal{C}$.
  - $\mathcal{A}$ sends a pair of messages (m_0, m_1) to the challenger.
- **Challenge phase:**
  - The challenger samples a bit $b \leftarrow\!\!\$\ \{0,1\}$.
  - The challenger computes $c \leftarrow \mathsf{Enc}(\mathsf{k}, m_b)$.
  - The challenger sends c to both $\mathcal{B}$ and $\mathcal{C}$.

Let b_1^* denotes the output of $\mathcal{B}$, and b_2^* denotes the output of $\mathcal{C}$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if $b_1^* = b$ and $b_2^* = b$. We then say that an unclonable encryption scheme has unclonability if no adversaries can win this game with probability significantly greater than $1/2$. That is, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{c} b_1^* = b \\ \wedge \\ b_2^* = b \end{array} : \begin{array}{l} b_1^* \leftarrow \mathcal{B}(|\mathfrak{k}^*\rangle_1, c), b_2^* \leftarrow \mathcal{C}(|\mathfrak{k}^*\rangle_2, c) \\ c \leftarrow \mathsf{Enc}(\mathsf{k}, m_b) \\ b \leftarrow\!\!\$\ \{0,1\} \\ (|\mathfrak{k}^*\rangle_{12}, (m_0, m_1)) \leftarrow \mathcal{A}(|\mathfrak{k}\rangle) \\ |\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{k}) \\ \mathsf{k} \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We provide an illustration of this anti-piracy property in Figure 4.14.

Remark 3. In the definition above, Bob and Charlie receive the same message. Remark that we can also define anti-piracy, with respect to a first distribution over pairs of messages to be encrypted, and to a second one over pairs of random bitstrings to be used as the randomness in the encryption algorithm. Such a more involved definition will be crucial in the next chapter (Chapter 5), when we investigate the links between single-decryptors and copy-protection.

Constructions

Georgiou and Zhandry (2020) showed how to turn an unclonable encryption scheme with unclonable-indistinguishability into a single-decryptor. As we do not know any unclonable encryption construction satisfying unclonable-indistinguishability, this transform unfortunately does not provide a construction for single-decryptor. However, we remark that, if the underlying unclonable encryption scheme has the simpler unclonability property, then this transform yields a single-decryptor with the regular anti-piracy security. This immediately yields such a single-decryptor scheme, using the unclonable encryption of Broadbent and Lord (2020) as the underlying scheme. We describe below the general transform of Georgiou and Zhandry (2020), and then the single-decryptor scheme resulting of this transform, applied to the Broadbent and Lord (2020) scheme.

The secret key of a single-decryptor scheme obtained using this transform, is made of a key from the underlying unclonable encryption scheme, as well as an n-long random bitstring (we set the message space $\mathcal{M} = \{0,1\}^n$). A quantum decryption key is then the quantum encryption of this random string. Encrypting a message consists in returning the unclonable encryption's key (not the random string), and the message one-time padded with the string. Finally, the decryption of such a ciphertext is done by decrypting the quantum encryption, yielding the random string, and using this string to uncompute the one-time pad and recover the message.

Construction 12: Single-Decryptor From Unclonable Encryption

Let $n = \mathsf{poly}(\lambda)$. Let $\mathsf{UE}.\langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\rangle$ be an unclonable encryption scheme, with message space $\mathcal{M} = \{0,1\}^n$.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample $\mathsf{k} \leftarrow \mathsf{UE.KeyGen}(1^\lambda)$, and $r \leftarrow\!\!\$\ \{0,1\}^n$.
- Return (k, r).

$\mathsf{QKeyGen}((\mathsf{k}, r))$:

- Prepare $|\mathsf{M}\rangle \leftarrow \mathsf{UE.Enc}(\mathsf{k}, r)$.
- Return $|\mathsf{M}\rangle$.

$\mathsf{Enc}((\mathsf{k}, r), m)$:

- Compute $c = m \oplus r$.
- Return (k, c).

$\mathsf{Dec}(|\mathsf{M}\rangle, (\mathsf{k}, c))$:

- Compute $r = \mathsf{UE.Dec}(\mathsf{k}, |\mathsf{M}\rangle)$.
- Return $c \oplus r$.

Anti-piracy security. The anti-piracy security follows directly from the unclonability of the underlying unclonable encryption scheme. Consider a triple of efficient adversaries Alice, Bob, and Charlie, who win the anti-piracy game for this single-decryptor with probability p. We construct another triple of efficient adversaries, Alex, Billy, and Clover, who win the unclonability game of the underlying unclonable encryption scheme with the same probability, which contradicts the unclonability property, and thus prove the anti-piracy one.

Alex, given a random quantum ciphertext (encrypting a message m), runs Alice on it to get a bipartite quantum state, that she shares among Billy and Clover. She also samples a random n-long bitstring c, and sends it to both Billy and Clover. Billy receives a key k, then runs Bob on his share of the bipartite state, the key, and this random string c. Note that there is a string m' such that $c = m \oplus m'$. With this notation, Bob's input corresponds to the single-decryptor's encryption of the message m'. He then returns m' with non-negligible probability, and Billy returns $c \oplus m'$. This analysis can actually be done for Billy and Clover simultaneously, yielding a correct answer for both of them with non-negligible probability, and finishing the reduction.

The proof of anti-piracy security (CPA-style) follows the same pattern.

Construction 13: Transform Applied To Broadbent and Lord's Construction

Let $n = \mathsf{poly}(\lambda)$.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample an n-long BB84 state description, that is $r \leftarrow\!\!\$\ \{0,1\}^n$, and $\theta \leftarrow\!\!\$\ \{0,1\}^n$ such that $|\theta| = n/2$.
- Sample $r' \leftarrow\!\!\$\ \{0,1\}^n$.
- Return (r, θ, r').

$\mathsf{QKeyGen}(r, \theta, r')$:

- Compute $x = r \oplus r'$.
- Return $|x^\theta\rangle$.

$\mathsf{Enc}((r, \theta, r'), m)$:

- Compute $c = m \oplus r'$.
- Return (r, θ, c).

$\mathsf{Dec}(|\mathsf{M}\rangle\ , (r, \theta, c))$:

- Measure $|\mathsf{M}\rangle$ in basis θ; let x denote the outcome.
- Return $c \oplus x \oplus r$.

Chapter

5 Towards Unclonable Cryptography in the Plain Model

In this chapter, we present our copy-protection of point functions and unclonable encryption constructions in the plain model. We also present independent results on tokenized signature schemes. This chapter is based on our following work: Chevalier, Hermouet, and Vu (2024b). We provide some supplementary materials for this chapter in Appendix A.

Chapter content

5.1 Introduction

Copy-protection of point functions has been constructed related to different oracles. The first construction we have is from Aaronson (2009) and was related to a quantum oracle. More than a decade later, Coladangelo, Majenz, and Poremba (2024) presented a concrete construction relying "only" on the quantum random oracle model. Finally, Ananth, Kaleoglu, Li, Liu, and Zhandry (2022) and Ananth, Kaleoglu, and Liu (2023) improved this result, presenting a construction with a standard security, still in the quantum random oracle model.

This quantum oracle model assumes the existence of a powerful random oracle, with arbitrary input and output spaces, that can be queried by anyone in a protocol. In particular, all the protocol's algorithms, and adversaries in the security definitions, can query it on any input, and even on superposition of classical inputs. This model turns out to be extremely useful, as it provides a real (or statistical) source of randomness, as opposed to the pseudorandomness commonly used in cryptography. While such an oracle provably cannot be implemented, it is still considered as a good heuristic regarding whether a protocol would be secure if we replace the oracle by a good cryptographic randomness generator, typically a hash function. However, again, quantum random oracles do not exist in real life, and therefore it is preferable to not use them (or any other oracles) at all, that is to define constructions in the so-called plain model. For this reason, in this chapter, we try to construct copy-protection of point functions in the plain model. This task has remained elusive for a long time, and is actually not completely solved at the time of writing this thesis. We therefore present our ideas to construct such a scheme, the resulting construction, and the bottlenecks: the ones we overcame, and the ones that still remain to be overcome.

The content of this chapter is based on two papers (Chevalier, Hermouet, and Vu (2023) and Chevalier, Hermouet, and Vu (2024b)). This chapter is articulated as follows. We first present in Sections 5.2 and 5.3 our constructions of copy-protection of point functions, and unclonable encryption, based on copy-protection of pseudorandom functions. Then, in Sections 5.4 and 5.5, we present new security definitions for copy-protection of

pseudorandom functions and single-decryptor, and show that an existing copy-protection scheme of pseudorandom functions — presented by Coladangelo, Liu, Liu, and Zhandry (2021) — satisfies this new security if the single-decryptor presented in the same paper satisfies the new single-decryptor security property. We turn to the security of this single-decryptor construction, and discuss the two main challenges in proving this new property. We finally show how to overcome the first challenge (Section 5.6), and formalize a conjecture that, if true, overcomes the second one (Section 5.7). As related contributions, we present in Section 5.8 two additional results on tokenized signature schemes: we define two new security properties of such schemes, and present a construction that satisfies them.

5.2 Copy-Protection: From Pseudorandom Functions to Point Functions

As our copy-protection of point functions is obtained from a copy-protection of pseudorandom random function in a black-box way, we start by briefly recalling the definition of these two unclonable primitives. We refer the reader to Section 4.6.1 and Appendix A.1 for more detailed definitions.

5.2.1 Copy-Protection of Point Functions

A copy-protection of point functions transforms a point function PF_y — defined by $\mathsf{PF}_y(x) = 0$ for all $x \neq y$ and $\mathsf{PF}_y(y) = 1$ — described by its point y, into a quantum state $|\text{☼}_y\rangle$ through a protection algorithm $\mathsf{Protect}$. It also provides an evaluation procedure Eval, to evaluate such a quantum encoding on any input x, to obtain $\mathsf{PF}_y(x)$, almost without perturbing the encoding, so that the evaluation can be performed a polynomial number of times.

Anti-piracy security. The anti-piracy security of a copy-protection of point functions scheme states that no efficient triple of adversaries Alice, Bob, and Charlie, can win the following game (illustrated in Figure 5.1) with a "good enough" probability, that we elaborate on below. In this game, a challenger sends a copy-protection $|\text{☼}_y\rangle$ of a random point function PF_y to Alice, who splits it, and shares the two halves between Bob and Charlie. The latter each receive a challenge input from the challenger, and must both return the correct evaluation of the function on this input in order to win the game.

Challenge distributions. Importantly, these challenges inputs are sampled from a specific challenge distribution — a parameter in the game. In this chapter, we consider three different challenge distributions:

- the product distribution, that either yields the point y to Bob, or another random input, and independently samples a challenge input for Charlie in the same way;
- the identical distribution, similar the product one, except that Bob and Charlie receive the same challenge;
- the non-colliding distribution, also similar to the product one, except that Bob and Charlie cannot both receive the point y, that is, they either both receive a random

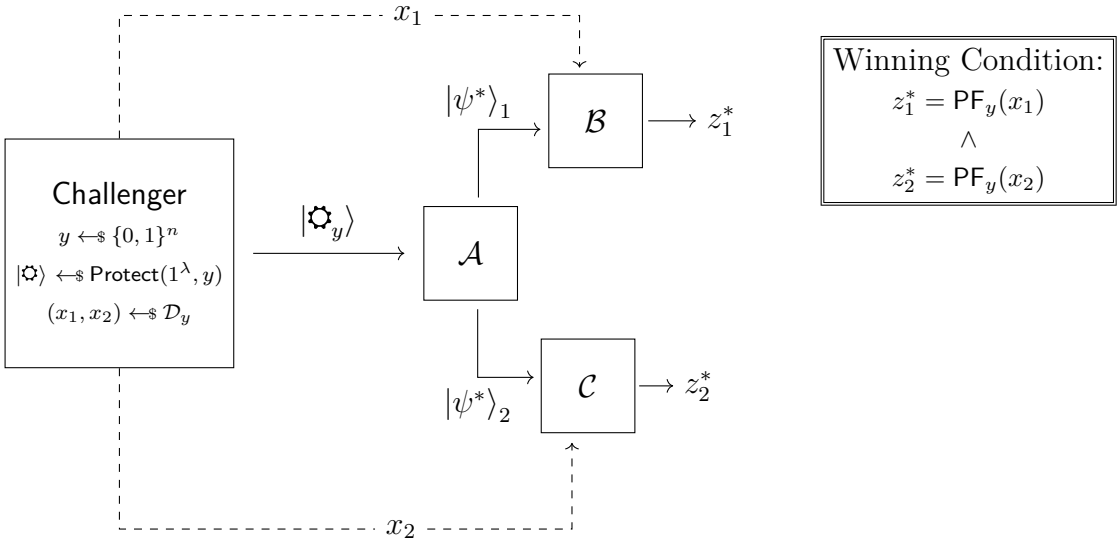

Figure 5.1: Anti-piracy security of a copy-protection scheme of point functions, with respect to a challenge distribution $\mathcal{D}$. This property states that no triple of efficient adversaries must win this game with probability significantly greater than $1/2$ if $\mathcal{D}$ is the product or identical distribution, or $2/3$ if it is the non-colliding distribution.

$\mathcal{D}_y^{prod}$	$\mathcal{D}_y^{id}$	$\mathcal{D}_y^{nc}$
(y, y)	(y, y)	
(y, x)		(y, x)
(x, y)		(x, y)
(x, x')	(x, x)	(x, x')

Figure 5.2: Product, identical, and non-colliding challenge distributions parametrized by a point $y \in \{0,1\}^n$ for some positive integer n. Each column represents one challenge distribution. Sampling from one of these distribution consists in sampling x and x' uniformly and independently at random from $\{0,1\}^n$, then picking one non-empty cell at random in the column corresponding to the chosen distribution, and finally yielding the cell.

independent input, or one receives y and the other receives a random input, each case happening with probability $1/3$.

These distributions define the maximum value for the aforementioned "good enough" probability. This probability, that we call *trivial probability*, is thus the one given by the following trivial strategy: Alice forwards the quantum encoding to Bob, who uses it to answer the challenge, while Charlie receives nothing and answers with the most probable output. This definition sets the trivial probability to $1/2$ for the product and identical distributions, and $2/3$ for the non-colliding one. We summarize these distributions in the table of Figure 5.2. In the rest of this section, and when clear from the context, we sometimes abuse the notation and simply write product distribution to denote the family of product distributions $\{\mathcal{D}_y^{prod}\}_y$. We do the same for identical and non-colliding distributions.

5.2.2 Copy-Protection of Pseudorandom Functions

In this subsection, we give a high-level definition of copy-protection of pseudorandom functions. We refer the reader to Appendix A.1 for a more complete definition.

Pseudorandom functions. We start by briefly recalling different properties of pseudorandom functions. Recall first that a family of pseudorandom functions $\{\mathsf{PRF}_\mathsf{k}\}_{\mathsf{k}\in\mathcal{K}}$, where every PRF_k has domain $\mathcal{X}$ and codomain $\mathcal{Z}$, is such that no efficient adversary can distinguish the behavior of a function from the one of a truly random function. More precisely, such an adversary is given oracle access to either a random pseudorandom function, or a truly random oracle with the same domain and codomain, must not be able to guess what function they were given better than a random guess.[1]

$$\left|\Pr_{f:\mathcal{X}\to\mathcal{Z}}\left[\mathcal{A}^{\mathcal{O}(f)}(1^\lambda)=1\right]-\Pr_{\mathsf{k}\leftarrow\$\mathcal{K}}\left[\mathcal{A}^{\mathcal{O}(\mathsf{PRF}_\mathsf{k}(\cdot))}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

We say that a pseudorandom functions family is *injective* if almost all PRF_k are injective. A pseudorandom functions family can also be *puncturing*, in which case there exists a puncturing procedure that takes a key k, and a point x, and returns a "punctured key" $\mathsf{k}\{x\}$. This punctured key can then be used in an evaluation procedure to compute PRF_k on any input $x'\neq x$. Furthermore, when the evaluation procedure is used on the punctured point, it returns a uniformly random bitstring. The security of a puncturing pseudorandom functions family states that, given a punctured key $\mathsf{k}\{x\}$, and its corresponding punctured point x, it is computationally hard to distinguish $\mathsf{PRF}_\mathsf{k}(x)$ from a random bitstring of same size.

$$\{\mathsf{k}\{x\},x,\mathsf{PRF}_\mathsf{k}(x)\}_{\mathsf{k}\leftarrow\$\mathcal{K}}\approx_c\{\mathsf{k}\{x\},x,z\}_{\substack{z\leftarrow\$\mathcal{Z}\\ \mathsf{k}\leftarrow\$\mathcal{K}}}\text{ for all }x\in\mathcal{X}$$

Finally, a pseudorandom functions family is said to be *extracting* if, even given the key, the image of a random input looks like random.

$$\{\mathsf{k},\mathsf{PRF}_\mathsf{k}(x)\}_{\substack{x\leftarrow\$\mathcal{X}\\ \mathsf{k}\leftarrow\$\mathcal{K}}}\approx\{\mathsf{k},z\}_{\substack{z\leftarrow\$\mathcal{Z}\\ \mathsf{k}\leftarrow\$\mathcal{K}}}$$

Remark that an extracting pseudorandom functions family is more generally defined with respect to some min-entropy quantity, and the equation above remains valid as long as the punctured point x is sampled from a distribution with at least this min-entropy.

Copy-protection of pseudorandom functions. A copy-protection scheme of pseudorandom functions is defined according to the general template presented in Section 4.6. More precisely, such a scheme is composed of a protection procedure, that produces quantum encoding $|\text{☼}\rangle$ of a pseudorandom functions family given its description (its key), and an evaluation algorithm that allows to evaluate a pseudorandom function on any input given its quantum encoding. The security asks that no adversaries Alice, Bob, and Charlie can win the following game with some non-negligible advantage over the trivial probability. Alice is given the quantum encoding of a random pseudorandom function, splits it, and sends the first and second halves to Bob and Charlie respectively. The latter are given a random input in the domain of the protected pseudorandom function, and have to guess the correct image. Note that, when the pseudorandom functions' codomain is large enough, the trivial probability in this game is negligible in the security parameter. We provide a formal definition of copy-protection of pseudorandom functions in Appendix A.1.

[1] Remark that pseudorandom functions families usually come with a key generation procedure that samples a random key. In particular, the underlying distribution is not necessarily uniform over $\mathcal{K}$. We abuse the notation and simply write $\mathsf{k}\leftarrow\$\,\mathcal{K}$ to denote this sampling procedure.

Indistinguishability anti-piracy security. Coladangelo, Liu, Liu, and Zhandry (2021) gave the first definition for copy-protection of pseudorandom functions, and, together with the regular anti-piracy security property defined above, they proposed a second property. This property is defined through a similar game, but instead of having to guess a random input's image, Bob and Charlie must distinguish between a "good" pair $(x, \mathsf{PRF}_\mathsf{k}(x))$, and a "bad" one (x, z) where z is sampled uniformly at random from the codomain. Note that the challenge pair given to Bob, and the one given to Charlie, are generated independently. As expected, the property asks that no adversaries must win this game with probability greater than random guessing. This property aims to capture the idea that, not only it should be impossible to split a protection of a pseudorandom function in two halves that both can evaluate the function correctly, but also that both these two halves cannot break the pseudorandomness of the function.

Reversed anti-piracy security. The copy-protection of pseudorandom functions that we need in our construction of copy-protection of point functions must satisfy a slightly different property, that we call reversed anti-piracy security. To understand the name, recall that in the indistinguishability anti-piracy security, Bob and Charlie are both asked to distinguish a good pair $(x, \mathsf{PRF}_\mathsf{k}(x))$ from a bad pair (x, z). We make the following three changes to this game:

- First, we change the definition of a bad pair: from a pair (x, z), it becomes $(x', \mathsf{PRF}_\mathsf{k}(x))$, for two independent inputs x and x'.
- Second, we make the game easier by giving the image part of the pairs to Alice. Bob and Charlie then only receive the preimage part x.
- Third, instead of having two independent challenge pairs — one for Bob, and one for Charlie — we ask the image part to be the same for both, and only the preimage part to be possibly different (subject to the challenge distribution we consider, as we explain below).

Finally, the reversed anti-piracy game (formally defined in Appendix A.1, and illustrated in Figure 5.4) is the following. Alice is given the protection of a random pseudorandom function, and the image of a random input $\mathsf{PRF}_\mathsf{k}(x)$. She splits the state, and sends the two halves to Bob and Charlie, who are then respectively given x_1 and x_2 — sampled from a challenge distribution — as challenges and must guess whether the challenge is x or not. We consider the same challenge distributions as for the copy-protection of point functions game, described in Figure 5.3.

5.2.3 Construction

We are now ready to present our construction for copy-protection of point functions in the plain model. Our idea is the following. Given a copy-protection scheme of pseudorandom functions $\{\mathsf{PRF}_\mathsf{k}\}_{\mathsf{k}\in\mathcal{K}}$ with domain $\{0,1\}^n$, a protection of a point y (an n-long bitstring) consists in a pair $\left(|\text{☼}_\mathsf{k}\rangle\,, z = \mathsf{PRF}_\mathsf{k}(y)\right)$ for a random key k. Evaluating such a pair on a point x then consists in computing the image of x under the pseudorandom function, using its quantum encoding, and then returning 0 or 1 depending on whether the outcome is z or not. Intuitively, being able to successfully split this quantum program to win in the anti-piracy game requires to be able to produce two quantum states that both can evaluate PRF_k,

$\mathcal{D}_x^{prod}$	$\mathcal{D}_x^{id}$	$\mathcal{D}_x^{nc}$
(x, x)	(x, x)	
(x, x')		(x, x')
(x', x)		(x', x)
(x', x'')	(x', x')	(x', x'')

Figure 5.3: Product, identical, and non-colliding challenge distributions parametrized by a pseudorandom random function's input $x \in \mathcal{X}$. Each column represents one challenge distribution. Sampling from one of these distributions consists in sampling x' and x'' uniformly and independently at random from $\mathcal{X}$, then picking one non-empty cell at random in the column corresponding to the chosen distribution, and finally yielding the cell.

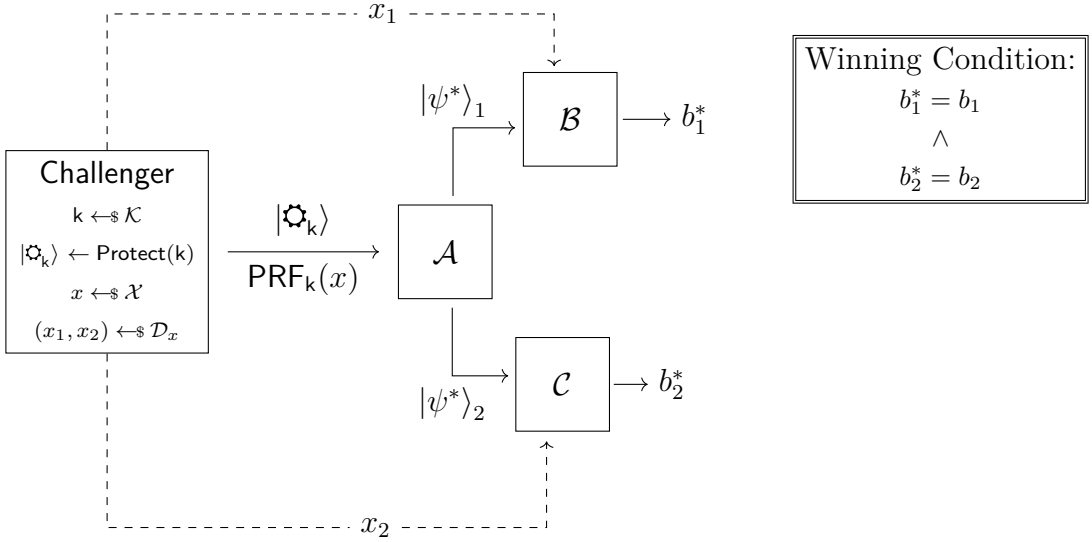

Figure 5.4: Reversed anti-piracy security of a copy-protection scheme of pseudorandom functions, with respect to a challenge distribution $\mathcal{D}$. b_1 indicates whether $x_1 = x$ or not, and similarly for b_2. This property states that no triple of efficient adversaries must win this game with probability significantly greater than $1/2$ if $\mathcal{D}$ is the product or identical distribution, or $2/3$ if it is the non-colliding distribution.

hence breaking the anti-piracy game of the copy-protection of pseudorandom functions. We provide below more intuition on the requirements for correctness and anti-piracy security of this transform.

Construction 14: Copy-Protection of Point Functions

Let $\{\mathsf{PRF}_\mathsf{k}\}_{\mathsf{k}\in\mathcal{K}}$ be a family of pseudorandom functions for some key space $\mathcal{K}$, and let $\mathsf{PRF}.\langle\mathsf{Protect},\mathsf{Eval}\rangle$ be a copy-protection scheme for this family.

$\mathsf{Protect}(1^\lambda, y)$:

- Sample $\mathsf{k} \leftarrow\!\!\$\ \mathcal{K}$.
- Prepare $|\text{⚙}_\mathsf{k}\rangle \leftarrow \mathsf{PRF.Protect}(\mathsf{k})$.
- Return $\left(|\text{⚙}_\mathsf{k}\rangle\,, \mathsf{PRF}_\mathsf{k}(y)\right)$.

$\mathsf{Eval}((|\text{⚙}_\mathsf{k}\rangle\,, z), x)$:

- Compute $z' \leftarrow \mathsf{PRF.Eval}(|\text{⚙}_\mathsf{k}\rangle\,, x)$.
- Return 1 if $z' = z$, otherwise return 0.

Correctness. Recall that correctness requires first that the evaluation on y of the protection of y returns 1 for all y, and then that the evaluation of x on the protection of y returns 0 for all y and $x \neq y$. Evaluating the protection of y on y returns 1 almost with certainty from the correctness of the underlying copy-protection scheme of pseudorandom functions, so the first point is satisfied. On the other hand, evaluating the protection of y on $x \neq y$, would return 1 if $\mathsf{PRF}_\mathsf{k}(x) = \mathsf{PRF}_\mathsf{k}(y)$. This means that the scheme does not have correctness if this happens with a non-negligible probability, when averaged over k, for a pair (x, y). However, provided that the pseudorandom functions' codomain is sufficiently large (more precisely it must be superpolynomial in the security parameter), this cannot happen. Indeed, if it were the case, an adversary given oracle access to either a pseudorandom function or a truly random function, and querying the oracle on x and y, would notice a significant difference depending on whether the oracle implements a pseudorandom function (in which case the outcome would be the same with non-negligible probability) or a truly random one (in which case the outcome is almost surely different). Such an adversary would then be able to break the security of the pseudorandom functions.

Theorem 11 (Correctness of Construction 14)**.** Assume the underlying copy-protection scheme of pseudorandom functions has correctness, and the size of the pseudorandom functions' codomain is superpolynomial in the security parameter. Then the copy-protection scheme of point functions defined in construction 14 has correctness.

Anti-piracy security. The anti-piracy security of our construction with respect to a challenge distribution is immediate if the underlying copy-protection of pseudorandom functions has reversed anti-piracy security with respect to this distribution. Indeed, the anti-piracy security game of this construction, with respect to a challenge distribution, is exactly the reversed anti-piracy game of the underlying copy-protection of pseudorandom functions, with respect to the same distribution.

Theorem 12 (Anti-Piracy of Construction 14)**.** Let $\mathcal{D}$ be a family of either product, identical, or non-colliding challenge distributions. Assume the underlying copy-protection scheme of pseudorandom functions has indistinguishability anti-piracy security with respect to $\mathcal{D}$. Then the copy-protection scheme of point functions defined in construction 14 has anti-piracy security with respect to $\mathcal{D}$.

5.3 Unclonable Encryption

In this section, we present our construction of unclonable encryption. This construction uses a copy-protection scheme of pseudorandom functions as a subroutine, and has unclonable-indistinguishability if the underlying copy-protection scheme has reversed anti-piracy security with respect to the identical distribution. We give below an informal presentation of unclonable encryption and its properties, then we present our construction and discuss its correctness and security.

5.3.1 Unclonable Encryption

An unclonable encryption scheme can be seen as a regular private encryption scheme, with a key generation procedure KeyGen, where the encryption procedure Enc produces quantum unclonable ciphertexts of classical messages, that can then be decrypted using the corresponding classical key, through the decryption procedure Dec. We present below private-key unclonable encryption for single-bit messages, but this primitive can be naturally extended to multi-bits messages, and also exists in a public-key version. We refer the reader to Section 4.4 for more details on this primitive.

Indistinguishability. An unclonable scheme has indistinguishability if no efficient adversary can distinguish the encryption of 1 from the encryption of 0. Many-time indistinguishability is defined similarly, except that the adversary is given as many ciphertexts as they want, each encrypting the same message (0 or 1), and must tell which message is encrypted. As mentioned in Section 4.4.3, it is possible to transform a one-time secure unclonable encryption scheme into a many-time secure one, simply by using a classical private encryption scheme, with many-time indistinguishability.

Unclonable-indistinguishability. The unclonable-indistinguishability property is defined through a "cloning game" (illustrated in Figure 5.5), similarly to copy-protection anti-piracy property. An adversary Alice is given a quantum ciphertext of one of two messages (0 or 1), splits it and shares it between two other adversaries Bob and Charlie. The latter are then given the key, and must both guess which message was encrypted in order to win the game. The unclonable-indistinguishability states that no adversaries can win this game with probability greater than 1/2.

5.3.2 Construction

The idea of our construction is the following. A key k_{ue} in this scheme is simply a random pseudorandom functions' input. The encryption of a bit m consists in first protecting a random pseudorandom function $\mathsf{PRF}_{\mathsf{k}_{prf}}$, then returning this protection, and either the image of the key k_{ue} under this pseudorandom function if $m = 1$, or a random string of

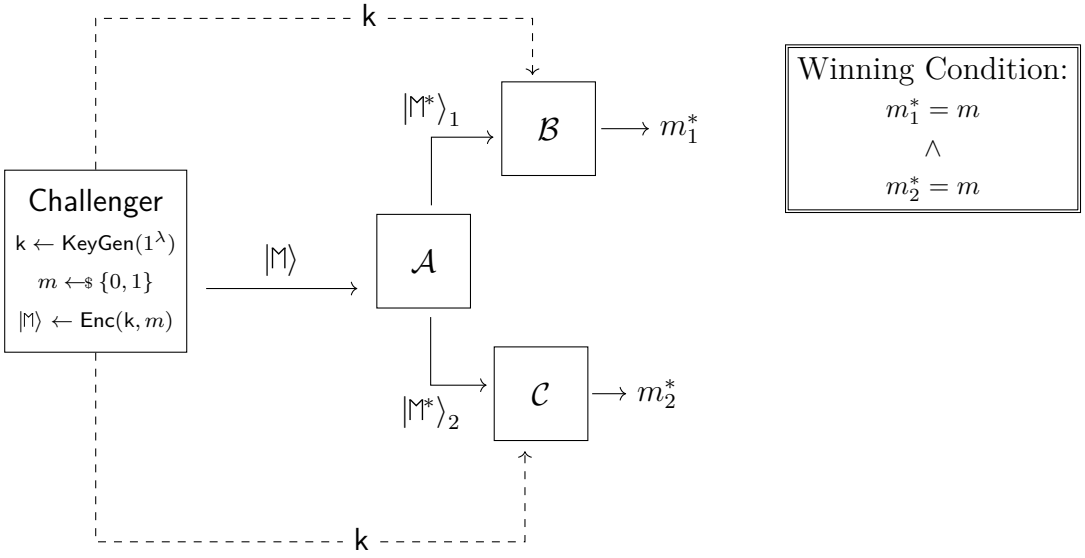

Figure 5.5: Unclonable-indistinguishability of an unclonable encryption scheme for single-bit messages. This property states that no efficient adversaries $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, can win this game with probability greater than $1/2$.

the size of the pseudorandom functions' images if $m = 0$. Then, in order to decrypt such a ciphertext given a key k_{ue}: simply evaluate the pseudorandom function on k_{ue} using the quantum protection part of the ciphertext, and checks whether the outcome is equal to the second part of the ciphertext (in this case return 1) or not (in this case return 0).

Construction 15: Unclonable Encryption From Copy-Protection of Pseudorandom Functions

Let $\{\mathsf{PRF}_\mathsf{k}\}_{\mathsf{k}\in\mathcal{K}}$ be a family of pseudorandom functions for some key space $\mathcal{K}$, domain $\{0,1\}^{nx}$, and codomain $\{0,1\}^{nz}$. Let $\mathsf{PRF}.\langle\mathsf{Protect}, \mathsf{Eval}\rangle$ be a copy-protection scheme for this family.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample $\mathsf{k}_{ue} \leftarrow\$ \{0,1\}^n$.
- Return k_{ue}.

$\mathsf{Enc}(\mathsf{k}_{ue}, m)$:

- Sample $\mathsf{k}_{prf} \leftarrow\$ \mathcal{K}$.
- Prepare $|\text{☼}\rangle \leftarrow \mathsf{PRF.Protect}(\mathsf{k}_{prf})$.
- If $m = 1$, let $z = \mathsf{PRF}(\mathsf{k}_{prf}, \mathsf{k}_{ue})$; otherwise, sample $z \leftarrow\$ \{0,1\}^{nz}$.
- Return $(|\text{☼}\rangle, z)$

$\mathsf{Dec}(\mathsf{k}_{ue}, (|\text{☼}\rangle, z))$:

- Compute $z' \leftarrow \mathsf{PRF.Eval}(|\text{☼}\rangle, \mathsf{k}_{ue})$.
- Return 1 if $z' = z$, otherwise return 0.

Correctness. The correctness of this scheme is immediate from the correctness of the underlying copy-protection scheme of pseudorandom functions, provided that the pseudorandom functions' codomain is large enough. Indeed, the probability of decrypting wrongly the message $m = 1$ is the probability of the evaluation of a protected pseudorandom function being incorrect, that is negligible if the underlying copy-protection scheme has correctness. Furthermore, the probability of decrypting wrongly the message $m = 0$ is the probability that two independent random $n_{\mathcal{Z}}$-long bitstrings are equal, that is also negligible if $n_{\mathcal{Z}}$ is large enough.

Theorem 13 (Correctness of Construction 15)**.** Assume the underlying copy-protection scheme of pseudorandom functions has correctness, and the size of the pseudorandom functions' codomain is superpolynomial in the security parameter. Then the unclonable encryption scheme defined in construction 15 has correctness.

Unclonable-indistinguishability. This scheme has unclonable indistinguishability if the underlying pseudorandom functions family is extracting, and if it has reversed anti-piracy security with respect to the identical distribution. To see that, remark that in the unclonable-indistinguishability game applied to this scheme, Alice receives $(|\text{✡}\rangle, z)$ where z is either $\mathsf{PRF}_{\mathsf{k}_{prf}}(\mathsf{k}_{ue})$ or a random string. The extracting property of the pseudorandom functions allows us to unnoticeably replace this random string by the pseudorandom function evaluation of a random string k'_{ue}. Doing this, the second part of the encryption that Alice receives is either the image of k_{ue} or of k'_{ue} under the pseudorandom function, and Bob and Charlie both receive k_{ue}. This is in fact, up to relabelling, equivalent to Alice receiving the image of k_{ue} as the second part of the ciphertext, and Bob and Charlie both receiving either k_{ue}, or k'_{ue}. This last case exactly corresponds to the indistinguishability anti-piracy game of the underlying copy-protection scheme, with respect to the identical distribution. The security thus follows directly.

Theorem 14 (Unclonable-Indistinguishability of Construction 15)**.** Assume the underlying copy-protection scheme of pseudorandom functions has reversed anti-piracy security with respect to the identical distribution. Then the unclonable encryption scheme defined in construction 15 has unclonable-indistinguishability.

5.4 Locking a Message with Coset States — A Single-Decryptor Construction

Now that we have presented our constructions of copy-protection of point functions and unclonable encryption, we need to find a copy-protection scheme of pseudorandom functions, with appropriate security. We use the only copy-protection scheme of pseudorandom functions in the literature, namely the one presented by Coladangelo, Liu, Liu, and Zhandry (2021). As the main argument regarding unclonability of this scheme is the same as the one regarding unclonability of a single-decryptor construction from the same paper, we present the single-decryptor first, and we present the copy-protection scheme in the next section. Before entering the details regarding single-decryptors, we give some intuition on this unclonability property, and see how to lock a message in a program, such that it can be unlocked only with the appropriate quantum key. We show how to leverage unclonable properties of coset states, as well as different sorts of obfuscation to achieve this purpose.

5.4.1 Compute-And-Compare Programs and Obfuscation

We first present the notions of compute-and-compare programs, the associated compute-and-compare obfuscation, and the notion of indistinguishable obfuscation. Note that we only give a high-level overview of these notions, we refer the reader to Section 3.3.6 for formal definitions.

A compute-and-compare program is parametrized by a function, a "lock-value", and a secret. On input an element in the function domain, the compute-and-compare program returns the secret if and only if the image of this element under the function is equal to the secret. Otherwise, it returns an error symbol.

$$\mathsf{CC}[f,\ell,m](x) = \begin{cases} m & \text{if } f(x)=\ell, \\ \bot & \text{otherwise .} \end{cases}$$

Consider a distribution $\mathcal{D}$ over pairs of the form $(\mathsf{CC}[f,\ell,s],\mathsf{aux})$ where aux is some (possibly quantum) auxiliary information on the program. Under the right conditions, the programs of this distribution's support can be obfuscated, in such a way that, even given its associated auxiliary information, an obfuscated program cannot be distinguished from a simulated program that returns the error symbol on every input. This obfuscation is called *compute-and-compare obfuscation*, and the corresponding obfuscation procedure is noted $\mathsf{CC\text{-}Obf}$. We often use the shorthand $\widetilde{\mathsf{CC}}[f,\ell,s]$ to denote the compute-and-compare obfuscation of the program $\mathsf{CC}[f,\ell,s]$.

$$\{\mathsf{CC\text{-}Obf}(\mathsf{CC}[f,\ell,s]),\mathsf{aux}\}_{\mathcal{D}} \approx_c \left\{\mathsf{Sim}(1^\lambda),\mathsf{aux}\right\}_{\mathcal{D}}$$

As mentioned, compute-and-compare obfuscation exists under the right conditions. More precisely, it exists when the aforementioned distribution is *unpredictable*. That is when, averaged over the distribution, no efficient adversary can guess the lock-value of a program given the program's function, and the associated auxiliary information.

$$\Pr[\mathcal{A}(f,\mathsf{aux})=\ell \ : \ (\mathsf{CC}[f,\ell,m],\mathsf{aux}) \leftarrow\!\$\ \mathcal{D}] \leq \mathsf{negl}(\lambda)$$

Indistinguishable obfuscation. Indistinguishable obfuscation, also known as best possible obfuscation, is another sort of obfuscation that states that the obfuscation of a circuit C_0 is indistinguishable from the obfuscation of any functionally equivalent circuit C_1 — that is, such that $\mathsf{C}_0(x)=\mathsf{C}_1(x)$ for any x. The corresponding obfuscation procedure is noted iO, and we often use the shorthand $\widehat{\mathsf{C}}$ to denote the indistinguishable obfuscated program C. We also sometimes write that a program obfuscated using indistinguishable obfuscation is iO-obfuscated.

$$\mathsf{iO}(\mathsf{C}_0) \approx_c \mathsf{iO}(\mathsf{C}_1) \text{ for all functionally equivalent } \mathsf{C}_0,\mathsf{C}_1.$$

As, when applicable, compute-and-compare obfuscation preserves the functionality of the program it obfuscates, the definition above tells us that applying indistinguishable obfuscation on a compute-and-compare program obfuscates it at least as well as a compute-and-compare obfuscation would. This corollary turns out to be handy when using coset states to lock a message in a program.

5.4.2 Coset States

We recall below important properties of coset states. We refer the reader to Section 4.2 for a more formal presentation.

Given a subspace A of $\mathbb{F}_2^n$, and two vectors s and s' in $\mathbb{F}_2^n$, the coset state $|A_{s,s'}\rangle$ is the superposition of all vectors of $A+s$ — the regular coset — and applying a Hadamard gate on it results in its dual coset state $|A^\perp_{s',s}\rangle$, a superposition of all vectors in $A^\perp + s'$ — the dual coset.

$$|A_{s,s'}\rangle = \frac{1}{\sqrt{|A|}} \sum_{a\in A} (-1)^{a\cdot s'} |a+s\rangle$$

These states have been introduces by Coladangelo, Liu, Liu, and Zhandry (2021), and satisfy two main properties[2]:

- *(direct product hardness)* no efficient adversary can extract a vector in the regular coset, and one in the dual coset from a coset state, even given access to an obfuscated membership program to these regular and dual cosets.
- *(monogamy-of-entanglement)* there is no way of splitting a coset state in two states such that an efficient adversary can extract a vector in the regular coset given the first state and a description of the subspace, and another efficient adversary can extract a vector in the dual coset given the second state and a description of the subspace. Similarly to direct product hardness, this task remains hard even when the adversaries are given obfuscated membership programs.

Membership programs. The aforementioned membership programs are defined as follows. Each coset state description (A, s, s') is associated to two membership programs P_{A+s} and $\mathsf{P}_{A^\perp+s'}$ for the regular and dual cosets respectively. A membership program is parametrized by a coset description and, on input a vector, returns 1 if the vector belongs to the coset, and 0 otherwise. Coladangelo, Liu, Liu, and Zhandry (2021) showed that using indistinguishable obfuscation on these programs, and giving them to the adversaries in the direct product hardness, and monogamy-of-entanglement properties, is enough to preserve the hardness of the underlying tasks of these properties.

$$\mathsf{P}_{A+s}(u) = \begin{cases} 1 & \text{if } u \in A+s \\ 0 & \text{otherwise.} \end{cases} \qquad \mathsf{P}_{A^\perp+s'}(u) = \begin{cases} 1 & \text{if } u \in A^\perp+s' \\ 0 & \text{otherwise.} \end{cases}$$

Canonical representation. It will be useful in the rest of this section to have a canonical way to represent a coset. As canonical representative of a coset $A+s$, we use the lexicographically smallest vector in $A+s$. We define Can_A as the function, parametrized by a subspace A, that on input a vector u, returns the canonical representative of the coset $A+u$. Importantly, for any coset $A+s$ and vector u, $\mathsf{Can}_A(u) = \mathsf{Can}_A(s)$ if and only if u belongs to $A+s$. This gives us an explicit way of checking membership in a coset $A+s$: on input a vector u, check whether $\mathsf{Can}_A(u) = \mathsf{Can}_A(s)$ and return 1 or 0 accordingly. [3] As mentioned above, the adversaries in the direct product hardness and monogamy-of-entanglement can be given an iO-obfuscated program that checks membership in the regular and dual cosets, without changing the hardness of the task.

[2]The monogamy-of-entanglement property has been conjectured by Coladangelo, Liu, Liu, and Zhandry (2021), and proven later by Culf and Vidick (2022).

[3]Remark that given a description of A, the function Can_A can be implemented efficiently.

5.4.3 Locking A Message with Coset States

We are now ready to present how to lock a message in a program with coset states. Consider a program $\mathsf{Q}_{m,r}$ with a pair of membership programs — corresponding to a random coset state — hardwired. The program has also a random bit r, and the message m to be locked, hardwired. On input a vector u, if $r = 0$, the program checks whether u is in the regular coset using the corresponding membership program. If $r = 1$, the program does the check with the dual coset. In any case, of the check passes, the program returns the message, otherwise it returns an error symbol.

A simple first idea. The idea is then to publish an iO-obfuscation of this program $\widehat{\mathsf{Q}}_{m,r}$, and its random coin r. It is then easy to see that, given this program and coin, and the corresponding coset state, one can recover the message with probability 1. To do that, apply a Hadamard gate to the coset state if $r = 1$, otherwise leave the state unchanged. It remains to run the program in superposition over the resulting state to get the message. Indeed, when $r = 0$, the state is a superposition of vectors in the regular coset, which all pass the test. When $r = 1$, the application of the Hadamard gate results in a superposition over vectors in a dual coset, which also all pass the test. On the other hand, given only the program and its coin, it is hard to extract the message, as we will see later on. We are actually more interested in a situation in between these two cases, where some party Bob is given a program $\widehat{\mathsf{Q}}_{m,r}$, its coin r, and some quantum information aux on the coset state description. Remark that the program $\mathsf{Q}_{m,r}$ is in fact functionally equivalent to the compute-and-compare program $\mathsf{CC}[\mathsf{Can}_A, \mathsf{Can}_A(s), m]$ if $r = 0$, or to $\mathsf{CC}[\mathsf{Can}_{A^\perp}, \mathsf{Can}_{A^\perp}(s), m]$ if $r = 1$. We then know that if compute-and-compare obfuscation is applicable, Bob does not have more chance to recover m from $\mathsf{Q}_{m,r}$ than he has to recover it from a simulated program that holds no information on m. That is, he is not able to recover m. Taking the contrapositive, it means that if Bob is able to recover the message, then there exists an efficient algorithm that extracts the lock-value $\mathsf{Can}_A(s)$ (or $\mathsf{Can}_{A^\perp}(s')$) from the function Can_A and aux. As Can_A is defined from the subspace's description A only, it means the algorithm extracts the lock-value from A and aux. Remark that this automatically proves that if Bob has no information on the coset apart from the program, he cannot recover m, as it would mean he could guess $\mathsf{Can}_A(s)$ or $\mathsf{Can}_{A^\perp}(s')$ from Can_A only, while Can_A holds no information on s or s'.

Using more coset states. In summary, we have shown a way to lock a message in a program with coset states. As the reader might think so far, this is a lot of trouble to only construct a simple encryption scheme. Actually, this method turns out to be way more interesting when considering two separated parties, both holding such a (program, coin) pair, and some (possibly entangled) auxiliary information on the coset states. We will see in the next subsection how Coladangelo, Liu, Liu, and Zhandry (2021) leveraged it to construct a single-decryptor. As it turns out to be useful later on, let us first extend the program $\mathsf{Q}_{m,r}$, by using κ pairs of membership programs — each corresponding to a random independent coset state — and κ coins. The program, on input κ vectors, performs κ membership tests similar to the one described above, and returns the message only if they all pass. Using the same arguments as above, we can argue that if an efficient adversary Bob who, given some auxiliary information aux, guesses a message locked in such a program, then there exists an efficient algorithm that extracts the lock-value of this extended program with non-negligible probability, from the description of the subspaces

$A_1, \ldots, A_\kappa$, and the auxiliary information aux. That is, a set of κ vectors $\ell_1, \ldots, \ell_\kappa$, where $\ell_i = \mathsf{Can}_{A_i}(s_i)$ if $r_i = 0$, or $\ell_i = \mathsf{Can}_{A_i^\perp}(s'_i)$ if $r_i = 1$.

Program 1: Program $\mathsf{Q}_{m,r}$

Hardwired: Membership programs $(\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i})$ for all $i \in [\![1, \kappa]\!]$; message m; random coins $r \in \{0,1\}^\kappa$.
Input: κ vectors $u_1, \ldots, u_\kappa$ in $\mathbb{F}_2^n$.

For $i \in \{1, \ldots, \kappa\}$:

- If $r_i = 0$ and $\widehat{\mathsf{P}}_{A_i+s_i}(u_i) = 0$: return $\perp$.
- Else if $r_i = 1$ and $\widehat{\mathsf{P}}_{A_i^\perp+s'_i}(u_i) = 0$: return $\perp$.
- Else: continue.

Return m.

5.4.4 Single-Decryptor

In this subsection, we present the single-decryptor construction of Coladangelo, Liu, Liu, and Zhandry (2021). We first define single-decryptors in a high-level, and refer the reader to Section 4.6.4 for more detailed information on this primitive.

