Numéro National de Thèse : 2020LYSEN054

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée par
l'École Normale Supérieure de Lyon

École Doctorale N°512
École Doctorale en Informatique et Mathématiques de Lyon

Discipline : Informatique

Soutenue publiquement le 19/10/2020, par :
Ida Tucker

Chiffrement fonctionnel et signatures distribuées fondés sur des fonctions de hachage à projection, l'apport des groupes de classes

Devant le jury composé de :

Abdalla Michel, Directeur de Recherche, CNRS, DIENS Paris	Rapporteur
Damgård Ivan, Professor, Aarhus University (Danemark)	Rapporteur
Agrawal Shweta, Associate Professor, I.I.T. Madras (Inde)	Examinatrice
Fouque Pierre-Alain, Professeur, Université Rennes 1	Examinateur
Ràfols Carla, Lectora Tenure Track, Universitat Pompeu Fabra (Espagne)	Examinatrice
Laguillaumie Fabien, Professeur, Université Claude Bernard Lyon 1	Directeur de thèse
Castagnos Guilhem, Maître de Conférences HDR, Université de Bordeaux	Co-directeur de thèse

Résumé

Un des enjeux actuels de la recherche en cryptographie est la mise au point de primitives cryptographiques avancées assurant un haut niveau de confiance. Dans cette thèse, nous nous intéressons à leur conception, en prouvant leur sécurité relativement à des hypothèses algorithmiques bien étudiées.

Mes travaux s'appuient sur la linéarité du chiffrement homomorphe, qui permet d'effectuer des opérations linéaires sur des données chiffrées. Précisément, je suis partie d'un tel chiffrement, introduit par Castagnos et Laguillaumie à CT-RSA'15, ayant la particularité d'avoir un espace des messages clairs d'ordre premier. Afin d'aborder une approche modulaire, j'ai conçu à partir de ce chiffrement des outils techniques (fonctions de hachage projectives, preuves à divulgation nulle de connaissances) qui offrent un cadre riche se prêtant à de nombreuses applications.

Ce cadre m'a d'abord permis de construire des schémas de chiffrement fonctionnel; cette primitive très expressive permet un accès mesuré à l'information contenue dans une base de données chiffrée. Puis, dans un autre registre, mais à partir de ces mêmes outils, j'ai conçu des signatures numériques à seuil, permettant de partager une clé secrète entre plusieurs utilisateurs, de sorte que ceux-ci doivent collaborer afin de produire des signatures valides. Ce type de signatures permet entre autres de sécuriser les portefeuilles