Single-decryptors. A (public) single-decryptor scheme can be seen as a classical encryption scheme, with quantum decryption keys. As in a classical (public) encryption scheme, a single-decryptor has a key generation procedure KeyGen sampling pairs of classical private and public keys. Here however, the private key here is not directly used for decryption, but rather allows for generating quantum decryption keys, used as input in the decryption algorithm Dec. The public keys serve for encrypting messages, analogously as in a classical scheme. The security of such a scheme states that the quantum secret keys must be unclonable in the sense that no triple of efficient adversaries, Alice, Bob, and Charlie, can win the following game with probability greater than some trivial probability that we mention in the next paragraph. Alice first receives a random public key from the challenger, and then is asked to send them back two messages. The challenger then sends Alice a quantum decryption key, that she splits and shares between Bob and Charlie. The latter are then given a challenge consisting in an encryption of one of the two messages sent by Alice, and they must both guess which message they have received in order to win the game.

Real-or-random anti-piracy security. In the original anti-piracy game, defined by Coladangelo, Liu, Liu, and Zhandry (2021), the encrypted messages given to Bob and Charlie are chosen independently, and encrypted with independent random coins. In this work, we consider a slightly different game. Alice only sends one message m to the challenger, and the challenges given to Bob and Charlie are either the encryption of m, or the encryption of a random message. Furthermore, we parametrize this *real-or-random anti-piracy* game by three distributions (described in Figure 5.6), similarly to copy-protection. With the product distribution, the encrypted messages given to Bob and Charlie are chosen independently, as well as the coins; with the identical distribution,

$\mathcal{D}^{prod}_{(m,r)}$	$\mathcal{D}^{id}_{(m,r)}$	$\mathcal{D}^{nc}_{(m,r)}$
$((m,r),(m,r))$	$((m,r),(m,r))$	
$((m,r),(m',r'))$		$((m,r),(m',r'))$
$((m',r'),(m,r))$		$((m',r'),(m,r))$
$((m',r'),(m'',r''))$	$((m',r'),(m',r'))$	$((m',r'),(m'',r''))$

Figure 5.6: Product, identical, and non-colliding challenge distributions used in real-or-random anti-piracy for single-decryptors, parametrized by a message $m \in \mathcal{M}$, and a set of random coins $r \in \{0,1\}^{\mathcal{R}}$. Each column represents one challenge distribution. Sampling from one of these distributions consists in sampling (m',r') and (m'',r'') uniformly and independently at random from $\mathcal{M} \times \{0,1\}^{\mathcal{R}}$, then picking one non-empty cell at random in the column corresponding to the chosen distribution, and finally yielding the cell.

Bob and Charlie receive the exact same challenge; and with the non-colliding distribution, Bob and Charlie cannot both receive the encryption of m. When parametrized with the non-colliding distribution, the adversaries must not win the game with probability greater than $2/3$, while it is still $1/2$ with the two other distributions. We provide a formal definition of this property in Appendix A.2, and an illustration of the real-or-random anti-piracy game in Figure 5.7.

5.4.5 Construction

We present below the single-decryptor construction of Coladangelo, Liu, Liu, and Zhandry (2021). The authors of this paper showed that it satisfies the original anti-piracy security, and we prove its real-or-random anti-piracy security with respect to the non-colliding distribution below. Proving this property turns out to be much more difficult when considering identical and product distributions, and we discuss it deeper in the end of this section.

Construction 16: Single-Decryptor Scheme [CLLZ21]

Let $n, \kappa = \mathsf{poly}(\lambda)$.

- $\mathsf{KeyGen}(1^\lambda)$:
  - Sample coset states' descriptions $\{A_i, s_i, s'_i\}_{i\in[\![1,\kappa]\!]}$ where each A_i is of dimension $n/2$.
  - Generate the iO-obfuscated membership programs for each coset $\{\widehat{\mathsf{P}}_{A_i+si}, \widehat{\mathsf{P}}_{A_i^\perp+s'i}\}_{i\in[\![1,\kappa]\!]}$.
  - Return $\{A_i, s_i, s'_i\}_{i\in[\![1,\kappa]\!]}$ as the secret key, and $\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'i}\}_{i\in[\![1,\kappa]\!]}$ as the public key.
- $\mathsf{QKeyGen}(\{A_i, s_i, s'_i\}_{i\in[\![1,\kappa]\!]})$:
  - Prepare $|\mathfrak{k}\rangle = \bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle$.
  - Return $|\mathfrak{k}\rangle$.

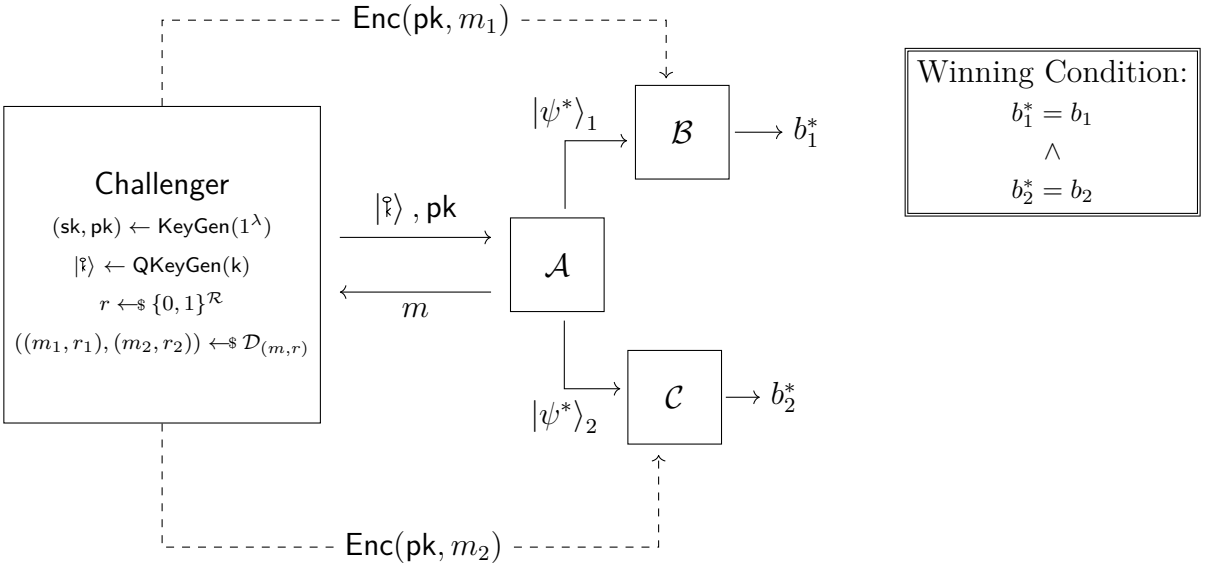

Figure 5.7: Real-or-random anti-piracy security of a single-decryptor, with respect to a challenge distribution $\mathcal{D}$. $b_1 = 0$ if $m_1 = m$ and 1 otherwise, and b_2 is defined analogously with m_2. This property states that no triple of efficient adversaries must win this game with probability significantly greater than $1/2$ if $\mathcal{D}$ is the product or identical distribution, or $2/3$ if it is the non-colliding distribution.

- $\mathsf{Enc}(\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'i}\}_{i\in[\![1,\kappa]\!]}, m)$:
  - Sample $r \leftarrow\$ \{0,1\}^{\mathcal{R}}$.
  - Generate an iO-obfuscated program $\widehat{\mathsf{Q}}_{m,r}$ of program $\mathsf{Q}_{m,r}$ described in program 1.
  - Return $\left(r, \widehat{\mathsf{Q}}_{m,r}\right)$.
- $\mathsf{Dec}\left(\bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle\ , \left(r, \widehat{\mathsf{Q}}_{m,r}\right)\right)$:
  - For all $i \in [\![1,\kappa]\!]$: if $r_i = 1$, apply $\mathsf{H}^{\otimes n}$ to $|A_{i,s_i,s'_i}\rangle$.
  - Let $|\mathfrak{k}'\rangle$ be the resulting state, run $\widehat{\mathsf{Q}}_{m,r}$ coherently on $|\mathfrak{k}'\rangle$. Let m denotes the outcome.
  - Uncompute the Hadamard gates above.
  - Return m.

The correctness of this scheme follows from the same argument as in the previous subsection.

Real-or-random anti-piracy security. The main idea to prove real-or-random anti-piracy security is based on the existence of the extractor we also discussed in the previous subsection. Assume that a triple of adversaries Alice, Bob, and Charlie, win the real-or-random anti-piracy game with probability greater than the trivial probability. The ciphertexts given to Bob and Charlie are both composed of a compute-and-compare

program, and they both receive a quantum state from Alice, that we can consider as auxiliary information on the coset states used in the game. A first thing to note is that, as compute-and-compare programs are the only objects containing information on the message, replacing them by simulated programs of the CC-obfuscation — containing no information on the message — prevents Bob and Charlie to guess their messages. It gives Bob and Charlie a way to distinguish between a real compute-and-compare program, and such a simulated program, meaning that they both have access to an efficient extractor for the lock-value of their respective programs. When the game is with respect to the non-colliding distribution, then the random coins r_1 and r_2 used for the encryption of Bob and Charlie's challenges are independent. It means that with overwhelming probability, there will be an index i such that $r_{1,i} = 0$, and $r_{2,i} = 1$. Remember that the lock-values returned by the extractors are the canonical representatives of the cosets. Then, the i-th lock-value extracted on Bob's side is the canonical representative of $A_i + s_i$, and the one on Charlie's side is the representative of $A_i^{\perp} + s'_i$. It means that Bob and Charlie have access to vectors in the regular i-th coset, and in the dual i-th coset respectively. Returning these vectors allow them to break the monogamy-of-entanglement of coset states, task proven to be impossible, thus proving the real-or-random anti-piracy security of this construction.

Theorem 15 (Correctness of Construction 16)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions. Then the single-decryptor scheme defined in construction 16 has correctness.

Theorem 16 (Real-Or-Random Anti-Piracy Security of Construction 16, with Respect to the Non-Colliding Distribution)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions. Then the single-decryptor scheme defined in construction 16 has anti-piracy security, with respect to the non-colliding distribution.

5.4.6 On the Need for a New Monogamy-Of-Entanglement Property

Recall that the security of the single-decryptor of construction 16 is based on the following argument. If Bob is able to guess correctly the encryption of which message he has received, then he can extract the lock-value of the encryption: a sequence of canonical representatives of cosets. Crucially, each vector in this sequence is either the representative of the corresponding regular coset, or of the dual one, and the one Bob received is decided by the challenge encryption. As we use the same argument for Charlie, when Bob and Charlie receive independent challenges — as this is the case when considering the non-colliding distribution — then it is almost certain that for some index, Bob receives a representative of a regular coset, and Charlie the representative of the corresponding dual coset, which allow them to win the monogamy-of-entanglement game for coset states.

Identical and product distributions. The same argument unfortunately does not work with identical and product distributions. With identical distribution indeed, the challenges are the same for Bob and Charlie, then they extract the same lock-values, and cannot break the monogamy-of-entanglement. The same problem also happens with product distribution, although Bob and Charlie receive the same challenge only with

probability 1/4. But we cannot rule out that the adversaries in the real-or-random anti-piracy game could have an advantage over the trivial probability only in this case, which prevents us to proceed with the reduction.

Monogamy-of-entanglement with identical basis. To overcome this first issue, we introduce a new monogamy-of-entanglement property, that we name monogamy-of-entanglement with identical basis. In the corresponding game, Bob and Charlie are asked to return a vector *belonging to the same coset* (regular or dual). What prevents adversaries to win the game with probability 1 is that Alice does not know in which coset the vectors returned by Bob and Charlie must belong: this information is revealed to Bob and Charlie only during the challenge phase. We prove that the best strategy for a triple of adversaries in this game is for Alice to make a random guess on this challenge coset, and measure the coset state accordingly. Such a strategy succeeds with probability negligibly close to 1/2. We also prove that, when parallelizing this game — with a polynomial number of coset states and an independent challenge coset picked for each one of them — the winning probability of any triple of adversaries drops to negligible. We present this new game in more details in Section 5.6.

5.4.7 Issues with Simultaneous Extraction

There is actually a more fundamental problem when proving the security of the single-decryptor (with respect to any challenge distribution) that we did not mention so far for sake of conciseness. As we mentioned above, the crux of the proof relies on the fact that, if Bob can guess which message has been encrypted given his "fake quantum key" (one half of the bipartite state generated by Alice), then there exists an efficient lock-value extractor *that uses the fake quantum key*. The reduction then proceeds as follows. Bob uses his extractor to obtain a lock-value, then Charlie uses his extractor to obtain a possibly different lock-value, and they both return an element of their lock-value as a candidate vector in the monogamy-of-entanglement game. The problem is that the two fake keys produced by Alice can be entangled and, as the extraction on Bob's side is in fact a measurement, this measurement might perturb Charlie's fake key, in a way that prevents it to be used to make a correct guess on Charlie's challenge. In this case, we can no longer ensures the existence of a lock-value extractor on Charlie's side, and the reduction cannot carry on.

Threshold implementation. Coladangelo, Liu, Liu, and Zhandry (2021), based on works of Zhandry (2020) and Aaronson, Liu, Liu, Zhandry, and Zhang (2021), overcome this problem using so-called "threshold implementations". In a nutshell, they represent the challenge phase as two tests, respectively performed on Bob's and Charlie's fake keys, each one being a mixture of projective measurements. Each projective measurement, parametrized by a message m and random coins r, first determines the probability that Bob (resp. Charlie) makes a correct guess when given the encryption of m, performed with random coins r, and returns 1 if this probability is greater than the trivial probability for the distribution we consider. With such a representation, they show that the bipartite state shared by Bob and Charlie collapses, conditioned on both test succeeding, to a state where both registers are in superposition over "successful fake keys", meaning that they will keep succeeding in the test no matter how many times we perform it. Then, applying the extractor on Bob's side might perturb the state on Charlie's side, but it

still remains a superposition over successful fake keys. Thus, Charlie can still make a correct guess, which ensures the existence of a lock-value extractor using his state, and we can finish the reduction. By applying this technique, we show that the single-decryptor of construction 16 has real-or-random anti-piracy security with respect to non-colliding distribution.

Identical and product distributions. This technique can however not be applied when considering identical and product distributions. Intuitively, the technique works in the non-colliding distribution because the message and coins (m, r) used for constructing Bob's and Charlie's challenges are independent in every case. On the other hand, they can be the same in the identical and product distributions.[4] In this case, extracting from Bob's register might collapse Charlie's state to a superposition of successful fake keys with respect to a single set of coins r only. As we need Charlie to be able to make a correct guess on average on every possible set of coins, the argument no longer follows.

Simultaneous compute-and-compare obfuscation. We have not been able to solve this issue so far, and copy-protection of point functions with respect to identical and product distributions, and unclonable encryption in the plain model, remain open problems. Nevertheless, we show that this issue can be reduced to how compute-and-compare obfuscation can be used in a non-local context. Recall that compute-and-compare obfuscation states that, loosely speaking, if the lock-value of a compute-and-compare program $\mathsf{CC}[f, \ell, s]$ cannot be extracted from its function, and some (quantum) auxiliary information, then there is a way to obfuscate this program such that it is indistinguishable from a simulated dummy program that returns $\perp$ on every input, even given the corresponding auxiliary information. We conjecture (conjectures 1 and 2) that this is still true when considering two compute-and-compare programs instead of one, and a bipartite auxiliary information. Namely, if their respective lock-values cannot be simultaneously extracted by two measurements acting on different registers of the auxiliary information state, then there is no pair of measurement, each given a different register of the auxiliary information state, that can simultaneously distinguish these programs from a dummy program as above. We elaborate on this question in Section 5.7, and we will see that it crucially depends on whether the randomness used for the compute-and-compare obfuscations if independent or not.

Theorem 17 (Real-Or-Random Anti-Piracy Security of Construction 16, with Respect to the Identical and Product Distributions)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions. Assume conjectures 1 and 2. Then the single-decryptor scheme defined in construction 16 has anti-piracy security, with respect to the identical and product distributions.

5.5 A Copy-Protection Scheme of Pseudorandom Functions in the Plain Model

Let us now turn to the construction of copy-protection of pseudorandom functions. In this section, we present the construction of Coladangelo, Liu, Liu, and Zhandry (2021),

[4] They are actually always the same in the identical distribution.

then show that it satisfies reversed anti-piracy security, with respect to any challenge distribution $\mathcal{D}$, provided that the single-decryptor of construction 16 has real-or-random anti-piracy with respect to $\mathcal{D}$. This immediately tells us that it has reversed anti-piracy security, with respect to the non-colliding distribution.

5.5.1 High-Level Description

To protect a pseudorandom function (described by a key k), the authors use the same technique as the one used in the previous section to lock a message with coset states. The idea is to obfuscate a program, with a key k and random coset membership programs hardwired. This program takes as input a pseudorandom function's input x, and vectors $u_1, \ldots, u_\kappa$, and checks whether each vector belongs to the corresponding (regular or dual) coset, similarly to the single-decryptor construction. If this is the case, the program returns $\mathsf{PRF}_\mathsf{k}(x)$, otherwise it returns an error symbol $\perp$. We use the κ first bits of the input x to decide whether to use the membership program for the regular or dual coset. The problem with this idea is that the obfuscated program could leak some information on the key, that Bob and Charlie could then use to answer their challenge. This can be overcome by using the "hidden trigger" technique introduced by Sahai and Waters (2014). Loosely speaking, this technique slightly modifies the program, and allows to argue that the challenges Bob and Charlie get in the game are independent of the key.

Hidden trigger technique. In a nutshell, the hidden trigger technique applied to this construction consists in generating a program R_k, with inputs x and $u_1, \ldots, u_\kappa$ as above, with two modes. In the *normal mode*, the program is equivalent to the one described above: it performs a sequence of membership checks, and returns the pseudorandom function evaluation of the input x if they all pass. In the *hidden trigger mode*, a program Q is extracted from x, and the outcome of R_k is $\mathsf{Q}(u_1, \ldots, u_\kappa)$. Deciding in which mode the program runs is done by testing whether x is a "trigger input" at the beginning of the program. If it is, the program runs in the hidden trigger mode, otherwise in the normal mode. We do not detail how this test, nor the program extraction are implemented in this section, and refer the curious reader to Appendix A.3 for more information. It is still important to understand the security of the construction to describe a few properties that these trigger inputs must have.

- The set of trigger inputs it sparse, which ensures that the program R_k implements almost perfectly the pseudorandom function.[5]
- There is an efficient (randomized) procedure to generate, given a program Q, a trigger input, whose extracted program is Q.
- Let x be a random input, r be a set of coins for the aforementioned trigger input generation procedure, and Q be a program functionally equivalent to the program $\mathsf{Q}_{\mathsf{PRF}_\mathsf{k}(x),r}$ defined in program 1. Then, the trigger input x', generated using the procedure above with randomness r and program Q, is indistinguishable from x, even given access to $\widehat{\mathsf{R}}_\mathsf{k}$, and a set of vectors that pass the membership tests. As a sanity check, remark that the outcome of R_k, when run on x or on x', is the same.

[5] More precisely, the generation of the trigger inputs test is randomized, and for each x, the probability that $\mathsf{R}_\mathsf{k}(x, u_1, \ldots, u_\kappa) = \mathsf{PRF}_\mathsf{k}(x)$ is negligibly close to 1 (provided that the vectors u_i all pass the membership tests).

In the former case, this is because x is almost certainly not a trigger input, hence the program is executed in the normal mode. In the latter, this is because x' is a trigger input, hence the program is executed in the hidden trigger mode and returns the outcome of $\mathsf{Q}_{\mathsf{PRF}_{\mathsf{k}}(x),r}(u_1,\ldots,u_\kappa)$ — where $u_1,\ldots,u_\kappa$ all pass the membership checks — which is exactly $\mathsf{PRF}_{\mathsf{k}}(x)$.

5.5.2 Construction

We present the construction of copy-protection of pseudorandom functions, and prove that we can reduce its reverse indistinguishability anti-piracy security to the real-or-random anti-piracy security of construction 16. As we mentioned above, we do not detail the implementation of the test for trigger inputs, nor the program extraction procedure. We simply denote these procedures as Test and $\mathsf{Extract}$, sampled from a family $\mathcal{D}_{HiddenTrigger}$. This construction protects a family of puncturable and extracting pseudorandom functions $\{\mathsf{PRF}_{\mathsf{k}}\}_{\mathsf{k}\in\mathcal{K}}$, with input space $\{0,1\}^{n_\mathcal{X}}$, and output space $\{0,1\}^{n_\mathcal{Z}}$, where $n_\mathcal{X}, n_\mathcal{Z} = \mathsf{poly}(\lambda)$, and $n_\mathcal{Z}$ is smaller enough than $n_\mathcal{X}$. Let $\kappa = \mathsf{poly}(\lambda)$, $\kappa < n_\mathcal{X}$. Let $n = \lambda$. Recall that for any program P, we use the notation $\widehat{\mathsf{P}} = \mathsf{iO}(\mathsf{P})$.

Construction 17: Copy-Protection of Pseudorandom Functions [CLLZ21]

$\mathsf{Protect}(1^\lambda, \mathsf{k})$:

- Sample κ n-long cosets states' descriptions $\{A_i, s_i, s'_i\}_{i\in[\![1,\kappa]\!]}$. Let $\mathsf{P}_{A_i+s_i}$ and $\mathsf{P}_{A_i^\perp+s'_i}$ be the corresponding membership programs for every i.
- Prepare the corresponding coset states $|A_{1,s_1,s'_1}\rangle \otimes \cdots \otimes |A_{\kappa,s_\kappa,s'_\kappa}\rangle$.
- Sample $(\mathsf{Test}, \mathsf{Extract}) \leftarrow\!\!\$\ \mathcal{D}_{HiddenTrigger}$.
- Generate the program R_{k}, defined in program 2, parametrized by the procedures Test and $\mathsf{Extract}$, and the obfuscated membership programs $\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}$.
- Return $|\text{⚙}_{\mathsf{k}}\rangle = \left(\bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle, \widehat{\mathsf{R}_{\mathsf{k}}}\right)$.

$\mathsf{Eval}\left(\left(\bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle, \widehat{\mathsf{R}_{\mathsf{k}}}\right), x\right)$:

- For $i \in [\![1,\kappa]\!]$:
  - if $x_i = 1$: apply a Hadamard gate H^n on $|A_{i,s_i,s'_i}\rangle$;
  - otherwise leave the state unchanged;
  - in any case, denote the resulting state $|\text{⚙}'_i\rangle$.
- Run the program $\widehat{\mathsf{R}_{\mathsf{k}}}$ coherently on $(x, |\text{⚙}'_1\rangle, \ldots, |\text{⚙}'_\kappa\rangle)$. Let z denotes the outcome.
- Uncompute the Hadamard gates above.
- Return z.

We now describe the program R, used in the encryption procedure.

Program 2: Program R

Hardwired: Trigger input test Test, program extraction procedure $\mathsf{Extract}$, programs $(\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^{\perp}+s'_i})$ for all $i \in [\![1, \kappa]\!]$.
Input: A bitstring $x \in \{0,1\}^{n_x}$, and κ vectors $u_1, \ldots, u_\kappa$ in $\mathbb{F}_2^n$:

$\mathsf{R}(x, u_1, \ldots, u_\kappa)$:

- If $\mathsf{Test}(\mathsf{x}) = 1$: *(Hidden Trigger Mode)*
  - Extract $\mathsf{Q} \leftarrow \mathsf{Extract}(x)$.
  - Return $\mathsf{Q}(u_1, \ldots, u_\kappa)$.

  Else: *(Normal Mode)*
  - For $i \in \{1, \ldots, \kappa\}$:
    - If $r_i = 0$ and $\widehat{\mathsf{P}}_{A_i+s_i}(u_i) = 0$: return $\perp$.
    - Else if $r_i = 1$ and $\widehat{\mathsf{P}}_{A_i^{\perp}+s'_i}(u_i) = 0$: return $\perp$.
    - Else: continue.
  - Return $\mathsf{PRF}_{1,\mathsf{k}_1}(x)$.

Correctness. As we mentioned above, the probability for x to be a trigger input is negligible for every input x. Then, the program R_k is almost always run in normal mode, and the correctness follows from the same argument as for construction 16.

Theorem 18 (Correctness of Construction 17)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Then the copy-protection scheme defined in construction 17 has correctness.

Reversed anti-piracy security. Proving that this construction has reversed anti-piracy security with respect to some challenge distribution $\mathcal{D}$ amounts to proving that the following game cannot be won with probability greater than the corresponding trivial winning probability. Alice is given a protected pseudorandom function, and an image $z = \mathsf{PRF}_\mathsf{k}(x)$, splits it and shares it between Bob and Charlie. The latter are then given challenges x_1 and x_2 respectively and must tell whether their challenge is x or not. As mentioned above, the hidden trigger technique allows us to replace x_1 by a trigger input whose extracted program is $\mathsf{Q}_{z,r}$ when $x_1 = x$, and by another trigger input whose extracted program is $\mathsf{Q}_{z',r}$ — where z' is a random image — when $x_1 \neq x$. We can do the same with x_2. Now, because the pseudorandom functions we consider are extracting, we can replace the image Alice receives by a random bitstring z, and change the challenges x_1 and x_2 accordingly. The resulting hybrid game is equivalent to the original one, from the adversaries' point of view. Crucially, in this last hybrid game, the challenges are completely independent of the pseudorandom function's key. Also, they can be constructed from a challenge in the real-or-random anti-piracy game of the single-decryptor scheme of construction 16, which allows us to do the reduction.

Reduction to real-or-random anti-piracy. Assume that a triple of adversaries Alice, Bob, and Charlie, wins the game above (with trigger inputs) with probability p greater

than the trivial winning probability. We construct a triple of adversaries Alex, Billy, and Clover, for the real-or-random anti-piracy game of the single-decryptor scheme of construction 16. In this game, Alex receives κ coset states — the quantum decryption key — and obfuscated membership programs for all of them — the public key. She samples a random message m from $\{0,1\}^{nz}$ and sends it to the challenger. She samples $(\mathsf{Test}, \mathsf{Extract}) \leftarrow\!\!\$\ \mathcal{D}_{HiddenTrigger}$, and uses these procedures, along with the membership programs, to generate the program R. Then she simulates Alice on the coset states, the message m, and the iO-obfuscation of R to receive a bipartite quantum state, that she shares between Billy and Clover, along with $(\mathsf{Test}, \mathsf{Extract})$. Billy receives a challenge $(r, \widehat{\mathsf{Q}}_{m_1,r})$: the encryption of m, or of another random message. He generates a trigger input whose extracted program is $\widehat{\mathsf{Q}}_{m_1,r}$. Note that a trigger input generated in this way follows the same distribution as the challenge received by Bob in the last hybrid game. Clover does the same with her challenge, and they respectively simulate Bob and Charlie on their respective trigger inputs to receive b_1^* and b_2^*, which they output. As the elements given to Alice, Bob, and Charlie, follow the same distribution as the ones they receive in the reversed anti-piracy game, Alex, Billy, and Clover win the real-or-random game with the same probability p, which finishes the reduction. We provide a formal proof in Appendix A.3.

Theorem 19 (Reversed Anti-Piracy Security of Construction 17)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Assume that the single-decryptor of construction 16 has correctness and real-or-random anti-piracy security with respect to a challenge distribution $\mathcal{D}$. Then the copy-protection scheme defined in construction 17 has reversed anti-piracy security, with respect to $\mathcal{D}$.

Given that we prove (Theorems 15 and 16) that this single-decryptor has real-or-random anti-piracy security with respect to non-colliding distribution, it gives us a copy-protection scheme of pseudorandom functions with reversed anti-piracy security with respect to the non-colliding distribution. It then yields, leveraging Theorem 12 a copy-protection scheme of point functions with anti-piracy security, with respect to the non-colliding distribution.

Corollary 1 (Reversed Anti-Piracy Security of Construction 17 with Respect to the Non-Colliding Distribution)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Then the copy-protection scheme defined in construction 17 has reversed anti-piracy security, with respect to the non-colliding distribution.

Corollary 2 (Anti-Piracy Security of Construction 14 with Respect to the Non-Colliding Distribution)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Then the copy-protection scheme defined in construction 14 has anti-piracy security, with respect to the non-colliding distribution.

Finally, our conjectures on simultaneous compute-and-compare obfuscation (conjectures 1 and 2) give us anti-piracy for these constructions with respect to the identical and product distributions, and unclonable encryption. We describe these conjectures in Section 5.7.

Corollary 3 (Reversed Anti-Piracy Security of Construction 17 with Respect to the Identical and Product Distributions)**.** Assume the existence of post-quantum indistinguishability

obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Assume conjectures 1 and 2. Then the copy-protection scheme defined in construction 17 has reversed anti-piracy security, with respect to the identical and product distributions.

Corollary 4 (Anti-Piracy Security of Construction 14 with Respect to the Identical and Product Distributions)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Assume conjectures 1 and 2. Then the copy-protection scheme defined in construction 14 has anti-piracy security, with respect to the identical and product distributions.

Corollary 5 (Unclonable-Indistinguishability of Construction 15)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions. Assume conjecture 1. Then the unclonable encryption scheme defined in construction 15 has unclonable-indistinguishability.

5.6 Monogamy-Of-Entanglement Game with Identical Basis

As mentioned in Section 5.4.6, we need a new monogamy-of-entanglement property for coset states in order to prove the real-or-random anti-piracy security of construction 16 in the identical and product distributions. We present in this section such a new monogamy-of-entanglement property. In this game, Alice is given a random coset state, splits it, and share it between Bob and Charlie. As in the original game, Bob and Charlie are then given the description of the subspace, but contrarily to it, they are not asked to return a vector in different cosets, but in the same one: either the regular coset or the dual one. If they can freely choose this coset, the task is easy, as Alice can simply measure the state in a random basis (computational or Hadamard) and sends the outcome to Bob and Charlie. We wonder whether it is still an easy task when the choice is not free: Bob and Charlie are instructed in which coset (regular or dual) the vector they return must belong to, and Alice does not know it in advance. If Alice applies the strategy mentioned above, the adversaries win the game with probability $1/2$. In this section, we prove that she cannot actually do better than this, meaning that the winning probability of any triple of adversaries in this game is upper-bounded by $1/2$. We formalize this new property below, and provide an illustration of this game in Figure 5.8.

Theorem 20 (Monogamy-Of-Entanglement With Identical Basis For Coset States (Statistical Version))**.** Define the following game, between a challenger and a triple of adversaries $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and parametrized by a security parameter λ. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate. Let $n = \lambda$.

- **Setup phase:**
  - The challenger samples an n-long coset state's description (A, s, s').
  - The challenger sends $|A_{s,s'}\rangle$ to $\mathcal{A}$.
- **Splitting phase:**

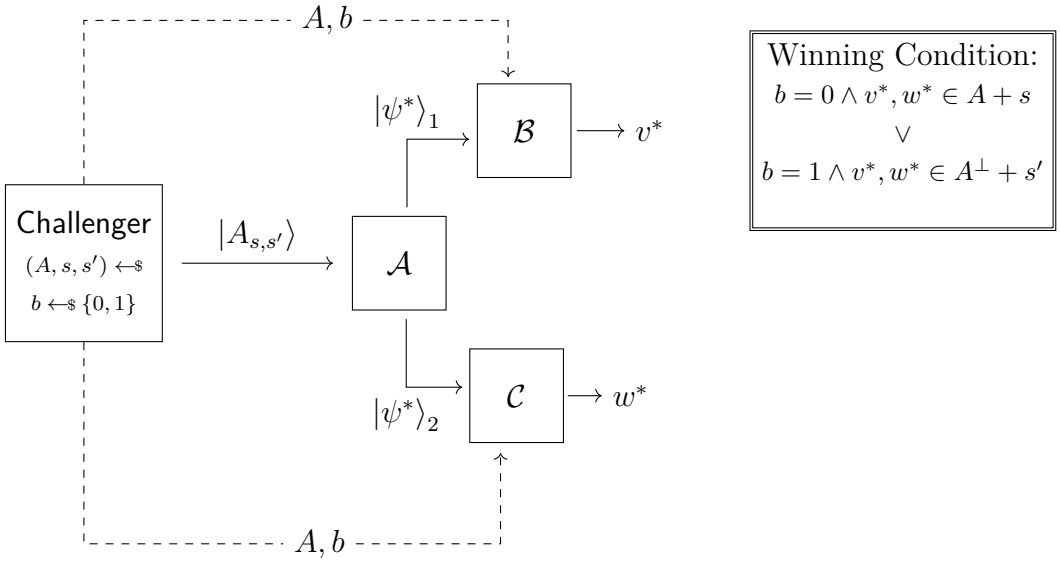

Figure 5.8: Monogamy-of-entanglement game with identical basis for coset states. (A, s, s') is a random n-long coset state's description. No triple of adversaries wins this game with probability non-negligibly greater than $1/2$.

  - $\mathcal{A}$ prepares a bipartite quantum state $|\psi^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\psi^*\rangle_1$ to $\mathcal{B}$ and $|\psi^*\rangle_2$ to $\mathcal{C}$.

- **Challenge phase:**
  - The challenger samples a bit $b \leftarrow\!\$\ \{0,1\}$.
  - The challenger sends A and b to both $\mathcal{B}$ and $\mathcal{C}$.

Let v^* denotes the output of $\mathcal{B}$, and w^* denotes the output of $\mathcal{C}$. They win if $b = 0$ and $v^*, w^* \in A + s$, or if $b = 1$ and $v^*, w^* \in A^\perp + s'$.

The monogamy-of-entanglement with identical basis property states that no triple of adversaries can win this game with non-negligible advantage over $1/2$. In other words, for any triple of adversaries $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$,

$$\Pr\left[\begin{array}{c|l} & v^* \leftarrow \mathcal{B}(|\psi^*\rangle_1, A, b) \\ b = 0 \wedge v^*, w^* \in A + s & w^* \leftarrow \mathcal{C}(|\psi^*\rangle_2, A, b) \\ \vee & |\psi^*\rangle_{12} \leftarrow \mathcal{A}(|A_{s,s'}\rangle) \\ b = 1 \wedge v^*, w^* \in A^\perp + s' & s, s' \leftarrow\!\$\ \mathbb{F}_2^n \\ & A \leftarrow\!\$\ \{A \in \mathbb{F}_2^{n \times n/2} \ : \ A \text{ is full-rank}\} \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

For our single-decryptor security purpose, we actually need to consider a parallel version of this game. That is, Alice is given κ coset states, and Bob and Charlie are instructed on the bases in which to return their κ vectors depending on κ random coins. We prove that the winning probability in this case drops exponentially with κ. Furthermore, as with the other coset states games we mentioned in this thesis, these upper-bounds are still true when the adversaries are computationally bounded and given obfuscated membership programs for the regular and dual cosets.

In the following, we provide a high-level proof of our monogamy-of-entanglement with identical basis game. A more detailed proof can be found in Appendix A.4.

5.6.1 Proof of Upper-Bound

We will actually study a "BB84 version" of this game. Using arguments from Culf and Vidick (2022), we can prove that winning the coset states version reduces to winning this BB84 version.

In this version, Alice receives a random n-long BB84 state $|x^\theta\rangle$, is asked to split it and share it between Bob and Charlie, who then both receive the basis θ, and a bit b, indicating whether they need to return $x_{|I}$ (if $b = 0$) or $x_{|\bar{I}}$ (if $b = 1$).[6] As with the coset states version, we want to prove that the winning probability of this game is upper-bounded by $1/2 + \mathsf{negl}(n)$.

Extended non-local games. Our proof follows the structure of the one of Tomamichel, Fehr, Kaniewski, and Wehner (2013). We first interpret this game as the following "extended non-local game". Bob and Charlie prepare a tripartite quantum state; we ask the first register to be n-qubits long. They send the first register to the challenger, Bob keeps the second, and Charlie the third. From this step, Bob and Charlie cannot communicate. The challenger measures the first register in a basis θ sampled at random; let x denotes the outcome. Then, the challenger sends θ, and a random bit b to both Bob and Charlie, who have to return $x_{|I_b}$, as above.

Given a triple of adversaries, Alice, Bob, and Charlie, winning the monogamy-of-entanglement game (BB84 version) — for n-qubits long BB84 state — with probability p, we can easily construct a pair of adversaries Billy, and Clover, winning the extended non-local game with the same probability p. Billy and Clover first prepare n EPR states $|\phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. Then they simulate Alice on the second halves of all these states to get a bipartite state $|\psi^*\rangle_{12}$ that they share between each other, and they send the first halves of the EPR states to the challenger. Then, when receiving the challenge (θ, b), Billy simulates Bob on $|\psi^*\rangle_1$ and the challenge, and returns the outcome. Clover does the same with Charlie and $|\psi^*\rangle_2$. The fact that the winning probability of this game is p follows from the following observation. The quantum state shared by the challenger, Bob, and Charlie before the challenger's measurement is

$$\sum_{r\in\{0,1\}^n} |r\rangle_0 \mathcal{A}(|r\rangle)_{12}$$

where the registers 0, 1, and 2, respectively denote the challenger's, Bob's, and Charlie's registers, and $\mathcal{A}$ is the algorithm representing Alice. After the challenger's measurement, by denoting x the outcome, the state is now

$$\begin{aligned}&\sum_{r\in\{0,1\}^n} |x^\theta\rangle\langle x^\theta| \, |r\rangle_0 \mathcal{A}(|r\rangle)_{12} \\ &= |x^\theta\rangle_0 \mathcal{A}(|x^\theta\rangle)_{12}\end{aligned}$$

Alice's (or $\mathcal{A}$'s) input distribution is then the same as the one in the monogamy-of-entanglement game, meaning that Bob's and Charlie's outcome will be both correct with probability p.

[6] We define $I = \{i \in [\![1, n]\!] \ : \ \theta_i = 0\}$. Recall that $x_{|I}$ is a bitstring whose components are x_i for all $i \in I$, and 0 everywhere else, and $x_{|\bar{I}}$ is defined analogously. In the following, we write $I_0 = I$, and $I_1 = \bar{I}$.

A first upper-bound. Our goal is now to upper-bound the winning probability of the extended non-local game. Note first that, using Naimark's dilatation theorem, we can assume without loss of generality that any pair of adversaries, Bob and Charlie, can be represented as two families of projective measurements $\{B^{\theta,b}\}_{\theta,b}$ (for Bob), and $\{C^{\theta,b}\}_{\theta,b}$ (for Charlie). With these notations, we can express the winning probability of the game as the quantity

$$\Pi_{\theta,b} = \sum_{x\in\{0,1\}^n} |x^\theta\rangle\langle x^\theta| \otimes B^{\theta,b}_{x_{|I_b}} \otimes C^{\theta,b}_{x_{|I_b}}$$

averaged over θ such that $|\theta| = n/2$, and $b \in \{0,1\}$.

Upper-bounding this expected value cannot be done simply using triangle inequality, as the bound would not be tight enough for our purpose. We rather use a lemma introduced by Tomamichel, Fehr, Kaniewski, and Wehner (2013) (Lemma 4), that allows us to state that the winning probability is upper-bounded by the quantity

$$\max_{\theta,b} \|\Pi_{\theta,b}\Pi_{\pi_{k,\alpha}(\theta,b)}\|$$

averaged over a family of orthogonal permutations $\{\pi_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ — where we denote by N the number of bases θ such that $|\theta| = n/2$.[7] We will prove later on the existence of a family of orthogonal permutations that satisfies the requirements we need.

Upper-bounding $\|\Pi_{\theta,b}\Pi_{\theta',1-b}\|$. Note that the quantity $\|\Pi_{\theta,b}\Pi_{\theta',1-b}\|$ is trivially upper-bounded by 1. Using this upper-bound for half of the permutations — say the $\{\pi_{k,0}\}_k$ — yields the following upper-bound on the winning probability:

$$\frac{1}{2} + \frac{1}{2N} \sum_{k\in[\![1,N]\!]} \max_{\theta,b} \|\Pi_{\theta,b}\Pi_{\pi_{k,1}(\theta,b)}\|$$

We then simply need to show that the second member of this equation is negligible.

A careful analysis (which we leave in Appendix A.4) shows that, when $b' = 1 - b$, the quantity $\|\Pi_{\theta,b}\Pi_{\theta',b'}\|$ depends on the number of indices $d(\theta,\theta')$ on which θ and θ' differ. More precisely, $\|\Pi_{\theta,b}\Pi_{\theta',b'}\|$ is upper-bounded by $2^{-d(\theta,\theta')/4}$. Thus, we want a family of permutations such that, for all k, the last bit of $\pi_{k,1}(\theta,b)$ is $1-b$ and $d(\theta,\theta')$ is large enough.

Finding the family of permutations. We build upon a result of Culf and Vidick (2022) to construct a family of permutations with the aforementioned properties. More concretely, Culf and Vidick define a mutually orthogonal family of permutations π_k, indexed by $1 \geq k \geq N$, with the latter property. We then define another family $\widetilde{\pi}_{k,b}$, indexed by $1 \geq k \geq N$ and $b \in \{0,1\}$, where $\widetilde{\pi}_{k,b}(\theta,b) = (\pi_k(\theta), 1-b)$. It is easy to see that this new family of permutations has both the former and the latter properties, and we prove that it is also a mutually orthogonal family.

With this family, the second member of the equation above is negligible, hence the winning probability in the monogamy-of-entanglement game (for both BB84 and coset versions) is upper-bounded by $1/2 + \mathsf{negl}(n)$.

[7] A family of permutations $\{\pi_k\}_k$ is orthogonal if for all x, $\pi_k(x) = \pi_{k'}(x)$ implies $k = k'$.

5.6.2 Computational Parallelized Version

As mentioned above, when this game is parallelized, the winning probability drops to negligible. The proof follows the same structure as the one above, and we give it in Appendix A.4. Furthermore, giving iO-obfuscated membership programs for the regular and dual cosets to *computationally bounded* adversaries does not significantly increase this probability. The proof uses the same arguments as the ones used in Section 5.8.4. We formalize this computational-parallelized version, and provide an illustration of it in Figure 5.9.

Theorem 21 (Monogamy-Of-Entanglement With Identical Basis For Coset States (Computational Parallelized Version))**.** Define the following game, between a challenger and a triple of adversaries $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and parametrized by a security parameter λ. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate. Let $n = \lambda$, and $\kappa = \mathsf{poly}(\lambda)$.

- **Setup phase:**
  - The challenger samples κ independent random n-long coset state's description $\{(A_i, s_i, s'_i)\}_{i\in[\![1,\kappa]\!]}$.
  - The challenger generates the corresponding iO-obfuscated programs $\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}$ for all $i \in [\![1,\kappa]\!]$.
  - The challenger sends $\bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle$, and $\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}$ to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite quantum state $|\psi^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\psi^*\rangle_1$ to $\mathcal{B}$ and $|\psi^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples a κ-long bitstring $r \leftarrow\!\!\$\ \{0,1\}^\kappa$.
  - The challenger sends $(A_i)_{i\in[\![1,\kappa]\!]}$ and r to both $\mathcal{B}$ and $\mathcal{C}$.

Let $(v_1^*, v_2^*, \ldots, v_\kappa^*)$ denotes the output of $\mathcal{B}$, and $(w_1^*, w_2^*, \ldots, w_\kappa^*)$ denotes the output of $\mathcal{C}$. They win if, for all $i \in [\![1,\kappa]\!]$, $r_i = 0$ and $v_i^*, w_i^* \in A_i + s_i$, or if $r_i = 1$ and $v_i^*, w_i^* \in A_i^\perp + s'_i$.

The monogamy-of-entanglement with identical basis property states that no triple of efficient adversaries can win this game with non-negligible probability. In other words, for any triple of QPT algorithms $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$,

$$\Pr\left[\begin{array}{c} \forall i \in [\![1,\kappa]\!] : \\ r_i = 0 \wedge v_i^*, w_i^* \in A_i + s_i \\ \vee \\ r_i = 1 \wedge v_i^*, w_i^* \in A_i^\perp + s'_i \end{array} \;:\; \begin{array}{l} (v_1^*, v_2^*, \ldots, v_\kappa^*) \leftarrow \mathcal{B}(|\psi^*\rangle_1, A, r) \\ (w_1^*, w_2^*, \ldots, w_\kappa^*) \leftarrow \mathcal{C}(|\psi^*\rangle_2, A, r) \\ |\psi^*\rangle_{12} \leftarrow \mathcal{A}\left(\bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle, \{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}\right) \\ r \leftarrow\!\!\$\ \{0,1\}^\kappa \\ s_i, s'_i \leftarrow\!\!\$\ \mathbb{F}_2^n\ \forall i \in [\![1,\kappa]\!] \\ A_i \leftarrow\!\!\$\ \{A \in \mathbb{F}_2^{n\times n/2} \;:\; A \text{ is full-rank}\}\ \forall i \in [\![1,\kappa]\!] \end{array}\right] \leq \mathsf{negl}(\lambda)$$

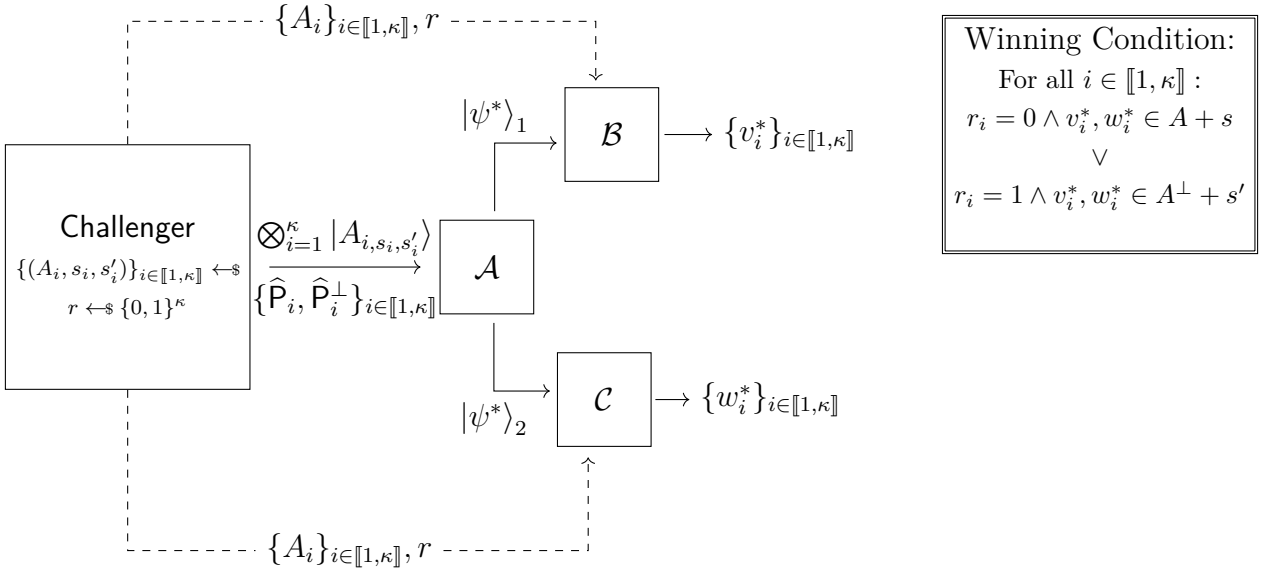

Figure 5.9: Computational κ-parallelized version of the monogamy-of-entanglement game with identical basis for coset states. All (A_i, s_i, s'_i) are random n-long coset state's descriptions. $\widehat{\mathsf{P}}_i$ and $\widehat{\mathsf{P}}_i^{\perp}$ respectively denote the iO-obfuscated membership programs $\mathsf{iO}(\mathsf{P}_{A_i+s_i})$ and $\mathsf{iO}(\mathsf{P}_{A_i^{\perp}+s'_i})$. No triple of adversaries can win this game with non-negligible probability.

5.7 Conjectures on Simultaneous Compute-and-Compare Obfuscation

In this section, we present our conjectures. We first give an overview of the conjectures, then we define them formally, and finally we discuss their relation to similar conjectures in a recent work of Ananth and Behera (2024).

5.7.1 Original Compute-And-Compare Obfuscation

Recall that compute-and-compare obfuscation (Section 3.3.8) works as follows. Let $\mathcal{D}$ be a distribution over pairs $(\mathsf{CC}[f,\ell,s], \mathsf{aux})$ where $\mathsf{CC}[f,\ell,s]$ is a compute-and-compare program with function f, lock-value ℓ, and secret s, and aux is a corresponding (possibly quantum) auxiliary information. $\mathcal{D}$ is said unlearnable if no efficient adversary can extract the lock-value of $\mathsf{CC}[f,\ell,s]$ from (f, aux) with non-negligible probability (averaged over pairs of program and auxiliary information sampled from $\mathcal{D}$). If $\mathcal{D}$ is unlearnable, then there exists an efficient procedure — called a compute-and-compare obfuscator — that takes as input a compute-and-compare program $\mathsf{CC}[f,\ell,s]$, and returns an obfuscated version of it: $\widetilde{\mathsf{CC}}[f,\ell,s]$. This obfuscated program is functionally equivalent to the original one, but is indistinguishable from a simulated dummy program that returns $\perp$ on every input.

5.7.2 Non-Local Context

We ask whether this still holds in a non-local context. More precisely, consider the two following tasks, which we call *simultaneous distinguishing* and *simultaneous predicting*.

Simultaneous predicting asks two players, Bob and Charlie, given a function associated to a compute-and-compare program, and a quantum state as auxiliary information on the program, to return the associated lock-value. Note that the function given to Bob and the one given to Charlie are not necessarily the same, and that the same goes for the quantum states they are given. Also, crucially, Bob's and Charlie's quantum states can be entangled. In simultaneous distinguishing, Bob and Charlie are given either an obfuscated compute-and-compare program, or the outcome of a simulated dummy program. As in simultaneous predicting, they are also given a quantum state each, and here, they are asked to tell whether they received the obfuscated program, or the simulated one.

These two tasks are parameterized by a distribution $\mathcal{D}$ over triples of the form $(\mathsf{CC}_1[f_1, \ell_1, s_1], \mathsf{CC}_2[f_2, \ell_2, s_2], |\psi\rangle_{12})$ — where the two first elements are compute-and-compare programs used to create the challenges in the challenge phase and the last one is the bipartite quantum state shared by Bob and Charlie. We say that such a distribution is simultaneously unpredictable if no adversaries can succeed in the associated simultaneous predicting task; and that simultaneous compute-and-compare obfuscation exists for this distribution if there is a compute-and-compare obfuscator with respect to which no adversaries can succeed in the associated simultaneous distinguishing task.[8] The question we ask now is:

Question. *Is there simultaneous compute-and-compare obfuscation for any simultaneous unpredictable distribution ?*

As discussed in Coladangelo, Liu, Liu, and Zhandry (2021), this question is far from trivial. Indeed, consider its contraposition: *if all candidate algorithms for simultaneous compute-and-compare obfuscation fail to obfuscate the programs as desired, does it mean that the distribution is simultaneously predictable for a certain pair of algorithms ?* Intuitively, the difficulty here stems from whether the challenges are independent or not: if they are, then one can analyze the two adversaries in the distinguishing game independently, and thus say that if the first adversary succeeds in their part of the task, then they can predict their lock-value, and that the same goes for the second adversary. If the challenges are not independent in the other hand, it is not clear what happens when the first adversary predicts the lock-value. Concretely indeed, the prediction is a measurement, Bob's measurement might perturb Charlie's register in a way that prevents the latter to predict his lock-value.

In this work, we break down this question in the following way. We consider two cases. In the first one, Bob and Charlie, in the simultaneous distinguishing task, either both receive an obfuscated program, or a simulated one, while in the second task, this choice is independent. In both tasks, the programs they receive are obfuscated using the same set of random coins.

5.7.3 Conjectures

We present the two tasks mentioned above. Both tasks are parameterized by a distribution $\mathcal{D}$ over triples of the form $(\mathsf{CC}[f_1, \ell_1, s_1], \mathsf{CC}[f_2, \ell_2, s_2], |\psi\rangle_{12})$, and played by two non-communicating players Bob and Charlie.

[8] Succeeding in these tasks mean that Bob and Charlie must both extract, or both distinguish correctly (depending on the task).

Simultaneous distinguishing task. A challenger samples a triple $(\mathsf{CC}[f_1, \ell_1, s_1], \mathsf{CC}[f_2, \ell_2, s_2], |\psi\rangle_{12})$ from $\mathcal{D}$. Then, the challenger sends $(f_1, |\psi\rangle_1)$ to Bob and $(f_2, |\psi\rangle_2)$ to Charlie. The latter succeed if Bob returns ℓ_1 and Charlie returns ℓ_2.

Simultaneous predicting task. We first define an *identical bit version* of this game. A challenger samples a triple of $(\mathsf{CC}[f_1, \ell_1, s_1], \mathsf{CC}[f_2, \ell_2, s_2], |\psi\rangle_{12})$ from $\mathcal{D}$, random coins r, and a bit b. The challenger computes $\mathsf{C}_1 \leftarrow \mathsf{CC\text{-}Obf}(\mathsf{CC}[f_1, \ell_1, s_1]; r)$ if $b = 0$, or $\mathsf{C}_1 \leftarrow \mathsf{Sim}()$ if $b = 1$, where Sim is an efficient simulator for compute-and-compare obfuscation. The challenger computes C_2 in the same way, using $\mathsf{CC}[f_2, \ell_2, s_2]$ instead of $\mathsf{CC}[f_1, \ell_1, s_1]$; importantly, the same random coins r are used for obfuscation of both programs. Finally, the challenger sends $(\mathsf{C}_1, |\psi\rangle_1)$ to Bob, and $(\mathsf{C}_2, |\psi\rangle_2)$ to Charlie. Bob and Charlie succeed they both return b.

We also define this task in the *product bit version*, in which the challenger samples two independent bits b_1 and b_2, deciding whether Bob and Charlie respectively are given an obfuscated program, or a simulated one.

We now state our two conjectures, illustrated in Figure 5.10.

Conjecture 1. Let $\mathcal{D}$ a distribution over triples of the form $(\mathsf{CC}[f_1, \ell_1, s_1], \mathsf{CC}[f_2, \ell_2, s_2], |\psi\rangle_{12})$. Assume that, for all QPT adversaries $(\mathcal{B}, \mathcal{C})$, the probability that $(\mathcal{B}, \mathcal{C})$ succeed in the simultaneous predicting task is negligible. Then, there exists a compute-and-compare obfuscator $\mathsf{CC\text{-}Obf}$ and its associated efficient simulator Sim with respect to which, no adversaries $(\mathcal{B}', \mathcal{C}')$ succeed in the simultaneous distinguishing task (identical bit version) with probability significantly greater than $1/2$.

Conjecture 2. Let $\mathcal{D}$ a distribution over triples of the form $(\mathsf{CC}[f_1, \ell_1, s_1], \mathsf{CC}[f_2, \ell_2, s_2], |\psi\rangle_{12})$. Assume that, for all QPT adversaries $(\mathcal{B}, \mathcal{C})$, the probability that $(\mathcal{B}, \mathcal{C})$ succeed in the simultaneous predicting task is negligible. Then, there exists a compute-and-compare obfuscator $\mathsf{CC\text{-}Obf}$ and its associated efficient simulator Sim with respect to which, no adversaries $(\mathcal{B}', \mathcal{C}')$ succeed in the simultaneous distinguishing task (product bit version) with probability significantly greater than $1/2$.

5.7.4 Related Work

In a recent work of Ananth and Behera (2024), the authors make a similar conjecture, this time on simultaneous Goldreich-Levin prediction. Roughly, the usual Goldreich-Levin theorem states that if f is a one way function (meaning that a random x is not predictable given $f(x)$), then no adversary can distinguish the dot product $x \cdot r$ from a random bit, given $f(x)$ — where x is a random input and r a random bitstring of same length. Ananth and Behera consider the simultaneous version of this task, that is, assuming that (x_1, x_2) are simultaneously unpredictable given $(f(x_1), f(x_2))$ (in the same sense as our definitions above), then $x_1 \cdot r_1$ and $x_2 \cdot r_2$ are simultaneously indistinguishable from two random strings — where the pairs (x_1, x_2), (r_1, r_2), and (b_1, b_2), are sampled from different types of distributions (uniform or identical), similarly as in our case — and they finally describe two conjectures. Note that, as there is a construction of compute-and-compare obfuscation from Coladangelo, Liu, Liu, and Zhandry (2021) (based on iO and hardness of LWE assumptions) that ultimately relies on the Goldreich-Levin theorem, we expect the conjectures of Ananth and Behera, combined with iO and LWE assumptions, to imply our conjectures.

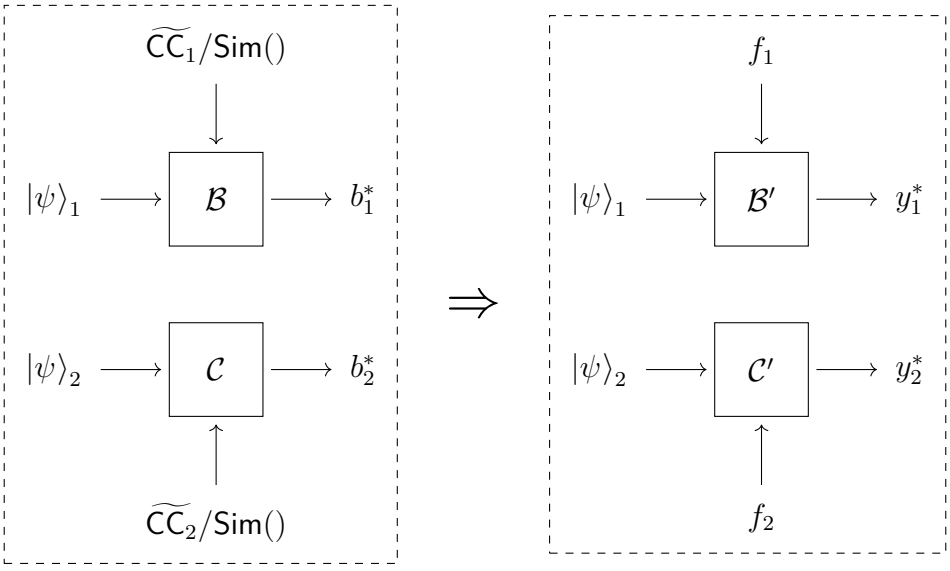

Figure 5.10: Contraposition of the conjectures: if $\mathcal{B}$ and $\mathcal{C}$ succeed in the simultaneous distinguishing task on the left with significant advantage over $1/2$, then there exist $\mathcal{B}'$ and $\mathcal{C}'$ succeeding in the simultaneous predicting task on the right with non-negligible probability. $\widetilde{\mathsf{CC}_1}$ and $\widetilde{\mathsf{CC}_2}$ represent the compute-and-compare obfuscation of CC_1 and CC_2 with the same random coins.

5.8 Tokenized Signature in the Plain Model

In this section, we present two new definitions for tokenized signature schemes, and show that the construction of Coladangelo, Liu, Liu, and Zhandry (2021) satisfies these constructions. Along the way, we present a new version of the direct product hardness property of coset states.

5.8.1 Tokenized Signatures

Recall that a public-key signature scheme has a key generation KeyGen procedure that generates a pair of keys: the verification key which is given to all parties, and allows for verifying signatures with the Verify procedure, and the secret key that is used to sign messages with the Sign procedure. A tokenized signature scheme is similar, except that the secret key allows for generating quantum tokens $|♙\rangle$ — with an additional $\mathsf{TokenGen}$ procedure — that can be used to sign *only one message*.

Weak and strong-unforgeability. Such a scheme has weak-unforgeability if no efficient adversary can produce two valid (message, signature) pairs given a verification key, and only one quantum token, such that the messages in the two pairs are different. We define strong-unforgeability, analogously as for classical signature schemes. That is, we relax the task mentioned above, and only ask the two pairs to be different. This means that an adversary can produce a message with two different valid signatures to succeed. More

precisely, for all QPT adversary $\mathcal{A}$, the following must hold:

$$\Pr\left[\begin{array}{lll} \mathsf{Verify}(\mathsf{vk}, m_1^*, sig_1^*) = 1 & & \\ \qquad \wedge & & (m_1^*, sig_1^*, m_2^*, sig_2^*) \leftarrow \mathcal{A}(\mathsf{vk}, |\text{♙}\rangle) \\ \mathsf{Verify}(\mathsf{vk}, m_2^*, sig_2^*) = 1 & : & |\text{♙}\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk}) \\ \qquad \wedge & & (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (m_1, sig_1) \neq (m_2, sig_2) & & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Unclonable unforgeability. Motivated by the non-local scenario considered for other unclonable primitives, we define the following unclonable unforgeability property for tokenized signature schemes. We consider a scenario where non-communicating Bob and Charlie want to share a single quantum token, in a way that allows both of them to sign a message of their choice, even when this message is *chosen after the token is split*. We capture this property with a game, played by a triple of adversaries Alice, Bob, and Charlie, where a challenger first sends a verification key vk, together with a corresponding signing token $|\text{♙}\rangle$, to Alice, who splits it — that is, she generates two states $|\text{♙}\rangle_1^*$ and $|\text{♙}\rangle_2^*$ (possibly entangled) — and sends the first and second states to Bob and Charlie respectively. Bob and Charlie are then given a random message to sign by the challenger, and must return a valid signature for this message. Thus, we relate how well they perform in the aforementioned task to the winning probability of this game, and, more importantly, to the difference between this winning probability and the "trivial" winning probability of this game. Here, the "trivial" winning probability is given by a strategy where Alice makes a random guess on the challenge message, sign this message using her token, and sends the signature to both Bob and Charlie. If her guess is correct, Bob and Charlie return the signature and win the game; if it is not, they return a random vector as a signature and lose the game almost with certainty.

Choosing the challenges. We directly see that, if the challenge messages are different, then, conditioned on the scheme having weak unforgeability, we already know that Alice, Bob, and Charlie cannot perform better than the trivial strategy. Indeed, otherwise, Alice would be able to produce two valid (message, signature) pairs with different messages, which is ruled out by the weak unforgeability property. Thus, we only consider the case where the challenger messages are the same. We propose two different ways of choosing the challenges. The first one — the search fashion — simply consists in sampling a message uniformly at random, and to send it to Bob and Charlie as the challenge. The second one — the decision fashion — is defined similarly as in the IND-CPA security of an encryption scheme. Alice, after she receives the quantum token and the verification key, is asked to send the challenger two messages of her choice. The challenger then samples uniformly at random one of these messages, and sends it to Bob and Charlie as the challenge. In the following, we consider a tokenized signature scheme for single-bit message. Note that in this case, the two ways of choosing the challenges are actually equivalent.

5.8.2 Definition

We formally define unclonable unforgeability of a tokenized signature scheme for single-bit message space below.

Define the following game, parameterized by a security parameter λ, and between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed

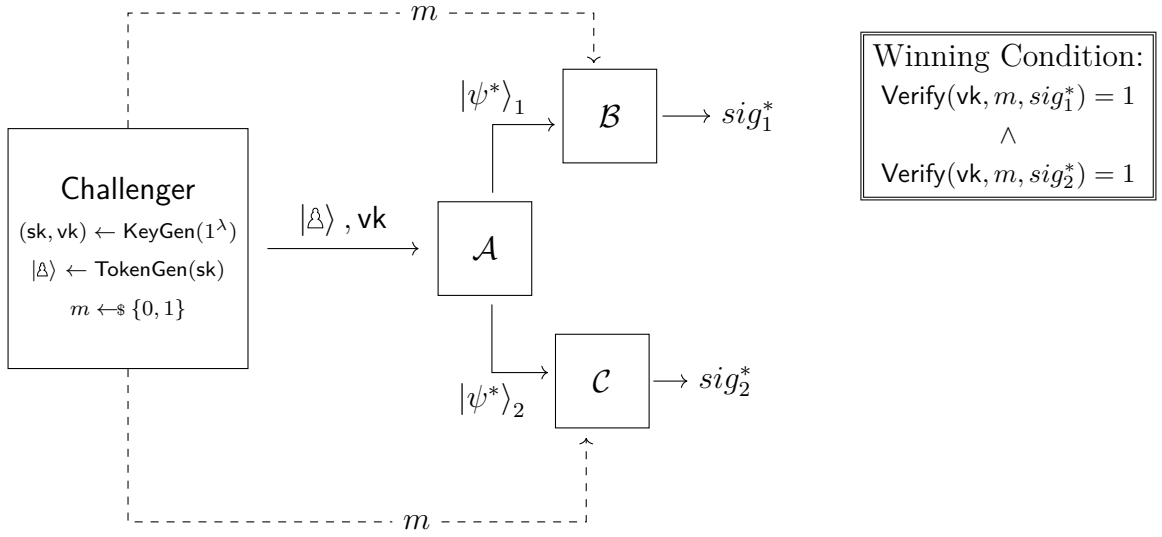

Figure 5.11: Unclonable unforgeability game for tokenized signatures for single-bit message space. A tokenized signature scheme has unclonable unforgeability if no triple of adversaries can win this game with non-negligible advantage over $1/2$.

to communicate.

- **Setup phase:**
  - The challenger samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and a message $m \in \{0,1\}$.
  - The challenger computes $|\text{♙}\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk})$.
  - The challenger sends $|\text{♙}\rangle$ and vk to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\text{♙}\rangle^*_{12}$.
  - $\mathcal{A}$ sends $|\text{♙}\rangle^*_1$ to $\mathcal{B}$, and $|\text{♙}\rangle^*_2$ to $\mathcal{C}$.
- **Challenge phase:** The challenger sends m to both $\mathcal{B}$ and $\mathcal{C}$.

Let sig^*_1 denotes the output of $\mathcal{B}$, and sig^*_2 denotes the output of $\mathcal{C}$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if $\mathsf{Verify}(\mathsf{vk}, m, sig^*_1) = 1$ and $\mathsf{Verify}(\mathsf{vk}, m, sig^*_2) = 1$. We then say that a tokenized signature scheme has unclonable unforgeability if no adversaries can win this game with probability significantly greater than $1/2$. That is, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{cc} & sig^*_1 \leftarrow \mathcal{B}(|\text{♙}\rangle^*_1, m), sig^*_2 \leftarrow \mathcal{C}(|\text{♙}\rangle^*_2, m) \\ \mathsf{Verify}(\mathsf{vk}, m, sig^*_1) = 1 & |\text{♙}\rangle^*_{12} \leftarrow \mathcal{A}(|\text{♙}\rangle, \mathsf{vk}) \\ \wedge & : \ |\text{♙}\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk}) \\ \mathsf{Verify}(\mathsf{vk}, m, sig^*_2) = 1 & m \leftarrow\!\$ \{0,1\} \\ & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We provide an illustration of this security property in Figure 5.11.

5.8.3 Construction

We present in the following the tokenized signature construction of Coladangelo, Liu, Liu, and Zhandry (2021). This construction relies on coset states and roughly works as follows. A secret key consists in a random coset state's description, and the corresponding verification key is the associated iO-obfuscated membership programs. Generating a quantum token for a secret key (A, s, s') then consists in preparing and returning the coset state $|A_{s,s'}\rangle$. To sign a single-bit message m, one measures the quantum token in the rectilinear basis if $m = 0$, or in the diagonal one if $m = 1$. The signature is the measurement outcome: a vector in the regular or dual coset, depending on m. Finally, verifying a signature consists in checking whether it belongs to the coset corresponding to the message, which can be performed with the membership programs. We present the construction formally in construction 18.

Construction 18: Tokenized Signature Scheme [CLLZ21]

Let $n = \mathsf{poly}(\lambda)$.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample an n-long random coset state's description (A, s, s').
- Generate the corresponding membership programs P_{A+s}, and $\mathsf{P}_{A^\perp+s'}$.
- Return (A, s, s') as the secret key, and the obfuscated membership programs $(\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'})$ as the public key.

$\mathsf{TokenGen}(A, s, s')$: Prepare and return $|A_{s,s'}\rangle$.

$\mathsf{Sign}(|A_{s,s'}\rangle, m)$:

- Apply $\mathsf{H}^{\otimes n}$ to $|A_{s,s'}\rangle$ if $m = 0$. Otherwise, leave the state unchanged. Let $|\text{♙}'\rangle$ denotes the resulting state.
- Measure $|\text{♙}'\rangle$ in the rectilinear basis, and return the outcome.

$\mathsf{Verify}((\widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'}), m, sig)$:

- If $m = 0$: return $\widehat{\mathsf{P}}_{A+s}(sig)$.
- If $m = 1$: return $\widehat{\mathsf{P}}_{A^\perp+s'}(sig)$.

The correctness of this scheme follows immediately from the functionality preserving property of iO-obfuscation, and the fact that measuring a coset state in the rectilinear (resp. diagonal) basis yields a vector in the regular (resp. dual) coset.

5.8.4 Direct Product Hardness with Identical Basis

To prove the weak-unforgeability of this construction, Coladangelo, Liu, Liu, and Zhandry (2021) leverage the direct product hardness of coset states, and the proof follows immediately. This property does not give us strong-unforgeability however: we need to prove a variant of it, that we name direct product hardness with identical basis. In this variant, the adversary Alice is still given a random coset state, but instead of having to return two

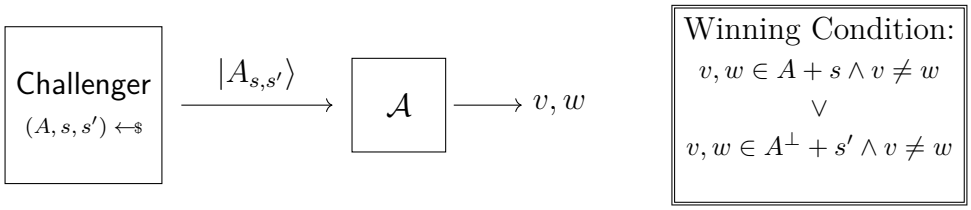

Figure 5.12: Direct product hardness with identical basis game for coset states. (A, s, s') is a random n-long coset state's description. No adversary wins this game with non-negligible probability in n.

vectors, respectively in the regular and dual cosets, she can return two vectors in the coset of her choice (regular or dual), as long as they are different. We illustrate this security notion in Figure 5.12.

As with the other coset states properties, we show that this task remains hard when the adversary is given polynomial queries to a membership oracle. We also prove a computational version of this game, where the adversary is computationally bounded, and given iO-obfuscation membership programs for the regular and dual cosets.

We give below high-level proofs of the statistical and computational versions. These proofs follow the structure of the ones of Coladangelo, Liu, Liu, and Zhandry (2021), with a significant modification in the computational one.

Statistical version. The proof is based on a result of Ben-David and Sattath (2023) that states that no adversary, given a subspace state $|A\rangle$, can output two non-zero vectors $v \neq w$, both in A or in $A^{\perp}$. Our statistical property follows easily, as given $|A\rangle$, one can sample two random vectors s and s', and prepare the state $|A_{s,s'}\rangle$, allowing to perfectly simulate an adversary for our coset state game. Then, it suffices to subtract s (or s') from this adversary's output to obtain two different non-zero vectors in A (or in $A^{\perp}$).

Remark 4. The same proof idea works when the adversary is provided with oracle access (limited to a polynomial number of queries) to membership programs for regular and dual cosets.

Computational version. The proof of the computational version aims to show that obfuscated membership programs do not give significant advantage to an efficient adversary In a nutshell, it consists in proving that the task of finding two different vectors, both in the regular or dual coset, given an n-long coset state and obfuscated membership programs, is equivalent of finding such a pair of vectors given only an $n/4$-long coset state.

Replacing membership programs. In the first part of the proof, we show that replacing an obfuscated program for the coset $A + s$ by one for the coset $B + s$ — where B is a random subspace of dimension $7n/8$, such that $A \subset B$ — does not change the winning probability of any adversary. This comes from iO-obfuscation, as these two programs are functionally equivalent on average. Indeed, the only difference occurs when the program is evaluated on a vector in $B \setminus A$, which can only happen with negligible probability as this set is sparse in $\mathbb{F}_2^n$. Using iO property again, we show that the latter program can be unnoticeably replaced by an obfuscated membership program for the coset $B + t$ — where t is the sum of s and a random vector in B — as $B + t$ and $B + s$ are the same coset. Applying the same arguments to the second membership program (for the dual coset), we

show that this program can be replaced by an obfuscated membership program for the coset $C^\perp + t'$, where C is a random subspace of smaller dimension such that $C \subset A$, and t' is the sum of s' and a random vector in C.

Ruling out a trivial attack. If we keep using the same proof structure as the one of Coladangelo, Liu, Liu, and Zhandry (2021), the next step would be to show that we can actually send B, C, t, t' in the clear to the adversary, without harming the hardness of the game. However, this argument does not work in our settings. Indeed, when B and C are given in the clear to the adversary, the latter can apply the following trivial strategy to win the game. First they measure the coset state in the rectilinear basis to obtain a vector v in A. Then, set $w = v + u_C$ where u_C is any non-zero vector in C. As C is a subset of A, the vector w belongs to A, hence (v, w) is a valid answer for the game.

We overcome this issue by showing that, if an adversary wins the last hybrid — in which they are given obfuscated membership programs for cosets $B + t$ and $C^\perp + t'$ — with non-negligible probability, then they can also win a variant of this hybrid where the winning condition asks the pair of vectors (v, w) to be such that $v - w$ does not belong to C or $B^\perp$. This is done by leveraging a lemma proved by Shmueli (2022b), which, when applied to our settings, roughly states that an adversary cannot consistently return a pair of vectors (v, w) such that $v - w$ is in C, and has to "miss" the subspace C, that is return (v, w) such that $v - w$ is in $A \setminus C$ with non-negligible probability. Note that a strategy similar as the one above can be applied to return two different vectors in $C^\perp + t'$. We prevent this strategy analogously by asking the pair of vectors not to belong to $B^\perp$.

This now allows us to send B, C, t, t' in the clear, as the aforementioned trivial strategies no longer produce a valid pair of vectors.

Sending B and C in the clear. We justify that B, C, t, t' can be sent in the clear by adapting the arguments of Coladangelo, Liu, Liu, and Zhandry (2021). We can assume without loss of generality that B is the subspace of vectors whose last $n/8$ entries are zero, and C is the subspace of vectors whose last $7n/8$ entries are zero. With such subspaces, A is now a random subspace of vectors whose last $7n/8$ entries are 0, and whose first $n/8$ entries are free. Note now that given a uniformly random coset state $|\widetilde{A}, \widetilde{s}, \widetilde{s}'\rangle$, for the ambient space $\mathbb{F}_2^{3n/4}$, and where $\widetilde{A}$ is of dimension $3n/8$, one can prepare a coset state $|A_{s,s'}\rangle$ where A has the aforementioned constraints on the entries. This is done by setting $|A_{s,s'}\rangle = |\mathbb{I}_{\widehat{s,s'}}\rangle \otimes |\widetilde{A}, \widetilde{s}, \widetilde{s}'\rangle \otimes |\mathring{s}\rangle$, where $\mathbb{I}$ is the identity matrix of $\mathbb{F}_2^{n/8}$, and $\widetilde{s}, \widetilde{s}', \mathring{s}$ are random vectors in $\mathbb{F}_2^{n/8}$. The resulting coset state $|A_{s,s'}\rangle$ is a uniformly n-long random coset state, subject to the last $7n/8$ entries of its vectors are 0, and the first $n/8$ are free. Following these observations, we reduce the task of succeeding in the hybrid where B, C, t, t' are given in clear — considering the ambient space $\mathbb{F}_2^n$, to the task of returning a pair of different vectors (v, w), both in the regular or in the dual coset, considering the ambient space $\mathbb{F}_2^{3n/4}$.

Chapter

6 Semi-Quantum Unclonable Cryptography

In this chapter, we present a remote state preparation for coset states, and show how to use it in a plug-and-play manner in existing unclonable primitive to obtain semi-quantum versions of these primitives. This chapter is based on our following work: Chevalier, Hermouet, and Vu (2023). We provide some supplementary materials for this chapter in Appendix B.

Chapter content

6.1 Introduction

Quantum technology is still at an early stage of its existence, and despite promising advances, a world where everyone owns their own quantum computer, and can communicate through quantum channels seems very distant. Instead, we can think of a closer world,

with a few "quantum parties" having access to quantum computers, and where most of the people are classical, and can only communicate through classical channels.

One question is then how it is possible to make the most of quantum capabilities with such strong restrictions. In particular, would it be possible to use unclonable cryptography, where parties are exchanging quantum states with various useful properties ? We propose a positive answer to this question, by constructing semi-quantum unclonable cryptography protocols, where the party who — in the quantum version — sends the quantum program (or key, token, etc...) is classical and instructs the receiver on how to construct such a quantum program, while preserving the unclonability of the protocol.

The content of this chapter is based on one paper (Chevalier, Hermouet, and Vu (2023)). In this chapter, we present our *remote coset states preparation* protocol, which we then use in a plug-and-play manner to construct so-called semi-quantum versions of unclonable cryptographic protocols based on coset states. In order to construct our remote coset states preparation protocol, we start with an idea of Shmueli (2022a) to use quantum fully homomorphic encryption to delegate preparation of coset states. As this delegating preparation does not preserve the monogamy-of-entanglement of coset states, we combine it with the self-testing and remote state preparation protocols for BB84 states of Gheorghiu, Metger, and Poremba (2022) and achieve our purpose.

This chapter is articulated as follows. We first present in Section 6.2 the protocol of Shmueli, and show why it does not fit our purpose. Then, in Section 6.3, we present the remote state preparation protocol of Gheorghiu, Metger, and Poremba. In Sections 6.4 and 6.5, we show how to combine these protocols to obtain a self-testing protocol for coset states, and based on that, a remote coset state preparation. Finally, in Section 6.6, we discuss semi-quantum unclonable cryptography, by defining semi-quantum copy-protection, and providing a construction based on our remote coset state preparation.

6.2 Delegating Preparation of Coset States — A First Idea Based on Homomorphic Encryption

Our goal in this chapter is to construct a remote coset state preparation protocol, that is a protocol in which a classical verifier instructs a quantum prover on how to construct a coset state. Furthermore, this protocol must preserve the unclonability of coset states, in particular the direct product hardness and monogamy-of-entanglement properties. Shmueli (2022a) presented a protocol to delegate preparation of coset states. However, as he pointed out, there is a simple attack on this protocol, and it does not preserve direct product hardness, nor monogamy-of-entanglement. In this section, we first define remote coset state preparation, then present Shmueli's protocol. We finally describe the attack mentioned above, showing that this protocol is not sufficient for our purpose.

6.2.1 Remote Coset State Preparation

We refer the reader to Section 4.2.2 for a detailed presentation of coset states. We define remote coset state preparation as an interactive protocol between two parties, that we call the verifier and the prover. In the end of the interaction, when the parties are honest, the verifier holds a coset state's description, and the prover the corresponding coset state. We want such a protocol to preserve direct product hardness and monogamy-of-entanglement properties of coset states. This is captured by two "semi-quantum games", one for each

property, similar to their original versions (Section 4.2.2), except that we replace the setup phase — in which the challenger samples the coset and sends it to the adversary — by the remote coset state preparation protocol. For technical reasons — that we explain in subsequent sections — our final protocol prepares a polynomial number κ of coset states. The definition is then adapted: in the semi-quantum games for direct product hardness and monogamy-of-entanglement, the adversaries receive κ coset states from the remote coset state preparation protocol, and must return a valid answer for each state to win the game.

Definition 38 (Remote Coset State Preparation Protocol)**.** Let λ a security parameter, and $n, \kappa \in \mathbb{N}$. A remote coset state preparation protocol is an interactive protocol $\mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle$ between a PPT verifier $\mathcal{P}$, and a QPT prover $\mathcal{P}$, with the following properties:

- $\left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, \left(\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\right)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle$. On input a security parameter, the verifier and the prover exchange a series of classical messages. After this interaction, the verifier outputs κ n-long coset state's descriptions $(A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}$, and the prover the corresponding quantum state $|\psi\rangle$. Both also output iO-obfuscated membership programs $\widehat{\mathsf{P}}_{A_i+s_i}$ and $\widehat{\mathsf{P}}_{A_i^\perp+s'_i}$ for every coset state description.

In the following, we simply write $\left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle$, to describe an execution of this protocol, when clear from the context.

In addition, we ask a remote coset state preparation to satisfy the following properties.

Correctness. The quantum state returned by an honest prover (that is, a prover who follows the protocol) must be negligibly close to the κ coset states corresponding to the verifier's descriptions. More precisely,

$$\Pr\left[|\psi\rangle \approx \bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle \ : \ \left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle\right] \geq 1 - \mathsf{negl}(\lambda)$$

Where $\approx$ is defined in Section 4.2.2.

Monogamy-of-entanglement and direct product hardness. We do not ask the quantum state output by any (efficient) malicious prover to be close to the expected one, conditioned on acceptance. Rather than that, we ask that no efficient cheating strategy in the remote coset state preparation protocol allow a prover to get sufficiently enough information on the coset state, allowing them to win the direct product hardness, or the monogamy-of-entanglement game with non-negligible probability. More precisely, we ask that no efficient prover win the following two games with non-negligible probability.

Theorem 22 (Semi-Quantum Direct Product Hardness)**.** The direct product hardness must be preserved when the preparation and sending of coset states is replaced by the remote coset state preparation. That is, for any QPT adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} \forall\, i \in [\![1,\kappa]\!] \\ v_i^* \in A_i + s_i \\ \wedge \\ w_i^* \in A_i^\perp + s'_i \end{array} \ : \ \begin{array}{l} v^*, w^* \leftarrow \mathcal{A}(|\psi\rangle\ , (\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i})_{i\in[\![1,\kappa]\!]}) \\ \left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{A}(1^\lambda)\rangle \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $v^* = (v_1^*, \ldots, v_\kappa^*)$, and $w^* = (w_1^*, \ldots, w_\kappa^*)$.

Theorem 23 (Semi-Quantum Monogamy-of-Entanglement)**.** Define the following game, between a challenger $\mathcal{V}$ and a triple of adversaries $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and parametrized by a security parameter λ, and a remote coset state preparation protocol RCP. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger and $\mathcal{A}$ execute the remote coset state preparation protocol $\left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{A}(1^\lambda)\rangle$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite quantum state $|\psi^*\rangle_{12}$.[1]
  - $\mathcal{A}$ sends $|\psi^*\rangle_1$ to $\mathcal{B}$ and $|\psi^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:** The challenger sends $(A_1, \dots, A_\kappa)$ to both $\mathcal{B}$ and $\mathcal{C}$.

Let $v^* = (v_1^*, \dots, v_\kappa^*)$ denotes the output of $\mathcal{B}$, and $w^* = (w_1^*, \dots, w_\kappa^*)$ denotes the output of $\mathcal{C}$. They win if and only if $v_i^* \in A_i + s_i$ and $w_i^* \in A_i^\perp + s'_i$ for all $i \in [\![1, \kappa]\!]$.

The semi-quantum monogamy-of-entanglement property states that no triple of efficient adversaries can win this game with non-negligible probability. In other words, for any triple of QPT algorithms $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$,

$$\Pr\left[\begin{array}{c} \forall\, i \in [\![1,\kappa]\!] \\ v_i^* \in A_i + s_i \\ \wedge \\ w_i^* \in A_i^\perp + s'_i \end{array} \;:\; \begin{array}{l} v^* \leftarrow \mathcal{B}(|\psi^*\rangle_1, (A_i)_{i\in[\![1,\kappa]\!]}), w^* \leftarrow \mathcal{C}(|\psi^*\rangle_2, (A_i)_{i\in[\![1,\kappa]\!]}) \\ |\psi^*\rangle_{12} \leftarrow \mathcal{A}(|\psi\rangle, (\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i})_{i\in[\![1,\kappa]\!]}) \\ \left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{A}(1^\lambda)\rangle \end{array}\right] \leq \mathsf{negl}(n)$$

We provide an illustration of this theorem in Figure 6.1.

6.2.2 A Protocol Based on Quantum Fully Homomorphic Encryption

We present the idea of Shmueli (2022a) for delegating coset states preparation. We first briefly introduce quantum fully homomorphic encryption.

Quantum fully homomorphic encryption. A fully homomorphic encryption scheme is a public key encryption scheme with an additional procedure allowing evaluation of a ciphertext on any (polynomial-size) circuit, given the public key. More precisely, when run on an encryption of a message m and a circuit C, this procedure returns an encryption of $\mathsf{C}(m)$.

For our purpose, we actually need a *quantum* fully homomorphic encryption scheme, where the circuits can be quantum. We consider schemes with the following structure. They are composed of a key generation procedure KeyGen that outputs a secret and a public key, an encryption procedure Enc for classical messages, and a decryption procedure Dec. So far, everything is classical. The quantum part comes with the evaluation procedure

[1] Similarly to the previous chapters, the quantum state prepared by the adversary can be a mixed state, contrarily to what the notation could suggest. In the rest of this chapter, for sake of simplicity, we abuse the notations and write quantum states prepared by adversaries with pure state notations (e.g. $|\psi^*\rangle_{12}$).

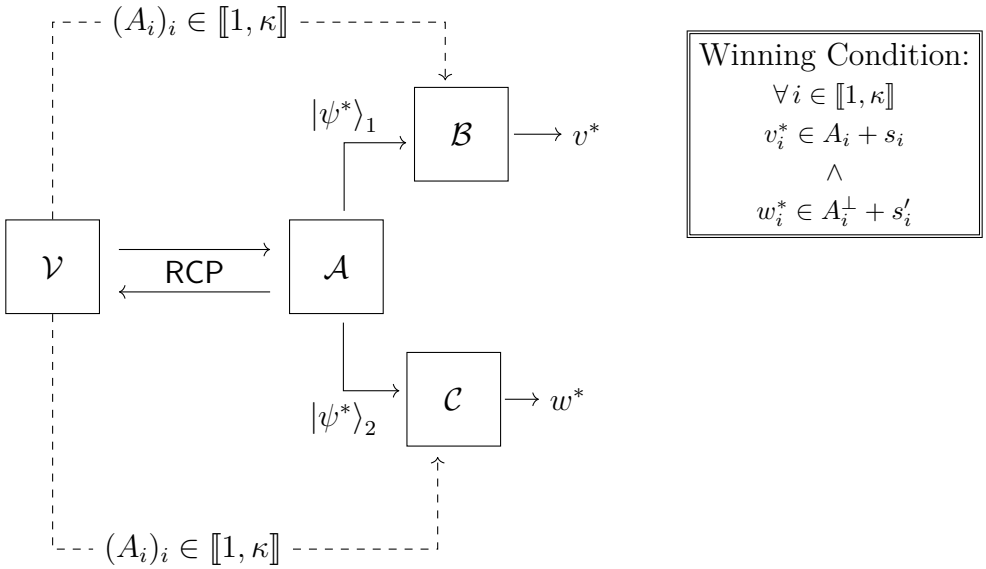

Figure 6.1: Semi-quantum monogamy-of-entanglement game for a remote coset state preparation protocol. This property states that no efficient adversaries win this game with non-negligible probability in λ.

Eval, that takes as input a public key, a state $|\psi\rangle^{(x,z)} = \mathsf{X}^x\mathsf{Z}^z|\psi\rangle$ (quantum one-time pad of $|\psi\rangle$ with keys x and z), the encryption of the keys $\mathsf{ct}_x = \mathsf{Enc}(\mathsf{pk}, x)$ and $\mathsf{ct}_z = \mathsf{Enc}(\mathsf{pk}, z)$, and a quantum circuit C. The procedure returns $|\psi'\rangle^{(x',z')}$: a quantum one-time padded evaluation of $\mathsf{C}(|\psi\rangle)$ with some keys x' and z', and $\mathsf{ct}_{x'}$ and $\mathsf{ct}_{z'}$: the encryption of these two keys. Whenever we use the evaluation procedure on the encryption of a classical message, we abuse the notation and simply write $\mathsf{Eval}(\mathsf{pk}, m^{(x)}, \mathsf{ct}_x, \mathsf{C})$, as applying or not a Z gate is equivalent. We refer the reader to Definition 18 for a more detailed definition.

The protocol. We are now ready to describe the protocol. The verifier first samples a random subspace $A \subset \mathbb{F}_2^\lambda$ of dimension $\lambda/2$ (where λ if a security parameter for the protocol). In the following, we use indifferently A to denote the subspace, or a matrix whose columns form a basis of the subspace. Then, they send a public key pk; a one-time pad masking of A with key x, $A^{(x)}$, as well as an encryption of x with key pk, ct_x, to the prover. The prover then use the evaluation procedure on these ciphertexts with a quantum circuit C that, on input a matrix A, returns the associated subspace state $|A\rangle$. The outcome of this evaluation is $\left(|A\rangle^{(s,s')}, \mathsf{ct}_s, \mathsf{ct}_{s'}\right)$. The prover then keeps $|A\rangle^{(s,s')}$ and sends ct_s and $\mathsf{ct}_{s'}$ to the verifier. The latter decrypts these two ciphertexts to obtain s and s', and finally outputs (A, s, s') as the coset state's description. Finally, recall that a coset state is in fact a quantum one-time padded subspace state. The prover returns the state $|A\rangle^{(s,s')}$, which is $|A_{s,s'}\rangle$. We write it in a more formal way below.

Protocol 1: Remote Coset State Preparation [Shmueli22]

Let λ a security parameter, and $n = \mathsf{poly}(\lambda)$. Let $\langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}\rangle$ be a quantum fully homomorphic encryption scheme, as defined above. This protocol is between a PPT verifier, and a QPT prover.

Verifier's setup phase:

- The verifier samples a random subspace $A \subset \mathbb{F}_2^n$, of dimension $n/2$.
- The verifier samples a pair of keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- The verifier computes $A^{(x)}$: the matrix A one-time padded with a random bitstring x.
- Finally, the verifier sends pk, $A^{(x)}$ and $\mathsf{ct}_x = \mathsf{Enc}(\mathsf{pk}, x)$ to the prover.

Prover's evaluation phase:

- The prover computes $\left(|A\rangle^{(s,s')}, \mathsf{ct}_s, \mathsf{ct}_{s'}\right) \leftarrow \mathsf{Eval}(\mathsf{pk}, A^{(x)}, \mathsf{ct}_x)$.
- The prover sends $\mathsf{ct}_s, \mathsf{ct}_{s'}$ to the verifier.
- The prover returns $|A\rangle^{(s,s')} = |A_{s,s'}\rangle$.

Verifier's decryption phase:

- The verifier computes $s \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_s)$ and $s' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_{s'})$.
- The verifier returns (A, s, s').

The attack. This protocol achieves correctness, as honest parties receive a coset state's description, and the corresponding coset state respectively.[2] This protocol unfortunately does not preserve the direct product hardness property, and hence does not preserve the monogamy-of-entanglement property either. We present a simple attack that allows a malicious prover to win the semi-quantum version of direct product hardness game with this protocol. When receiving $(A^{(x)}, \mathsf{ct}_x)$, the prover generates two circuits, C_y and $\mathsf{C}'_{y'}$ — parametrized by random bitstrings $y, y' \leftarrow\!\!\$\ \{0,1\}^{n/2}$ — that, on input A, return Ay and $A^\perp y$ respectively. Evaluating $(A^{(x)}, \mathsf{ct}_x)$ on such circuits yields a vector $v = Ay + s$ for the first circuit, and a vector $w = A^\perp y' + s'$ for the second, where s and s' are the one-time pad's keys. The evaluation also yields the encryption of s and s', that the prover can send to the verifier. The verifier then obtain the coset state's description (A, s, s'), and the prover holds $v \in A + s$, and $w \in A^\perp + s'$, allowing them to win the direct product hardness with probability 1.

This attack is possible because the verifier does not check whether the prover honestly perform the evaluation, and actually, we do not have any way of verifying classically this evaluation. Interestingly however, the prover does not have the state $|A_{s,s'}\rangle$ in their registers when applying the cheating strategy, and actually they cannot have information on it due to the semantic security of the quantum fully homomorphic encryption scheme. Based on this observation, our idea is to construct and use a *self-testing protocol* for coset states, whose role is to ensure that the prover does have a coset state corresponding to the vectors they send to the verifier in their registers. Our hope is that no malicious prover can cheat in the resulting protocol, while succeeding in the self-testing protocol.

[2]Note that, although the parties do not output obfuscated membership programs in this protocol, it is easy to extend it, simply by letting the verifier generate the programs and send them to the prover.

6.3 Self-Testing Protocol for BB84 States

Our goal is to construct a self-testing protocol for coset states, and to combine it with the "unsecure" remote preparation protocol of the previous section to achieve "secure" remote coset state preparation. Gheorghiu and Vidick (2019) proposed a self-testing protocol for a single BB84 state, and Gheorghiu, Metger, and Poremba (2022) extended it to handle a polynomial number of BB84 states in tensor product. We described in Section 4.2.2 that coset states and BB84 states share many similarities. In particular, an n-long BB84 state is actually an n-long coset state. Our idea is then to use the protocols mentioned above as starting point. In this section, we informally describe the protocols of Gheorghiu and Vidick (2019) and Gheorghiu, Metger, and Poremba (2022).

6.3.1 Extended Noisy Trapdoor Claw-Free Function

The main ingredient of these protocols is the extended noisy trapdoor claw-free functions. This consists in two indistinguishable families of functions: a family of noisy trapdoor claw-free functions and a family of trapdoor injective functions — we will simply write claw-free functions and injective functions in the following. We provide an informal presentation of these families, and refer the curious reader to Mahadev (2018, Section 4) for a detailed definition.

Claw-free functions. A family $\mathcal{F}$ of claw-free functions consists in pairs of injective functions (f_0, f_1), both with domain $\mathcal{X}$ and codomain $\mathcal{Y}$. There must exist an efficient sampling procedure, that samples a pair of functions from $\mathcal{F}$, along with a trapdoor for these functions, that allows to efficiently invert them through a procedure $\mathsf{Inv}_{\mathcal{F}}$. That is, for all triples (f_0, f_1, t), all bits b, and all images y in the support of f_b:

$$\mathsf{Inv}_{\mathcal{F}}(t, b, y) = x_b \text{ such that } f_b(x_b) = y$$

In the following, we write $(f_0, f_1, t) \leftarrow\!\!\$\ \mathcal{F}$ to denote an execution of this sampling procedure. Whenever we do not need the trapdoor, we omit it and simply write $(f_0, f_1) \leftarrow\!\!\$\ \mathcal{F}$. For each pair, f_0 and f_1 have the same support, meaning that every image y in this support has exactly one preimage x_0 for f_0, and one preimage x_1 for f_1. This pair (x_0, x_1) is called a claw for y. Such family is claw-free if it is not possible to efficiently come up with such a claw, that is:

$$\Pr\left[f_0(x_0) = f_1(x_1) \ : \ \begin{array}{l}(x_0, x_1) \leftarrow \mathcal{A}(f) \\ f = (f_0, f_1) \leftarrow\!\!\$\ \mathcal{F}\end{array}\right] = \mathsf{negl}(\lambda)$$

A family of claw-free functions must also satisfy an adaptive hardcore bit property: it must not be possible to efficiently find (x_0, d, c) or (x_1, d, c) satisfying the equation $d \cdot (x_0 \oplus x_1) = c$ — where the x_0 and x_1 of the equation form a claw.

$$\Pr\left[d \cdot (x_0 \oplus x_1) = c \ : \ \begin{array}{l}x_{1-b} = f_{1-b}^{-1}(f_b(x_b)) \\ (b, x_b, d, c) \leftarrow \mathcal{A}(f) \\ f = (f_0, f_1) \leftarrow\!\!\$\ \mathcal{F}\end{array}\right] = \mathsf{negl}(\lambda)$$

Injective functions. The main difference between claw-free functions and injective functions is that the pairs of functions in the latter have disjoint supports. A family $\mathcal{G}$ of injective functions consists of pairs of injective functions (g_0, g_1), both with domain $\mathcal{X}$

and codomain $\mathcal{Y}$. There also must exist an efficient sampling procedure, that samples a pair of functions from $\mathcal{G}$, along with a trapdoor for these functions. As the supports of g_0 and g_1 are disjoint, the efficient inverting procedure $\mathsf{Inv}_{\mathcal{G}}$ is slightly different from $\mathsf{Inv}_{\mathcal{F}}$. More precisely, for all triples (g_0, g_1, t), and all images y in the support of g_0 or g_1, $\mathsf{Inv}_{\mathcal{G}}(t, y)$ yields the unique pair (b, x_b) such that $f_b(x_b) = y$. Similarly to claw-free functions, we write $(g_0, g_1, t) \leftarrow\!\!\$\ \mathcal{G}$ or simply $(g_0, g_1) \leftarrow\!\!\$\ \mathcal{G}$ to denote an execution of this sampling procedure.

Extended noisy trapdoor claw-free functions. An extended noisy trapdoor claw-free function family is then a pair of claw-free and injective functions families $(\mathcal{F}, \mathcal{G})$ as defined above, associated with efficient procedures to sample either in $\mathcal{F}$ or in $\mathcal{G}$. Furthermore, it must be computationally infeasible to tell apart a pair sampled from $\mathcal{F}$, from one sampled from $\mathcal{G}$.

$$\{(f_0, f_1) \ : \ (f_0, f_1) \leftarrow\!\!\$\ \mathcal{F}\} \approx_c \{(g_0, g_1) \ : \ (g_0, g_1) \leftarrow\!\!\$\ \mathcal{G}\}$$

Mahadev (2018) showed how to construct such a family assuming the hardness of the LWE problem.

6.3.2 Committing Using Claw-Free and Injective Functions

In the remote state preparation protocol we describe in the next subsection, the prover is asked to pass a series of tests, in which they have to "commit" on a qubit, and measure it and a specific way. We describe in this subsection what committing on a qubit means in this context, and what interesting measurements the prover can perform on a committed qubit.

Claw-free functions. Assume the prover is given a pair of claw-free functions (f_0, f_1). Consider the unitary operator U defined as $\mathsf{U}\,|b\rangle\,|x\rangle\,|y\rangle = |b\rangle\,|x\rangle\,|y \oplus f_b(x)\rangle$ — we call the three registers the bit, preimage, and image registers respectively. Running this operator with $|+\rangle$ as the bit register, $|+\cdots+\rangle$ as the preimage one, and $|0\rangle$ as the image one, results in a superposition over the images in the supports of f_0 and f_1 in the image register, and the associated preimages in the preimage one. If the prover performs this operation, and then measures the image register and obtains y as the outcome, the bit and preimage registers collapse to

$$|0\rangle\,|x_0\rangle + |1\rangle\,|x_1\rangle$$

where (x_0, x_1) is a claw for y.[3] This procedure is called a commitment using the claw-free functions (f_0, f_1).

Measuring the collapsed bit and preimage registers in the computational basis yields a pair (b, x_b) where $f_b(x_b) = y$, while measuring them in the Hadamard basis yields $(d \cdot (x_0 \oplus x_1), d)$. To see that, note that measuring the preimage register in the Hadamard basis, and obtaining d collapses the bit register to the following state:

$$\begin{aligned}
&(-1)^{d \cdot x_0}\,|0\rangle + (-1)^{d \cdot x_1}\,|1\rangle \\
=&(-1)^{d \cdot x_0}\left(|0\rangle + (-1)^{d \cdot (x_0 \oplus x_1)}\,|1\rangle\right) \\
=&\pm\,\mathsf{H}\,|d \cdot (x_0 \oplus x_1)\rangle
\end{aligned}$$

[3] For sake of readability, we omit the normalization factor in this part.

Note that while the prover can choose to learn either (b, x_b) or $(d \cdot (x_0 \oplus x_1), d)$, they cannot learn both due to the adaptive hardcore bit property of claw-free functions.

Injective functions. The commitment using a pair of injective functions (g_0, g_1) is done in the same way as with the claw-free ones. After measuring of the preimage register, the two first registers collapse to

$$|b\rangle\,|x_b\rangle$$

such that $g_b(x_b) = y$.

Measuring this state in the computational basis yields a pair (b, x_b) such that $f_b(x_b) = y$. Note that, as the two registers are in a tensor product, measuring the bit register in the computational basis will always return this b, regardless on which measurement we perform on the preimage register. This will be handy in the following.

6.3.3 Self-Testing and Remote Preparation of BB84 States

The remote state preparation protocols of Gheorghiu and Vidick (2019) and Gheorghiu, Metger, and Poremba (2022) both use the same idea: executing a number of self-testing protocols, before finally asking the prover to prepare their state. In a nutshell, a self-testing protocol is an interactive protocol in which, after having exchanged a series of classical messages (some tests) with a prover, a verifier is convinced that the prover's register at the beginning of the protocol contains a particular quantum state. The verifier is in fact convinced that the prover has this state, *up to a local isometry*, that is, an (invertible) operation acting only on this state. This is actually impossible to prevent, as even if the verifier sends the quantum state directly to the prover, the latter can always perform such isometry and still succeeds in every subsequent tests. Note also that, in such a protocol, the prover's state is destroyed in the process.

In the protocol of Gheorghiu and Vidick (2019), the verifier instructs a prover on how to construct a BB84 state in a blind and verifiable way. The idea of this protocol is to run a series of *test rounds*, that act as self-testing and whose purpose is to ensure that the prover behaves honestly. Once the verifier is convinced that the prover behaves honestly, they run a *preparation round*, in the end of which the prover holds a BB84 state, and the verifier holds its description. Importantly, a malicious prover cannot distinguish a test round from a preparation round, meaning that they cannot know when to behave honestly to pass the tests, and when to cheat in the state preparation. We describe these two types of rounds in the following.

Preparation round. In a preparation round, the verifier starts by sampling a random bit θ, then samples a pair of injective functions, or a pair of claw-free functions, with their respective trapdoor, depending on whether $\theta = 0$ or 1. The verifier then sends this pair to the prover and asks them to perform the commitment mentioned above, and to send them the corresponding image y. Note that the prover does not have to know — and actually cannot know — whether they receive an injective pair or a claw-free pair to perform the commitment. The verifier then asks the prover to measure the preimage register in the Hadamard basis, and to send them the outcome d. The verifier inverts y with the trapdoor, obtaining (b, x_b) or (x_0, x_1) depending on θ, then computes $v = b$ if $\theta = 0$, or $v = d \cdot (x_0 \oplus x_1)$ if $\theta = 1$. Finally, the prover returns their bit register as the prepared state, and the verifier returns (v, θ) as its description. From the previous

subsection, this protocol has correctness, meaning that an honest prover's bit register is $|v^\theta\rangle$ (up to a global phase).

Test round. A test round begins in the same way as a preparation one: the verifier sends a random pair of claw-free or injective functions to the prover, who performs a commitment and sends back y. At this point, two cases can happen, either the verifier decides that the round is a *preimage round*, in which case they ask the prover to measure their bit and preimage registers in the computational basis, and to send the outcomes (b, d). Or they decide it is a *Hadamard round*, in which case they ask the prover to measure the preimage register in the Hadamard basis and to send the outcome d as before, and then to measure the bit register in the computational basis if $\theta = 0$, or in the Hadamard one if $\theta = 1$, and to send the outcome.[4] The verifier finally performs a consistency check depending on the type of round they picked. In the preimage round case, the verifier checks whether $f_b(x_b) = y$, where x_b is obtained by inverting y. In the Hadamard one, the verifier checks the same if $\theta = 0$, and if $\theta = 1$, the verifier checks whether $d \cdot (x_0 \oplus x_1) = v$, where (x_0, x_1) is obtained by inverting y. These consistency checks ensure that the prover is not trying to cheat in the protocol.

Multi qubits test round. As we need it later on, we formally describe below an n-fold parallel repetition of the test round described above, using either only claw-free pairs, or injective pairs for all n instances. Each instance of this protocol is slightly different from the single-qubit test-round protocol. Indeed, we assume that an honest prover holds n qubits $|\pm\rangle$ in the beginning of the protocol, to be used as their bit registers, instead of $|+\rangle$ qubits for the single qubit test round. More precisely, the n qubits are $\mathsf{H}|s'\rangle = \bigotimes_{i=1}^{n} \mathsf{H}|s'_i\rangle$, and s' is given as input to the verifier. These modifications are needed for our final remote coset state preparation protocol, and do not modify the security (we speak here of *rigidity*) of the self-testing protocol. Informally, this means the following. Consider an efficient prover passing the test in this protocol with probability close to 1 Then, this prover's behavior is close to an honest prover's behavior. We leave the proof of the rigidity of this protocol in Appendix B.3.2.

This protocol's description assume honest verifier and prover.

Protocol 2: Test Round for BB84 States

This is an interactive protocol between a PPT verifier, and a QPT prover.

Parameters: A security parameter λ, an integer $n = \mathsf{poly}(\lambda)$. A family of extended noisy trapdoor claw-free functions $(\mathcal{F}, \mathcal{G})$. A unitary U, parametrized by functions (h_0, h_1), defined by $\mathsf{U}|b\rangle|x\rangle|y\rangle = |b\rangle|x\rangle|y \oplus h_b(x)\rangle$.

Inputs: The verifier is given as input a one-time pad key pair $(s, s') \in \mathbb{F}_n^2 \times \mathbb{F}_n^2$. The prover initially holds a quantum state $|\phi\rangle$. For an honest prover, $|\phi\rangle = \mathsf{H}|s'\rangle$.

Setup:

- The verifier samples a bit $\theta \leftarrow\!\!\$\ \{0, 1\}$.
- The verifier samples n pairs of functions and the corresponding trapdoors $(h_{0,i}, h_{1,i}, t_i)$, from $\mathcal{F}$ if $\theta = 0$, or from $\mathcal{G}$ if $\theta = 1$.

[4] In practice, the verifier decides to perform a Hadamard round or a test round at random (with probability 1/2 for each round type).

– The verifier sends $((h_{0,i}, h_{1,i}))_{i\in[\![1,n]\!]}$ to the prover.

Commitment:

– For $i \in [\![1, n]\!]$:

∗ The prover applies U (parametrized by $(h_{0,i}, h_{1,i})$) to $\phi_i \otimes |+\cdots+\rangle\langle\cdot| \otimes |0\ldots0\rangle\langle\cdot|$.[a] [b]

∗ The prover measures the third register. Let y_i denotes the outcome, and ϕ'_i, ψ'_i be the first and second registers after the measurement.

– The prover sends $y_1, \ldots, y_n$ to the verifier.

Round type:

– With probability 1/2, the verifier decides to perform a preimage round, and with probability 1/2, the verifier decides to perform a Hadamard round.

– The verifier sends the decided round type ("preimage round" or "Hadamard round") to the prover.

(Preimage round) - Measurements:

– The prover measures all $\phi'_1, \ldots, \phi'_n$ registers, and all $\psi'_1, \ldots, \psi'_1$ registers in the computational basis. Let $b_1, \ldots, b_n$, and $d_1, \ldots, d_n$, denote the corresponding measurement outcomes.

– The prover sends these outcomes $b_1, \ldots, b_n$, and $d_1, \ldots, d_n$, to the verifier.

(Preimage round) - Consistency check:

– For $i \in [\![1, n]\!]$: The verifier checks whether $h_{b_i,i}(d_i) = y_i$.

– The verifier aborts if any of the checks fail.

(Hadamard round) - Preimage registers measurements:

– The prover measures all $\psi'_1, \ldots, \psi'_n$ registers in the Hadamard basis. Let $d_1, \ldots, d_n$ denote the corresponding outcomes.

– The prover sends $d_1, \ldots, d_n$ to the verifier.

– Let $\phi''_1, \ldots, \phi''_n$ be the bit registers after the measurements.

(Hadamard round) - Basis disclosure: The verifier sends θ to the prover.

(Hadamard round) - Bit registers measurements:

– The prover measures all bit registers $\phi''_1, \ldots, \phi''_n$ in the computational basis if $\theta = 0$, or in the Hadamard basis if $\theta = 1$. In both case, let $v_1, \ldots, v_n$ denote the corresponding outcomes.

– The prover sends $v_1, \ldots, v_n$ to the verifier.

(Hadamard round) - Consistency check:

- If $\theta = 0$:
  - For $i \in [\![1, n]\!]$:
    - The verifier computes $(b'_i, x_{b'_i,i}) \leftarrow \mathsf{Inv}_{\mathcal{G}}(t_i, y_i)$.
    - The verifier checks whether $b_i = b'_i$.
- If $\theta = 1$:
  - For $i \in [\![1, n]\!]$:
    - The verifier computes

$$x_{0,i} \leftarrow \mathsf{Inv}(t_i, 0, y_i),$$

$$x_{1,i} \leftarrow \mathsf{Inv}(t_i, 1, y_i).$$

    - The verifier checks whether $v_i \oplus s'_i = d_i \cdot (x_{0,i} \oplus x_{1,i})$.
- The verifier returns aborts if any of the checks fails.

[a] We use the shorthand $|x\rangle\langle\cdot|$ to denote $|x\rangle\langle x|$.
[b] ϕ_i denotes the i-th qubit of $|\phi\rangle$

We show in the following section how to adapt this protocol to handle coset states.

6.4 Self-Testing Coset States

In the protocol we described above, an honest prover initially holds a state $|\pm \cdots \pm\rangle$ in their register, and every qubit of this state is used as the bit register in a (slightly modified) single qubit test round. We describe below what happens if we use an n-long coset state as bit registers. More precisely, the i-th qubit of the coset state is used as the bit register of the i-th commitment, and we measure all the bit registers in the same basis (that is, we use either only claw-free functions, or only injective functions for all bit registers).

To see that, first remark that any i-th qubit of a coset state can be written — up to reordering the registers — as

$$\alpha_0 |0\rangle |\psi_i^0\rangle + \alpha_1 |1\rangle |\psi_i^1\rangle$$

Then, committing using this i-th qubit as the bit register (we still use $|+\cdots+\rangle$ as the preimage register, and $|0\rangle$ as the image register) yields an image y and collapses the whole system on

$$\begin{cases} |b\rangle |\psi_i^b\rangle |x_b\rangle & \text{if the function pair is an injective one or} \\ \alpha_0 |0\rangle |\psi_i^0\rangle |x_0\rangle + \alpha_1 |1\rangle |\psi_i^1\rangle |x_1\rangle & \text{if it is a claw-free one} \end{cases}$$

where in the first line, (b, x_b) is the inverse of y, and in the second one, (x_0, x_1) are the claw for y.

Now, measuring the preimage register in the Hadamard basis (we only consider

Hadamard round when using coset states) yields some d, and collapses the system to

$$\begin{cases} & |b\rangle\,|\psi_i^b\rangle & \text{if the function pair is an injective one or} \\ & \alpha_0\,|0\rangle\,|\psi_i^0\rangle + (-1)^{d\cdot(x_0\oplus x_1)}\alpha_1\,|1\rangle\,|\psi_i^1\rangle & \\ = & \mathsf{H}\,(\alpha_0\,|0\oplus u_i\rangle\,|\psi_i^0\rangle + \alpha_1\,|1\oplus u_i\rangle\,|\psi_i^1\rangle) & \text{if it is a claw-free one} \end{cases} \tag{6.1}$$

where u_i denotes $d\cdot(x_0\oplus x_1)$.

An important observation is that the state $|\psi_i^b\rangle$ is the superposition of all $n-1$ long vectors x such that the vector $(x_1\dots x_{i-1}\|b\|x_i\dots x_n)$ belongs to $A+s$. A verifier receiving the measurement outcome $(v_i)_i$ of all the bit registers, measured in the computational basis when the functions are injective, or in the Hadamard basis when they are claw-free, can then perform the following checks. Assume a measurement is performed on all the bit registers — in the computational basis when the functions are injective, or in the Hadamard basis when they are claw-free — yielding a bit v_i for each register i. Let $v=(v_1,v_2,\dots,v_n)$. When the measurements are done in the Hadamard basis, we denote u as $(u_1,u_2,\dots,u_n)$, where each u_i is the $d\cdot(x_0\oplus x_1)$ from Equation (6.1). Then, it is clear from the observation and Equation (6.1) that such a vector v obtained by measuring all the bit registers in the rectilinear basis (which corresponds to the first line of the equation) is in $A+s$, while one obtained by measuring them in the Hadamard basis (which corresponds to the second line) is in $A^\perp+s'+u$.

6.4.1 Test Round for Coset States

From the observations above, we can now define a test round for coset states. This round is similar to the regular one we described in the last section. Assume that the prover holds a coset state $|A_{s,s'}\rangle$, and the verifier its description (A,s,s') at the beginning of the round. In a test round for coset states, the verifier only decides to perform Hadamard round, the interaction between the prover and the verifier are then the same as the interaction in a test round for BB84 states. The only modifications are that the honest prover uses the coset state instead of a series of $|\pm\rangle$ state as bit registers, and that the verifier applies different consistency checks: if the pair of functions is an injective one, they check whether the vector v returned by the prover is in $A+s$, and if the pair is a claw-free one, they compute $u_i=d\cdot(x_0\oplus x_1)$ by inverting the image y sent by the prover, and then check whether $v+u$ is in $A^\perp+s'$. We prove in Appendix B.3.3 that this protocol has rigidity, similarly to protocol 2. That is, any prover passing the test with probability close to 1 must behave similarly to an honest prover.

We formally describe this test round for coset states below. This protocol's description assumes honest verifier and prover.

Protocol 3: Test round for Coset States
This is an interactive protocol between a PPT verifier, and a QPT prover.
Parameters: A security parameter λ, an integer $n=\mathsf{poly}(\lambda)$. A family of extended noisy trapdoor claw-free functions $(\mathcal{F},\mathcal{G})$. A unitary U, parametrized by functions (h_0,h_1), defined by $\mathsf{U}\,|b\rangle\,|x\rangle\,|y\rangle=|b\rangle\,|x\rangle\,|y\oplus h_b(x)\rangle$.
Inputs: The verifier is given an n-long coset state's description (A,s,s'). The prover initially holds a quantum state $|\phi\rangle$. For an honest prover, $|\phi\rangle=|A_{s,s'}\rangle$.

Setup:

- The verifier samples a bit $\theta \leftarrow\!\!\$\ \{0,1\}$.
- The verifier samples n pairs of functions and the corresponding trapdoors $(h_{0,i}, h_{1,i}, t_i)$, from $\mathcal{F}$ if $\theta = 0$, or from $\mathcal{G}$ if $\theta = 1$.
- The verifier sends $((h_{0,i}, h_{1,i}))_{i[\![1,n]\!]}$ to the prover.

Commitment:

- For $i \in [\![1,n]\!]$:
  - Let ϕ_i denotes the i-th qubit of $|A_{s,s'}\rangle$.
  - The prover applies U (parametrized by $(h_{0,i}, h_{1,i})$) to $\phi_i \otimes |+\cdots+\rangle\langle\cdot| \otimes |0\ldots0\rangle\langle\cdot|$.[a]
  - The prover measures the third register. Let y_i denotes the outcome, and ϕ'_i, ψ'_i be the first and second registers after the measurement.
- The prover sends $y_1, \ldots, y_n$ to the verifier.

Round type: The verifier sends "Hadamard round" to the prover.

Preimage registers measurements:

- The prover measures all $\psi'_1, \ldots, \psi'_n$ registers in the Hadamard basis. Let $d_1, \ldots, d_n$ denote the corresponding outcomes.
- The prover sends $d_1, \ldots, d_n$ to the verifier.
- Let $\phi''_1, \ldots, \phi''_n$ be the bit registers after the measurements.

Basis disclosure: The verifier sends θ to the prover.

Bit registers measurements:

- The prover measures all bit registers $\phi''_1, \ldots, \phi''_n$ in the computational basis if $\theta = 0$, or in the Hadamard basis if $\theta = 1$. In both cases, let $v_1, \ldots, v_n$ denote the corresponding outcomes.
- The prover sends $v_1, \ldots, v_n$ to the verifier.

Consistency check:

- Let v denotes the vector $(v_1, \ldots, v_n)$.
- If $\theta = 0$: The verifier checks that $v \in A + s$.
- If $\theta = 1$:
  - For $i \in [\![1,n]\!]$: The verifier computes

$$x_{0,i} \leftarrow \mathsf{Inv}_{\mathcal{F}}(t_i, 0, y_i),$$
$$x_{1,i} \leftarrow \mathsf{Inv}(t_i, 1, y_i),$$
$$u_i = d_i \cdot (x_{0,i} \oplus x_{1,i})$$

.

* Let u denotes the vector $(u_1, \ldots, u_n)$.
* The verifier checks that $v + u \in A^{\perp} + s'$.

– If the check does not pass, the verifier aborts.

[a]We use the shorthand $|x\rangle\langle\cdot|$ to denote $|x\rangle\langle x|$.

6.4.2 Self-Testing Protocol for Coset States

Given this test round protocol for coset states, and the multi qubits one for BB84 states that we described in the previous section, we define a self-testing protocol for coset states. In a nutshell, the verifier in this protocol runs test rounds for BB84 states, until they are convinced that the prover does not cheat. Once the verifier is convinced, they run one test round for coset states. We prove in Appendix B.3.4 a soundness statement for this protocol. That is a prover succeeding in the protocol with probability close to 1 (that is, such that the verifier never aborts) must hold a state close to $|A_{s,s'}\rangle$, up to a local isometry, in their register at the beginning of the last test round. We describe formally the behavior of honest verifier and prover below.

Protocol 4: Self-Testing of Coset States

This is an interactive protocol between a PPT verifier, and a QPT prover.

Parameters: A security parameter λ, integers $n, M = \mathsf{poly}(\lambda)$.

Inputs: The verifier is given as input an n-long coset state's description (A, s, s'); M^2 pairs of one-time pad keys $(s_i, s'_i)_{i \in [\![1, M^2]\!]}$, where each $s_i, s'_i \in \{0,1\}^n$; and an integer $j \in [\![1, M^2]\!]$, corresponding to the position of the coset state in an honest prover's register. The prover initially holds a quantum state $|\phi\rangle$. For an honest prover, $|\phi\rangle = \bigotimes_{i=1}^{M^2+1} |\phi\rangle_i = \bigotimes_{i=1}^{j-1} \mathsf{H}\,|s'_i\rangle \otimes |A_{s,s'}\rangle \otimes \bigotimes_{i=j}^{M^2} \mathsf{H}\,|s'_i\rangle$.

- The verifier samples $B \leftarrow\!\$\, [\![M, M^2 - 1]\!]$.
- The verifier and the prover perform B instances of protocol 2. For each instance, the verifier samples an index $i \in [\![1, M^2 + 1]\!] \setminus \{j\}$ that has not been used for a previous instance, uses (s_i, s'_i) as input, and tells the prover to use $|\phi\rangle_i$ as input. The verifier aborts if any of these instances aborts. Otherwise, they continue.
- The verifier and the prover perform an instance of protocol 3. The verifier uses (A, s, s') as input, and tells the prover to use $|\phi\rangle_j$ as input. The prover aborts if this execution aborts.

The reason why the verifier instructs the prover which part of their state they must use as input is that in the final protocol, the prover blindly prepares quantum states $|\phi\rangle$ of the form described above: they know that each of these states consists of M^2 $|\pm \cdots \pm\rangle$ states, and one coset state $|A_{s,s'}\rangle$, but do not know the position of the coset state.

On the soundness of the self-testing. Remark that the self-testing protocol described above only has inverse-polynomial soundness. This is because a malicious prover can always make a random guess on which of their registers will be used for the preparation protocol, and not prepare this register's state honestly. As there are a polynomial number of registers, this behavior is unnoticed with inverse-polynomial probability. Remark that the same argument also applies to the self-testing protocols of Gheorghiu and Vidick

(2019) and Gheorghiu, Metger, and Poremba (2022), which therefore also only satisfy inverse-polynomial soundness. In our case, our ultimate goal is to construct a remote coset state preparation protocol preserving direct product hardness and monogamy-of-entanglement properties. We show in the following sections that a self-testing protocol with inverse-polynomial security is enough.

6.5 Remote Coset State Preparation

We show in this section how to use Shmueli's idea of "unsecure" remote coset state preparation (protocol 1) to make the prover prepare $|\pm\cdots\pm\rangle$ states and coset states without knowing which state they are preparing. We can then plug this preparation to protocol 4 to ensure that the prover does not know whether they are performing test rounds for BB84 states, or for coset states.

6.5.1 Hiding a Coset State Round Among BB84 Rounds

Consider the following quantum circuit C, defined as follows. On input any non-zero matrix A, describing a subspace A, $\mathsf{C}(A)$ returns the subspace state $|A\rangle$. On input a matrix filled with zeros (denoted simply as 0 in the following), $\mathsf{C}(0)$ returns n qubits $|+\rangle$. As already mentioned, running C on an encryption of A with quantum fully homomorphic encryption yields a coset state $|A_{s,s'}\rangle$, together with encryptions of s and s'. Running this program on an encryption of 0 yields (up to a global phase) $\mathsf{H}\,|s'\rangle = \bigotimes_{i=1}^{n} \mathsf{H}\,|s'_i\rangle$: n qubits in either $|+\rangle$ or $|-\rangle$ state, which we write $|\pm\cdots\pm\rangle$ in the following.

Following this observation, and the fact that the correctness and soundness of the self-testing protocol of Gheorghiu, Metger, and Poremba (2022) are preserved if the prover uses $|\pm\cdots\pm\rangle$ instead of $|+\cdots+\rangle$ as bit registers, our idea is to instruct the prover to construct their own input for the test rounds. Concretely, to run a test round in the BB84 version, the verifier first samples a pair of secret and public keys of a quantum fully homomorphic encryption scheme, then sends the public key, along with the one-time padded encryption of 0 to the prover. The honest prover then homomorphically evaluates C on this encryption, and uses the outcome state as input in protocol 2. To run a test round in the coset states version, the verifier does the same, except that they send an encryption of a random subspace A, of dimension $n/2$, instead of an encryption of 0. It is important to notice that, when doing the tests this way, a malicious prover cannot tell whether they are performing a BB84 version, or a coset state version. This is implied by the semantic security of the quantum fully homomorphic encryption scheme, and allows us to carry the security of the self-testing protocol of Gheorghiu, Metger, and Poremba (2022) to our settings.

6.5.2 Remote Coset States Preparation for Coset States

We present below our final remote coset state preparation protocol. In this protocol, the verifier makes the prover prepare κM^2 states of the form $|\pm\cdots\pm\rangle$, and 2κ coset states. The verifier and the prover then execute κ instances of protocol 4 with for each instance, M^2 $|\pm\cdots\pm\rangle$ states, and one coset state, picked at random by the verifier. Once all tests pass and these input states are consumed, the verifier tells the prover to output the remaining κ coset states, and outputs their descriptions. This κ among 2κ cut-and-choose protocol

allows us to leverage the quantum sample-and-estimate formalism of Bouman and Fehr (2010) to prove that a malicious prover passing all tests must have prepared *at least one* coset state. This result however, only holds with probability inverse polynomially close to 1, and not probability negligibly close to 1 as we would like. The reason is that the underlying self-testing protocol (protocol 4) only has inverse polynomial soundness. We prove that this does not prevent our remote coset state preparation protocol to preserve the direct product hardness and monogamy-of-entanglement properties. Intuitively, this protocol allows a malicious prover to cheat without being caught (up to inverse polynomial probability) and prepare a state that differs from a coset state, but this cheating strategy does not provide them useful information for the direct product hardness, and monogamy-of-entanglement games.

Protocol 5: Remote Coset State Preparation

Let λ be a security parameter, $\kappa = \lambda$, and $n, M = \mathsf{poly}(\lambda)$. Let $\langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval} \rangle$ be a quantum fully homomorphic encryption protocol. This is an interactive protocol between a PPT verifier, and a QPT prover.

Setup:

(BB84 instances)

- The verifier samples κM^2 pairs of keys $(\mathsf{sk}^i_j, \mathsf{pk}^i_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$ for every $i \in [\![1, \kappa]\!]$, $j \in [\![1, M^2]\!]$.
- The verifier samples κM^2 pairs of one-time pad keys (x^i_j, z^i_j), where $x^i_j, z^i_j \in \{0,1\}^{n^2/2}$, for $i \in [\![1, \kappa]\!]$, and $j \in [\![1, M^2]\!]$.
- The verifier computes $\mathsf{ct}_{x^i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}^i, x^i_j)$ and $\mathsf{ct}_{z^i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}^i, z^i_j)$ for every $i \in [\![1, \kappa]\!]$, $j \in [\![1, M^2]\!]$.
- The verifier computes $0^{i,(x^i_j, z^i_j)}_j$: the one-time pad encryption of the 0 matrix in $\mathbb{F}_2^n$ with keys x^i_j and z^i_j for every $i \in [\![1, \kappa]\!]$, $j \in [\![1, M^2]\!]$.

(Coset instances)

- The verifier samples 2κ pairs of keys $(\mathsf{sk}^i, \mathsf{pk}^i) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- The verifier samples 2κ subspaces A^i of dimension $n/2$, and 2κ pairs of one-time pad keys (x^i, z^i).
- The verifier computes $\mathsf{ct}_{x^i} \leftarrow \mathsf{Enc}(\mathsf{pk}^i, x^i)$ and $\mathsf{ct}_{z^i} \leftarrow \mathsf{Enc}(\mathsf{pk}^i, z^i)$ for every $i \in [\![1, 2\kappa]\!]$.
- The verifier computes $A^{i,(x^i, z^i)}$: the one-time pad encryption of A^i with keys x^i and z^i, for every $i \in [\![1, 2\kappa]\!]$.

State preparation:

- The verifier shuffles all the tuples

$$\left\{\left(\mathsf{pk}^i, A^{i,(x^i,z^i)}, \mathsf{ct}_{x^i}, \mathsf{ct}_{z^i}\right)\right\}_{i \in [\![1,2\kappa]\!]} \cup \left\{\left(\mathsf{pk}^i_j, 0^{i,(x^i_j, z^i_j)}_j, \mathsf{ct}_{x^i_j}, \mathsf{ct}'_{z^i_j}\right)\right\}_{i \in [\![1,\kappa]\!], j \in [\![M^2]\!]}$$

and sends them to the prover.

– Let C the quantum circuit defined as $\mathsf{C}(0) = |+\cdots+\rangle$ and $\mathsf{C}(A) = |A\rangle$ for $A \neq 0$.

– The prover computes

$$\left(|A_{i,s_i,s'_i}\rangle, \mathsf{ct}_{s^i}, \mathsf{ct}_{s'^i}\right) \leftarrow \mathsf{Eval}(\mathsf{pk}^i, A^{i,(x^i,z^i)}, \mathsf{ct}_{x^i}, \mathsf{ct}_{z^i}, \mathsf{C})$$

$$\left(\mathsf{H}\,|s'^i_j\rangle, \mathsf{ct}_{s^i_j}, \mathsf{ct}_{s'^i_j}\right) \leftarrow \mathsf{Eval}(\mathsf{pk}^i_j, 0_j^{i,(x^i_j,z^i_j)}, \mathsf{ct}_{x^i_j}, \mathsf{ct}'_{z^i_j}, \mathsf{C})$$

for all $i \in [\![1, 2\kappa]\!], j \in [\![1, M^2]\!]$.[5]

– The prover sends all the $(\mathsf{ct}_{s^i}, \mathsf{ct}_{s'^i})$ and all the $(\mathsf{ct}_{s^i_j}, \mathsf{ct}_{s'^i_j})$ to the verifier.

– The verifier computes

$$s^i \leftarrow \mathsf{Dec}(\mathsf{sk}^i, \mathsf{ct}_{s^i}),$$

$$s'^i \leftarrow \mathsf{Dec}(\mathsf{sk}^i, \mathsf{ct}_{s'^i}),$$

$$s^i_j \leftarrow \mathsf{Dec}(\mathsf{sk}^i_j, \mathsf{ct}_{s^i_j}),$$

$$s'^i_j \leftarrow \mathsf{Dec}(\mathsf{sk}^i_j, \mathsf{ct}_{s'^i_j})$$

for every $i \in [\![1, \kappa]\!]$, $j \in [\![1, M^2]\!]$.

Self-testing:

– For $i \in [\![\kappa]\!]$: The verifier and the prover run protocol 4 with (A^i, s^i, s'^i) and $\{(s^i_j, s'^i_j)\}_{j\in[\![1,M^2]\!]}$ as verifier's input, and $|\phi\rangle = |A_{i,s_i,s'_i}\rangle \otimes \bigotimes_{j=1}^{M^2} \mathsf{H}\,|s'^i_j\rangle$ (not necessarily in this order) as prover's input.[6] If any instance aborts, the verifier aborts.

Output:

– For $i \in [\![\kappa+1, 2\kappa]\!]$: The verifier prepares the obfuscated membership programs $\widehat{\mathsf{P}}_{A^i+s^i}, \widehat{\mathsf{P}}_{A^{(i)\perp}+s'^i}$, then send them to the prover.

– The verifier outputs $\{(A^i, s^i, s'^i)\}_{i\in[\![\kappa+1,2\kappa]\!]}$.

– The prover outputs $\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![\kappa+1,2\kappa]\!]}$.

– Both also output $\{\widehat{\mathsf{P}}_{A^i+s^i}, \widehat{\mathsf{P}}_{A^{(i)\perp}+s'^i}\}_{i\in[\![\kappa+1,2\kappa]\!]}$.

[5] Note that, because of the shuffling, the prover does not know which instances are BB84 ones, and which are coset states ones.

We prove in Appendix B.3.5 that the initial state of any prover succeeding in the protocol with probability close to 1 must be inverse-polynomially close to $|A_{s,s'}\rangle$. While this inverse-polynomially closeness is not usually considered secure enough, we show that this is does not prevent our protocol to preserve the monogamy-of-entanglement and direct product hardness properties of coset states.

Theorem 24 (Informal)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, and quantum hardness of the LWE problem. Then

protocol 5 has correctness, and it preserves direct product hardness and monogamy-of-entanglement properties.

We present below the proof idea for the monogamy-of-entanglement. The same proof structure can be applied for direct product hardness. The first argument is that the κ among 2κ sample-and-estimate allows us to say that, with inverse polynomial probability, at least one of the registers of a prover passing all tests really contains a coset state $|A_{s,s'}\rangle$. Then, consider the following hybrid, in which the verifier is now quantum. The verifier asks the prover, at the end of the remote coset state preparation, to send them one of them register (the index is sampled at random by the verifier). The verifier then checks that the state sent by the prover is the expected coset state (they can do it efficiently with the description of the coset state), and returns it if it is. If not, they abort the protocol. The rest of the game consists in playing the monogamy-of-entanglement game with this single state (the $\kappa - 1$ others are ignored). As mentioned above, one of the state of the prover is $|A_{s,s'}\rangle$ with inverse polynomial probability q. As this state is chosen by the verifier in the hybrid game with probability $1/\kappa$, a triple of adversaries winning the game with non-negligible probability p also wins this hybrid with probability at least $\frac{pq}{\kappa}$, which is also non-negligible.

It remains to show that this hybrid cannot be won with non-negligible probability. This is done by first removing the secret key of the quantum fully homomorphic encryption, using both subspace obfuscation techniques of Coladangelo, Liu, Liu, and Zhandry (2021), and complexity leveraging of Shmueli (2022a). Then, we construct a last hybrid, in which the verifier, instead of sending back the coset state to the prover, samples another coset state at random and sends it. The monogamy-of-entanglement game is then played with this new coset state, and we leverage the monogamy-of-entanglement property to state that no adversaries can win this game with non-negligible probability, concluding the proof.

We refer the interested reader to Appendix B.4 for a complete proof.

6.6 Semi-Quantum Copy-Protection

Recall that our goal is to construct semi-quantum constructions of unclonable primitives. We show in this section how to use our remote coset state preparation protocol in a plug-and-play manner to construct a semi-quantum version of an unclonable primitive given a regular construction of this primitive. As an example, we show how to construct a semi-quantum copy-protection scheme from construction 14, using construction 17 as backbone underlying copy-protection of pseudorandom functions. We only consider the non-colliding challenge distribution for the sake of simplicity, but note that the definitions, construction, and proofs below can easily be adapted to other distributions. The difference between a regular, and a semi-quantum copy-protection scheme, is that in the latter version, the vendor is classical, and instructs the client on how to construct their protected quantum program. The anti-piracy of regular copy-protection must be preserved, in the sense that we define a semi-quantum version of the regular anti-piracy game, and no malicious client should have any advantage in this semi-quantum version, compared to the regular one.

6.6.1 Semi-Quantum Copy-Protection of Point Functions

We formally define in this section semi-quantum copy-protection scheme for point functions.

Definition 39 (Semi-Quantum Copy-Protection of Point Functions)**.** A semi-quantum copy-protection scheme of a point functions family $\mathcal{F} = \{\mathsf{PF}_y\}_{y\in\{0,1\}^n}$ is composed of an interactive protocol $\mathsf{Protect}$ and an algorithm Eval defined in the following way:

- $|\text{✿}\rangle \leftarrow \mathsf{Protect}\langle\mathcal{V}(1^\lambda, y), \mathcal{P}(1^\lambda)\rangle$. The protection interactive protocol is between a PPT vendor $\mathcal{V}$ and a QPT client $\mathcal{P}$. The vendor takes as input a security parameter, and a point $y \in \{0,1\}^n$. The client takes as input a security parameter, and outputs a quantum encoding $|\text{✿}\rangle$ of PF_y.
- $z \leftarrow \mathsf{Eval}(|\text{✿}\rangle, x)$. The evaluation algorithm takes as input a quantum encoding $|\text{✿}\rangle$, and an input x in $\{0,1\}^n$, and outputs a bit z.

In addition, a semi-quantum copy-protection scheme of point functions must satisfy the following properties.

Correctness. The correctness of a semi-quantum copy-protection scheme of point functions is adapted from the correctness definition of a regular scheme. That is, for all $y, x \in \{0,1\}^n$, and for all honest PPT vendor $\mathcal{V}$ and QPT client $\mathcal{P}$:

$$\Pr\left[\mathsf{Eval}(|\text{✿}\rangle, x) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} : |\text{✿}\rangle \leftarrow \mathsf{Protect}\langle\mathcal{V}(1^\lambda, y), \mathcal{P}(1^\lambda)\rangle\right] \geq 1 - \mathsf{negl}(\lambda)$$

Anti-piracy security with respect to non-colliding distribution. This definition is defined through a game, parametrized by a security parameter λ, and between a challenger and a triple of QPT algorithms $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples a point $y \leftarrow\!\!\$\ \{0,1\}^n$.
  - The challenger and the client run the protection protocol: $|\text{✿}\rangle \leftarrow \mathsf{Protect}\langle \mathcal{V}(1^\lambda, y), \mathcal{A}(1^\lambda)\rangle$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\text{✿}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\text{✿}^*\rangle_1$ to $\mathcal{B}$, and $|\text{✿}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples $(x_1, x_2) \leftarrow \mathcal{D}^{nc}_y$. That is, the challenger samples two independent random bitstrings $x', x'' \leftarrow\!\!\$\ \{0,1\}^n$; then sets $(x_1, x_2) = (y, x')$, or $(x_1, x_2) = (x', y)$, or $(x_1, x_2) = (x', x'')$, with probability 1/3 for each case.
  - The challenger sends x_1 to $\mathcal{B}$, and x_2 to $\mathcal{C}$.

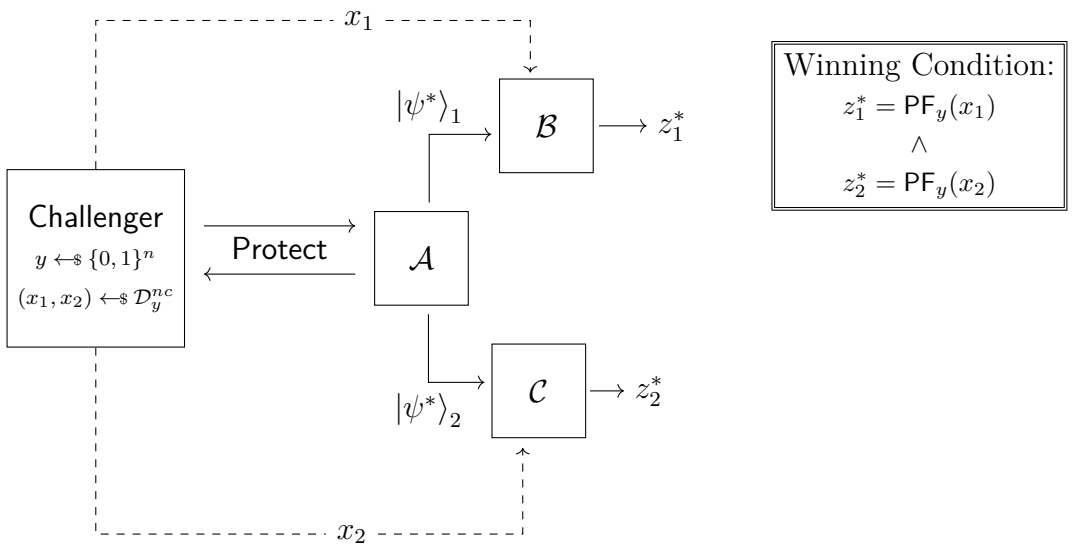

Figure 6.2: Anti-piracy security of a semi-quantum copy-protection scheme of point functions, with respect to the non-colliding distribution. This property states that no triple of efficient adversaries must win this game with probability significantly greater than $2/3$.

Let z_1^* denotes the output of $\mathcal{B}$, and z_2^* denotes the output of $\mathcal{C}$. $\mathcal{B}$ makes a correct guess if $z_1^* = \mathsf{PF}_y(x_1)$, that is if $x_1 = y$ and $z_1^* = 1$ or if $x_1 \neq y$ and $z_1^* = 0$. Similarly, $\mathcal{C}$ makes a correct guess if $z_2^* = \mathsf{PF}_y(x_2)$, that is if $x_2 = y$ and $z_2^* = 1$ or if $x_2 \neq y$ and $z_2^* = 0$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ make a correct guess. We say that a copy-protection scheme of point functions has anti-piracy security with respect to the product distribution if no triple of QPT adversaries wins this game with probability significantly greater than $2/3$.

In other words, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{cc} & z_1^* \leftarrow \mathcal{B}(|\text{✪}^*\rangle_1, x_1); z_2^* \leftarrow \mathcal{C}(|\text{✪}^*\rangle_2, x_2) \\ z_1^* = \mathsf{PF}_y(x_1) & (x_1, x_2) \leftarrow\$ \mathcal{D}_y^{nc} \\ \wedge \quad : & |\text{✪}^*\rangle_{12} \leftarrow \mathcal{A}(|\text{✪}\rangle) \\ z_2^* = \mathsf{PF}_y(x_2) & |\text{✪}\rangle \leftarrow \mathsf{Protect}\langle \mathcal{V}(1^\lambda, y), \mathcal{A}(1^\lambda)\rangle \\ & y \leftarrow\$ \{0,1\}^n \end{array}\right] \leq \frac{2}{3} + \mathsf{negl}(\lambda)$$

We provide an illustration of this game in Figure 6.2.

6.6.2 Construction

We present below a construction of semi-quantum copy-protection scheme of point functions. This construction is based on our black-box construction 14, where we use construction 17 as the underlying copy-protection of pseudorandom functions. The protection algorithm is replaced by an interactive protocol, in which we use our remote coset state preparation to make the scheme semi-quantum.

Construction 19: Semi-Quantum Copy-Protection of Point Functions

Let λ a security parameter, and $n = \mathsf{poly}(\lambda)$. Let RCP be the remote coset state preparation of protocol 5. Let $\mathcal{D}_{HiddenTrigger}$ be a family of Test and $\mathsf{Extract}$ procedures as defined in Section 5.5. Let $\{\mathsf{PRF}_k\}_{k\in\mathcal{K}}$ a family of puncturable and extracting pseudorandom functions, with input space $\{0,1\}^{n_x}$, and output space $\{0,1\}^{n_z}$, where

$n_{\mathcal{X}}, n_{\mathcal{Z}} = \mathsf{poly}(\lambda)$, and $n_{\mathcal{Z}}$ is smaller enough than $n_{\mathcal{X}}$. Let $\kappa = \mathsf{poly}(\lambda)$, $\kappa < n_{\mathcal{X}}$. Recall that for any program P, we use the notation $\widehat{\mathsf{P}} = \mathsf{iO}(\mathsf{P})$.

$\mathsf{Protect}\langle\mathcal{V}(1^\lambda, y), \mathcal{P}(1^\lambda)\rangle$:

- $\mathcal{V}$ and $\mathcal{P}$ run $\left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle$ to prepare κ n-long coset states.
- $\mathcal{V}$ samples $\mathsf{k} \leftarrow\!\!\$\ \mathcal{K}$.
- $\mathcal{V}$ samples $(\mathsf{Test}, \mathsf{Extract}) \leftarrow\!\!\$\ \mathcal{D}_{HiddenTrigger}$.
- $\mathcal{V}$ generates the program R_k, defined in program 2, parametrized by the procedures Test and $\mathsf{Extract}$, and the obfuscated membership programs $\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}$.
- $\mathcal{V}$ sends $\widehat{\mathsf{R}_\mathsf{k}} = \mathsf{iO}(\mathsf{R}_\mathsf{k})$, and $z = \mathsf{PRF}_\mathsf{k}(y)$ to $\mathcal{P}$.

$\left(\left(\bigotimes_{i=1}^{\kappa} |A_{i,s_i,s'_i}\rangle, \widehat{\mathsf{R}_\mathsf{k}}, z\right), x\right)$:

- For $i \in [\![1, \kappa]\!]$:
  - if $x_i = 1$: apply a Hadamard gate H^n on $|A_{i,s_i,s'_i}\rangle$;
  - otherwise leave the state unchanged;
  - in any case, denote the resulting state $|\text{☼}'_i\rangle$.
- Run the program $\widehat{\mathsf{R}_\mathsf{k}}$ coherently on $(x, |\text{☼}'_1\rangle, \ldots, |\text{☼}'_\kappa\rangle)$. Let z' denotes the outcome.
- Uncompute the Hadamard gates above.
- Return 1 if $z' = z$, and 0 otherwise.

Correctness and anti-piracy security. The correctness of this construction follows directly from the correctness of the original construction, and the one of the remote coset state preparation protocol (protocol 5).

Theorem 25 (Correctness of Construction 19)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions. Then the semi-quantum copy-protection scheme defined in construction 19 has correctness.

The anti-piracy security follows from the fact that the remote coset state preparation protocol preserves the monogamy-of-entanglement property. The proof follows the one of the original construction: the hybrids are defined similarly (preparing and sending coset states in the original hybrids are replaced by the remote coset state preparation), and the final reduction is made with the semi-quantum version (using protocol 5) of the monogamy-of-entanglement game, instead of the original one.

The proof follows the same structure as the one for the original construction's (construction 14) security. The same arguments allow us to reduce the anti-piracy security of this game to the real-or-random anti-piracy security with respect to the non-colliding

distribution of the semi-quantum equivalent of the single-decryptor construction of construction 16. We formally describe this semi-quantum version below. Remark that in this semi-quantum single-decryptor, the original key generation and quantum key generation procedures are replaced by a single interactive protocol $\mathsf{QKeyGen}$ between the classical vendor $\mathcal{V}$ and the quantum client $\mathcal{P}$.

Construction 20: Semi-Quantum Single-Decryptor Scheme

Let $n, \kappa = \mathsf{poly}(\lambda)$.

- $\mathsf{QKeyGen}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle$:
  - $\mathcal{V}$ and $\mathcal{P}$ run $\left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}, |\psi\rangle\right) \leftarrow \mathsf{RCP}\langle\mathcal{V}(1^\lambda), \mathcal{P}(1^\lambda)\rangle$ to prepare κ n-long coset states.
  - $\mathcal{V}$ generates the obfuscated membership programs $\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}$, and send them to $\mathcal{P}$.
  - Set $\mathsf{sk} = \left((A_i, s_i, s'_i)_{i\in[\![1,\kappa]\!]}\right)$, $\mathsf{pk} = \{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}$, and $|\mathfrak{k}\rangle = |\psi\rangle$.
- $\mathsf{Enc}(\{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'i}\}_{i\in[\![1,\kappa]\!]}, m)$:
  - Sample $r \leftarrow\!\!\$\ \{0,1\}^{\mathcal{R}}$.
  - Generate an iO-obfuscated program $\widehat{\mathsf{Q}}_{m,r}$ of program $\mathsf{Q}_{m,r}$ described in program 1.
  - Return $\left(r, \widehat{\mathsf{Q}}_{m,r}\right)$.
- $\mathsf{Dec}\left(|\mathfrak{k}\rangle, \left(r, \widehat{\mathsf{Q}}_{m,r}\right)\right)$:
  - For all $i \in [\![1,\kappa]\!]$: if $r_i = 1$, apply $\mathsf{H}^{\otimes n}$ to the i-th register of $|\mathfrak{k}\rangle$.
  - Let $|\mathfrak{k}'\rangle$ be the resulting state, run $\widehat{\mathsf{Q}}_{m,r}$ coherently on $|\mathfrak{k}'\rangle$. Let m denotes the outcome.
  - Uncompute the Hadamard gates above.
  - Return m.

Recall that in Section 5.4.5, the main argument regarding the security of the single-decryptor is that, in the real-or-random anti-piracy security game, we can replace the compute-and-compare programs in the encryption challenges by simulated programs without any information on the message, making sure that Bob and Charlie cannot both guess which message has been encrypted. Observe that we can actually use the same replacement in this semi-quantum version: assume there exists a triple of adversaries Alice, Bob, and Charlie, such that Bob and Charlie both distinguish compute-and-compare programs from simulated ones. Then, we construct another triple of adversaries, Alex, Billy, and Clover, that win the semi-quantum monogamy-of-entanglement game of coset states with non-negligible probability, leading to a contradiction. Billy and Clover use Bob and Charlie to extract the lock-value of their respective compute-and-compare program (we know they can do this, as Bob and Charlie can both distinguish between their respective compute-and-compare program, and the simulated one). Because we are considering the

non-colliding distribution, the random coins r_1 and r_2 used for Bob and Charlie's challenges respectively will be such that $r_{1,i} = 0$ and $r_{2,i} = 1$ for some index i with overwhelming probability. The corresponding lock-values are then vectors in $A + s \times A^{\perp} + s'$. Billy and Clover simply have to output these lock-values to win the game. This contradicts the semi-quantum monogamy-of-entanglement property of coset states, therefore proving the security of our construction.

Theorem 26 (Anti-Piracy Security of Construction 19)**.** Assume the existence of post-quantum indistinguishability obfuscation, one-way functions, compute-and-compare obfuscation for the class of unpredictable distributions. Then the copy-protection scheme defined in construction 19 has anti-piracy security, with respect to the non-colliding distribution.

We stress that semi-quantum versions of constructions whose security is based on the direct-product hardness or monogamy-of-entanglement of coset states can be achieved in the same way, by simply replacing the preparation and sending of coset states by our remote coset state preparation protocol.

Bibliography

Aaronson, Scott (July 2009). "Quantum Copy-Protection and Quantum Money". In: *2009 24th Annual IEEE Conference on Computational Complexity*. DOI: 10.1109/ccc.2009.42. URL: http://dx.doi.org/10.1109/CCC.2009.42.

Aaronson, Scott and Paul Christiano (2012). "Quantum money from hidden subspaces". In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pp. 41–60.

Aaronson, Scott, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang (2021). "New approaches for quantum copy-protection". In: *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*. Springer, pp. 526–555.

Ananth, Prabhanjan and Amit Behera (2024). "A modular approach to unclonable cryptography". In: *Annual International Cryptology Conference*. Springer, pp. 3–37.

Ananth, Prabhanjan and Fatih Kaleoglu (2021). "Unclonable encryption, revisited". In: *Theory of Cryptography Conference*. Springer, pp. 299–329.

Ananth, Prabhanjan, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry (2022). "On the feasibility of unclonable encryption, and more". In: *Annual International Cryptology Conference*. Springer, pp. 212–241.

Ananth, Prabhanjan, Fatih Kaleoglu, and Qipeng Liu (Aug. 2023). "Cloning Games: A General Framework for Unclonable Primitives". In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Heidelberg, pp. 66–98. DOI: 10.1007/978-3-031-38554-4_3.

Ananth, Prabhanjan and Rolando L. La Placa (Oct. 2021). "Secure Software Leasing". In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Heidelberg, pp. 501–530. DOI: 10.1007/978-3-030-77886-6_17.

Ananth, Prabhanjan, Alexander Poremba, and Vinod Vaikuntanathan (2023). "Revocable cryptography from learning with errors". In: *Theory of Cryptography Conference*. Springer, pp. 93–122.

Barak, Boaz, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang (Aug. 2001). "On the (Im)possibility of Obfuscating Programs". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, pp. 1–18. DOI: 10.1007/3-540-44647-8_1.

Bartusek, James, Sanjam Garg, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts (2023). "Obfuscation and outsourced computation with certified deletion". In: *Cryptology ePrint Archive*.

Bartusek, James and Dakshita Khurana (Aug. 2023). "Cryptography with Certified Deletion". In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya.

Vol. 14085. LNCS. Springer, Heidelberg, pp. 192–223. DOI: 10.1007/978-3-031-38554-4_7.

Bartusek, James, Dakshita Khurana, Giulio Malavolta, Alexander Poremba, and Michael Walter (2023). "Weakening assumptions for publicly-verifiable deletion". In: *Theory of Cryptography Conference*. Springer, pp. 183–197.

Bartusek, James, Dakshita Khurana, and Alexander Poremba (2023). "Publicly-verifiable deletion via target-collapsing functions". In: *Annual International Cryptology Conference*. Springer, pp. 99–128.

Behera, Amit, Or Sattath, and Uriel Shinar (2021). "Noise-tolerant quantum tokens for mac". In: *arXiv preprint arXiv:2105.05016*.

Ben-David, Shalev and Or Sattath (2023). "Quantum tokens for digital signatures". In: *Quantum* 7, p. 901.

Bennett, Charles and Gilles Brassard (Jan. 1984). "Quantum cryptography: Public key distribution and coin tossing". In: vol. 560, pp. 175–179. DOI: 10.1016/j.tcs.2011.08.039.

Bennett, Charles H., Gilles Brassard, Seth Breidbard, and Stephen Wiesner (1982). "Quantum Cryptography, or Unforgeable Subway Tokens". In: *Advances in Cryptology: Proceedings of CRYPTO '82*. Plenum, pp. 267–275.

Boneh, Dan and Brent Waters (Dec. 2013). "Constrained Pseudorandom Functions and Their Applications". In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15.

Bouman, Niek J and Serge Fehr (2010). "Sampling in a quantum population, and applications". In: *Annual Cryptology Conference*. Springer, pp. 724–741.

Boyle, Elette, Shafi Goldwasser, and Ioana Ivan (Mar. 2014). "Functional Signatures and Pseudorandom Functions". In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Heidelberg, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29.

Brakerski, Zvika, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick (Oct. 2018). "A Cryptographic Test of Quantumness and Certifiable Randomness from a Single Quantum Device". In: *59th FOCS*. Ed. by Mikkel Thorup. IEEE Computer Society Press, pp. 320–331. DOI: 10.1109/FOCS.2018.00038.

Broadbent, Anne and Eric Culf (2023). "Uncloneable Cryptographic Primitives with Interaction". In: *arXiv preprint arXiv:2303.00048*.

Broadbent, Anne and Rabib Islam (Nov. 2020). "Quantum Encryption with Certified Deletion". In: *TCC 2020, Part III*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12552. LNCS. Springer, Heidelberg, pp. 92–122. DOI: 10.1007/978-3-030-64381-2_4.

Broadbent, Anne, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram (2021). "Secure software leasing without assumptions". In: *Theory of Cryptography Conference*. Springer, pp. 90–120.

Broadbent, Anne and Sébastien Lord (2020). "Uncloneable Quantum Encryption via Oracles". In: Leibniz International Proceedings in Informatics (LIPIcs) 158. Ed. by Steven T. Flammia, 4:1–4:22. ISSN: 1868-8969.

Brodutch, Aharon, Daniel Nagaj, Or Sattath, and Dominique Unruh (2014). "An adaptive attack on Wiesner's quantum money". In: *arXiv preprint arXiv:1404.1507*.

Cakan, Alper, Vipul Goyal, Chen-Da Liu-Zhang, and João Ribeiro (2024). "Unbounded leakage-resilience and intrusion-detection in a quantum world". In: *Theory of Cryptography Conference*. Springer, pp. 159–191.

Chevalier, Céline, Paul Hermouet, and Quoc-Huy Vu (2023). "Semi-quantum copy-protection and more". In: *Theory of Cryptography Conference*. Springer, pp. 155–182.

— (2024a). "Security Models and Cryptographic Protocols in a Quantum World". In: *Foundations and Trends in Theoretical Computer Science*. To appear.

— (2024b). "Towards Unclonable Cryptography in the Plain Model". In: https://eprint.iacr.org/2023/1825. URL: https://eprint.iacr.org/2023/1825.

Cohen, Aloni, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs (2016). "Watermarking cryptographic capabilities". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 1115–1127.

Coladangelo, Andrea, Jiahui Liu, Qipeng Liu, and Mark Zhandry (Aug. 2021). "Hidden Cosets and Applications to Unclonable Cryptography". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, pp. 556–584. DOI: 10.1007/978-3-030-84242-0_20.

Coladangelo, Andrea, Christian Majenz, and Alexander Poremba (2024). "Quantum copy-protection of compute-and-compare programs in the quantum random oracle model". In: *Quantum* 8, p. 1330.

Coladangelo, Andrea and Or Sattath (July 2020). "A Quantum Money Solution to the Blockchain Scalability Problem". In: *Quantum* 4, p. 297. ISSN: 2521-327X. DOI: 10.22331/q-2020-07-16-297. URL: http://dx.doi.org/10.22331/q-2020-07-16-297.

Conde Pena, Marta, Raul Duran Diaz, Jean-Charles Faugere, Luis Hernandez Encinas, and Ludovic Perret (2019). "Non-quantum cryptanalysis of the noisy version of Aaronson–Christiano's quantum money scheme". In: *IET Information Security* 13.4, pp. 362–366.

Culf, Eric and Thomas Vidick (2022). "A monogamy-of-entanglement game for subspace coset states". In: *Quantum* 6, p. 791.

Deutsch, David and Richard Jozsa (1992). "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907, pp. 553–558.

Diffie, Whitfield and Martin E Hellman (1976). "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6, pp. 644–654.

Farhi, Edward, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor (2012). "Quantum money from knots". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 276–289.

Feynman, Richard P (2018). "Simulating physics with computers". In: *Feynman and computation*. cRc Press, pp. 133–153.

Georgiou, Marios and Mark Zhandry (2020). *Unclonable Decryption Keys*. Cryptology ePrint Archive, Report 2020/877. https://ia.cr/2020/877.

Gheorghiu, Alexandru, Tony Metger, and Alexander Poremba (2022). "Quantum cryptography with classical communication: parallel remote state preparation for copy-protection, verification, and more". In: *arXiv preprint arXiv:2201.13445*.

Gheorghiu, Alexandru and Thomas Vidick (2019). "Computationally-secure and composable remote state preparation". In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, pp. 1024–1033.

Goldreich, Oded, Shafi Goldwasser, and Silvio Micali (Oct. 1984). "How to Construct Random Functions (Extended Abstract)". In: *25th FOCS*. IEEE Computer Society Press, pp. 464–479. DOI: 10.1109/SFCS.1984.715949.

Gottesman, Daniel (2002). "Uncloneable encryption". In: *arXiv preprint quant-ph/0210062*.

Håstad, Johan, Russell Impagliazzo, Leonid A. Levin, and Michael Luby (1999). "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal on Computing* 28.4, pp. 1364–1396.

Hiroka, Taiga, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa (Dec. 2021). "Quantum Encryption with Certified Deletion, Revisited: Public Key, Attribute-Based, and Classical Communication". In: *ASIACRYPT 2021, Part I*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13090. LNCS. Springer, Heidelberg, pp. 606–636. DOI: 10.1007/978-3-030-92062-3_21.

— (2022). *Certified Everlasting Functional Encryption*. Cryptology ePrint Archive, Report 2022/969. https://eprint.iacr.org/2022/969.

Johnston, Nathaniel, Rajat Mittal, Vincent Russo, and John Watrous (2016). "Extended non-local games and monogamy-of-entanglement games". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 472.2189.

Kane, Daniel M, Shahed Sharif, and Alice Silverberg (2022). "Quantum money from quaternion algebras". In: *Mathematical Cryptology* 2.1, pp. 60–83.

Kane, Daniel M. (2018). *Quantum Money from Modular Forms*. arXiv: 1809.05925 [quant-ph].

Kiayias, Aggelos, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias (Nov. 2013). "Delegatable pseudorandom functions and applications". In: *ACM CCS 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, pp. 669–684. DOI: 10.1145/2508859.2516668.

Kitagawa, Fuyuki and Ryo Nishimaki (2023). "One-out-of-Many Unclonable Cryptography: Definitions, Constructions, and More". In: *Theory of Cryptography Conference*. Springer, pp. 246–275.

Kitagawa, Fuyuki, Ryo Nishimaki, and Takashi Yamakawa (2021). "Secure software leasing from standard assumptions". In: *Theory of Cryptography Conference*. Springer, pp. 31–61.

Kwiat, Paul, Harald Weinfurter, Thomas Herzog, Anton Zeilinger, and Mark A Kasevich (1995). "Interaction-free measurement". In: *Physical Review Letters* 74.24, p. 4763.

Liu, Jiahui, Hart Montgomery, and Mark Zhandry (Apr. 2023a). "Another Round of Breaking and Making Quantum Money: How to Not Build It from Lattices, and More". In: *EUROCRYPT 2023, Part I*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14004. LNCS. Springer, Heidelberg, pp. 611–638. DOI: 10.1007/978-3-031-30545-0_21.

— (2023b). *Another round of breaking and making quantum money: How to not build it from lattices, and more*. Springer.

Lutomirski, Andrew (2010). "An online attack against Wiesner's quantum money". In: *arXiv preprint arXiv:1010.0256*.

Lutomirski, Andrew, Scott Aaronson, Edward Farhi, David Gosset, Jonathan A. Kelner, Avinatan Hassidim, and Peter W. Shor (Jan. 2010). "Breaking and Making Quantum Money: Toward a New Quantum Cryptographic Protocol". In: *ICS 2010*. Ed. by Andrew Chi-Chih Yao. Tsinghua University Press, pp. 20–31.

Mahadev, Urmila (2018). "Classical verification of quantum computations". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, pp. 259–267.

Manin, Yuri I. (1980). "Computable and uncomputable". In: *Uspekhi Matematicheskikh Nauk* 35.6, pp. 219–220.

Merkle, Ralph C (1978). "Secure communications over insecure channels". In: *Communications of the ACM* 21.4, pp. 294–299.

Metger, Tony and Thomas Vidick (Jan. 2021). "Self-Testing of a Single Quantum Device Under Computational Assumptions". In: *ITCS 2021*. Ed. by James R. Lee. Vol. 185. LIPIcs, 19:1–19:12. DOI: 10.4230/LIPIcs.ITCS.2021.19.

Nakamoto, Satoshi (2008). "Bitcoin". In: *A peer-to-peer electronic cash system* 21260.

Pastawski, Fernando, Norman Y Yao, Liang Jiang, Mikhail D Lukin, and J Ignacio Cirac (2012). "Unforgeable noise-tolerant quantum tokens". In: *Proceedings of the National Academy of Sciences* 109.40, pp. 16079–16082.

Radian, Roy and Or Sattath (2019). "Semi-quantum money". In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pp. 132–146.

Regev, Oded (May 2005). "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, pp. 84–93. DOI: 10.1145/1060590.1060603.

Roberts, Bhaskar (Oct. 2021). "Security Analysis of Quantum Lightning". In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Heidelberg, pp. 562–567. DOI: 10.1007/978-3-030-77886-6_19.

Sahai, Amit and Brent Waters (2014). "How to use indistinguishability obfuscation: deniable encryption, and more". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 475–484.

Shmueli, Omri (2022a). "Public-key quantum money with a classical bank". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 790–803.

— (2022b). "Semi-quantum tokenized signatures". In: *Annual International Cryptology Conference*. Springer, pp. 296–319.

Shor, Peter W (1994). "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee, pp. 124–134.

— (1999). "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2, pp. 303–332.

Simon, D.R. (1994). "On the power of quantum computation". In: pp. 116–123. DOI: 10.1109/SFCS.1994.365701.

Tomamichel, Marco, Serge Fehr, Jedrzej Kaniewski, and Stephanie Wehner (Oct. 2013). "A monogamy-of-entanglement game with applications to device-independent quantum cryptography". In: *New Journal of Physics* 15.10, p. 103002. DOI: 10.1088/1367-2630/15/10/103002. URL: https://dx.doi.org/10.1088/1367-2630/15/10/103002.

Wiesner, Stephen (1983). "Conjugate coding". In: *ACM Sigact News* 15.1, pp. 78–88.

Wilde, Mark M (2011). "From classical to quantum Shannon theory". In: *arXiv preprint arXiv:1106.1445*.

Zhandry, Mark (May 2019). "Quantum Lightning Never Strikes the Same State Twice". In: *EUROCRYPT 2019, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. LNCS. Springer, Heidelberg, pp. 408–438. DOI: 10.1007/978-3-030-17659-4_14.

— (2020). "Schrödinger's pirate: How to trace a quantum decoder". In: *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part III 18*. Springer, pp. 61–91.

— (2021). "Quantum lightning never strikes the same state twice. or: quantum money from cryptographic assumptions". In: *Journal of Cryptology* 34, pp. 1–56.

Zhandry, Mark (2023). *Quantum Money from Abelian Group Actions*. Cryptology ePrint Archive, Paper 2023/1097. https://eprint.iacr.org/2023/1097. URL: https://eprint.iacr.org/2023/1097.

Appendix

A Towards Unclonable Cryptography in the Plain Model - Supplementary Materials

A.1 Copy-Protection of Pseudorandom Functions

We define in this section copy-protection of pseudorandom functions. We refer the reader to Definition 8 for a definition of pseudorandom functions.

A.1.1 Definitions

Definition 40 (Copy-Protection of Point Functions)**.** Consider a family of pseudorandom functions $\mathcal{F} = \{\mathsf{PRF}_\mathsf{k}\}_{\mathsf{k}\in\mathcal{K}}$ with key space k. All PRF_k have domain $\mathcal{X}$ and codomain $\mathcal{Z}$. For sake of simplicity, we denote $\mathsf{k} \leftarrow\!\!\$\, \mathcal{K}$ the key sampling procedure for this family. Note that this sampling does not have to be uniform.

A copy-protection scheme of $\mathcal{F}$ is composed of two algorithms $\langle \mathsf{Protect}, \mathsf{Eval}\rangle$ defined in the following way:

- $|\text{✡}\rangle \leftarrow \mathsf{Protect}(1^\lambda, \mathsf{k})$. The protection algorithm takes as input a security parameter, and a key $\mathsf{k} \in \mathcal{K}$, and outputs a quantum encoding of PRF_k: $|\text{✡}\rangle$.
- $z \leftarrow \mathsf{Eval}(|\text{✡}\rangle\, , x)$. The evaluation algorithm takes as input a quantum encoding $|\text{✡}\rangle$, and an input x in $\mathcal{X}$, and outputs an image z in $\mathcal{Z}$.

In addition, a copy-protection scheme of pseudorandom functions must satisfy the following properties.

Correctness. The correctness of a copy-protection scheme of pseudorandom functions is defined as follows. That is, for all $\mathsf{k} \in \mathcal{K}$, and $x \in \mathcal{X}$,

$$\Pr\Big[\mathsf{Eval}(|\text{✡}\rangle\, , x) = \mathsf{PRF}_\mathsf{k}(x) \;:\; |\text{✡}\rangle \leftarrow \mathsf{Protect}(1^\lambda, \mathsf{k})\Big] \geq 1 - \mathsf{negl}(\lambda)$$

Reversed anti-piracy security. We define our new definition of reversed anti-piracy security. This definition is defined through a game, parametrized by a security parameter λ, and between a challenger and a triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. It is also parametrized by a challenge distribution $\mathcal{D} = \{\mathcal{D}_x\}_{x\in\mathcal{X}}$, where each distribution $\mathcal{D}_x$ is over pairs of inputs for the pseudorandom functions. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**
  - The challenger samples a key $\mathsf{k} \leftarrow\!\!\$\ \mathcal{K}$, and an input $x \leftarrow\!\!\$\ \mathcal{X}$.
  - The challenger computes $|\text{✿}\rangle \leftarrow \mathsf{Protect}(1^\lambda, \mathsf{k})$.
  - The challenger computes $z = \mathsf{PRF}_\mathsf{k}(x)$.
  - The challenger sends $|\text{✿}\rangle$ and z to $\mathcal{A}$.
- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\text{✿}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\text{✿}^*\rangle_1$ to $\mathcal{B}$, and $|\text{✿}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples $(x_1, x_2) \leftarrow \mathcal{D}_x$.
  - The challenger sends x_1 to $\mathcal{B}$, and x_2 to $\mathcal{C}$.

Let b_1^* denotes the output of $\mathcal{B}$, and b_2^* denotes the output of $\mathcal{C}$. $\mathcal{B}$ makes a correct guess if $b_1^* = 1$ and $x_1 = x$, or if $b_1^* = 0$ and $x_1 \neq x$. Similarly, $\mathcal{C}$ makes a correct guess if $b_2^* = 1$ and $x_2 = x$, or if $b_2^* = 0$ and $x_2 \neq x$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ make a correct guess. We say that a copy-protection scheme of pseudorandom functions has reversed anti-piracy security with respect to the challenge distribution $\mathcal{D}$ if no triple of QPT adversaries wins this game with probability significantly greater than the trivial probability $p_\mathcal{D}$ corresponding to $\mathcal{D}$ - typically, this trivial probability is $1/2$ for the identical and product challenge distributions, and $2/3$ for the non-colliding distribution.

In other words, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{l} (b_1^* = 1 \wedge x_1 = x) \vee (b_1^* = 0 \wedge x_1 \neq x) \\ \wedge \\ (b_2^* = 1 \wedge x_2 = x) \vee (b_2^* = 0 \wedge x_2 \neq x) \end{array} : \begin{array}{l} b_1^* \leftarrow \mathcal{B}(|\text{✿}^*\rangle_1, x_1); b_2^* \leftarrow \mathcal{C}(|\text{✿}^*\rangle_2, x_2) \\ (x_1, x_2) \leftarrow\!\!\$\ \mathcal{D}_x \\ |\text{✿}^*\rangle_{12} \leftarrow \mathcal{A}(|\text{✿}\rangle, \mathsf{PRF}_\mathsf{k}(x)) \\ x \leftarrow\!\!\$\ \mathcal{X} \\ |\text{✿}\rangle \leftarrow \mathsf{Protect}(1^\lambda, \mathsf{k}) \\ \mathsf{k} \leftarrow\!\!\$\ \mathcal{K} \end{array}\right] \leq p_\mathcal{D} + \mathsf{negl}(\lambda)$$

A.2 Real-Or-Random Anti-Piracy Security for Single-Decryptors

In this section, we define our new real-or-random anti-piracy security for single-decryptors.

Let $\langle \mathsf{KeyGen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ be a public-key single-decryptor with message space $\mathcal{M}$. Let $\mathcal{R}$ be the space of random coins used for the encryption. Define the following game, parametrized by a security parameter λ, and between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. It is also parametrized by a challenge distribution $\mathcal{D} = \{\mathcal{D}_{(m,r)}\}_{m \in \mathcal{M}, r \in \mathcal{R}}$, where each distribution $\mathcal{D}_{(m,r)}$ is over pairs of (message, random coins) pairs. During the game, $\mathcal{B}$ and $\mathcal{C}$ are not allowed to communicate.

- **Setup phase:**

- The challenger samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- The challenger prepares $|\mathfrak{k}\rangle \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$.
- The challenger sends $|\mathfrak{k}\rangle$ and pk to $\mathcal{A}$.

- **Splitting phase:**
  - $\mathcal{A}$ prepares a bipartite state $|\mathfrak{k}^*\rangle_{12}$.
  - $\mathcal{A}$ sends $|\mathfrak{k}^*\rangle_1$ to $\mathcal{B}$, and $|\mathfrak{k}^*\rangle_2$ to $\mathcal{C}$.
  - $\mathcal{A}$ sends a message m to the challenger.

- **Challenge phase:**
  - The challenger samples $r \leftarrow\!\!\$\ \mathcal{R}$, and $((m_1, r_1), (m_2, r_2)) \leftarrow\!\!\$\ \mathcal{D}_{(m,r)}$.
  - The challenger computes $c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1; r_1)$, and $c_2 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_2; r_2)$. [1]
  - The challenger sends c_1 to $\mathcal{B}$ and c_2 to $\mathcal{C}$.

Let b_1^* denotes the output of $\mathcal{B}$, and b_2^* denotes the output of $\mathcal{C}$. $\mathcal{B}$ makes a correct guess if $b_1^* = b$ and $m_1 = m$ or if $b_1^* = 0$ and $m_1 \neq m$. Similarly, $\mathcal{C}$ makes a correct guess if $b_2^* = b$ and $m_2 = m$ or if $b_2^* = 0$ and $m_2 \neq m$ We say that a single-decryptor scheme has real-or-random anti-piracy security with respect to the challenge distribution $\mathcal{D}$ if no triple of QPT adversaries wins this game with probability significantly greater than the trivial probability $p_{\mathcal{D}}$ corresponding to $\mathcal{D}$ - typically, this trivial probability is $1/2$ for the identical and product challenge distributions, and $2/3$ for the non-colliding distribution.

In other words, if for all triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:

$$\Pr\left[\begin{array}{c} (b_1^* = 1 \wedge m_1 = m) \vee (b_1^* = 0 \wedge m_1 \neq m) \\ \wedge \\ (b_2^* = 1 \wedge m_2 = m) \vee (b_2^* = 0 \wedge m_2 \neq m) \end{array} : \begin{array}{l} b_1^* \leftarrow \mathcal{B}(|\text{✡}^*\rangle_1, \mathsf{Enc}(\mathsf{pk}, m_1; r_1)) \\ b_2^* \leftarrow \mathcal{C}(|\text{✡}^*\rangle_2, \mathsf{Enc}(\mathsf{pk}, m_2; r_2)) \\ ((m_1, r_1), (m_2, r_2)) \leftarrow\!\!\$\ \mathcal{D}_{(m,r)} \\ r \leftarrow\!\!\$\ \mathcal{R} \\ (|\text{✡}^*\rangle_{12}, m) \leftarrow \mathcal{A}(|\text{✡}\rangle, \mathsf{pk}) \\ |\text{✡}\rangle \leftarrow \mathsf{Protect}(1^\lambda, \mathsf{k}) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda) \end{array}\right] \leq p_{\mathcal{D}} + \mathsf{negl}(\lambda)$$

A.3 Proof of Theorem 19

In this section, we formally prove Theorem 19. That is, that construction 17 has reversed anti-piracy security with respect to a challenge distribution $\mathcal{D}$ if construction 16 has real-or-random anti-piracy security with respect to $\mathcal{D}$. This section is mostly an adaptation of our paper: Chevalier, Hermouet, and Vu (2024b), and some parts are taken verbatim.

A.3.1 [CLLZ21] Construction

We start by presenting the complete construction of the copy-protection scheme of pseudorandom functions of Coladangelo, Liu, Liu, and Zhandry (2021). In particular, we explicitly give the Test and Extract procedures.

[1] Recall that $\mathsf{Enc}(\mathsf{pk}, m_1; r_1)$ means that we use r_1 as the random tape in the Enc algorithm.

Let n be a polynomial in λ; we define $n_{\mathcal{X},0}, n_{\mathcal{X},1}, n_{\mathcal{X},2}$ such that $n_{\mathcal{X}} = n_{\mathcal{X},0} + n_{\mathcal{X},1} + n_{\mathcal{X},2}$ and $n_{\mathcal{X},2} - n_{\mathcal{X},0}$ is large enough. For this construction, we need three pseudorandom functions:

- A puncturable extracting pseudorandom function $\mathsf{PRF}_{1,\mathsf{k}_1} : \{0,1\}^{n_{\mathcal{X}}} \to \{0,1\}^{n_{\mathcal{Z}}}$ with error $2^{-\lambda-1}$ for min-entropy $n_{\mathcal{X}}$, where $n_{\mathcal{Z}}$ is a polynomial in λ and $n_{\mathcal{X}} \geq n_{\mathcal{Z}} + 2\lambda + 4$.
- A puncturable injective pseudorandom function $\mathsf{PRF}_{2,\mathsf{k}_2} : \{0,1\}^{n_{\mathcal{X},2}} \to \{0,1\}^{n_{\mathcal{X},1}}$ with failure probability $2^{-\lambda}$, with $n_{\mathcal{X},1} \geq 2n_{\mathcal{X},2} + \lambda$.
- A puncturable pseudorandom function $\mathsf{PRF}_{3,\mathsf{k}_3} : \{0,1\}^{n_{\mathcal{X},1}} \to \{0,1\}^{n_{\mathcal{X},2}}$.

Let $\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3$ the key spaces for these three pseudorandom functions.

Construction 21: Pseudorandom Function Copy-Protection

$\mathsf{Protect}(1^\lambda, \mathsf{k})$:

- Sample $n_{\mathcal{X},0}$ random coset states $\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}$, where each subspace $A_i \subseteq \mathbb{F}_2^n$ if of dimension $\frac{n}{2}$.
- For each coset state $|A_{i,s_i,s'_i}\rangle$, prepare the obfuscated membership programs $\widehat{\mathsf{P}}_i^0 = \mathsf{iO}(A_i + s_i)$ and $\widehat{\mathsf{P}}_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$.
- Sample $\mathsf{k}_i \leftarrow\!\!\$\, \mathcal{K}_i$ for $i \in \{1,2,3\}$.
- Prepare the program $\widehat{\mathsf{R}} \leftarrow \mathsf{iO}(\mathsf{R})$, where R is described in Figure A.1.
- Return $|\text{✿}_{\mathsf{k}}\rangle = \left(\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}, \widehat{\mathsf{R}}\right)$.

$\mathsf{Eval}(1^\lambda, |\text{✿}_{\mathsf{k}}\rangle, x)$:

- Parse $|\text{✿}_{\mathsf{k}}\rangle = \left(\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}, \widehat{\mathsf{R}}\right)$.
- Parse x as $x^{(0)}\|x^{(1)}\|x^{(2)}$, where each $x^{(i)}$ is of length $n_{\mathcal{X},i}$.
- For each $i \in [\![1, n_{\mathcal{X},0}]\!]$, if $x_i^{(0)} = 1$, apply $\mathsf{H}^{\otimes n}$ to $|A_{i,s_i,s'_i}\rangle$; if $x_i^{(0)} = 0$, leave the state unchanged.
- Let σ be the resulting state (which can be interpreted as a superposition over tuples of $n_{\mathcal{X},0}$ vectors). Run $\widehat{\mathsf{R}}$ coherently on input x and σ, and measure the final output register to obtain z.
- Return z.

A.3.2 Proof of Reversed Anti-Piracy Security

We start by defining some notations.

Notations. In the proof, we sometimes parse $x \in \{0,1\}^{n_{\mathcal{X}}}$ as $(x^{(0)}, x^{(1)}, x^{(2)})$ such that $x = x^{(0)}\|x^{(1)}\|x^{(2)}$ (where $\cdot\|\cdot$ is the concatenation operator) and the length of $x^{(i)}$ is $n_{\mathcal{X},i}$ for $i \in \{0,1,2\}$.

Hardcoded: Keys $(\mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3) \in \mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{K}_3$, programs $\mathsf{P}_i^0, \mathsf{P}_i^1$ for all $i \in [\![1, n_{\mathcal{X},0}]\!]$.
On input $x = x^{(0)} \| x^{(1)} \| x^{(2)}$ and vectors $v_0, v_1, \cdots, v_{n_{\mathcal{X},0}}$ where each $v_i \in \mathbb{F}_2^n$, do the following:

1. (**Hidden Trigger Mode**) If $\mathsf{PRF}_{3,\mathsf{k}_3}(x^{(1)}) \oplus x^{(2)} = x^{(0)} \| \mathsf{Q}'$ and $x^{(1)} = \mathsf{PRF}_{2,\mathsf{k}_2}(x^{(0)} \| \mathsf{Q}')$: treat Q' as a classical circuit and output $\mathsf{Q}'(v_1, \cdots, v_{n_{\mathcal{X},0}})$.

2. (**Normal Mode**) If for all $i \in [\![1, n_{\mathcal{X},0}]\!]$, $\mathsf{P}_i^{x_i}(v_i) = 1$, then output $\mathsf{PRF}_{1,\mathsf{k}_1}(\mathsf{k}_1, x)$. Otherwise, output $\bot$.

Figure A.1: Program R.

Given as input $x^{(0)} \in \{0,1\}^{n_{\mathcal{X},0}}$, $z \in \{0,1\}^{n_{\mathcal{Z}}}$, $\mathsf{k}_2, \mathsf{k}_3 \in \mathcal{K}_2 \times \mathcal{K}_3$ and cosets $\{A_{i,s_i,s_i'}\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}$:

1. Let Q be the program which, given $v_0, \ldots, v_{n_{\mathcal{X},0}}$, returns z if $R_i^{x_{0,i}}(v_i) = 1$ for all i or $\bot$ otherwise.

2. $x'^{(1)} \leftarrow \mathsf{PRF}_{2,\mathsf{k}_2}(x^{(0)} \| \mathsf{Q})$;

3. $x'^{(2)} \leftarrow \mathsf{PRF}_{3,\mathsf{k}_3}(x'^{(1)}) \oplus (x^{(0)} \| \mathsf{Q})$;

4. Return $x^{(0)} \| x'^{(1)} \| x'^{(2)}$.

Figure A.2: $\mathsf{GenTrigger}$ *procedure.*

We proceed with both proofs through a sequence of hybrids. For any pair of hybrids (G_i, G_j), we say that G_i is *negligibly close to* G_j if for every triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$, the probability that $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ wins G_i is negligibly close to the probability that they win G_j.

Procedure. We define the $\mathsf{GenTrigger}$ procedure (Figure A.2) which, given an input's prefix $x^{(0)}$ and a pseudorandom function image z returns a so-called *trigger input* x' that: passes the "Hidden Trigger" condition of the program R. Although this procedure also takes as input pseudorandom function keys $\mathsf{k}_2, \mathsf{k}_3$ and coset states descriptions, we will abuse notation and only write $\mathsf{GenTrigger}(x^{(0)}, y)$ when it is clear from the context. We will also write $\mathsf{GenTrigger}(x^{(0)}; \mathsf{Q})$ - where Q is a program - to denote the same procedure using Q instead of the program normally defined in step 1.

Trigger's inputs lemma. The following lemma is taken from Coladangelo, Liu, Liu, and Zhandry (2021, Lemma 7.17).

Lemma 2. Assuming post-quantum iO and one-way functions, any efficient QPT algorithm $\mathcal{A}$ cannot win the following game with non-negligible advantage:

- A challenger samples $\mathsf{k}_i \leftarrow\!\!\$\ \mathcal{K}_i$ for $i \in \{1,2,3\}$, and prepares a quantum key $|\text{⛭}_{\mathsf{k}_1}\rangle := (\{|A_{i,s_i,s_i'}\rangle\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}, \mathsf{iO}(\mathsf{R}))$ (recall that R has keys $\mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3$ hardcoded).

- The challenger then samples a random input $x_1 \leftarrow \{0,1\}^{n_{\mathcal{X}}}$; let $z_1 \leftarrow \mathsf{PRF}_{1,\mathsf{k}_1}(x_1)$ and computes $x'_1 \leftarrow \mathsf{GenTrigger}(x_1^{(0)}, z_1)$.
- Similarly, the challenger samples a random input $x_2 \leftarrow \{0,1\}^{n_{\mathcal{X}}}$; let $z_2 \leftarrow \mathsf{PRF}_{1,\mathsf{k}_1}(x_2)$ and computes $x'_2 \leftarrow \mathsf{GenTrigger}(x_2^{(0)}, z_1)$.
- The challenger flips a coin b, and sends either $\left(|\text{✿}_{\mathsf{k}_1}\rangle, x_1, x_2\right)$ or $\left(|\text{✿}_{\mathsf{k}_1}\rangle, x'_1, x'_2\right)$ to $\mathcal{A}$, depending on the value of the coin.

$\mathcal{A}$ wins if it guesses b correctly.

Game G_0: This is the reversed anti-piracy security game, with respect to the challenge distribution $\mathcal{D}$.

- **Setup phase:**
  - The challenger samples $n_{\mathcal{X},0}$ random cosets $\{A_i, s_i, s'_i\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}$, and prepares the associated coset states $\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}$ and the obfuscated membership programs $\{(\widehat{\mathsf{P}}_i^0, \widehat{\mathsf{P}}_i^1)\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}$.
  - The challenger samples $\mathsf{k}_i \leftarrow \mathcal{K}_i$ for $i \in \{1,2,3\}$ and generates the obfuscated program $\widehat{\mathsf{R}} \leftarrow \mathsf{iO}(\mathsf{R})$.
  - The challenger samples $x \leftarrow\!\!\$\ \{0,1\}^{n_{\mathcal{X}}}$ and computes $z := \mathsf{PRF}_{1,\mathsf{k}_1}(x)$.
  - Finally, the challenger sends $|\text{✿}_{\mathsf{k}_1}\rangle := \left(\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,n_{\mathcal{X},0}]\!]}, \widehat{\mathsf{R}}\right)$ and z to $\mathcal{A}$.
- **Splitting phase:** $\mathcal{A}$ prepares a bipartite quantum state $|\text{✿}^*\rangle_{12}$, then sends $|\text{✿}^*\rangle_1$ to $\mathcal{B}$ and $|\text{✿}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples two inputs $x_1, x_2 \leftarrow\!\!\$\ \mathcal{D}_x$.
  - The challenger sends x_1 to $\mathcal{B}$, and x_2 to $\mathcal{C}$.

  Let b_1^* denotes the output of $\mathcal{B}$, and b_2^* denotes the output of $\mathcal{C}$. $\mathcal{B}$ makes a correct guess if $b_1^* = 1$ and $x_1 = x$, or if $b_1^* = 0$ and $x_1 \neq x$. Similarly, $\mathcal{C}$ makes a correct guess if $b_2^* = 1$ and $x_2 = x$, or if $b_2^* = 0$ and $x_2 \neq x$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ make a correct guess.

Game G_1: In this game, we replace the challenges x_1 and x_2 by their trigger inputs for both $\mathcal{B}$ and $\mathcal{C}$. More precisely, the challenge phases become the following.

- **Challenge phase:**
  - The challenger samples two inputs $x_1, x_2 \leftarrow\!\!\$\ \mathcal{D}_x$.
  - The challenger computes $z_1 = \mathsf{PRF}_{1,\mathsf{k}_1}(x_1)$ and $z_2 = \mathsf{PRF}_{2,\mathsf{k}_2}(x_2)$.
  - The challenger sends $\mathsf{GenTrigger}(x_1^{(0)}, z_1)$ to $\mathcal{B}$, and $\mathsf{GenTrigger}(x_2^{(0)}, z_2)$ to $\mathcal{C}$.

The trigger's inputs lemma (Lemma 2) implies that G_1 is negligibly close to G_0.

Game G_2: In this game, we replace z (in the setup phase) and z_1, z_2 (in the challenge phases) by uniformly random strings. Since all the inputs have enough min-entropy $n_{\mathcal{X},1} + n_{\mathcal{X},2} \geq m + 2\lambda + 4$ and PRF_1 is extracting, the images are statistically close to independently random bitstrings. Thus, G_2 is negligibly close to G_1.

Game G_3: This game has exactly the same distribution as that of G_2. We only change the order in which some values are sampled, and recognize that certain procedures become identical to encryption in the single-decryptor scheme $\langle \mathsf{SD.KeyGen}, \mathsf{SD.QKeyGen}, \mathsf{SD.Enc}, \mathsf{SD.Dec} \rangle$ from construction 16. Thus, the probability of winning in G_3 is the same as in G_2.

- **Setup phase:**
  - The challenger runs $\mathsf{SD.KeyGen}(1^\lambda)$ to obtain $n_{\mathcal{X},0}$ random cosets $\{A_i, s_i, s'_i\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}$, the associated coset states $\{|A_{i,s_i,s'_i}\rangle\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}$ and the obfuscated membership programs $\{(\mathsf{P}^0_i, \mathsf{P}^1_i)\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}$. Let $|\mathfrak{k}_{\mathsf{sk}}\rangle := \{|A_{i,s_i,s'_i}\rangle\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}$.
  - The challenger samples $\mathsf{k}_i \leftarrow\!\!\$\ \mathcal{K}_i$ for $i \in \{1, 2, 3\}$ and generates the obfuscated program $\widehat{\mathsf{R}} \leftarrow \mathsf{iO}(\mathsf{R})$.
  - The challenger samples $z \leftarrow\!\!\$\ \{0,1\}^{n_z}$ and sends $|\mathfrak{k}_{\mathsf{k}_1}\rangle := \left(\{|A_{i,s_i,s'_i}\rangle\}_{i \in [\![1, n_{\mathcal{X},0}]\!]}, \widehat{\mathsf{R}}\right)$ and z to $\mathcal{A}$.
- **Splitting phase:** $\mathcal{A}$ prepares a bipartite quantum state $|\mathfrak{k}^*\rangle_{12}$, then sends $|\mathfrak{k}^*\rangle_1$ to $\mathcal{B}$ and $|\mathfrak{k}^*\rangle_2$ to $\mathcal{C}$.
- **Challenge phase:**
  - The challenger samples random coins $r \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$ for encryption.
  - The challenger samples two (message, random coins) pairs $((z_1, r_1), (z_2, r_2)) \leftarrow\!\!\$\ \mathcal{D}_{(z,r)}$.
  - The challenger computes $(x_1, \mathsf{Q}_1) \leftarrow \mathsf{Enc}(\mathsf{pk}, z_1; r_1)$ and $(x_2, \mathsf{Q}_2) \leftarrow \mathsf{Enc}(\mathsf{pk}, z_2; r_2)$.
  - The challenger sends $\mathsf{GenTrigger}(x_1^{(0)}, z_1)$ to $\mathcal{B}$ and $\mathsf{GenTrigger}(x_2^{(0)}, z_2)$ to $\mathcal{C}$.

  Let b_1^* denotes the output of $\mathcal{B}$, and b_2^* denotes the output of $\mathcal{C}$. $\mathcal{B}$ makes a correct guess if $b_1^* = 1$ and $z_1 = z$, or if $b_1^* = 0$ and $z_1 \neq z$. Similarly, $\mathcal{C}$ makes a correct guess if $b_2^* = 1$ and $z_2 = z$, or if $b_2^* = 0$ and $z_2 \neq z$. $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ make a correct guess.

Reduction from single-decryptor's piracy game for the product distribution. We reduce the game G_3 to the real-or-random anti-piracy game of the single-decryptor (construction 16), both with respect to the challenge distribution $\mathcal{D}$. Assume that there exists a triple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ who wins the last hybrid G_3 with respect to the product distribution with advantage δ. We construct a QPT adversary $(\mathcal{A}', \mathcal{B}', \mathcal{C}')$ who wins the real-or-random anti-piracy game of the single-decryptor scheme of construction 16 with the same advantage δ.

- $\mathcal{A}'$, on input a quantum key $|\mathfrak{k}_{\mathsf{sk}}\rangle$ and the associated public key pk:
  - samples $\mathsf{k}_i \leftarrow \mathcal{K}_i$ for $i \in \{1, 2, 3\}$ and use these keys and pk to prepare the obfuscated program $\widehat{\mathsf{R}} \leftarrow \mathsf{iO}(\mathsf{R})$;
  - samples $z \leftarrow\!\!\$\ \{0,1\}^{n_z}$;

  - runs $\mathcal{A}$ on $(\rho_{\mathsf{sk}}, \widehat{\mathsf{R}}, z)$ to get $|\mathfrak{k}^*\rangle_{12}$;
  - then prepares $|\mathfrak{k}^{**}\rangle_1 := |\mathfrak{k}^*\rangle_1 \otimes |\mathsf{k}_2, \mathsf{k}_3\rangle$ and $|\mathfrak{k}^{**}\rangle_2 := |\mathfrak{k}^*\rangle_2 \otimes |\mathsf{k}_2, \mathsf{k}_3\rangle$;
  - and finally sends $|\mathfrak{k}^{**}\rangle_1$ to $\mathcal{B}$, $|\mathfrak{k}^{**}\rangle_2$ to $\mathcal{C}$, and the message z to the challenger.
- $\mathcal{B}'$, on input $|\mathfrak{k}^{**}\rangle_1$ and a ciphertext (r, Q):
  - computes $x' \leftarrow \mathsf{GenTrigger}(r; \mathsf{Q})$;
  - runs $\mathcal{B}$ on $(|\mathfrak{k}^*\rangle_1, x')$ and returns the outcome.
- $\mathcal{C}'$ is defined similarly as $\mathcal{B}'$ by replacing $|\mathfrak{k}^{**}\rangle_1$ by $|\mathfrak{k}^{**}\rangle_2$.

The adversary $(\mathcal{A}', \mathcal{B}', \mathcal{C}')$ perfectly simulates $(\mathcal{A}, \mathcal{B}, \mathcal{C})$, and thus $(\mathcal{A}', \mathcal{B}', \mathcal{C}')$ wins the real-or-random anti-piracy security of the single-decryptor scheme with the same advantage δ, which concludes the proof.

A.4 Proof of Theorems 20 and 21

In this section, we formally define our monogamy-of-entanglement property, and prove its hardness. We also define a computational and parallel version of this property, and prove its hardness. This section is taken verbatim from Chevalier, Hermouet, and Vu (2024b).

In this section, we present a new monogamy-of-entanglement game for coset states and prove an upper-bound on the probability of winning this game. Along the way, we present a BB84 version of this game with the same upper-bound.

A.4.1 The Coset Version

Definition 41 (Monogamy-of-Entanglement Game with Identical Basis (Coset Version))**.** This game is between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ - where $\mathcal{B}$ and $\mathcal{C}$ are not communicating, and is parametrized by a security parameter λ.

- The challenger samples a subspace $A \leftarrow \{0,1\}^{\lambda \times \frac{\lambda}{2}}$ and two vectors $(s, s') \leftarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$. Then the challenger prepares the coset state $|A_{s,s'}\rangle$ and sends $|A_{s,s'}\rangle$ to $\mathcal{A}$.
- $\mathcal{A}$ prepares a bipartite quantum state σ_{12}, then sends σ_1 to $\mathcal{B}$ and σ_2 to $\mathcal{C}$.
- The challenger samples $b \leftarrow \{0,1\}$, then sends (A, b) to both $\mathcal{B}$ and $\mathcal{C}$.
- $\mathcal{B}$ returns u_1 and $\mathcal{C}$ returns u_2.

We say that $\mathcal{B}$ makes a correct guess if $(b = 0 \ \wedge \ u_1 \in A + s)$ or if $(b = 1 \ \wedge \ u_1 \in A^\perp + s')$. Similarly, we say that $\mathcal{C}$ makes a correct guess if $(b = 0 \ \wedge \ u_2 \in A + s)$ or if $(b = 1 \ \wedge \ u_2 \in A^\perp + s')$. We say that $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ makes a correct guess. For any triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$ for this game, we note $\mathsf{MoE}_{coset}(1^\lambda, \mathcal{A}, \mathcal{B}, \mathcal{C})$ the random variable indicating whether $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win the game or not.

We note that there is a trivial way for a triple of adversaries to win this game with probability 1/2, by applying the following strategy. $\mathcal{A}$ samples a random bit b^*. $\mathcal{A}$ measures $|A_{s,s'}\rangle$ in the computational basis if $b^* = 0$, or in the Hadamard basis if $b^* = 1$. In both cases, $\mathcal{A}$ sends the outcome u to both $\mathcal{B}$ and $\mathcal{C}$. Regardless of the value of A and b, $\mathcal{B}$ and $\mathcal{C}$ both return u. Because when $b^* = b$ (which happens with probability 1/2), the outcome of the measurement is a vector of the expected coset space, the adversaries win the game with probability 1/2. In the rest of this section we prove that no triple of adversaries can actually win the game with a probability significantly greater than 1/2.

Theorem 27. There exists a negligible function $\mathsf{negl}(\cdot)$ such that, for any triple of algorithms $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$, $\Pr\left[\mathsf{MoE}_{coset}(1^\lambda, \mathcal{A}, \mathcal{B}, \mathcal{C}) = 1\right] \leq 1/2 + \mathsf{negl}(\lambda)$.

The proof of this theorem is given in subsequent sections.

A.4.2 The BB84 Version

We introduce below the BB84 version of this game. We show in the following that it is sufficient to study the BB84 version (which is simpler) to prove Theorem 27, as any triple of adversaries for the BB84 version can be turned into a triple of adversaries for the coset version without changing the probability of winning.

Notations. Through all Appendix A.4.2 and Appendix A.4.3, we use the following notations. Let $n \in \mathbb{N}$, we note $\Theta_n := \{\theta \in \{0,1\}^n \ : \ |\theta| = n/2\}$ - where $|\cdot|$ denotes the Hamming weight - and $N := \binom{n}{n/2}$. Thus, Θ_λ has exactly N elements.

Definition 42 (Monogamy-of-Entanglement Game with Identical Basis (BB84 Version))**.** This game is between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ - where $\mathcal{B}$ and $\mathcal{C}$ are non-communicating, and is parametrized by a security parameter λ. An illustration of this game is depicted in Figure A.3.

- The challenger samples $x \leftarrow \{0,1\}^\lambda$ and $\theta \leftarrow \Theta_\lambda$. Then the challenger prepares the state $|x^\theta\rangle := \bigotimes_{i \in [\![1,\lambda]\!]} \mathsf{H}^{\theta_i} |x_i\rangle$ and sends $|x^\theta\rangle$ to $\mathcal{A}$.
- $\mathcal{A}$ prepares a bipartite quantum state σ_{12}, then sends σ_1 to $\mathcal{B}$ and σ_2 to $\mathcal{C}$.
- The challenger samples $b \leftarrow \{0,1\}$, then sends (θ, b) to both $\mathcal{B}$ and $\mathcal{C}$.
- $\mathcal{B}$ returns x_1 and $\mathcal{C}$ returns x_2.

Let $x_{T_b} := \{x_i \mid \theta_i = b\}$. We say that $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win the game if $x_1 = x_2 = x_{T_b}$. For any triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$ for this game, we note $\mathsf{MoE}_{BB84}(1^\lambda, \mathcal{A}, \mathcal{B}, \mathcal{C})$ the random variable indicating whether $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win the game or not.

We note that the trivial strategy for the coset version can be easily adapted for the BB84 one. Hence, the greatest probability of winning this game is also lower bounded by 1/2.

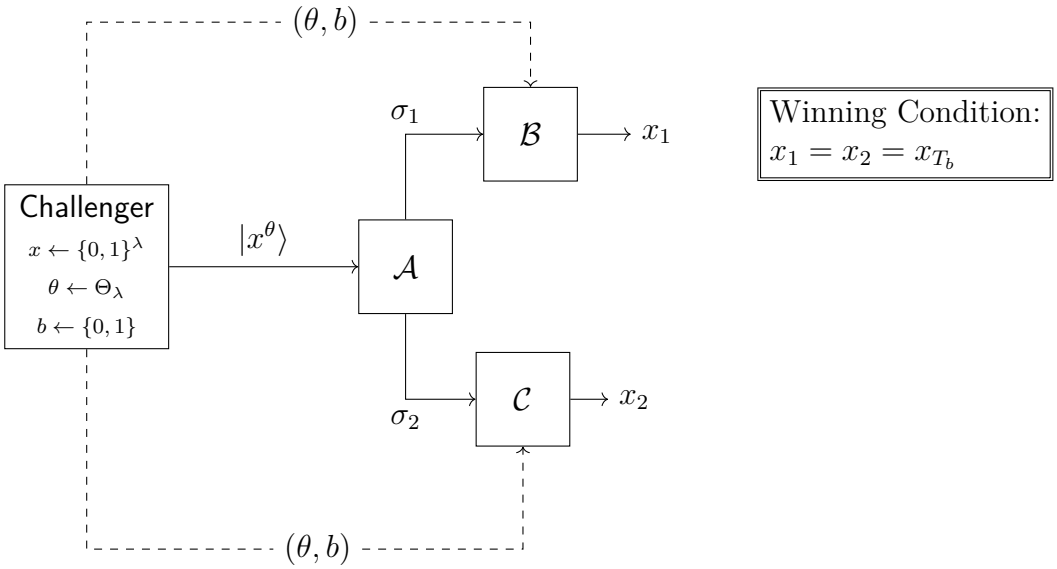

Figure A.3: Monogamy-of-Entanglement Game with Identical Basis (BB84 Version)

Theorem 28. There exists a negligible function $\mathsf{negl}(\cdot)$ such that, for any triple of algorithms $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$, $\Pr\left[\mathsf{MoE}_{BB84}(1^\lambda, \mathcal{A}, \mathcal{B}, \mathcal{C}) = 1\right] \leq 1/2 + \mathsf{negl}(\lambda)$.

Proof of Theorem 27 follows similarly as that of Culf and Vidick (2022), in which the winning probability of cloning adversaries in the monogamy-of-entanglement game of coset states reduces to the winning probability of the adversaries in the game of BB84 states. We thus provide the proof of Theorem 28 below.

A.4.3 Proof of Theorem 28

This proof follows the same structure as Culf and Vidick (2022). We can separate the proof in four main steps.

1. In the first step, we define the *extended non-local game* Johnston, Mittal, Russo, and Watrous (2016) associated the monogamy-of-entanglement game (BB84 version), and show that the greatest winning probability of the monogamy game is the same as the one of this extended non-local game. This step allows us to use a technique from Tomamichel, Fehr, Kaniewski, and Wehner (2013) to bound the winning probability.

2. In the second step, we express any strategy for this extended non-local game with security parameter $n \in \mathbb{N}$ as a tripartite quantum state ρ_{012} as well as two families of projective measurements, $\{B^{\theta,b}\}$ and $\{C^{\theta,b}\}$, both indexed by $\theta \in \Theta_n$ and $b \in \{0,1\}$. We define the projector $\Pi_{\theta,b} = \sum_{x\in\{0,1\}^n} |x\rangle\langle x|^\theta \otimes B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}$ such that the winning probability of this strategy is $p_{win} = \mathbb{E}_{\theta,b}\left[\mathsf{Tr}\left(\Pi_{\theta,b}\ \rho_{012}\right)\right]$. Then, we show the following upper-bound:

$$\begin{aligned} p_{win} &\leq \frac{1}{2N} \sum_{\substack{1\leq k\leq N \\ \alpha\in\{0,1\}}} \max_{\theta,b} \left\| \Pi_{\theta,b} \Pi_{\pi_{k,\alpha}(\theta\|b)} \right\| \\ &= \frac{1}{2} + \frac{1}{2N} \sum_{1\leq k\leq N} \max_{\theta,b} \left\| \Pi_{\theta,b} \Pi_{\pi_{k,1}(\theta\|b)} \right\| \end{aligned}$$

where $\{\pi_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ is a family of permutations to be defined later in the proof.

3. In the third step, we show that the quantity $\|\Pi_{\theta,b}\ \Pi_{\theta',b'}\|$ is upper-bounded by a small quantity as long as $b' \neq b$.

4. Finally, in the fourth step, we show that there exists a family of permutations such that, when $\alpha = 0$, $\pi_{k,\alpha}(\theta, b) = (\theta', b')$ for some θ' and $b' \neq b$, and conclude the proof.

Step 1: extended non-local game. We define the following extended non-local game, and show that any triple of adversaries that win the monogamy-of-entanglement game with same basis (BB84 version) with probability p can be turned into another triple of adversaries that win this extended non-local game with the same probability p.

Definition 43 (Extended Non-Local Game)**.** This game is between a challenger and two adversaries $\mathcal{A}$ and $\mathcal{B}$, and is parametrized by a security parameter λ.

- $\mathcal{B}$ and $\mathcal{C}$ jointly prepare a quantum state ρ_{012} - where ρ_0 is a λ-qubits quantum state, then send ρ_0 to the challenger. $\mathcal{B}$ and $\mathcal{C}$ keep ρ_1 and ρ_2 respectively. From this step $\mathcal{B}$ and $\mathcal{C}$ cannot communicate.

- The challenger samples $\theta \leftarrow \Theta_n$ and $b \leftarrow \{0,1\}$. Then, for all $i \in [\![1,\lambda]\!]$, the challenger measures the i^{th} qubit of ρ_0 in computational basis if $\theta_i = 0$ or in Hadamard basis if $\theta_i = 1$. Let $m \in \{0,1\}^n$ denote the measurement outcome. Finally, the challenger sends (θ, b) to $\mathcal{B}$ and $\mathcal{C}$.

- $\mathcal{B}$ returns m_1 and $\mathcal{C}$ returns m_2.

Let $m_{T_b} := \{m_i \mid \theta_i = b\}$. We say that $(\mathcal{B},\mathcal{C})$ win the game if $m_1 = m_2 = m_{T_b}$.

Lemma 3. Let $n \in \mathbb{N}$ and $(\mathcal{A},\mathcal{B},\mathcal{C})$ a triple of adversaries for the monogamy-of-entanglement game (Definition 42) parametrized by n, that win with probability p_n. Then there exists a quantum state ρ_{012} and a pair of adversaries $(\mathcal{A}'_1,\mathcal{A}'_2)$ for the extended non-local game (Definition 43) that win with the same probability p_n.

Proof. Consider a triple of adversaries for the monogamy-of-entanglement game (Definition 42), parametrized by $n \in \mathbb{N}$, that win with probability p_n. We can model these adversaries as a CPTP map $\Phi : \mathcal{H}_0 \to \mathcal{H}_1 \times \mathcal{H}_2$, and POVMs families $\{B^{\theta,b}\}$ and $\{C^{\theta,b}\}$, both indexed by $\theta \in \Theta_n$ and $b \in \{0,1\}$. Then we have

$$p_n = \mathop{\mathbb{E}}_{\substack{\theta\in\Theta_n\\ b\in\{0,1\}}} \mathop{\mathbb{E}}_{x\in\{0,1\}^n} Tr\left[(B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}})\Phi(|x^\theta\rangle\langle x^\theta|)\right].$$

The strategy for the extended non-local game is as follows. $\mathcal{B}$ and $\mathcal{C}$ prepare the bipartite state $\rho_{00'} = \bigotimes_{1\leq i\leq n} |\phi^+\rangle\langle\phi^+|$ where ϕ^+ denotes the EPR state $(|00\rangle + |11\rangle)/\sqrt{2}$, and where ρ_0 (resp. $\rho_{0'}$) is composed of the first halves (resp. second halves) of these EPR states. Then, they apply Φ to $\rho_{0'}$. Let ρ_{012} denotes the resulting state. They send ρ_0 to the challenger, $\mathcal{B}$ keeps ρ_1 and $\mathcal{C}$ keeps ρ_2. Later, when $\mathcal{B}$ receives (θ, b), from the challenger, $\mathcal{B}$ applies the POVM $B^{\theta,b}$ to ρ_1 and returns the outcome. $\mathcal{C}$ does the same with POVM $C^{\theta,b}$ and ρ_2. The probability of winning of such strategy is then

$$p'_n = \mathop{\mathbb{E}}_{\substack{\theta\in\Theta_n\\ b\in\{0,1\}}} \sum_{x\in\{0,1\}^n} Tr\left[\left(|x^\theta\rangle\langle x^\theta| \otimes B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\rho_{012}\right]. \tag{A.1}$$

We do the following calculation.

$$\begin{aligned}
Tr\left[\left(|x^\theta\rangle\langle x^\theta| \otimes B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\rho_{012}\right] &= \frac{1}{2^n}\sum_{r,r'\in\{0,1\}^n} Tr\left[\left(|x^\theta\rangle\langle x^\theta| \otimes B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\left(|r\rangle\langle r'| \otimes \Phi\left(|r\rangle\langle r'|\right)\right)\right] \\
&= \frac{1}{2^n}\sum_{r,r'\in\{0,1\}^n} \langle r|x^\theta\rangle\langle x^\theta|r'\rangle\, Tr\left[\left(B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\Phi\left(|r\rangle\langle r'|\right)\right] \\
&= \frac{1}{2^n}\sum_{r,r'\in\{0,1\}^n} Tr\left[\left(B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\Phi\left(|r\rangle\langle r|x^\theta\rangle\langle x^\theta|r'\rangle\langle r'|\right)\right] \\
&= \frac{1}{2^n} Tr\left[\left(B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\Phi\left(\frac{1}{2^n}\sum_{r\in\{0,1\}^n}|r\rangle\langle r|\;|x^\theta\rangle\langle x^\theta|\frac{1}{2^n}\sum_{r'\in\{0,1\}^n}|r'\rangle\langle r'|\right)\right] \\
&= \frac{1}{2^n} Tr\left[\left(B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}\right)\Phi\left(|x^\theta\rangle\langle x^\theta|\right)\right]
\end{aligned}$$

By plugging this result into Equation (A.1), we get $p'_n = p_n$, which concludes the proof. □

Step 2: first upper-bound of the winning probability. We prove an upper-bound for the extended non-local game above. We need the following lemma.

Lemma 4 (Lemma 2 of Tomamichel, Fehr, Kaniewski, and Wehner (2013))**.** Let $\Pi_1, \ldots, \Pi_n$ be projective positive semi-definite operators on a Hilbert space, and $\{\pi_i\}_{i\in[\![1,n]\!]}$ be a set of orthogonal permutations for some integer n. Then

$$\left\|\sum_{i=1}^n \Pi_i\right\| \leq \sum_{i=1}^n \max_{j\in[\![1,n]\!]} \left\|\Pi_j \Pi_{\pi_i(j)}\right\|$$

Let $\left(\{B^{\theta,b}\}_{\theta\in\Theta_n,b\in\{0,1\}}, \{C^{\theta,b}\}_{\theta\in\Theta_n,b\in\{0,1\}}, \rho_{012}\right)$ be a strategy for the extended non-local game. Using Naimark's dilation theorem, we can assume without loss of generality that the $B^{\theta,b}$ and $C^{\theta,b}$ are all projective. Let $\Pi_{\theta,b}$ be the following projector: $\Pi_{\theta,b} := \sum_{x\in\{0,1\}^n} |x\rangle\langle x|^\theta \otimes B^{\theta,b}_{x_{T_b}} \otimes C^{\theta,b}_{x_{T_b}}$. Then the winning probability of this strategy is

$$\begin{aligned}
p_{win} &= \mathop{\mathbb{E}}_{\theta\in\Theta_n,b\in\{0,1\}} \mathsf{Tr}\left(\Pi_{\theta,b}\;\rho_{012}\right) \\
&\leq \mathop{\mathbb{E}}_{\theta\in\Theta_n,b\in\{0,1\}} \|\Pi_{\theta,b}\| \\
&\leq \frac{1}{2N}\sum_{\substack{1\leq k\leq N \\ \alpha\in\{0,1\}}} \max_{\theta,b} \left\|\Pi_{\theta,b}\Pi_{\pi_{k,\alpha}(\theta,b)}\right\| \qquad \text{(A.2)}
\end{aligned}$$

where the first inequality follows from the definition of the norm and the second from Lemma 4; and where $\{\pi_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ is a family of mutually orthogonal permutations.

Step 3: upper-bound of $\|\Pi_{\theta,b}\Pi_{\theta',1-b}\|$. In this part, we show that for all $(\theta,\theta') \in \Theta_n$ and all $b \in \{0,1\}$, we can upper-bound $\|\Pi_{\theta,b}\Pi_{\theta',1-b}\|$ by a small quantity.

Let $(\theta,\theta') \in \Theta_n^2$ and $b \in \{0,1\}$. Note $R := \{i \in [\![1,N]\!] \;:\; \theta_i \neq \theta'_i\}$, $T := \{i \in [\![1,N]\!] \;:\; \theta_i = b\}$, $T' := \{i \in [\![1,N]\!] \;:\; \theta'_i = 1-b\}$ and $S := \{i \in R \;:\; \theta_i = b \text{ and } \theta'_i = 1-b\}$. We define $\bar{\mathrm{P}}$ and $\bar{\mathrm{Q}}$ as follows:

$$\begin{aligned}
\bar{\mathrm{P}} &:= \sum_{x_T\in\{0,1\}^T} \mathsf{H}^b|x_S\rangle\langle x_S|\mathsf{H}^b \otimes \mathbb{I}_{\bar{S}} \otimes B^{\theta,b}_{x_T} \otimes \mathbb{I}_C \\
\bar{\mathrm{Q}} &:= \sum_{x_{T'}\in\{0,1\}^{T'}} \mathsf{H}^{1-b}|x_S\rangle\langle x_S|\mathsf{H}^{1-b} \otimes \mathbb{I}_{\bar{S}} \otimes C^{\theta',1-b}_{x_{T'}} \otimes \mathbb{I}_B
\end{aligned}$$

where $|x_S\rangle\langle x_S|$ denotes the subsystem of $|x_T\rangle\langle x_T|$ whose indices belong to S, and $\mathbb{I}_{\bar{S}}$ denotes the rest of the system.

Remark that we have:

$$\begin{aligned}\|\Pi_{\theta,b}\Pi_{\theta',1-b}\|^2 &= \|\Pi_{\theta',1-b}\Pi_{\theta,b}\Pi_{\theta',1-b}\| \\ &\leq \|\Pi_{\theta',1-b}\bar{\mathrm{P}}\Pi_{\theta',1-b}\| \\ &= \|\bar{\mathrm{P}}\Pi_{\theta',1-b}\bar{\mathrm{P}}\| \\ &\leq \bar{\mathrm{P}}\bar{\mathrm{Q}}\bar{\mathrm{P}}\end{aligned}$$

where we have the first line because $\Pi_{\theta,b}$ is a projection, the second because $\Pi_{\theta,b} \leq \bar{\mathrm{P}}$, the third because $\Pi_{\theta,b}$ and $\bar{\mathrm{P}}$ are projections and the last because $\Pi_{\theta',1-b} \leq \bar{\mathrm{Q}}$.

Consider now the quantity $\bar{\mathrm{P}}\bar{\mathrm{Q}}\bar{\mathrm{P}}$. We compute the following upper-bound for $\bar{\mathrm{P}}\bar{\mathrm{Q}}\bar{\mathrm{P}}$:

$$\begin{aligned}\bar{\mathrm{P}}\bar{\mathrm{Q}}\bar{\mathrm{P}} &= \sum_{\substack{x_T,z_T\in\{0,1\}^T \\ y_{T'}\in\{0,1\}^{T'}}} \mathsf{H}^b|x_S\rangle\langle x_S|\mathsf{H}^b\mathsf{H}^{1-b}|y_S\rangle\langle y_S|\mathsf{H}^{1-b}\mathsf{H}^b|z_S\rangle\langle z_S|\mathsf{H}^b \otimes \mathbb{I}_{\bar{S}} \otimes B^{\theta,b}_{x_T}\, B^{\theta,b}_{z_T} \otimes C^{\theta',1-b}_{y_{T'}} \\ &= \sum_{\substack{x_T\in\{0,1\}^T \\ y_{T'}\in\{0,1\}^{T'}}} \mathsf{H}^b|x_S\rangle\langle x_S|\mathsf{H}^b\mathsf{H}^{1-b}|y_S\rangle\langle y_S|\mathsf{H}^{1-b}\mathsf{H}^b|x_S\rangle\langle x_S|\mathsf{H}^b \otimes \mathbb{I}_{\bar{S}} \otimes B^{\theta,b}_{x_T} \otimes C^{\theta',1-b}_{y_{T'}} \\ &= 2^{-|S|} \sum_{\substack{x_T\in\{0,1\}^T \\ y_{T'}\in\{0,1\}^{T'}}} \mathsf{H}^b|x_S\rangle\langle x_S|\mathsf{H}^b \otimes \mathbb{I}_{\bar{S}} \otimes B^{\theta,b}_{x_T} \otimes C^{\theta',1-b}_{y_{T'}} \\ &= 2^{-|S|} \sum_{x_T\in\{0,1\}^T} \mathsf{H}^b|x_S\rangle\langle x_S|\mathsf{H}^b \otimes \mathbb{I}_{\bar{S}} \otimes B^{\theta,b}_{x_T} \otimes \mathbb{I}_C\end{aligned}$$

where the first equality comes from $B^{\theta,b}_{x_T}\, B^{\theta,b}_{z_T} = B^{\theta,b}_{x_T}$ if $x_T = z_T$ and 0 otherwise; the second comes from $\langle x_S|\,\mathsf{H}^b\mathsf{H}^{1-b}|y_S\rangle\langle y_S|\mathsf{H}^{1-b}\mathsf{H}^b\,|x_S\rangle = |\langle x_S|\,\mathsf{H}\,|y_S\rangle|^2 = 2^{-|S|}$ for all $x_T, y_{T'}$ and the third from $\sum_{y_{T'}} C^{\theta',1-b}_{y_{T'}} = \mathbb{I}_C$. Notice that we can assume without loss of generality that $|S|$ is larger than $|R|/2$: if it is not the case, we just swap the roles of θ and θ'. Thus, by linearity and from $\sum_{x_T} B^{\theta,b}_{x_T} = \mathbb{I}_B$, it comes $\|\bar{\mathrm{P}}\bar{\mathrm{Q}}\bar{\mathrm{P}}\| \leq 2^{-|S|} \leq 2^{-|R|/2}$ hence

$$\|\Pi_{\theta,b}\Pi_{\theta',1-b}\| \leq 2^{-|R|/4} \tag{A.3}$$

Remark 5. Remark that, when considering $\|\Pi_{\theta,b}\Pi_{\theta',b}\|$ instead, we have $S = \emptyset$. Thus, the reasoning above yields the trivial upper-bound

$$\|\Pi_{\theta,b}\Pi_{\theta',b}\| \leq 1 \tag{A.4}$$

Step 4: finding the permutation family. In this part, we construct a family of mutually orthogonal permutations $\{\pi_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ such for all $k \in [\![1, N]\!]$, $\pi_{k,0}$ "flips" the last input's bit and $\pi_{k,1}$ leaves it unchanged.

We use the following lemma, proven in Culf and Vidick (2022).

Lemma 5 (Lemma 3.4 of Culf and Vidick (2022))**.** Let n be an even integer, $\Theta_n := \{\theta \in \{0,1\}^n \;:\; |\theta| = n/2\}$ and $N = \binom{n}{n/2}$. Then there is a family of N mutually orthogonal permutations $\{\tilde{\pi}_k\}_{k\in[\![1,N]\!]}$ of Θ_n such that the following holds. For each $i \in [\![1, n/2]\!]$, there are exactly $\binom{n/2}{i}^2$ permutations $\tilde{\pi}_k$ such that the number of positions at which θ and $\tilde{\pi}_k(\theta)$ are both 1 is $n/2 - i$.

We prove the following corollary.

Corollary 6. Let n be an even integer, $\Theta_n := \{\theta \in \{0,1\}^n \ : \ |\theta| = n/2\}$ and $N = \binom{n}{n/2}$. Then there is a family of $2N$ mutually orthogonal permutations $\{\pi_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ of $\Theta_n \times \{0,1\}$ such that the two following properties hold.

- For each $i \in [\![1, n/2]\!]$, there are exactly $\binom{n/2}{i}^2$ permutations $\pi_{k,0}$ such that the number of positions at which θ and θ' are both 1 is $n/2 - i$ (i.e. θ and θ' differ in $2i$ positions).
- If $\alpha = 0$, then $b' = 1 - b$. Otherwise, $b' = b$.

where we use the notation $(\theta' \| b') := \pi_{k,\alpha}(\theta \| b)$.

Proof. Let $\{\tilde{\pi}_k\}_{k\in[\![1,N]\!]}$ be a family of orthogonal permutations promised in Lemma 5. Define the family $\{\pi_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ as follows. For all $k \in [\![1, N]\!]$:

$$\pi_{k,0}(\theta||b) = \tilde{\pi}_k(\theta)||(1-b)$$
$$\pi_{k,1}(\theta||b) = \tilde{\pi}_k(\theta)||b$$

The two properties follow directly by construction. It remains to prove that these $2N$ permutations are mutually orthogonal. Assume $\pi_{k,\alpha}(\theta) = \pi_{k',\alpha'}(\theta)$. Then we have $\alpha = \alpha'$, and $\tilde{\pi}_k(\theta) = \tilde{\pi}_{k'}(\theta)$, hence $k = k'$ because $\{\tilde{\pi}_k\}_k$ is a family of orthogonal permutations. □

Concluding the proof. We make use of the following lemma from Culf and Vidick (2022).

Lemma 6 (Lemma 3.6 of Culf and Vidick (2022))**.** Let $n \geq 2$ an integer, and note $N = \binom{n}{n/2}$. Then we have

$$\frac{1}{N}\sum_{i=0}^{n/2}\binom{n/2}{i}^2 2^{-i/2} \leq \sqrt{e}\left(\cos\frac{\pi}{8}\right)^n$$

The rest of the proof follows easily. We first rewrite Equation (A.2) as

$$p_{win} \leq \frac{1}{2N}\sum_{k=1}^{N}\max_{\theta,b}\left\|\Pi_{\theta,b}\Pi_{\pi_{k,1}(\theta,b)}\right\| + \frac{1}{2N}\sum_{k=1}^{N}\max_{\theta,b}\left\|\Pi_{\theta,b}\Pi_{\pi_{k,0}(\theta,b)}\right\|$$

Then, by plugging the permutation's family of Corollary 6, and using the upper-bounds proved in Equation (A.3) and Equation (A.4), it comes

$$\begin{aligned} p_{win} &\leq \frac{1}{2} + \frac{1}{2N}\sum_{i=1}^{n/2} 2^{-i/2} \\ &\leq \frac{1}{2} + \frac{\sqrt{e}}{2}\left(\cos\frac{\pi}{8}\right)^n. \end{aligned}$$

A.4.4 Computational Version

We provide below a computational version of the monogamy-of-entanglement with identical basis. The only difference is that the adversaries are given access to obfuscated membership programs for the coset space and its dual. This game is still hard to win with probability significantly greater than 1/2 if we make the assumption that the adversaries are polynomially bounded. The proof of this statement follows directly from the proof of hardness of the computational version of the regular monogamy-of-entanglement game Coladangelo, Liu, Liu, and Zhandry (2021).

Definition 44 (Computational Monogamy-of-Entanglement Game with Identical Basis (Coset Version))**.** This game is between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ - where $\mathcal{B}$ and $\mathcal{C}$ are not communicating, and is parametrized by a security parameter λ.

- The challenger samples a subspace $A \leftarrow \{0,1\}^{\lambda \times \frac{\lambda}{2}}$ and two vectors $(s, s') \leftarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$. Then the challenger prepares the coset state $|A_{s,s'}\rangle$ as well as two obfuscated membership programs $\widehat{\mathsf{P}}_{A+s} := \mathsf{iO}(A+s)$ and $\widehat{\mathsf{P}}_{A^\perp+s'} := \mathsf{iO}(A^\perp+s')$ and sends $\left(|A_{s,s'}\rangle, \widehat{\mathsf{P}}_{A+s}, \widehat{\mathsf{P}}_{A^\perp+s'}\right)$ to $\mathcal{A}$.
- $\mathcal{A}$ prepares a bipartite quantum state σ_{12}, then sends σ_1 to $\mathcal{B}$ and σ_2 to $\mathcal{C}$.
- The challenger samples $b \leftarrow \{0,1\}$, then sends (A, b) to both $\mathcal{B}$ and $\mathcal{C}$.
- $\mathcal{B}$ returns u_1 and $\mathcal{C}$ returns u_2.

For $i \in \{1,2\}$, we say that $\mathcal{A}_i$ makes a correct guess if $(b = 0 \ \wedge \ u_i' \in A + s)$ or if $(b = 1 \ \wedge \ u_i' \in A^\perp + s')$. We say that $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ makes a correct guess. For any triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$ for this game, we note $\mathsf{MoE}_{coset(comp)}(1^\lambda, \mathcal{A}, \mathcal{B}, \mathcal{C})$ the random variable indicating whether $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win the game or not.

Theorem 29. There exists a negligible function $\mathsf{negl}(\cdot)$ such that, for any triple of QPT algorithms $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$, $\Pr\left[\mathsf{MoE}_{coset(comp)}(1^\lambda, \mathcal{A}, \mathcal{B}, \mathcal{C}) = 1\right] \leq 1/2 + \mathsf{negl}(\lambda)$.

A.4.5 Parallel Repetition of the Game

For our proof of anti-piracy of copy-protection, we actually need a parallel version of this game, where the challenger samples $\kappa \in \mathbb{N}$ independent cosets and an independent basis choice for each coset; and the adversaries are supposed to return a vector in the correct space for all the cosets to win the game. We show that the winning probability of this game is negligible.

Definition 45 (κ-Parallel Computational Monogamy-of-Entanglement Game with Identical Basis (Coset Version))**.** This game is between a challenger and a triple of adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ - where $\mathcal{B}$ and $\mathcal{C}$ are not communicating, and is parametrized by a security parameter λ.

- The challenger samples κ subspaces $\{A_i\}_{i \in [\![1,\kappa]\!]}$ and κ pairs of vectors $\{(s_i, s_i')\}_{i \in [\![1,\kappa]\!]}$ where $A_i \leftarrow \{0,1\}^{\lambda \times \frac{\lambda}{2}}$ and $(s_i, s_i') \leftarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$ for all $i \in [\![1, \kappa]\!]$. Then the challenger

prepares the coset states $\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,\kappa]\!]}$ as well as the associated obfuscated membership programs $\widehat{\mathsf{P}}_{A_i+s_i} := \mathsf{iO}(A_i + s_i)$ and $\widehat{\mathsf{P}}_{A_i^\perp+s'_i} := \mathsf{iO}(A_i^\perp + s'_i)$ for $i \in [\![1,\kappa]\!]$; and sends $\left(\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\![1,\kappa]\!]}, \{\widehat{\mathsf{P}}_{A_i+s_i}, \widehat{\mathsf{P}}_{A_i^\perp+s'_i}\}_{i\in[\![1,\kappa]\!]}\right)$ to $\mathcal{A}$.

- $\mathcal{A}$ prepares a bipartite quantum state σ_{12}, then sends σ_1 to $\mathcal{B}$ and σ_2 to $\mathcal{C}$.
- The challenger samples $r \leftarrow \{0,1\}^\kappa$, then sends $\{A_i\}_{i\in[\![1,\kappa]\!]}$ and r to both $\mathcal{B}$ and $\mathcal{C}$.
- $\mathcal{B}$ returns κ vectors $\{u_i\}_{i\in[\![1,\kappa]\!]}$ and $\mathcal{C}$ returns κ vectors $\{u'_i\}_{i\in[\![1,\kappa]\!]}$.

We say that $\mathcal{B}$ makes a correct guess if $(r_i = 0 \ \wedge \ u_i \in A_i + s_i)$ or if $(r_i = 1 \ \wedge \ u_i \in A_i^\perp + s'_i)$ for all $i \in [\![1,\kappa]\!]$. Similarly, we say that $\mathcal{C}$ makes a correct guess if $(r_i = 0 \ \wedge \ u'_i \in A_i + s_i)$ or if $(r_i = 1 \ \wedge \ u'_i \in A_i^\perp + s'_i)$ for all $i \in [\![1,\kappa]\!]$. We say that $(\mathcal{A},\mathcal{B},\mathcal{C})$ win the game if both $\mathcal{B}$ and $\mathcal{C}$ makes a correct guess. For any triple of adversaries $(\mathcal{A},\mathcal{B},\mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$ for this game, we note $\kappa - \mathsf{MoE}_{coset(comp)}(1^\lambda,\mathcal{A},\mathcal{B},\mathcal{C})$ the random variable indicating whether $(\mathcal{A},\mathcal{B},\mathcal{C})$ win the game or not.

Theorem 30. There exists a negligible function $\mathsf{negl}(\cdot)$ such that, for any triple of QPT algorithms $(\mathcal{A},\mathcal{B},\mathcal{C})$ and any security parameter $\lambda \in \mathbb{N}$, $\Pr\left[\kappa - \mathsf{MoE}_{coset(comp)}(1^\lambda,\mathcal{A},\mathcal{B},\mathcal{C}) = 1\right] \leq \mathsf{negl}(\lambda)$.

Comparison with Cakan, Goyal, Liu-Zhang, and Ribeiro (2024). In Cakan, Goyal, Liu-Zhang, and Ribeiro (2024), the authors also present a new monogamy-of-entanglement game for coset states. Their game is similar too our parallel version except that, instead of receiving the same challenge bitstring r, $\mathcal{B}$ and $\mathcal{C}$ receive respectively r_1 and r_2, two independently sampled challenge bitstrings, and must answer accordingly. Note that the hardness of the parallel version of our game can be proven using lemma 18 of Ananth, Kaleoglu, and Liu (2023) on their game[2]. We still provide a direct proof for this theorem in Appendix A.4.6 for completeness. We emphasize that for the single-instance version, however, the same lemma cannot be applied.

A.4.6 Proof of Parallel Version of the Monogamy Game

In this subsection, we prove Theorem 30. We do it by proving that a parallel version of the BB84 version of the monogamy game has negligible security, as the coset version follows as for the single instance. As the proof follows the same structure as the one of Theorem 28, we only describe here the important steps of the proof.

Step 1: extended non-local game. We first describe the extended non-local game for this parallel version of the game. This game is between a challenger and two adversaries $\mathcal{A}$ and $\mathcal{B}$, and is parametrized by a security parameter λ and a number of repetitions $\kappa := \mathsf{poly}(\lambda)$.

- $\mathcal{B}$ and $\mathcal{C}$ jointly prepare a quantum state ρ_{012} - where ρ_0 is composed of κ λ-qubits registers, denoted as $\rho_0^1,\ldots,\rho_0^\kappa$ - then send ρ_0 to the challenger. $\mathcal{B}$ and $\mathcal{C}$ keep ρ_1 and ρ_2 respectively. From this step $\mathcal{B}$ and $\mathcal{C}$ cannot communicate.

[2]We thank Alper Çakan and Vipul Goyal for pointing out this shorter proof.

- For $j \in [\![1,\kappa]\!]$, the challenger samples $\theta^j \leftarrow \Theta_n$, then the challenger samples $r \leftarrow \{0,1\}^\kappa$. Then, for all $i \in [\![1,\lambda]\!]$ and $j \in [\![1,\kappa]\!]$, the challenger measures the i^{th} qubit of ρ_0^j in computational basis if $\theta_i^j = 0$ or in Hadamard basis if $\theta_i^j = 1$. Let $m^j \in \{0,1\}^n$ denote the measurement outcome for every j. Finally, the challenger sends $\theta := (\theta^1, \dots, \theta^\kappa)$ and r to $\mathcal{B}$ and $\mathcal{C}$.
- $\mathcal{B}$ returns $\{m_1^j\}_{j\in[\![1,\kappa]\!]}$ and $\mathcal{C}$ returns $\{m_2^j\}_{j\in[\![1,\kappa]\!]}$.

Let $m^j_{T_{r_j}} := \{m_i^j \mid \theta_i^j = r_j\}$. We say that $(\mathcal{B},\mathcal{C})$ win the game if $m_1^j = m_2^j = m_{T_{r_j}}$ for all $j \in [\![1,\kappa]\!]$.

Step 2: first upper-bound. Let $\theta = (\theta^1, ..., \theta^\kappa)$, we define $\Pi_{\theta,r} := \bigotimes_{j=1}^{\kappa} \sum_{x\in\{0,1\}^n} |x\rangle\langle x|^{\theta^j} \otimes B^{\theta,r}_{x_{T_r}} \otimes C^{\theta,r}_{x_{T_r}}$. We then prove in the same way as in Theorem 28 that

$$p_{win} \leq \frac{1}{(2N)^\kappa} \sum_{\substack{k=k_1\|\dots\|k_\kappa \\ 1\leq k_j\leq N\ \forall j \\ \alpha\in\{0,1\}^\kappa}} \max_{\theta,r} \left\|\Pi_{\theta,r}\Pi_{\pi_{k,\alpha}(\theta,r)}\right\|$$

where $\{\pi_{k,\alpha}\}$ is a family of mutually orthogonal permutations indexed by $k = k_1\|\dots\|k_\kappa$ - where each $k_j \in [\![1,N]\!]$ - and $r \in \{0,1\}^\kappa$.

Step 3: upper-bound of $\|\Pi_{\theta,r}\Pi_{\theta',\bar{r}}\|$. Let $\theta = (\theta^1, ..., \theta^\kappa)$ and $\theta' = (\theta'^1, ..., \theta'^\kappa)$ where each θ^j and θ'^j belongs to Θ_n. Let $r \in \{0,1\}^\kappa$. For every $j \in [\![1,\kappa]\!]$, note $R^j := \{i \in [\![1,N]\!] \ : \ \theta_i^j \neq \theta_i'^j\}$, $T^j := \{i \in [\![1,N]\!] \ : \ \theta_i^j = r_j\}$, $T'^j := \{i \in [\![1,N]\!] \ : \ \theta_i'^j = 1 - r_j\}$ and $S^j := \{i \in R \ : \ \theta_i^j = r_j \text{ and } \theta_i'^j = 1 - r_j\}$. We define $\bar{\mathrm{P}}$ and $\bar{\mathrm{Q}}$ as follows:

$$\bar{\mathrm{P}} = \sum_{\substack{j\in[\![1,\kappa]\!] \\ x_{T^j}\in\{0,1\}^{T^j}}} \bigotimes_{j=1}^{\kappa} \mathsf{H}^{r_j}|x_{S^j}\rangle\langle x_{S^j}|\mathsf{H}^{r_j} \otimes \mathbb{I}_{\bar{S}^j} \otimes B^{\theta,r}_{x_T} \otimes \mathbb{I}_C$$

$$\bar{\mathrm{Q}} = \sum_{\substack{j\in[\![1,\kappa]\!] \\ x_{T'^j}\in\{0,1\}^{T'^j}}} \bigotimes_{j=1}^{\kappa} \mathsf{H}^{1-r_j}|x_{S^j}\rangle\langle x_{S^j}|\mathsf{H}^{1-r_j} \otimes \mathbb{I}_{\bar{S}^j} \otimes \mathbb{I}_B \otimes C^{\theta',1-\bar{r}}_{x_{T'}}$$

where $T := T^1\|\dots\|T^\kappa$, $|x_{S^j}\rangle\langle x_{S^j}|$ denotes the subsystem of $|x_{T^j}\rangle\langle x_{T^j}|$ whose indices belong to S^j, and $\mathbb{I}_{\bar{S}^j}$ denotes the rest of the system.

Following the same reasoning as in Theorem 28 (step 3), it comes

$$\|\Pi_{\theta,r}\Pi_{\theta',\bar{r}}\| \leq 2^{-\frac{\sum_j |R^j|}{4}}$$

Step 4: finding the permutation family. Let $\{\pi^\star_{k,\alpha}\}_{k\in[\![1,N]\!],\alpha\in\{0,1\}}$ denotes the permutation family defined in step 4 of Theorem 28. We define the permutation family $\{\pi_{k,\beta}\}$ - indexed by $k = k_1\|\dots\|k_\kappa$ where each $k_j \in [\![1,N]\!]$ and $\beta \in \{0,1\}^\kappa$ - as $\pi_{k,r}(\theta_1\|\dots\|\theta_\kappa, r) = \pi^\star_{k_1,\beta_1}(\theta_1,r_1)\|\dots\|\pi^\star_{k_\kappa,\beta_\kappa}(\theta_\kappa,r_\kappa)$. It is easy to see that this family is orthogonal and has the same required properties as in the single instance proof, that is that for every $j \in [\![1,\kappa]\!]$ and $i \in [\![1,n/2]\!]$, there are exactly $\binom{n/2}{i}^2$ permutations $\pi_{k,0}$ such

that the number of positions at which θ^j and θ'^j are both 1 is $n/2 - i$ (i.e. $|R^j| = 2^i$). Using this set of permutations we have:

$$\begin{aligned}
p_{win} &\leq \frac{1}{(2N)^\kappa} \sum_{\substack{k=k_1\|\ldots\|k_\kappa \\ \beta\in\{0,1\}^\kappa}} \max_{\substack{\theta=\theta_1\|\ldots,\|\theta_\kappa \\ r\in\{0,1\}^\kappa}} \|\Pi_{\theta,r}\Pi_{\theta',r'}\| \\
&= \frac{1}{(2N)^\kappa} \sum_{w=0}^{\kappa} \sum_{\substack{k=k_1\|\ldots\|k_\kappa \\ \beta\in\{0,1\}^\kappa,|\beta|=w}} \max_{\substack{\theta=\theta_1\|\ldots,\|\theta_\kappa \\ r\in\{0,1\}^\kappa}} \|\Pi_{\theta,r}\Pi_{\theta',r'}\| \\
&\leq \frac{1}{(2N)^\kappa} \sum_{w=0}^{\kappa} \binom{\kappa}{w} \left(\sum_{\ell=0}^{n/2} \binom{n/2}{\ell}^2 2^{-\ell/2}\right)^w \\
&= \frac{1}{(2N)^\kappa} \left(1 + \sum_{\ell=0}^{n/2} \binom{n/2}{\ell}^2 2^{-\ell/2}\right)^\kappa \\
&\leq \frac{1}{(2N)^\kappa} \left(1 + \binom{n/2}{n/4}^2 \sum_{\ell=0}^{n/2} 2^{-\ell/2}\right)^\kappa \\
&= \frac{1}{(2N)^\kappa} \left(1 + \binom{n/2}{n/4}^2 \frac{1 - 2^{-n/4-1/2}}{1 - 2^{-1/2}}\right)^\kappa
\end{aligned}$$

Where in the first equality, we split the sum over the possible weights of β; the first inequality comes from Corollary 6; we obtain the second equality by applying the binomial theorem; the second inequality comes from $\binom{n}{k} \leq \binom{n}{n/2}$ for all n, k; and the last inequality comes from the fact that the sum is the sum of a geometric series.

Using both Stirling approximation and asymptotic development of logarithm, we get that the logarithm of this last inequality decreases linearly in k, meaning that the upper bound is negligible in n which concludes the proof.

Appendix

B Semi-Quantum Unclonable Cryptography - Supplementary Materials

Most of this appendix is taken verbatim from our work (Chevalier, Hermouet, and Vu (2023)). We provide the detailed proofs for the different security properties of protocols 2 to 5. For convenience, we also provide in Appendix B.2 the definitions and protocols in the way they are presented in the original paper (Chevalier, Hermouet, and Vu (2023)), and useful preliminaries in Appendix B.1.

B.1 Preliminaries

In this section, we define decoding maps for extended trapdoor claw-free functions, provide lemmas used in the proofs of this appendix, and describe the sample-and-estimate framework of Bouman and Fehr (2010).

B.1.1 Extended Trapdoor Claw-free Functions

Our remote state preparation protocol is based on a cryptographic primitive called extended noisy trapdoor claw free function families (ENTCF families), which are defined in Mahadev (2018, Section 4) and can be constructed from the Learning with Errors assumption Regev (2005) and Brakerski, Christiano, Mahadev, Vazirani, and Vidick (2018). We use the same notation as in Mahadev (2018, Section 4), with the exception that we write $\mathcal{K}_0$ instead of $\mathcal{K}_{\mathcal{G}}$ and $\mathcal{K}_1$ instead of $\mathcal{K}_{\mathcal{F}}$. In addition, we also define the following functions for convenience:

Definition 46 (Decoding maps, Metger and Vidick (2021, Definition 2.1))**.**

1. For a key $k \in \mathcal{K}_0 \cup \mathcal{K}_1$, an image $y \in \mathcal{Y}$, a bit $b \in \{0,1\}$, and a pre-image $x \in \mathcal{X}$, we define $\mathsf{Chk}(k,y,b,x)$ to return 1 if $y \in \mathrm{Supp}(f_{k,b}(x))$, and 0 otherwise. (This definition is as in Mahadev (2018, Definition 4.1 and 4.2).)

2. For a key $k \in \mathcal{K}_0$ and a $y \in \mathcal{Y}$, we define $\hat{b}(k,y)$ by the condition $y \in \cup_x \mathrm{Supp}\Big(f_{k,\hat{b}(k,y)}(x)\Big)$. (This is well-defined because $f_{k,0}$ and $f_{k,1}$ form an injective pair.)

3. For a key $k \in \mathcal{K}_0 \cup \mathcal{K}_1$ and a $y \in \mathcal{Y}$, we define $\hat{x}_b(k,y)$ by the condition $y \in \mathrm{Supp}(f_{k,b}(\hat{x}_b(k,y)))$, and $\hat{x}_b(k,y) = \perp$ if $y \notin \cup_x \mathrm{Supp}(f_{k,b}(x))$. For $k \in \mathcal{K}_0$, we also use the shorthand $\hat{x}(k,y) \coloneqq \hat{x}_{\hat{b}(k,y)}(k,y)$.

4. For a key $k \in \mathcal{K}_1$, a $y \in \mathcal{Y}$, and a $d \in \{0,1\}^w$, we define $\hat{u}(k,y,d)$ by the condition $d \cdot (\hat{x}_0(k,y) \oplus \hat{x}_1(k,y)) = \hat{u}(k,y,d)$.

The above decoding maps applied to vector inputs are understood to act in an element-wise fashion. For example, for $\vec{k} \in \mathcal{K}_1^{\times n}, \vec{y} \in \mathcal{Y}^{\times n}$, and $\vec{d} \in \{0,1\}^{w \times n}$, we denote by $\hat{u}(\vec{k},\vec{y},\vec{d}) \in \{0,1\}^n$ the string defined by $\left(\hat{u}(\vec{k},\vec{y},\vec{d})\right)_i := \hat{u}(k_i,y_i,d_i)$.

B.1.2 Sampling in a Quantum Population

In this section, we describe a generic framework presented in Bouman and Fehr (2010) for analyzing cut-and-choose strategies applied to quantum states.

Classical Sampling Strategies

Let $q := (q_1,\ldots,q_n) \in \Omega^n$ be a string of length n. We consider the problem of estimating the relative Hamming weight of a substring $\omega(q|_{\bar{t}})$ by only looking at the substring $q|_t$ of q, for a subset $t \subset [\![1,n]\!]$. We consider sampling strategies $\Psi := (P_T, P_S, f)$, where P_T is an (independently sampled) distribution over subsets $t \subseteq [\![1,n]\!]$, P_S is a distribution over seeds $s \in S$, and $f : \{(t,v) : t \subset [\![1,n]\!], v \in \Omega^t\} \times S \to \mathbb{R}$ is a function that takes the subset t, the substring v, and a seed s, and outputs an estimate for the relative Hamming weight of the remaining string. For a fixed subset t, seed s, and a parameter δ, define $B^\delta_{t,s}(\Psi) \subseteq \Omega^n$ as

$$B^\delta_{t,s} := \{b \in \Omega^n : |\omega(b|_{\bar{t}}) - f(t, b|_t, s)| < \delta\}.$$

Then we define the *classical error probability* of strategy Ψ as follows.

Definition 47 (Classical Error Probability)**.** The classical error probability of a sampling strategy $\Psi := (P_T, P_S, f)$ is defined as the following value, parameterized by $0 < \delta < 1$:

$$\varepsilon^\delta_{\mathsf{classical}}(\Psi) := \max_{q \in \Omega^n} \Pr_{t \leftarrow P_T, s \leftarrow P_S}\left[q \notin B^\delta_{t,s}(\Psi)\right].$$

Quantum Sampling Strategies

Now, let $A := A_1,\ldots,A_n$ be an n-partite quantum system where the state space of each system A_i equals $\mathcal{H}_{A_i} = \mathbb{C}^d$ with $d = |\Omega|$, and let $\{|a\rangle\}_{a\in\Omega}$ be a fixed orthonormal basis of $\mathbb{C}^d$. A may be entangled with another system E, and we write the purified state on A and E as $|\psi\rangle_{AE}$. We consider the problem of testing whether the state on A is close to the all-zero reference state $|0\rangle_{A_1}\ldots|0\rangle_{A_n}$. There is a natural way to apply any sampling strategy $\Psi = (P_T, P_S, f)$ to this setting: sample t, s according to P_T, P_S, measure subsystems A_i for $i \in [\![1,t]\!]$ in basis $\{|a\rangle\}_a$ to observe $q|_t \in \Omega^{|t|}$, and compute an estimate $f(t, q|_t, s)$.

In order to analyze the effect of this strategy, we first consider the mixed state on registers T (holding the subset t), S (holding the seed s), and A, E that results from sampling t and s according to $P_{TS} := P_T P_S$

$$\rho_{TSAE} := \sum_{t,s} P_{TS}(t,s)\,|t,s\rangle\langle t,s|_{TS} \otimes |\psi\rangle\langle\psi|_{AE}.$$

Next, we compare this state to an *ideal* state, parameterized by $0 < \delta < 1$, of the form

$$\widetilde{\rho}_{TSAE} := \sum_{t,s} P_{TS}(t,s)\,|t,s\rangle\langle t,s|_{TS} \otimes |\widetilde{\psi}^{ts}\rangle\langle\widetilde{\psi}^{ts}|_{AE} \text{ with } |\psi^{ts}\rangle_{AE} \in \mathsf{span}\left(B^\delta_{t,s}\right) \otimes \mathcal{H}_E,$$

where

$$\mathsf{span}\left(B^{\delta}_{t,s}\right) \coloneqq \mathsf{span}\left(\{|b\rangle : b \in B^{\delta}_{t,s}\}\right) = \mathsf{span}\left(\{|b\rangle : |\omega(b|_{\bar{t}}) - f(t, b|_t, s)| < \delta\}\right).$$

That is, $\widetilde{\rho}_{TSAE}$ is a state such that it holds *with certainty* that the state on registers $A|_{\bar{t}}E$, after having measured $A|_t$ and observing $q|_t$, is in a superposition of states with relative Hamming weight δ-close to $f(t, q|_t, s)$. This leads us to the definition of the *quantum error probability* of strategy Ψ.

Definition 48 (Quantum Error Probability)**.** The quantum error probability of a sampling strategy $\Psi \coloneqq (P_T, P_S, f)$ is defined as the following value, parameterized by $0 < \delta < 1$:

$$\varepsilon^{\delta}_{\mathsf{quantum}}(\Psi) \coloneqq \max_{\mathcal{H}_E} \max_{|\psi\rangle_{AE}} \min_{\widetilde{\rho}_{TSAE}} \Delta\left(\rho_{TSAE}, \widetilde{\rho}_{TSAE}\right),$$

where the first max is over all finite-dimensional registers E, the second max is over all state $|\psi\rangle_{AE}$ and the min is over all ideal state $\widetilde{\rho}_{TSAE}$ of the form described above.

Finally, we relate the classical and quantum error probabilities.

Theorem 31 (Bouman and Fehr (2010))**.** For any sampling strategy Ψ and $\delta > 0$,

$$\varepsilon^{\delta}_{\mathsf{quantum}}(\Psi) \leq \sqrt{\varepsilon^{\delta}_{\mathsf{classical}}(\Psi)}.$$

Remark 6. The results presented here immediately generalize from the all-zero reference state $|0\rangle \dots |0\rangle$ to an arbitrary reference state $|\varphi\rangle_A$ of the form $|\varphi\rangle_A = U_1 |0\rangle \dots U_n |0\rangle$ for unitary operators U_i acting on $\mathbb{C}^d$. Indeed, the generalization follows simply by a suitable change of basis, defined by the U_i's.

In this work, we will only need to analyze one simple sample-and-estimate strategy $\Psi_{\mathsf{uniform}} \coloneqq (P_T, P_S, f)$, where P_T is the uniform distribution over subsets $t \subseteq [\![1, n]\!]$, P_S is empty and $f(t, q|_t) = \omega(q|_t)$. That is, f receives a uniformly random subset $q|_t$ of q, and outputs the relative Hamming weight of $q|_t$ as its guess for the relative Hamming weight of $q|_{\bar{t}}$. The classical error probability of this strategy can be bound using Hoeffding inequalities, which is done in Bouman and Fehr (2010, Appendix B.3), where it is shown to be bounded by $4\exp(\frac{-n\delta^2}{32})$ for parameter δ. Thus, we have the following corollary of Theorem 31.

Corollary 7. The quantum error probability of Ψ_{uniform} with parameter δ is

$$\varepsilon^{\delta}_{\mathsf{quantum}}(\Psi_{\mathsf{uniform}}) \leq 2\exp(\frac{-n\delta^2}{64}).$$

B.1.3 Properties of the State-Dependent Distance

A feature of the state-dependent distance is that if two operators are close in the state-dependent distance, we can replace one operator by the other *acting on either side of the state*.

Lemma 7 (Replacement lemma Metger and Vidick (2021, Lemma 2.21))**.** Let $\psi \in \mathrm{Pos}(\mathcal{H})$, and $A, B, C \in \mathcal{L}(\mathcal{H})$. If $A \approx_{\epsilon,\psi} B$ and $\|C\|_\infty = O(1)$, then

$$\mathrm{Tr}[CA\psi] \approx_{\epsilon^{1/2}} \mathrm{Tr}[CB\psi]\ , \tag{B.1}$$
$$\mathrm{Tr}[AC\psi] \approx_{\epsilon^{1/2}} \mathrm{Tr}[BC\psi]\ . \tag{B.2}$$

Lemma 8 (Metger and Vidick (2021, Lemma 2.22))**.** Let $A, B \in \mathcal{L}(\mathcal{H})$ be linear operators, $C \in \mathcal{L}(\mathcal{H})$ a linear operator with constant operator norm, and $\psi \in \mathrm{Pos}(\mathcal{H})$ with $\mathrm{Tr}[\psi] \leq 1$. Then, the following holds:

$$A \approx_{\epsilon,\psi} B \implies A\psi C \approx_{\epsilon} B\psi C \quad \text{and} \quad C\psi A^{\dagger} \approx_{\epsilon} C\psi B^{\dagger}. \tag{B.3}$$

The following lemma allows us to replace computationally indistinguishable states with one another in the state-dependent distance. This means that if two states are computationally indistinguishable and a state-dependent operator relation holds for one of the states, we can "lift" this relation to the other state, provided the operators are efficient.

Lemma 9 (Lifting lemma Metger and Vidick (2021, Lemma 2.25))**.** Let $\psi, \psi' \in \mathcal{D}(\mathcal{H})$ such that $\psi \overset{c}{\approx}_{\delta} \psi'$. Let $\mathcal{H}'$ be another Hilbert space with $\dim(\mathcal{H}') \geq \dim(\mathcal{H})$. For this case, let $\psi, \psi' \in \mathcal{D}(\mathcal{H}')$ such that $\psi \overset{c}{\approx}_{\delta} \psi'$. Let A be an efficient binary observable on $\mathcal{H}$, B an efficient binary observable on $\mathcal{H}'$, and $V : \mathcal{H} \to \mathcal{H}'$ an efficient isometry. Then:

$$VAV^{\dagger} \approx_{\epsilon,\psi} B \implies VAV^{\dagger} \approx_{\epsilon^{1/4}+\delta,\psi'} B. \tag{B.4}$$

Finally, we recall some further miscellaneous properties of the state-dependent distance.

Lemma 10 (Metger and Vidick (2021, Lemma 2.18))**.** Let $\psi_i \in \mathrm{Pos}(\mathcal{H})$ for $i \in \{1, \ldots, n\}$ with constant n, and $A, B \in \mathcal{L}(\mathcal{H})$. Define $\psi = \sum_i \psi_i$. Then:

$$\forall i \in [\![1, n]\!] : \; A \approx_{\epsilon,\psi_i} B \text{ iff } A \approx_{\epsilon,\psi} B \tag{B.5}$$

Lemma 11 (Metger and Vidick (2021, Lemma 2.24))**.** Let $\mathcal{H}_1, \mathcal{H}_2$ be Hilbert spaces with $\dim(\mathcal{H}_1) \leq \dim(\mathcal{H}_2)$ and $V : \mathcal{H}_1 \to \mathcal{H}_2$ an isometry. Let A and B be binary observables on $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, $\psi \in \mathrm{Pos}(\mathcal{H}_1)$, and $\epsilon \geq 0$. Then for any $b \in \{0, 1\}$:

$$V^{\dagger}BV \approx_{\epsilon,\psi} A \implies V^{\dagger}B^{(b)}V \approx_{\epsilon,\psi} A^{(b)}, \tag{B.6}$$

$$B \approx_{\epsilon,V\psi V^{\dagger}} VAV^{\dagger} \implies B^{(b)} \approx_{\epsilon,V\psi V^{\dagger}} VA^{(b)}V^{\dagger}. \tag{B.7}$$

B.2 Definitions and Protocols

In this section, we introduce our protocol for remote hidden coset state preparation. We first give a definition of completeness and soundness in Appendix B.2.1. Our construction is given in Appendix B.2.2.

B.2.1 Definitions

Definition 49 (Remote Coset State Preparation Protocol)**.** A remote coset state preparation protocol is an interactive classical communication protocol between a PPT verifier (or sender, denoted as V) and a QPT prover (or receiver, denoted as P) such that at the end of the protocol, the verifier obtains a list $T \subset \mathbb{N}$ of classical description of cosets $\{S_i, \alpha_i, \beta_i\}_{i \in T}$ and the prover outputs a quantum state ψ. The two parties also obtain a common output which is obfuscated membership checking programs of $S_i + \alpha_i$ and $S_i^{\perp} + \beta_i$ for all $i \in T$.

We denote an execution of the protocol as $(\{S_i, \alpha_i, \beta_i\}_{i\in T}, \psi, \{P_{0,i}, P_{1,i}\}_{i\in T}) \leftarrow \langle \mathsf{P}(1^\lambda), \mathsf{V}(1^\lambda)\rangle$, where $P_{0,i}$ is an obfuscated membership checking program of $S_i + \alpha_i$ and $P_{1,i}$ is an obfuscated membership checking program of $S_i^\perp + \beta_i$. Note that $\{P_{0,i}, P_{1,i}\}_{i\in T}$ is the common output of both parties. When it is clear from the context, we omit the common output and just write $(\{S_i, \alpha_i, \beta_i\}_{i\in T}, \psi) \leftarrow \langle \mathsf{P}(1^\lambda), \mathsf{V}(1^\lambda)\rangle$.

The protocol is *correct* if the protocol does not abort and at the end of the execution, there exists a negligible function $\varepsilon(\lambda)$ such that

$$\Pr\left[\psi \approx_\varepsilon \bigotimes_{i\in T} |S_{i,\alpha_i,\beta_i}\rangle\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over randomness of the verifier V.

We now formally define the notions of soundness of remote coset state preparation protocol. We will give two different definitions: one for the monogamy-of-entanglement property (Definition 50), and another for the direct product hardness property (Definition 51).

Definition 50 (Monogamy-of-Entanglement Soundness)**.** Let $(\{S_i, \alpha_i, \beta_i\}_{i\in T}, \psi) \leftarrow \langle \mathsf{P}_\lambda(\rho_\lambda), \mathsf{V}(1^\lambda)\rangle$ be an execution of a remote coset state preparation protocol between a QPT prover $\mathsf{P} = \{\mathsf{P}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ and a PPT verifier V, after which V outputs $\{S_i, \alpha_i, \beta_i\}_{i\in T}$ and P outputs a state ψ. The prover (now modeled as a triple algorithm $(\mathsf{P}, \mathcal{B}, \mathcal{C})$) then interacts with the verifier in the following monogamy game.

1. The prover applies a CPTP map to split ψ into a bipartite state ψ_{BC}; it sends the register B to $\mathcal{B}$ and the register C to $\mathcal{C}$. No communication is allowed between $\mathcal{B}$ and $\mathcal{C}$ after this phase.

2. Question. The verifier sends the description of $\{S_i\}_{i\in T}$, to both $\mathcal{B}$ and $\mathcal{C}$.

3. Answer. $\mathcal{B}$ returns $s_1^{(i)} \in \mathbb{F}_2^n$ and $\mathcal{C}$ returns $s_2^{(i)} \in \mathbb{F}_2^n$ for all $i \in T$.

The prover $(\mathsf{P}, \mathcal{B}, \mathcal{C})$ wins if and only if $s_1^{(i)} \in S_i + \alpha_i$ and $s_2^{(i)} \in S_i^\perp + \beta_i$ for all $i \in T$. Let $\mathsf{SMCosetMonogamy}(\mathsf{P}, \lambda)$ be a random variable which takes the value 1 if the game above is won by the prover $(\mathsf{P}, \mathcal{B}, \mathcal{C})$, and takes the value 0 otherwise.

The protocol is secure if the winning probability of any QPT adversary is negligible. Formally, for any QPT malicious prover, the protocol is *computationally sound with the monogamy-of-entanglement property* if

$$\Pr[\mathsf{SMCosetMonogamy}(\mathsf{P}, \lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Definition 51 (Direct Product Soundness)**.** Let $(\{S_i, \alpha_i, \beta_i\}_{i\in T}, \psi) \leftarrow \langle \mathsf{P}_\lambda(\rho_\lambda), \mathsf{V}(1^\lambda)\rangle$ be an execution of a remote coset state preparation protocol between a QPT prover $\mathsf{P} = \{\mathsf{P}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ and a PPT verifier V, after which V outputs $\{S_i, \alpha_i, \beta_i\}_{i\in T}$ and P outputs a state ψ. The prover then outputs $\{(v_i, w_i)_{i\in T}\}$. The prover wins if and only if for all $i \in T$, either:

(i) $(v_i, w_i) \in (A_i + s_i) \times (A_i + s_i)$ and $v_i \neq w_i$;

(ii) or $(v_i, w_i) \in (A_i^\perp + s_i') \times (A_i^\perp + s_i')$ and $v_i \neq w_i$;

(iii) or $(v_i, w_i) \in (A_i + s_i) \times (A_i^\perp + s_i')$.

Let $\mathsf{SMDirectProduct}(\mathsf{P}, \lambda)$ be a random variable which takes the value 1 if the game above is won by the prover P, and takes the value 0 otherwise.

The protocol is secure if the winning probability of any QPT adversary is negligible. Formally, for any QPT malicious prover, the protocol is *computationally sound with the direct product hardness property* if

$$\Pr[\mathsf{SMDirectProduct}(\mathsf{P}, \lambda) = 1] \leq \mathsf{negl}(\lambda).$$

B.2.2 Construction

Notation. Our protocol 6 and protocol 7 will be (almost) a parallel repetition of a sub-protocol. We make use of vector notation to denote tuples of items corresponding to the different copies of the sub-protocol. For example, if each of the n parallel sub-protocols requires a key k_i, we denote $\vec{k} = (k_1, \ldots, k_n)$. A function that takes as input a single value can be extended to input vectors in the obvious way: for example, if f takes as input a single key k, then we write $f(\vec{k})$ for the vector $(f(k_1), \ldots, f(k_n))$. We will also use $\vec{0}$ and $\vec{1}$ for the bit strings consisting only of 0 and 1, respectively (and whose length will be clear from the context), and $\vec{1^i} \in \{0,1\}^n$ for the bit string whose i-th bit is 1 and whose remaining bits are 0. Let n the length of a vector in a coset state (i.e., if $v \in A$ then $|v| = n$). In our constructions below, we set $n \coloneqq 2\lambda$.

Ingredients. Our constructions use the following building blocks:

- A quantum hybrid fully homomorphic encryption scheme $\mathsf{QFHE} \coloneqq \langle \mathsf{KeyGen}, \mathsf{QOTP}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec} \rangle$, with sub-exponential advantage security.
- A post-quantum secure indistinguishability obfuscation scheme iO.
- A post-quantum secure extended noisy trapdoor claw-free function (ENTCF) family $(\mathcal{F}, \mathcal{G})$.

Our main protocol's construction is given in protocol 10. The protocol involves two parties: a QPT prover (or receiver, denoted as P), and a PPT verifier (or sender, denoted as V).

Protocol 6: Semi-Quantum Protocol: BB84 Test Round

Input. The verifier initially receives Pauli keys (α, β) with $\alpha, \beta \in \{0,1\}^n$ as private inputs.

1. The verifier selects a uniformly random basis $\theta \leftarrow\!\!\$\ \{0,1\}$, where 0 corresponds to the computational and 1 to the Hadamard basis.
2. The verifier samples keys and trapdoors $\{(k_i, t_i)\}_{i=1}^n$ by computing $(k_i, t_i) \leftarrow \mathsf{Gen}_{\mathcal{K}_\theta}(1^\lambda)$. The verifier then sends $\{k_i\}_{i=1}^n$ to the prover (but keeps the trapdoors $\{t_i\}_{i=1}^n$ private).
3. The verifier receives $\{y_i\}_{i=1}^n$ where $y_i \in \mathcal{Y}$ from the prover.
4. The verifier selects a round type $\in \{\text{pre-image round}, \text{Hadamard round}\}$ uniformly at random and sends the round type to the prover.
   (a) For a *pre-image round*: the verifier receives $\{(b_i, x_i)\}_{i=1}^n$ from the prover, with

$b_i \in \{0,1\}$, and $x_i \in \mathcal{X}$. The verifier sets $\texttt{flag}_{\texttt{bb84}} \leftarrow \texttt{flag}_{\texttt{Pre}}$ and aborts if $\mathsf{Chk}(k_i, t_i, b_i, x_i) = 0$ for any $i \in [\![1, n]\!]$.

(b) For a *Hadamard round*: the verifier receives $\{d_i\}_{i=1}^n$ from the prover with $d_i \in \{0,1\}^w$ (for some w depends on the security parameter λ). The verifier sends $q = \theta$ to the prover, and receives answers $\{v_i\}_{i=1}^n$ with $v_i \in \{0,1\}$. The verifier performs the following:

- If $q = \theta = 0$, set $\texttt{flag}_{\texttt{bb84}} \leftarrow \texttt{flag}_{\texttt{Had}}$ and abort if $\hat{b}(k_i, y_i) \neq v_i$ for some $i \in [\![1, n]\!]$.
- If $q = \theta = 1$, set $\texttt{flag}_{\texttt{bb84}} \leftarrow \texttt{flag}_{\texttt{Had}}$ and abort if $\hat{u}(k_i, y_i, d_i) \neq v_i \oplus \beta_i$ for some $i \in [\![1, n]\!]$.

Protocol 7: Semi-Quantum Protocol: Coset-state Test Round

Input. The verifier initially receives a subspace $A \subseteq \mathbb{F}_2^n$ and Pauli keys (α, β) with $\alpha, \beta \in \{0,1\}^n$ as private inputs.

1. The verifier selects a uniformly random basis $\theta \leftarrow\!\$\ \{0,1\}$, where 0 corresponds to the computational and 1 to the Hadamard basis.
2. The verifier samples keys and trapdoors $\{(k_i, t_i)\}_{i=1}^n$ by computing $(k_i, t_i) \leftarrow \mathsf{Gen}_{\mathcal{K}_\theta}(1^\lambda)$. The verifier then sends $\{k_i\}_{i=1}^n$ to the prover (but keeps the trapdoors $\{t_i\}_{i=1}^n$ private).
3. The verifier receives $\{y_i\}_{i=1}^n$ where $y_i \in \mathcal{Y}$ from the prover.
4. The verifier sends "Hadamard round" as the round type to the prover.
5. The verifier receives $\{d_i\}_{i=1}^n$ from the prover with $d_i \in \{0,1\}^w$ (for some w depends on the security parameter λ). The verifier sends $q = \theta$ to the prover, and receives answers $\{v_i\}_{i=1}^n$ with $v_i \in \{0,1\}$.

   The verifier performs the following:

   - If $q = \theta = 0$, let $\vec{v} \coloneqq v_1 \dots v_n$. Set $\texttt{flag}_{\texttt{coset}} \leftarrow \texttt{flag}_{\texttt{Had}}$ and abort if $\vec{v} \notin A + \alpha$.
   - If $q = \theta = 1$, let $s_i \leftarrow v_i \oplus \hat{u}(k_i, y_i, d_i)$ and let $s \coloneqq s_1 \dots s_n$. Set $\texttt{flag}_{\texttt{coset}} \leftarrow \texttt{flag}_{\texttt{Had}}$ and abort if $\vec{s} \notin A^\perp + \beta$.

Protocol 8: Semi-Quantum Protocol: Self-Testing

Let M^2 the maximum number of test rounds (for $M \in \mathbb{N}$).

Input. The verifier initially receives a subspace $A \subseteq \mathbb{F}_2^n$ and Pauli keys (α', β') and $\{(\alpha_i, \beta_i)\}_{i=1}^{M^2}$ with $\alpha', \beta', \alpha_i, \beta_i \in \{0,1\}^n$ as private inputs. Note that (A, α', β') corresponds to one coset-state instance, and $\{(\alpha_i, \beta_i)\}_{i=1}^{M^2}$ corresponds to M^2 BB84 instances.

1. The verifier privately samples $B \leftarrow\!\$\ [\![1, M-1]\!]$ (this determines the number of BB84 test rounds that will be performed).
2. The verifier performs BM executions of protocol 6 (with corresponding private inputs $\{(\alpha_i, \beta_i)\}$) with the prover. The verifier aborts if protocol 6 aborts for some

execution.

3. The verifier privately samples $R \leftarrow\!\$\ [\![1, M]\!]$ and executes protocol 6 with the prover $R-1$ times (with corresponding private inputs $\{(\alpha_i, \beta_i)\}$). Then the verifier executes protocol 7 with the prover (with private inputs (A, α', β')) and aborts if protocol 7 aborts.

Protocol 9: Semi-Quantum Protocol: Self-Testing (with Soundness Amplification)

Let $N := \lambda$ the number of iterations.
Input. The verifier initially receives $\{(A_i, \alpha'_i, \beta'_i)\}_{i=1}^{N}$ and $\{(\alpha_i, \beta_i)\}_{i=1}^{NM^2}$ as private inputs. Each tuple in the first set corresponds to a coset-state instance, and each tuple in the second set corresponds to a BB84 instance.
The verifier and the prover sequentially run protocol 8 N times as follows.

1. For each run, the verifier and the prover interactively run protocol 8 with one coset state instance $(A_i, \alpha'_i, \beta'_i)$ and M^2 BB84 instances $\{(\alpha_i, \beta_i)\}_{i=1}^{M^2}$, each is picked uniformly at random from the input sets. (If some instance has been picked before, it will be excluded).

2. The verifier aborts unless protocol 8 does not abort in all N iterations.

Protocol 10: Semi-Quantum Protocol: Main Protocol

Verifier's preparation.

1. **Coset-state instances.** For each $i \in [\![1, 2N]\!]$, the verifier samples a random $\frac{n}{2}$-dimensional subspace $S_i \subseteq \mathbb{F}_2^n$, described by a matrix $\mathbf{M}_{S_i} \in \{0,1\}^{\frac{n}{2}\times n}$. Samples Pauli keys $p_{\alpha_i} \leftarrow\!\$\ \{0,1\}^{\frac{n^2}{2}}$ to encrypt $\mathbf{M}_{S_i}^{p_{\alpha_i}} \leftarrow \mathsf{QFHE.QOTP}(p_{\alpha_i}, \mathbf{M}_{S_i})$, and then $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{QFHE.KeyGen}(1^\lambda, 1^{\ell(\lambda)})$ for some polynomial $\ell(\cdot)$, $\mathsf{ct}_i \leftarrow \mathsf{QFHE.Enc}(\mathsf{pk}_i, p_{\alpha_i})$.

2. **n-qubit BB84 instances.** For each $i \in [\![1, NM^2]\!]$, the verifier samples Pauli keys $p_{\alpha_i} \leftarrow\!\$\ \{0,1\}^{\frac{n^2}{2}}$ to encrypt $\mathbf{M}_0^{p_{\alpha_i}} \leftarrow \mathsf{QFHE.QOTP}(p_{\alpha_i}, \mathbf{M}_0)$ (here, $\mathbf{M}_0$ is the all-zero vector of length $\frac{n^2}{2}$), and then $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{QFHE.KeyGen}(1^\lambda, 1^{\ell(\lambda)})$, $\mathsf{ct}_i \leftarrow \mathsf{QFHE.Enc}(\mathsf{pk}_i, p_{\alpha_i})$.

3. For each index $i \in [\![1, 2N + NM^2]\!]$, the verifier picks uniformly at random one instance from either the set of (encrypted) coset states or the set of (encrypted) n-qubit BB84 states prepared above. For each index i, denote the i-th instance as $(\mathsf{pk}_i, \mathbf{M}^{p_{\alpha_i}}, \mathsf{ct}_i)$ with secrets (sk_i, S_i). (If this instance is from the set of n-qubit BB84 states, we understand that $S_i = \mathbf{M}_0$.)

4. The verifier sends $\{\mathsf{pk}_i, \mathbf{M}^{p_{\alpha_i}}, \mathsf{ct}_i\}_{i=1}^{2N+NM^2}$ to the prover.

Prover's homomorphic evaluation

5. Let C the quantum circuit that for an input matrix $\mathbf{M} \in \{0,1\}^{\frac{n}{2}\times n}$, outputs a uniform superposition of its row span, except that if $\mathbf{M} = \mathbf{M}_0$, it outputs a uniform superposition of all vectors in the space $\mathbb{F}_2^n$. The prover homomorphically evaluates C for each $i \in [\![1, 2N + NM^2]\!]$: $(|S_{i,\alpha_i,\beta_i}\rangle, \mathsf{ct}_{i,\alpha_i,\beta_i}) \leftarrow \mathsf{QFHE.Eval}\Big(\mathsf{pk}_i, (\mathbf{M}^{p_{\alpha_i}}, \mathsf{ct}_i), C\Big)$, saves the quantum part $|S_{i,\alpha_i,\beta_i}\rangle$ and sends the classical part $\mathsf{ct}_{i,\alpha_i,\beta_i}$ to the verifier.

Self-testing for the prover.

6. For each $i \in [\![1, 2N+NM^2]\!]$, the verifier decrypts $(\alpha_i, \beta_i) \leftarrow \mathsf{QFHE.Dec}(\mathsf{sk}_i, \mathsf{ct}_{i,\alpha_i,\beta_i})$. For all coset-state instances, if $\alpha_i \in S_i$, the protocol is terminated.

7. The verifier then runs protocol 9 with these NM^2 prepared BB84 instances and N coset-state instances, where each coset-state instance is picked uniformly at random among $2N$ prepared instances. (If some instance has been picked before, it will be excluded). It aborts if protocol 9 aborts.

Coset-state generation.

8. The verifier samples a random $\frac{n}{2}$-dimensional coset $(\hat{S}, \hat{\alpha}, \hat{\beta}) \subseteq \mathbb{F}_2^n$ independently.[a] Let $\mathbf{M}_{\hat{S}}, \mathbf{M}_{\hat{S}^\perp} \in \{0,1\}^{\frac{n}{2}\times n}$ bases for $\hat{S}$ and $\hat{S}^\perp$, respectively.

9. Let T the set of indexes of the remaining N instances of the coset-states which have not been used in the self-testing protocol above. For each $i \in T$, the verifier does the following:

   (a) Let $\mathbf{M}_{S_i^\perp} \in \{0,1\}^{\frac{n}{2}\times n}$ a basis for $S_i^\perp$ (as a matrix). Compute indistinguishability obfuscations $P_{0,i} \leftarrow \mathsf{iO}\left(\mathsf{iO}(\mathbf{M}_{S_i} + \alpha_i) \vee \mathsf{iO}(\mathbf{M}_{\hat{S}} + \hat{\alpha})\right)$ and $P_{1,i} \leftarrow \mathsf{iO}\left(\mathsf{iO}(\mathbf{M}_{S_i^\perp} + \beta_i) \vee \mathsf{iO}(\mathbf{M}_{\hat{S}^\perp} + \hat{\beta})\right)$, all with appropriate padding.[b]

   (b) Record $\{(\alpha_i, \beta_i, S_i)\}_{i\in T}$.

   (c) Send T and $\{P_{0,i}, P_{1,i}\}_{i\in T}$ to the prover.

The output of the prover is $\{P_{0,i}, P_{1,i}, |S_{i,\alpha_i,\beta_i}\rangle\}_{i\in T}$ where $|T| = N$.

[a] This step is merely an artifact that we will need later for the security proof.

[b] Here, we understand that for any two programs C, C' with binary output, $\mathsf{iO}(C \vee C')(x)$ outputs $C(x) \vee C'(x)$.

B.3 Rigidity and soundness of protocols 6 to 8

In this section, we first prove rigidity statements of protocols 6 and 7. Then, we prove the soundness of the self-testing protocol (protocol 8). Finally, we prove the soundness of the final protocol (protocol 10).

The rigidity argument we establish in this section for protocol 8 is based on the n-fold parallel rigidity proof from Gheorghiu, Metger, and Poremba (2022). We will make frequent use of some technical lemmas from the proof of that paper.

B.3.1 Modeling a General Prover

Devices

We model the actions of a general prover by a "device". This formalizes all possible actions that can be taken by the prover to compute his answers to the verifier in protocol 6 and protocol 7. By Naimark's theorem, up to adding dimensions to the prover's Hilbert space, we can assume without loss of generality that the prover only performs projective measurements (instead of more general POVMs).

Definition 52 (Devices Gheorghiu, Metger, and Poremba (2022))**.** A device $D := (S, \Pi, M, P)$ is specified by the following:

1. A set $S = \{\psi^{(\vec{\theta})}\}_{\vec{\theta}\in\{0,1\}^n}$ of states $\psi^{(\vec{\theta})} \in \mathcal{D}(\mathcal{H}_D \otimes \mathcal{H}_Y)$, where $\dim(\mathcal{H}_Y) = |\mathcal{Y}|^n$ and the states are classical on $\mathcal{H}_Y$:

$$\psi^{(\vec{\theta})} = \sum_{\vec{y}\in\mathcal{Y}^n} \psi_{\vec{y}}^{(\vec{\theta})} \otimes |\vec{y}\rangle\langle\vec{y}|_Y \,. \tag{B.8}$$

   In the context of protocol 6 and protocol 7, $\psi^{(\vec{\theta})}$ is the prover's state after returning $\vec{y}$ for the case where the verifier makes basis choices $\vec{\theta}$.[1] Each $\psi^{(\vec{\theta})}$ also implicitly depends on the specific keys chosen by the verifier (not just the basis choice $\vec{\theta}$); all the statements we make hold on average over key choices (for a fixed basis choice $\vec{\theta}$). Furthermore, since protocol 6 and protocol 7 are actually used as sub-protocols in a bigger protocol (protocol 10), $\psi^{(\vec{\theta})}$ also depends on all messages exchanged (before the executions of these sub-protocols) in protocol 10; for clarity we suppress this dependence from the notation, as we will see later these dependencies do not affect the rigidity proofs of these sub-protocols.

2. In the case of protocol 6, a projective measurement Π on $\mathcal{H}_D \otimes \mathcal{H}_Y$:

$$\Pi = \left\{ \Pi^{(\vec{b},\vec{x})} = \sum_{\vec{y}} \Pi_{\vec{y}}^{(\vec{b},\vec{x})} \otimes |\vec{y}\rangle\langle\vec{y}|_Y \right\}_{\vec{b}\in\{0,1\}^n;\, \vec{x}\in\mathcal{X}^n} . \tag{B.9}$$

   This is the measurement used by the prover to compute his answer $(\vec{b}, \vec{x})$ in the pre-image challenge.

3. In the case of protocol 7, Π is the identity operator $\mathcal{I}$ on $\mathcal{H}_D \otimes \mathcal{H}_Y$. This is because in protocol 7, there is no pre-image challenge.

4. A projective measurement M on $\mathcal{H}_D \otimes \mathcal{H}_Y$:

$$M = \left\{ M^{(\vec{d})} = \sum_{\vec{y}} M_{\vec{y}}^{(\vec{d})} \otimes |\vec{y}\rangle\langle\vec{y}|_Y \right\}_{\vec{d}\in\{0,1\}^{w\times n}} . \tag{B.10}$$

   This is the measurement used by the prover to compute his answer $\vec{d}$ in the Hadamard challenge. We use an additional Hilbert spaces $\mathcal{H}_R$ to record the outcomes of measuring M and write the post-measurement state after applying M to $\psi^{(\vec{\theta})}$ as

$$\sigma^{(\vec{\theta})} := \sum_{\vec{y},\vec{d}} M_{\vec{y}}^{(\vec{d})} \psi_{\vec{y}}^{(\vec{\theta})} M_{\vec{y}}^{(\vec{d})} \otimes |\vec{y}, \vec{d}\rangle\langle\vec{y}, \vec{d}|_{YR} \,. \tag{B.11}$$

[1] In protocol 6, the only two basis choices are $\vec{\theta} = \vec{0}$ and $\vec{\theta} = \vec{1}$. However, $\psi^{(\vec{\theta})}$ is still well-defined as the state that the prover (who is defined in terms of the quantum circuits he runs on a given input) would prepare if given keys of basis choice $\vec{\theta}$, even though this never occurs in protocol 6. This is different from protocol 7, as it is crucial for the verifier's procedure in protocol 7 to use only $\vec{0}$ or $\vec{1}$ as the basis choice. Otherwise the protocol would be "undefined".

5. A set $P = \{P_q\}$, where for each $q \in \{0,1\}$, P_q is a projective measurement on $\mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R$:

$$P_q = \left\{ P_q^{(\vec{v})} = \sum_{\vec{y},\vec{d}} P_{q,\vec{y},\vec{d}}^{(\vec{v})} \otimes |\vec{y},\vec{d}\rangle\langle\vec{y},\vec{d}|_{YR} \right\}_{\vec{v}\in\{0,1\}^n} . \tag{B.12}$$

   In the context of protocol 6 and protocol 7, given question q, the prover will measure $\{P_q^{(\vec{v})}\}$ and return the outcome $\vec{v}$ as his answer.

Definition 53 (Efficient devices)**.** A device is called *efficient* if the states $\psi^{(\vec{\theta})}$ can be prepared efficiently and the measurements Π, M, and P_q can be performed efficiently.

Success Probabilities of a Device

During the self-testing protocol (protocol 8), the verifier applies certain checks to the answers given by the prover. If the prover fails these checks, the verifier sets a flag to $\texttt{flag}_{\texttt{Pre}}$ or $\texttt{flag}_{\texttt{Had}}$ then aborts. Here, we define the probabilities that the prover passes these checks and relate these probabilities in both protocols protocol 6 and protocol 7.

Definition 54 (Success probabilities)**.** For any device $D \coloneqq (S, \Pi, M, P)$ we define $\gamma_P(D_{\texttt{bb84}})$ as the device's failure probability in a pre-image round, $\gamma_H(D_{\texttt{bb84}})$ as the failure probability in a Hadamard round in protocol 6 and $\gamma_H(D_{\texttt{coset}})$ as the failure probability in a Hadamard round in protocol 7:

$$\gamma_P(D_{\texttt{bb84}}) \coloneqq \Pr\left[\texttt{flag}_{\texttt{bb84}} = \texttt{flag}_{\texttt{Pre}} \mid \text{round type} = \text{pre-image round}\right], \tag{B.13}$$

$$\gamma_H(D_{\texttt{bb84}}) \coloneqq \Pr\left[\texttt{flag}_{\texttt{bb84}} = \texttt{flag}_{\texttt{Had}} \mid \text{round type} = \text{Hadamard round}\right], \tag{B.14}$$

$$\gamma_H(D_{\texttt{coset}}) \coloneqq \Pr\left[\texttt{flag}_{\texttt{coset}} = \texttt{flag}_{\texttt{Had}}\right]. \tag{B.15}$$

Next, we give the definition of a perfect prover in protocol 6. Informally, a perfect prover is accepted by the verifier in a pre-image round with probability negligibly close to 1.

Definition 55 (Perfect device in protocol 6)**.** We call a device D *perfect* if $\gamma_P(D_{\texttt{bb84}}) = \mathsf{negl}(\lambda)$.

The following lemma says that for any device in protocol 6 that has a non-negligible failure probability in the pre-image test, there is another perfect device that is "close" to the original one in the sense that its measurements are the same as for the original device and its states only differ by $O(\gamma_P(D))$. By using this lemma, for the rest of the rigidity proof, it suffices to only consider perfect devices: for any arbitrary device, we can first make a reduction to the corresponding perfect device at the cost of incurring an approximation error of $O(\gamma_P(D))$, and then apply our soundness proof to the perfect device.

Lemma 12 (Gheorghiu, Metger, and Poremba (2022, Lemma 4.9))**.** Let $D = (S, \Pi, M, P)$ be an efficient device in protocol 6 with $\gamma_P(D_{\texttt{bb84}}) < 1$, where $S = \left\{\psi^{(\vec{\theta})}\right\}$. Then there exists an efficient *perfect* device $D' = (S', \Pi, M, P)$, which uses the same measurements Π, M, P and whose states $S' = \left\{\psi'^{(\vec{\theta})}\right\}$ satisfy for any $\vec{\theta} \in \{0,1\}^n$:

$$\psi'^{(\vec{\theta})} \approx_{\gamma_P(D_{\texttt{bb84}})} \psi^{(\vec{\theta})} . \tag{B.16}$$

Proof. The proof of this lemma uses essentially the same technique to that of Metger and Vidick (2021, Lemma 4.13), which in turn based on Mahadev (2018, Claim 7.2). We give a sketch of the proof for correctness. A construction of D' is as follows. D' first prepares the states $\psi^{(\vec{\theta})}$ as D does, then applies the efficient unitary U_Π associated with the measurement Π:

$$|0\rangle\langle 0|_R \otimes \psi^{(\vec{\theta})} \xmapsto{U_\Pi} |\vec{b},\vec{x}\rangle\langle\vec{b},\vec{x}|_R \otimes \Pi^{(\vec{b},\vec{x})}\psi^{(\vec{\theta})}\Pi^{(\vec{b},\vec{x})}. \tag{B.17}$$

Now D' coherently evaluates the (efficient) Chk-function on the Y-register of $\Pi^{(\vec{b},\vec{x})}\psi^{(\vec{\theta})}\Pi^{(\vec{b},\vec{x})}$ and the new register containing (b_i, x_i) for all $i \in [\![1, n]\!]$. If Chk succeeds, D' applies $U_\Pi^\dagger$ to the state, traces out the ancillary register R, and uses this as $\psi'^{(\vec{\theta})}$. Otherwise, D' repeats the process up to polynomially (in the security parameter) many times, and aborts if the Chk procedure never succeeds. Since $\gamma_P(D_{\mathtt{bb84}})$ is defined as the maximum failure probability of the pre-image test, and the Chk procedure fails if the pre-image check fails on any qubit, the probability of the Chk procedure failing is at most $n \cdot \gamma_P(D_{\mathtt{bb84}}) = O(\gamma_P(D_{\mathtt{bb84}}))$ by a union bound.

If $1-\gamma_P(D_{\mathtt{bb84}})$ is negligible, the trace distance bound between $\psi^{(\vec{\theta})}$ and $\psi'^{(\vec{\theta})}$ is trivially satisfied. If $1-\gamma_P(D_{\mathtt{bb84}})$ is non-negligible, the probability that Chk fails polynomially many times is negligible. Furthermore, by definition of the ENTCF family, the Chk procedure requires only the function key and not the trapdoor, which implies that it can be computed efficiently by the prover D'. It means that D' is efficient and perfect.

Fix $\vec{\theta}$. By Definition 3, we need to show $\left\|\psi'^{(\vec{\theta})} - \psi^{(\vec{\theta})}\right\|_1 \approx_{\gamma_P(D_{\mathtt{bb84}})^{1/2}} 0$. Since the probability of the Chk to succeed is at least $1-O(\gamma_P(D_{\mathtt{bb84}}))$, by the gentle measurement lemma (Wilde (2011)), the post-measurement state after Chk has succeeded is $O(\gamma_P(D_{\mathtt{bb84}})^{1/2})$-close in trace distance to $U_\Pi(|0\rangle\langle 0|_R \otimes \psi^{(\vec{\theta})})U_\Pi^\dagger$. Because the trace distance is unitarily invariant, this implies that the state $\psi'^{(\vec{\theta})}$ is also $O(\gamma_P(D_{\mathtt{bb84}})^{1/2})$-close in trace distance to $\psi^{(\vec{\theta})}$. □

B.3.2 Rigidity Proof of protocol 6

The rigidity proof of protocol 6 follows identically from that of Gheorghiu, Metger, and Poremba (2022). In this section, we recall definitions and related technical lemmas from Gheorghiu, Metger, and Poremba (2022) that are needed for our proof later. The main difference lies in the last verification procedure, in which our verification procedure also involves the Pauli keys from the QFHE. However, one can easily inspect their proof and see that this difference does not change most part of the proof. This essentially follows from the fact that the one-time pads (and generally, the homomorphic enryption) are independent of all the messages and verifier's secrets in the execution of protocol 6, it only is used in the verification of the verifier as its secret input. When the difference appears, we will re-prove the lemma with respect to our protocol.

Definition 56 (Observables)**.** For a device $D \coloneqq (S, \Pi, M, P)$ with projective measurements

as in Definition 52 and $\vec{\beta} \in \{0,1\}^n$, we define the following binary observables:

$$Z_i = \sum_{\vec{v}} (-1)^{v_i} P_0^{(\vec{v})}\,, \tag{B.18}$$

$$X_i = \sum_{\vec{v}} (-1)^{v_i} P_1^{(\vec{v})}\,, \tag{B.19}$$

$$\tilde{X}_i = \sum_{\vec{v},\vec{y},\vec{d}} (-1)^{\beta_i \oplus v_i \oplus \hat{u}(k_i,y_i,d_i)} P_{1,\vec{y},\vec{d}}^{(\vec{v})} \otimes |\vec{y},\vec{d}\rangle\langle\vec{y},\vec{d}|_{YR}\,. \tag{B.20}$$

We further use the following notation for products of observables: for $\vec{a} \in \{0,1\}^n$, we define

$$Z(\vec{a}) \coloneqq Z_1^{a_1} \dots Z_n^{a_n} = \sum_{\vec{v}} (-1)^{\vec{a}\cdot\vec{v}} P_0^{(\vec{v})}\,, \tag{B.21}$$

and likewise for $X(\vec{a})$ and $\tilde{X}(\vec{a})$. It is easy to see that

$$\tilde{X}(\vec{a})_{\vec{y},\vec{d}} = (-1)^{\vec{a}\cdot\left(\vec{\beta}\oplus\hat{u}(\vec{k},\vec{y},\vec{d})\right)} X(\vec{a})_{\vec{y},\vec{d}}. \tag{B.22}$$

Remark 7. $\tilde{X}_i$ is not an observable that an efficient prover can implement because it depends on $\hat{u}(k,y,d)$, which requires the trapdoor information to be computed efficiently, and the Pauli key β, which the prover only has an encryption of it. Intuitively, while X_i describes the prover's answer, $\tilde{X}_i$ describes whether that answer is accepted by the verifier.

Definition 57 (Partial post-measurement states)**.** For $k \in \mathcal{K}_0 \cup \mathcal{K}_1$, $v \in \{0,1\}$ and $\beta \in \{0,1\}$ define the set $V_{\beta,k,v} \subseteq \mathcal{Y} \times \{0,1\}^w$ by the following condition:

$$(y,d) \in V_{\beta,k,v} \text{ iff } \begin{cases} \hat{b}(k,y) = v & \text{if } k \in \mathcal{K}_0\,, \\ \hat{u}(k,y,d) = v \oplus \beta & \text{if } k \in \mathcal{K}_1\,. \end{cases} \tag{B.23}$$

Then for $\vec{\beta},\vec{k},\vec{\theta},\vec{v}$ we define

$$\sigma^{(\vec{\beta},\vec{\theta},\vec{v})} = \sum_{y_1,d_1 \in V_{\beta_1,k_1,v_1}} \cdots \sum_{y_n,d_n \in V_{\beta_n,k_n,v_n}} \sigma_{\vec{y},\vec{d}}^{(\vec{\theta})} \otimes |\vec{y},\vec{d}\rangle\langle\vec{y},\vec{d}|\,. \tag{B.24}$$

Further for $\vec{a} \in \{0,1\}^n$ we define

$$\sigma^{(\vec{\beta},\vec{\theta},v,\vec{a})} \coloneqq \sum_{\vec{v}:\,\vec{v}\cdot\vec{a}=v} \sigma^{(\vec{\beta},\vec{\theta},\vec{v})}\,. \tag{B.25}$$

Remark 8. In the following, once $\vec{\beta}$ is fixed, we can drop $\vec{\beta}$ from these notations and simply write $\sigma^{(\vec{\theta},\vec{v})}$ and $\sigma^{(\vec{\theta},v,\vec{a})}$. The reason is that as we explained above, the involvement of $\vec{\beta}$ is primarily a technicality needed because of our protocol construction, but does not affect the modular proofs we present here. Another way to see it is to consider $\vec{\beta}$ as a part of the trapdoor information $\vec{t}$. Then we can write $\hat{u}'(k,y,d) \coloneqq \hat{u}(k,y,d) \oplus \beta$ and define $(y,d) \in V_{k,v}$ if $\hat{u}'(k,y,d) = v$ when $k \in \mathcal{K}_1$. For any statement involving these states, we understand that there is some $\vec{\beta}$ known by the verifier and these states are defined with respect to this $\vec{\beta}$.

Intuitively, when $\vec{\theta} = \vec{0}$, then for any $\vec{a} \in \{0,1\}^n$, $\sigma^{(\vec{0},v,\vec{a})}$ is that part of the state $\sigma^{(\vec{0})}$ for which the honest device would receive outcome v when measuring the observable $Z(\vec{a})$. The following lemma shows what outcomes a successful device must produce when measuring the observables from Definition 56 on the partial post-measurement states from Definition 57.

Lemma 13 (Gheorghiu, Metger, and Poremba (2022, Corollary 4.18))**.** Consider an efficient device $D = (S, \Pi, M, P)$ and a bit $v \in \{0,1\}$.

1. For any $\vec{\theta}, \vec{a} \in \{0,1\}^n$ such that $\theta_i = 0$ if $a_i = 1$, then:

$$Z(\vec{a}) \approx_{\gamma_H(D_{\mathsf{bb84}}),\sigma^{(\vec{\theta},v,\vec{a})}} (-1)^v \mathcal{I}\,. \tag{B.26}$$

2. For any $\vec{\theta}, \vec{a} \in \{0,1\}^n$ such that $\theta_i = 1$ if $a_i = 1$, then:

$$X(\vec{a}) \approx_{\gamma_H(D_{\mathsf{bb84}}),\sigma^{(\vec{\theta},v,\vec{a})}} (-1)^v \mathcal{I}\,. \tag{B.27}$$

Next, we define isometries $\tilde{V}, V$ which can be shown to map the prover's observables to the corresponding Pauli observables.

Definition 58 (Rounding isometries Gheorghiu, Metger, and Poremba (2022))**.** For a device D with associated Hilbert space $\mathcal{H}_D$ and $\vec{y} \in \mathcal{Y}^{\times n}$, $d \in \{0,1\}^{w\times n}$, we define the isometry $\tilde{V}_{y,d} : \mathcal{H}_D \to \mathcal{H}_D \otimes \mathcal{H}_A \otimes \mathcal{H}_Q$ by the following action on an arbitrary state $|\varphi\rangle_D$:

$$\tilde{V}_{\vec{y},\vec{d}}|\varphi\rangle_D \coloneqq \mathop{\mathbb{E}}_{\vec{a},\vec{b}\in\{0,1\}^n} \left(\left(\tilde{X}(\vec{a})_{\vec{y},\vec{d}} Z(\vec{b})_{\vec{y},\vec{d}}\right)_D \otimes \left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_A\right)|\varphi\rangle_D \otimes \left(|\Phi^+\rangle^{\otimes n}\right)_{AQ}\,, \tag{B.28}$$

where $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ denotes an EPR pair, and $\left(|\Phi^+\rangle^{\otimes n}\right)_{AQ}$ is distributed between A and Q such that every EPR pair has one qubit in either system. We can combine the different $V_{y,d}$ into one isometry

$$\tilde{V} \coloneqq \sum_{\vec{y},\vec{d}} \tilde{V}_{\vec{y},\vec{d}} \otimes |\vec{y},\vec{d}\rangle\langle\vec{y},\vec{d}| : \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R \to \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R \otimes \mathcal{H}_A \otimes \mathcal{H}_Q\,. \tag{B.29}$$

We similarly define

$$V_{\vec{y},\vec{d}}|\varphi\rangle_D \coloneqq \mathop{\mathbb{E}}_{\vec{a},\vec{b}\in\{0,1\}^n} \left(\left(X(\vec{a})_{\vec{y},\vec{d}} Z(\vec{b})_{\vec{y},\vec{d}}\right)_D \otimes \left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_A\right)|\varphi\rangle_D \otimes \left(|\Phi^+\rangle^{\otimes n}\right)_{AQ} \tag{B.30}$$

and

$$V \coloneqq \sum_{\vec{y},\vec{d}} V_{\vec{y},\vec{d}} \otimes |\vec{y},\vec{d}\rangle\langle\vec{y},\vec{d}|\,. \tag{B.31}$$

The following lemma relates $\tilde{V}$ and V.

Lemma 14. For any keys $\vec{k} \in \mathcal{K}_1^n$ and $\vec{\beta} \in \{0,1\}^n$:

$$V_{\vec{y},\vec{d}} = \sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{\beta}\right)_A \otimes \sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{\beta}\right)_Q \tilde{V}_{\vec{y},\vec{d}}. \tag{B.32}$$

Proof. For any state $|\varphi\rangle_D$, we have:

$$\begin{aligned}
&\sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{\beta}\right)_A \otimes \sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{\beta}\right)_Q \tilde{V}_{\vec{y},\vec{d}}|\varphi\rangle_D \\
&= \mathop{\mathbb{E}}_{a,b\in\{0,1\}^n} \left(\tilde{X}(\vec{a})_{\vec{y},\vec{d}} Z(\vec{b})_{\vec{y},\vec{d}}\right)_D |\varphi\rangle_D \otimes \\
&\qquad \left[\left(\sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{\beta}\right)\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_A \otimes \sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{\beta}\right)_Q \left(|\Phi^+\rangle^{\otimes n}\right)_{AQ}\right]
\end{aligned}$$

Repeatedly using that $(\sigma_Z)_A |\Phi^+\rangle_{AQ} = (\sigma_Z)_Q |\Phi^+\rangle_{AQ}$:

$$= \mathop{\mathbb{E}}_{a,b\in\{0,1\}^n} \left(\tilde{X}(\vec{a})_{\vec{y},\vec{d}} Z(\vec{b})_{\vec{y},\vec{d}}\right)_D |\varphi\rangle_D \otimes \left[\left(\sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right)\sigma_X(\vec{a})\sigma_Z(\vec{b})\sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right)\right)_A \left(|\Phi^+\rangle^{\otimes n}\right)_{AQ}\right]$$

Since $\sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right)\sigma_X(\vec{a})\sigma_Z(\vec{b})\sigma_Z\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right) = (-1)^{\vec{a}\cdot\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right)}\sigma_X(\vec{a})\sigma_Z(\vec{b})$:

$$= \mathop{\mathbb{E}}_{a,b\in\{0,1\}^n} \left((-1)^{\vec{a}\cdot\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right)}\tilde{X}(\vec{a})_{\vec{y},\vec{d}} Z(\vec{b})_{\vec{y},\vec{d}}\right)_D |\varphi\rangle_D \otimes \left[\left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_A \left(|\Phi^+\rangle^{\otimes n}\right)_{AQ}\right]$$

Recalling from Definition 56 that $(-1)^{\vec{a}\cdot\left(\hat{u}(\vec{k},\vec{y},\vec{d})\oplus\vec{\beta}\right)}\tilde{X}(\vec{a})_{\vec{y},\vec{d}} = X(\vec{a})_{\vec{y},\vec{d}}$:

$$= \mathop{\mathbb{E}}_{a,b\in\{0,1\}^n} \left(X(\vec{a})_{\vec{y},\vec{d}} Z(\vec{b})_{\vec{y},\vec{d}}\right)_D |\varphi\rangle_D \otimes \left[\left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_A \left(|\Phi^+\rangle^{\otimes n}\right)_{AQ}\right]$$
$$= V|\varphi\rangle_D \,. \qquad \square$$

We then show that the isometry $\tilde{V}$ maps the observables $\tilde{X}(\vec{a})Z(\vec{b})$ to the corresponding Pauli observables.

Lemma 15 (Gheorghiu, Metger, and Poremba (2022, Lemma 4.28))**.** For an efficient perfect device $D = (S, \Pi, M, P)$ and any $\vec{a}, \vec{b} \in \{0,1\}^n$ we have

$$\mathrm{Tr}\left[\tilde{V}^\dagger \left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_Q^\dagger \tilde{V}\tilde{X}(\vec{a})_{DYR} Z(\vec{b})_{DYR}\sigma^{(\vec{1})}_{DYR}\right] \approx_{n^{1/2}\gamma_H(D_{\mathsf{bb84}})^{1/8}} 1\,. \tag{B.33}$$

By combining Lemma 14 and Lemma 15 we can show that the isometry V maps the observables $X(\vec{a})Z(\vec{b})$ to the corresponding Pauli observables.

Lemma 16 (Gheorghiu, Metger, and Poremba (2022, Proposition 4.29))**.** For an efficient perfect device $D = (S, \Pi, M, P)$ and any $\vec{a}, \vec{b} \in \{0,1\}^n$ we have

$$VX(\vec{a})Z(\vec{b})V^\dagger \approx_{n^{1/2}\gamma_H(D_{\mathsf{bb84}})^{1/8}, V\sigma^{(\vec{1})}V^\dagger} \left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_Q \otimes \mathcal{I}_{YRDA}\,. \tag{B.34}$$

B.3.3 Rigidity Proof of protocol 7

Having established a characterization of the prover's observables $X(\vec{a})Z(\vec{b})$ in protocol 6, we now use this to characterize the prover's behavior in protocol 7.

Step 1: Modeling. First, we introduce the corresponding notion of post-measurement states for an efficient device of protocol 7. Note that the two protocols are identical from the prover's point of view when the round type is the Hadamard round, and the marginal observables from Definition 56 are defined for Hadamard round. Thus we can use the same notation of marginal observables from Definition 56 (in particular, we only need the efficient observables $X(\vec{a})$ and $Z(\vec{b})$) for an efficient device in protocol 7.

Definition 59. For $\vec{k} \in (\mathcal{K}_0 \cup \mathcal{K}_1)^n$, $\vec{v} \in \{0,1\}^n$ and $A \subseteq \mathbb{F}_2^n$, $\vec{\alpha}, \vec{\beta} \in \{0,1\}^n$ define the set $V_{A,\vec{\alpha},\vec{\beta},\vec{k},\vec{v}} \subseteq \mathcal{Y}^n \times \{0,1\}^{w\times n}$ by the following condition:

$$(\vec{y},\vec{d}) \in V_{A,\vec{\alpha},\vec{\beta},\vec{k},\vec{v}} \text{ iff } \begin{cases} \hat{b}(\vec{k},\vec{y}) = \vec{v} \in A + \vec{\alpha} & \text{if } \vec{k} \in \mathcal{K}_0^n\,, \\ \hat{u}(\vec{k},\vec{y},\vec{d}) \oplus \vec{v} \in A^\perp + \vec{\beta} & \text{if } \vec{k} \in \mathcal{K}_1^n\,. \end{cases} \tag{B.35}$$

Then for $\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{\theta} \in \{\vec{0}, \vec{1}\}, \vec{v}$ we define

$$\sigma^{(A,\vec{\alpha},\vec{\beta},\vec{\theta},\vec{v})} = \sum_{\vec{y},\vec{d} \in V_{A,\vec{\alpha},\vec{\beta},\vec{k},\vec{v}}} \sigma^{(\vec{\theta})}_{\vec{y},\vec{d}} \otimes |\vec{y},\vec{d}\rangle\langle\vec{y},\vec{d}| \,. \tag{B.36}$$

By the same argument as in Remark 8, we can write $\sigma^{(\vec{\theta},\vec{v})}$ for simplicity.

We note that different from Definition 57, we only consider two basis choices $\vec{\theta} = \vec{0}$ or $\vec{\theta} = \vec{1}$, whereas the post-measurement states in Definition 57 can be defined with respect to any basis choice. Similar to Lemma 13, we analyze what outcomes a successful device must produce when measuring the observables from Definition 56 on the post-measurement states from Definition 59.

Lemma 17. For any efficient device $D = (S, \Pi, M, P)$, a coset state description (A, α, β):

$$\sum_{\vec{v} \in S_0} \mathrm{Tr}\left[Z_i^{(v_i)} \sigma^{(\vec{0},\vec{v})}\right] \approx_{\gamma_H(D_{\mathsf{coset}})} 1 \,, \tag{B.37}$$

$$\sum_{\vec{v} \in S_1} \mathrm{Tr}\left[X_i^{(v_i)} \sigma^{(\vec{1},\vec{v})}\right] \approx_{\gamma_H(D_{\mathsf{coset}})} 1 \,, \tag{B.38}$$

where $S_0 \coloneqq A + \alpha$ and $S_1 \coloneqq A^\perp + \beta - \hat{u}(\vec{k}, \vec{y}, \vec{d})$.

Proof. We first prove Equation (B.37). Since the case $q = \theta = 0$ occurs with probability $1/2$ in protocol 7, the device's failure probability in this case can be at most $2\gamma_H(D_{\mathsf{coset}})$. Furthermore, since the device only succeeds if $v_i = \hat{b}(k_i, y_i)$ and $\vec{v} \in A + \alpha$ for all $i \in [\![1, n]\!]$ in the protocol, it means that the device succeeds with probability at least $1 - 2\gamma_H(D)$. Now comparing the definition of $\sigma^{(\vec{0},\vec{v})}$ with the verifier's checks in the protocol, this means that for all $i \in [\![1, n]\!]$:

$$\sum_{v \in S_0} \mathrm{Tr}\left[Z_i^{(v_i)} \sigma^{(\vec{0},\vec{v})}\right] \geq 1 - 2\gamma_H(D) \,.$$

For the inequality in the other direction, we note that since $Z_i^{(v_i)}$ is a projector, we immediately have

$$\sum_{\vec{v} \in S_0} \mathrm{Tr}\left[Z_i^{(v_i)} \sigma^{(\vec{0},\vec{v})}\right] \leq \sum_{\vec{v} \in S_0} \mathrm{Tr}\left[\sigma^{(\vec{0},\vec{v})}\right] = \mathrm{Tr}\left[\sigma^{(\vec{0})}\right] = 1,$$

finishing the proof of Equation (B.37).

The proof of Equation (B.38) is completely analogous, combining with the fact that if $\vec{v} + \hat{u}(\vec{k}, \vec{y}, \vec{d}) \in A^\perp + \beta$ iff $\vec{v} \in A^\perp + \beta - \hat{u}(\vec{k}, \vec{y}, \vec{d})$. □

Step 2: Relating protocol 6 and protocol 7. We relate the prover's operators and states in protocol 6 and protocol 7 by the following lemmas.

Lemma 18. For any efficient devices D, D' with the notation given in Definition 52. Assume that D is a device of protocol 6 with corresponding states $(\psi^{(\vec{\theta})}, \sigma^{(\vec{\theta})})$ and D' is a device of protocol 7 with corresponding states $(\psi'^{(\vec{\theta}')}, \sigma'^{(\vec{\theta}')})$. Then

$$\psi^{(\vec{\theta})} \overset{c}{\approx}_0 \psi'^{(\vec{\theta}')} \,, \tag{B.39}$$

and

$$\sigma^{(\vec{\theta})} \overset{c}{\approx}_0 \sigma^{(\vec{\theta}')} \,. \tag{B.40}$$

Proof. At the beginning of each protocol's execution: in protocol 6, the device's state is (encrypted) BB84 states, while in protocol 7, the device's state is (encrypted) coset states. Furthermore, note that executing protocol 6 or protocol 7 does not require the secret key of the QFHE encryption scheme. Equation (B.39) then follows directly from semantic security of the QFHE encryption scheme.

In protocol 7, the verifier never sends a "pre-image round" challenge. In protocol 6, the round type is chosen uniformly at random, so with probability $\frac{1}{2}$, the round type is "Hadamard round". In this case, the execution of two protocols are identical from the prover's point of view. Since the prover is efficient, Equation (B.40) also follows. □

We then obtain the following relation between the success probabilities of devices in protocol 6 and protocol 7.

Corollary 8. For any efficient device $D \coloneqq (S, \Pi, M, P)$:

$$\gamma_H(D_{\texttt{bb84}}) \overset{c}{\approx}_0 2\gamma_H(D_{\texttt{coset}}). \tag{B.41}$$

Remark 9. Due to the relation in Equation (B.41) and the definition of the "$\approx$"-notation (Definition 3), from now on, we drop the subscript and simply write $\gamma_H(D)$ when it is clear from the context.

Combining Corollary 8 and Lemma 18, using the same isometry V defined in Definition 58, we can "lift" the approximate-equality relations described in Lemma 16 for an efficient device in protocol 6 to an efficient device in protocol 7.

Lemma 19. For an efficient perfect device $D = (S, \Pi, M, P)$ in protocol 7 and any $\vec{a}, \vec{b} \in \{0,1\}^n$ we have

$$VX(\vec{a})Z(\vec{b})V^\dagger \approx_{n^{1/8}\gamma_H(D)^{1/32}, V\sigma^{(\vec{1})}V^\dagger} \left(\sigma_X(\vec{a})\sigma_Z(\vec{b})\right)_Q \otimes \mathcal{I}_{YRDA}\,. \tag{B.42}$$

Proof. The lemma follows directly from the lifting lemma (Item 6 of Lemma 9) and the fact that the isometry V and the operators X, Z are efficient. Using the notation from Lemma 9, we have $\delta = 0$, $\varepsilon = n^{1/2}\gamma_H(D)^{1/8}$, the isometry is V, the observable A is $X(\vec{a})Z(\vec{b})$, the observable B is $\sigma_X(\vec{a})\sigma_z(\vec{b}) \otimes \mathcal{I}$. The two states are $V\sigma'^{(\vec{1})}V^\dagger$ of a device in protocol 6 and $V\sigma^{(\vec{1})}V^\dagger$ of a device in protocol 7. □

Step 3: Rigidity. We first prove the following technical lemma.

Lemma 20. For an efficient device $D = (S, \Pi, M, P)$, a coset state description (A, α, β):

$$\sum_{\vec{v}\in S_0} |\vec{v}\rangle\langle\vec{v}| \otimes (\sigma^{(v_i)}_{Z,i})_Q \approx_{\varepsilon, \sum_{\vec{v}'\in S_0} |\vec{v}'\rangle\langle\vec{v}'|\otimes V\sigma^{\vec{0},\vec{v}'}V^\dagger} \mathcal{I}\,, \tag{B.43}$$

$$\sum_{\vec{v}\in S_1} |\vec{v}\rangle\langle\vec{v}| \otimes (\sigma^{(v_i)}_{X,i})_Q \approx_{\varepsilon, \sum_{\vec{v}'\in S_1} |\vec{v}'\rangle\langle\vec{v}'|\otimes V\sigma^{\vec{1},\vec{v}'}V^\dagger} \mathcal{I}\,, \tag{B.44}$$

where $S_0 = A + \alpha$, $S_1 = A^\perp + \beta - \hat{u}(\vec{k}, \vec{y}, \vec{d})$ and the approximation factor ε will be clarified later in the proof.

Proof. We first prove the first statement. It is easy to check that $\sum_{\vec{v}\in V} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma_{Z,i}^{(v_i)}\right)_Q$ is a projector, so we can expand the definition of the state-dependent distance and compute:

$$\begin{aligned}
&\mathrm{Tr}\left[\left(\sum_{\vec{v}\in S_0} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma_{Z,i}^{(v_i)}\right)_Q - \mathcal{I}\right)^\dagger \left(\sum_{\vec{v}\in S_0} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma_{Z,i}^{(v_i)}\right)_Q - \mathcal{I}\right) \sum_{\vec{v}'\in S_0} |\vec{v}'\rangle\langle\vec{v}'| \otimes V\sigma^{(\vec{0},\vec{v}')}V^\dagger\right] \\
&= \mathrm{Tr}\left[\left(\mathcal{I} - \sum_{\vec{v}\in S_0} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma_{Z,i}^{(v_i)}\right)_Q\right) \sum_{\vec{v}'\in S_0} |\vec{v}'\rangle\langle\vec{v}'| \otimes V\sigma^{(\vec{0},\vec{v}')}V^\dagger\right] \\
&= 1 - \sum_{\vec{v}\in S_0} \mathrm{Tr}\left[\left(|\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma_{Z,i}^{(v_i)}\right)_Q\right) \sum_{\vec{v}'\in S_0} |\vec{v}'\rangle\langle\vec{v}'| \otimes V\sigma^{(\vec{0},\vec{v}')}V^\dagger\right] \\
&= 1 - \sum_{\vec{v}\in S_0} \mathrm{Tr}\left[\left(\sigma_{Z,i}^{(v_i)}\right)_Q V\sigma^{(\vec{0},\vec{v})}V^\dagger\right],
\end{aligned}$$

To show the first part of the lemma, we need to show that

$$\sum_{\vec{v}\in S_0} \mathrm{Tr}\left[\left(\sigma_{Z,i}^{(v_i)}\right)_Q V\sigma^{(\vec{0},\vec{v})}V^\dagger\right] \approx_\varepsilon 1\,. \tag{B.45}$$

For this, recall from Lemma 19 that we have

$$VZ_iV^\dagger \approx_{n^{1/8}\gamma_H(D)^{1/32}, V\sigma^{(\vec{1})}V^\dagger} (\sigma_{Z,i})_Q \otimes \mathcal{I}_{YRDA}\,. \tag{B.46}$$

For shorthand, write $\gamma := n^{1/8}\gamma_H(D)^{1/32}$. Since V and Z_i are efficient, by the lifting lemma (Lemma 9) and the fact that $\sigma^{(\vec{0})} \overset{c}{\approx}_0 \sigma^{(\vec{1})}$, this implies that:

$$VZ_iV^\dagger \approx_{\gamma^{1/4}, V\sigma^{(\vec{0})}V^\dagger} (\sigma_{Z,i})_Q \otimes \mathcal{I}_{YRDA}\,. \tag{B.47}$$

Using Lemma 10 and Lemma 11, we get:

$$\sum_{\vec{v}\in S_0} VZ_i^{(v_i)}V^\dagger \approx_{\gamma^{1/4}, \sum_{\vec{v}\in S_0} V\sigma^{(\vec{0},\vec{v})}V^\dagger} \sum_{\vec{v}\in S_0} \left(\sigma_{Z,i}^{(v_i)}\right)_Q \otimes \mathcal{I}_{YRDA}\,. \tag{B.48}$$

Using the replacement lemma (Lemma 7), we obtain

$$\begin{aligned}
\sum_{\vec{v}\in S_0} \mathrm{Tr}\left[\left(\sigma_{Z,i}^{(v_i)}\right)_Q V\sigma^{(\vec{0},v_i,\vec{1}^i)}V^\dagger\right] &\approx_{\gamma^{1/8}} \sum_{\vec{v}\in S_0} \mathrm{Tr}\left[VZ_i^{(v_i)}V^\dagger V\sigma^{(\vec{0},\vec{v})}V^\dagger\right] && \text{(B.49)} \\
&= \sum_{\vec{v}\in S_0} \mathrm{Tr}\left[Z_i^{(v_i)}\sigma^{(\vec{0},\vec{v})}\right] && \text{(B.50)} \\
&\approx_{\gamma_H(D)} 1\,, && \text{(B.51)}
\end{aligned}$$

where the last line follows from Equation (B.37). Set $\varepsilon := \gamma^{1/8}$, this finishes the proof of the first statement.

For the second statement, we can perform the same calculation, but use Equation (B.38). □

Lemma 21. For an efficient perfect device $D = (S, \Pi, M, P)$, a coset state description (A, α, β) and $\vec{\theta} \in \{\vec{0}, \vec{1}\}$, there exists a set of subnormalized states $\{\rho_i^{(\vec{\theta},\vec{v})}\}_{\vec{v}\in S_i}$ where S_i for $i \in \{0,1\}$ are defined as in Lemma 20 such that

$$\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \approx_{2n\varepsilon} \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left((\mathsf{H}^{\otimes n})^i |\vec{v}\rangle\langle\vec{v}| (\mathsf{H}^{\otimes n})^i\right)_Q \otimes \rho_i^{(\vec{\theta},\vec{v})}\,, \tag{B.52}$$

where $i = 0$ if $\vec{\theta} = \vec{0}$ and $i = 1$ if $\vec{\theta} = \vec{1}$.

Proof. We define the shorthand

$$M(\theta) = \begin{cases} Z & \text{if } \theta = 0\,, \\ X & \text{if } \theta = 1\,. \end{cases}$$

Applying Lemma 20 and Lemma 8 to get

$$\begin{aligned} &\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \\ &\approx_\varepsilon \left(\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma^{(v_1)}_{M(\theta_1),1}\right)_Q\right) \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \left(\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma^{(v_1)}_{M(\theta_1),1}\right)_Q\right) \end{aligned}$$

We repeat this for the remaining indices $j = 2, \ldots, n$. Since there are in total n steps, the total approximation error will be $n\varepsilon$. We then have

$$\begin{aligned} &\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \\ &\approx_{n\varepsilon} \left(\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma^{(v_1)}_{M(\theta_1),1}\right)_Q\right) \cdots \left(\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma^{(v_n)}_{M(\theta_n),n}\right)_Q\right) \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \\ &\qquad \left(\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma^{(v_1)}_{M(\theta_1),1}\right)_Q\right) \cdots \left(\sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\sigma^{(v_n)}_{M(\theta_n),n}\right)_Q\right) \\ &= \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left(\prod_j \sigma^{(v_j)}_{M(\theta_j),j}\right)_Q V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \left(\prod_j \sigma^{(v_j)}_{M(\theta_j),j}\right)_Q . \end{aligned}$$

Now noting that $\prod_j \sigma^{(v_j)}_{M(\theta_j),j} = (\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i$, we obtain

$$\begin{aligned} &= \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left((\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i\right)_Q \otimes \left(\langle v| (\mathsf{H}^{\otimes n})^i\right)_Q V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \left((\mathsf{H}^{\otimes n})^i |v\rangle\right)_Q \\ &= \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left((\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i\right)_Q \\ &\qquad \otimes \mathsf{Tr}_Q\left[\left((\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i\right)_Q V\sigma^{(\vec{\theta},\vec{v})}V^\dagger \left((\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i\right)_Q\right] \end{aligned}$$

Analogously to how we added the factors $\prod_j \sigma^{(v_j)}_{M(\theta_j),j}$ in a previous step, we can now replace the factors $((\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i)_Q$ inside the partial trace by identity, resulting in

$$\approx_{2n\varepsilon} \sum_{\vec{v}\in S_i} |\vec{v}\rangle\langle\vec{v}| \otimes \left((\mathsf{H}^{\otimes n})^i|\vec{v}\rangle\langle\vec{v}|(\mathsf{H}^{\otimes n})^i\right)_Q \otimes \mathsf{Tr}_Q\left[V\sigma^{(\vec{\theta},\vec{v})}V^\dagger\right] .$$

We then obtain the desired statement by defining

$$\rho_i^{(\vec{\theta},\vec{v})} := \mathsf{Tr}_Q\left[V\sigma^{(\vec{\theta},\vec{v})}V^\dagger\right] , \tag{B.53}$$

with $i = 0$ if $\vec{\theta} = \vec{0}$ and $i = 1$ if $\vec{\theta} = \vec{1}$. □

What Lemma 21 says is that up to an isometry, with inverse polynomial error, the device's state must be (information-theoretically) close to a mixed state of vectors in

S_i, tensored with an auxiliary state $\rho_i^{(\vec{\theta},\vec{v})}$. We note that it is not hard to show that $\rho_0^{(\vec{0},\vec{v})} \overset{c}{\approx}_0 \rho_1^{(\vec{1},\vec{v})}$. (Though it is not necessary for our soundness proof.)

Furthermore, from the statement of Lemma 21, for a fixed efficient device D, if we run protocol 7 "coherently" in superposition, then

(i) when $\vec{\theta} = \vec{0}$, the device's state must be in superposition of all vectors in S_0, that is $|A + \alpha\rangle$,

(ii) when $\vec{\theta} = \vec{0}$, the device's state must be in superposition of all vectors in S_1. By applying a correction (XOR-ing the register Q with $\hat{u}(\vec{k}, \vec{y}, \vec{d})$), the state would be $|A^{\perp} + \beta\rangle$.

Thus, with the verifier in protocol 7, we obtain efficient projective measurements to characterize the prover's initial state. Formally, let O_0 be the following process: run protocol 7 in superposition (without measuring any intermediate messages such as y, d, v) with the basis choice $\vec{\theta} = \vec{1}$ and check if the register Q at the end of the protocol is $|A + \alpha\rangle$. O_1 is defined analogously for $\vec{\theta} = \vec{1}$, and it applies a correction by XORing the register Q with $\hat{u}(\vec{k}, \vec{y}, \vec{d})$ and check if the register Q at the end is $|A^{\perp} + \beta\rangle$. We obtain the main technical lemma.

Lemma 22. For any efficient device D, the initial state of the device ψ must be close to (up to some inverse polynomial error) $|A_{\alpha,\beta}\rangle \otimes \rho$:

$$\psi \approx_{4n\varepsilon} |A_{\alpha,\beta}\rangle \otimes \rho. \tag{B.54}$$

Proof. Let U_0 and U_1 be the efficient unitaries corresponding to operators O_0 and O_1 defined above. Fix a device D. We first apply $U_0\psi$ and record the output to an ancilla register. If the output is 1, apply the inverse $U_0^{\dagger}$ to obtain ψ'. Finally apply $U_1\psi'$. If the output is 1, by the definition of U_i (and O_i), the lemma follows. Note that for each application of U_i, the approximation error is $2n\varepsilon$ which comes from Lemma 21. □

B.3.4 Rigidity Proof of protocol 8

We are now ready to prove the rigidity of protocol 8, namely that any efficient quantum prover that does not cause the protocol to abort must have the initial state close to a hidden coset state.

Lemma 23. For any $\lambda \in \mathbb{N}$, there exist choices $M = \mathsf{poly}(\lambda)$ and $\delta = 1/\mathsf{poly}(\lambda)$ such that if the verifier executes protocol 8 with an efficient quantum prover whose success probability is lower-bounded by an inverse polynomial, the following holds. Let (A, α, β) the private input of the verifier for the coset instance. Denoting the probability that the protocol does not abort as $\Pr[\top]$, and let ψ the initial state of the prover. Then, with probability $\Pr[\top]$, we have

$$\psi \overset{c}{\approx}_{\varepsilon} |A_{\alpha,\beta}\rangle \otimes \rho, \tag{B.55}$$

for some auxiliary state ρ, and the approximation error ε is inverse polynomial on the security parameter λ.

Proof. Essentially, we can see protocol 8 as a cut-and-choose protocol in which the number of evaluation instances is 1 and the number of check instances is $M^2 - 1$. We then can reduce this lemma to Lemma 22 using the same argument as in Gheorghiu, Metger, and Poremba (2022, Theorem 4.33). We omit the details. □

Remark 10. We make few comments on the inverse polynomial soundness.[2] First of all, what the soundness lemma (Lemma 23) says is effectively the same as a typical self-testing statement, which is that: if the prover succeeds with probability $1-\varepsilon$ in the protocol, the state it used in the protocol must be, up to an isometry, $\mathsf{poly}(\varepsilon)$-close to ideal (in our setting, the closeness is measured by computational distinguishability rather than trace distance, as in typical self-testing settings). Now, in practice, we would have to estimate ε by doing many runs of the protocol. In particular, we would need about $1/\varepsilon^2$ repetitions to have high (that is, $1-\mathsf{negl}(\lambda)$) confidence that the prover's success probability is $1-\varepsilon$. This implies that if we want ε to be negligible, we would have to do superpolynomial-many repetitions of the protocol and since this is not efficient, we are limited to $\varepsilon = 1/\mathsf{poly}(\lambda)$. It is from doing this $1/\varepsilon^2$ repetitions that we go from the original self-testing statement (Lemma 22) to the statement that characterizes the prover's state in the actual protocol.

B.3.5 Self-Testing Protocol Soundness

We state and prove in this section the following proposition.

Proposition 1. For any $\lambda \in \mathbb{N}$, there exist choices $M = \mathsf{poly}(\lambda)$ and $\delta = 1/\mathsf{poly}(\lambda)$ such that if the verifier executes protocol 10 with an efficient quantum prover whose success probability is lower-bounded by an inverse polynomial, the following holds. We denote by ϕ^T_{SVP} the verifier and prover's joint final state at the end of protocol 10, where T is the set of coset states obtained by the verifier, S is set to $|\bot\rangle\langle\bot|$ by the verifier if the protocol aborts and $|\top\rangle\langle\top|$ otherwise, V is the register in which the verifier records the set T, and P is the prover's registers. Then, denoting the probability of success as $\Pr[\top]$, and writing

$$\phi^T_{SVP} = \Pr[\top]|\top\rangle\langle\top|_S \otimes \phi^T_{VP|\top} + (1-\Pr[\top])|\bot\rangle\langle\bot| \otimes \phi^T_{VP|\bot}.$$

Then there exists a coset instance (A, α, β) in T such that the state $\phi^T_{VP|\top}$ conditioned on acceptance satisfies:

$$\phi^T_{VP|\top} \overset{c}{\approx}_{1/\mathsf{poly}(\lambda)} |T\rangle\langle T|_V \otimes |A_{\alpha,\beta}\rangle\langle A_{\alpha,\beta}| \otimes \rho, \tag{B.56}$$

for some auxiliary state ρ.

Proof. Since in the final protocol (protocol 10), we run N instances over $2N$ possible instances of the self-testing protocol (protocol 8) (in the cut-and-choose fashion), we can invoke techniques developed in Bouman and Fehr (2010) to relate quantum sampling to classical sampling and conclude Proposition 1.

In particular, consider the following interaction between a quantum prover P and a challenger V.

1. P and V jointly execute protocol 10. Let $\overline{T}$ be the set of N indices chosen uniformly at random by V in N runs of the self-testing protocol.

2. Let X_i be the outcome of each of N runs of the self-testing protocol. V verifies that $X_i = \mathsf{accept}$ for all $i \in \overline{T}$, and aborts otherwise.

This is a natural quantum analogue of the following classical sampling experiment (Bouman and Fehr (2010, Example 1)) on a length-$2N$ bitstring X to test if X is close to the all-zero string:

[2] We thank Alexandru Gheorghiu for providing us this insightful comments.

1. randomly select a size-N subset $\overline{T} \subset [\![1, 2N]\!]$,

2. compute $\omega(X|_{\overline{T}})$, and accept if the estimate vanishes and else reject.

Noting that this sample-and-estimate strategy is exactly the Ψ_{uniform} strategy described at the end of Appendix B.1.2, we have by Corollary 7 that the quantum error probability of this strategy is bounded by $2\exp(\frac{-n\delta^2}{64})$, for $\delta = 1/2$. By the definition of quantum error probability (Definition 48), this means that, with overwhelming probability over $\overline{T}$, the state of the prover P in the remaining set T also satisfies Equation (B.54). Indeed, by changing of basis, this reduces to the question of testing if the state of the prover before running the self-testing protocol is close to the all-zero state. Then the quantum sample-and-estimate technique tells us that the state of the prover must be supported on vectors with relative Hamming distance $< 1/2$, and it means there must be at least 1 bit in string which is 0. If this is the case, it corresponds (up to some inverse polynomial error) to the coset state $|A_{\alpha,\beta}\rangle$ in Equation (B.54). This completes the proof of the proposition. □

B.4 Proof of Semi-Quantum Monogamy-Of-Entanglement Property of Protocol 5

In this section, we prove that our remote coset states preparation preserves the monogamy-of-entanglement property. Formally, we prove soundness of protocol 10.

B.4.1 Monogamy-of-Entanglement Soundness of protocol 10

We now formally define the notion of soundness for our protocol, which is described as a coset semi-quantum monogamy game.

Theorem 32. protocol 10 is computationally sound, according to Definition 50.

Proof. Let $\mathsf{P} = \{\mathsf{P}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ a quantum polynomial time adversary that succeeds in the game $\mathsf{SMCosetMonogamy}$ with some non-negligible probability $\varepsilon = \{\varepsilon_\lambda\}_{\lambda\in\mathbb{N}}$. Let $(\{S_i, \alpha_i, \beta_i\}_{i\in T}, \psi) \leftarrow \langle \mathsf{P}_\lambda(\rho_\lambda), \mathsf{V}(1^\lambda)\rangle$. This means that $\mathsf{P} = (\mathsf{P}, \mathcal{B}, \mathcal{C})$ is able to output a pair $(s_1^{(i)}, s_2^{(i)}) \in (S_i + \alpha_i) \times (S_i^\perp + \beta_i)$ for all $i \in T$ in the monogamy game defined in Definition 50.

Let $\delta' \in (0, 1]$ the sub-exponential security level of the QFHE (that is, any QPT adversary cannot break the semantic security of the QFHE with advantage bigger than $2^{\lambda^{\delta'}}$), and denote $\delta := \frac{\delta'}{2}$.

We next describe a sequence of hybrid experiments.[3]

Game G_0: This is the original experiment.

We define G_0 as the original attack, where P interacts with the verifier in protocol 10 and wins the monogamy game $\mathsf{SMCosetMonogamy}$. We say G_0 is successful if $\mathsf{SMCosetMonogamy}(\mathsf{P}, \lambda) = 1$. The experiment G_0 is thus successful with probability ε.

Game G_1: Changing the success definition of the experiment.

Pick a random index $i \in T$, for shorthand, denote this coset instance as (S, α, β), and the adversary's corresponding output in the monogamy game is (s_1, s_2). In the current

[3]Some hybrids follow from the proof strategy given in Shmueli (2022a).

hybrid, the experiment is defined to be successful if $s_1 \in S + \alpha$ and $s_2 \in S^\perp + \beta$. In particular, in the current hybrid, we only consider the monogamy game for a random instance among $|T|$ coset instances. (The other instances are not considered). Apparently, G_1 is successful with probability at least ε. From now on, we only consider this coset instance in later hybrids, and all the changes are only applied to this instance.

Game G_2: Injecting quantum communication into the interaction between the prover and the verifier.

This hybrid is identical to G_1 except that now we consider the verifier as a QPT algorithm instead of a PPT algorithm, and we make an additional round of interaction using quantum communication in the protocol. (Think about the verifier now as a QPT challenger of the experiment.) In particular, right after the last step of protocol 10 (step 9c), we ask the prover to send the coset state $|S_{\alpha,\beta}\rangle$ to the verifier. Denote this state as $|\$\rangle$. The verifier then does the following:

- Verify the received coset state:
  (a) Checks that the output qubit of the computation $\mathsf{iO}(S+\alpha)(|\$\rangle)$[4] is 1.
  (b) Execute Hadamard transform $\mathsf{H}^{\otimes\lambda}$ on $|\$\rangle$ to obtain $|\$'\rangle$ and then check the output qubit of the computation $\mathsf{iO}(S^\perp+\beta)(|\$'\rangle)$ is 1.
- If any of these checks returns 0, abort and declare the game as a failure.
- Execute $\mathsf{H}^{\otimes\lambda}$ again on $|\$'\rangle$ to obtain $|\$''\rangle$ and send $|\$''\rangle$ back to the prover.

From Proposition 1, it follows that with probability at least $1/|T|$, the adversary's output state ϕ is inverse polynomially ϵ-close to $|S_{\alpha,\beta}\rangle \otimes \rho$ for some auxiliary state ρ. It means that when it is asked, the adversary can always send a state $|\$\rangle$ that is inverse polynomially ϵ-close to $|S_{\alpha,\beta}\rangle$ to the challenger.

Note that the quantum verification described above executes only on the register containing $|\$\rangle$ and thus commutes with any other quantum operation on a register entangled with it at the point where P finishes executing the real protocol protocol 10. Thus after finishing the above additional interaction, the adversary's state is unchanged, if the verification passed.

The probability that the adversary does not fail in the experiment is $1-\epsilon$. It is then clear that, for any adversary that wins the G_1 with probability ε, it wins G_2 with probability at least $\varepsilon' := \varepsilon(1-\epsilon)/|T|$. Thus, the success probability of G_2 is ε' for some non-negligible ε'.

Game G_3: Removing subspace information from obfuscated circuits.

This hybrid is identical to G_2, with the only difference is that when the verifier returns the obfuscations P_0, P_1 in the last step of protocol 10 (Step 9c), the obfuscations are changed: We sample two random $(\lambda-\lambda^\delta)$-dimensional subspaces $T_0, T_1 \subseteq \mathbb{F}_2^\lambda$ subjected to $T_1^\perp \subseteq S \subseteq T_0$. The verifier uses $\mathsf{iO}(T_0+\alpha)$ instead of $\mathsf{iO}(S+\alpha)$, and $\mathsf{iO}(T_1+\beta)$ instead of $\mathsf{iO}(S^\perp+\beta)$.

It is easy to see that any QPT distinguisher between G_2 and G_3 can be transformed into a QPT distinguisher between obfuscations of the original functions $S+\alpha, S^\perp+\beta$ and

[4]We are running a classical function on a quantum input, which can be interpreted as running a classical function in superposition.

obfuscations of $T_0 + \alpha, T_1 + \beta$. By the subspace hiding property of indistinguishability obfuscators, the success probabilities of G_2 and G_3 are thus negligibly close. Thus the successful probability of G_3 is at least $\varepsilon' - \mathsf{negl}(\lambda)$.

Game G_4: Computing the obfuscations with less information on α, β.

This hybrid is identical to G_3, with a modification in the way we check membership in each of the cosets: Let B_0 a basis for T_0, and B_1 a basis for $T_1^\perp$, and let $y_\alpha, y_\beta \in \{0,1\}^{\lambda-\lambda^\delta}$ defined as $y_\alpha := B_0 \cdot \alpha$ and $y_\beta := B_1 \cdot \beta$. $\mathsf{iO}(T_0 + \alpha)$ is changed to be an obfuscation of a circuit that for an input $u \in \{0,1\}^\lambda$ checks whether $B_0 \cdot u = y_\alpha$. $\mathsf{iO}(T_1 + \beta)$ is changed to be an obfuscation of a circuit that for an input $u \in \{0,1\}^\lambda$ checks whether $B_1 \cdot u = y_\beta$.

One can verify that the functionality of the obfuscated circuits $\mathsf{iO}(T_0 + \alpha)$, $\mathsf{iO}(T_1 + \beta)$ did not change, and thus by the security of the indistinguishability obfuscation schemes, the distributions are indistinguishable and the success probability of G_4 is $\varepsilon' - \mathsf{negl}(\lambda)$.

Game G_5: Reordering the sampling process of the subspaces S, T_0, T_1.

This hybrid is identical to G_4, except that we change the order of the subspaces sampling process. In the previous hybrid, we sample a random $\frac{\lambda}{2}$-dimensional subspace $S \subseteq \mathbb{F}_2^\lambda$ then two random $(\lambda - \lambda^\delta)$-dimensional subspaces T_0, T_1 subjected to $T_1^\perp \subseteq S \subseteq T_0$. In the current hybrid, we first sample two random $(\lambda - \lambda^\delta)$-dimensional subspaces $T_0, T_1 \subseteq \mathbb{F}_2^n$ subjected to $T_1^\perp \subseteq T_0$, then sample a random $\frac{\lambda}{2}$-dimensional subspace $S \subseteq \mathbb{F}_2^n$ subjected to $T_1^\perp \subseteq S \subseteq T_0$.

Since the distribution of (S, T_0, T_1) in both hybrids are identical, the success probability of G_5 is $\varepsilon' - \mathsf{negl}(\lambda)$.

Game G_6: Fixing the subspace T_0, T_1.

In the subspace sampling process described in the previous hybrid, T_0 and T_1 are sampled before everything else. Thus we can perform an averaging argument on the sampling of T_0, T_1 to take the samples that maximize the success probability of the previous hybrid. Fix these samples of T_0, T_1 and define G_6 with respect to these samples. It is clear that the success probability of G_6 is $\varepsilon' - \mathsf{negl}(\lambda)$.

Game G_7: Removing the QFHE secret key from the reduction.

This hybrid is identical to G_6 with one change: In step 6, when the verifier decrypts the QFHE classical part to get the Pauli keys α, β, the current hybrid does not decrypt to get α, β and instead it samples uniformly random $\alpha', \beta' \in \{0,1\}^\lambda$ and computes $y'_\alpha := B_0 \cdot \alpha', y'_\beta := B_1 \cdot \beta'$. The verifier then use these strings as y_α, y_β in the construction of the obfuscations $\mathsf{iO}(T_0 + \alpha)$, $\mathsf{iO}(T_1 + \beta)$, respectively.

We note that this change is only done for the specific coset instance under the consideration, for the other instances, the verifier still decrypts normally using the corresponding QFHE secret key.

Since α', β' are chosen uniformly at random, for fixed bases B_0, B_1, y'_α, y'_β are also uniformly random. Observe that conditioned on the probabilistic event $y'_\alpha = y_\alpha$ and $y'_\beta = y_\beta$ (for which to happen, the probability is exactly $2^{-2\lambda^\delta}$), the current and previous hybrids distribute identically. It follows that the success probability in G_7 is at least $2^{-2\lambda^\delta} \cdot (\varepsilon' - \mathsf{negl}(\lambda)) > 2^{-3\lambda^\delta}$.

Game G_8: Clearing all given knowledge on S and reducing to the original monogamy-of-entanglement game.

This hybrid is identical to G_7, except that we make two additional changes as follows.

- In the additional quantum communication round that we added after the end of protocol 10 (see hybrid G_2), instead of sending back the original state $|\$\rangle$, the verifier send $|\hat{S}_{\hat{\alpha},\hat{\beta}}\rangle$. Recall that the coset $(\hat{S}, \hat{\alpha}, \hat{\beta})$ is the one the verifier sampled independently in step 8.
- In the step 2 in the monogamy game (Definition 50), when the challenger (i.e., the verifier) sends the description of the subspace S to both adversaries $\mathcal{B}, \mathcal{C}$, it sends $\hat{S}$ instead.
- Consequently, the winning condition is changed to be that $\mathcal{B}$ outputs a vector in $\hat{S} + \hat{\alpha}$ and $\mathcal{C}$ outputs a vector in $\hat{S}^{\perp} + \hat{\beta}$.

We make few observations on the distribution in the current hybrid. First, in order to execute G_8, there is no need to know the secret key (corresponding to the coset instance under the consideration) of the QFHE scheme. However, one needs to care when invoking the semantic security of the QFHE, because even there is no need for the secret key, the adversary is still given a "predicate" check on the ciphertext, that is the obfuscation. Thus, to use the security of the QFHE, it is necessary to use two plaintexts such that the obfuscation evaluation on the ciphertext of these two plaintexts are identical. Our obfuscations $(P_{0,i}, P_{1,i})$ were generated so that this condition is satisfied.

Secondly, the obfuscation distribution does not change from the description above, and we can see that in the previous hybrid, the adversary obtains a quantum one-time pad encryption of $|S\rangle$, while in the current hybrid, the adversary obtains a quantum one-time pad of $|\hat{S}\rangle$. More precisely, the adversary in the current hybrid receives an encryption of $|\hat{S}\rangle$ that is $|\hat{S}_{\hat{\alpha},\hat{\beta}}\rangle$ and an encryption of some Pauli keys (α, β) that are different from $(\hat{\alpha}, \hat{\beta})$ with overwhelming probability. But because of the semantic security of $\mathsf{QFHE.Enc}$, this is indistinguishable from having $|\hat{S}_{\hat{\alpha},\hat{\beta}}\rangle$ and an actual encryption of $(\hat{\alpha}, \hat{\beta})$.

From these observations, it follows that we can invoke the security of the QFHE to argue the indistinguishability of the current and previous hybrids, and in particular the indistinguishability between their success probabilities. Using the sub-exponential-advantage security of the QFHE, we have the success probability of G_8 is $> 2^{-3\lambda^{\delta}} - 2^{-2\lambda^{\delta'}} > 2^{-3\lambda^{\delta}-1}$.

At this point of the proof, we can reduce the success probability of an adversary in G_8 to the monogamy-of-entanglement game. We note that the coset game can achieve sub-exponentially negligible security, say $2^{-4\lambda^{\delta}}$, if we assume sub-exponential security of the building blocks (i.e., the indistinguishability obfuscation scheme). Now, any QPT adversary of G_8 can be used to construct a QPT adversary for the coset game as follows. Specifically, the reduction receives a challenge coset state $|\hat{S}_{\hat{\alpha},\hat{\beta}}\rangle$ and the obfuscated membership checking programs $\mathsf{iO}(\hat{S}+\hat{\alpha}), \mathsf{iO}(\hat{S}^{\perp}+\hat{\beta})$ from its challenger in the coset game in. The reduction runs protocol 10 with the adversary. Note that the reduction (playing the role of the verifier in protocol 10) only needs $\mathsf{iO}(\hat{S}+\hat{\alpha})$ and $\mathsf{iO}(\hat{S}^{\perp}+\hat{\beta})$ to perfectly simulate the protocol with the adversary. Furthermore, it uses $|\hat{S}_{\hat{\alpha},\hat{\beta}}\rangle$ in the experiment described above instead of generating the state on its own, when it needs to send a coset state back to the adversary. When the reduction receives $\hat{S}$ from its challenger, it sends

$\hat{S}$ to $\mathcal{B}, \mathcal{C}$, and finally the reduction outputs whatever $\mathcal{B}$ and $\mathcal{C}$ output. (Formally, the reduction now consists of two non-communicating reductions, each interacts with $\mathcal{B}$ and $\mathcal{C}$ respectively.) This is exactly in contradiction to strong monogamy-of-entanglement security as we presented above. □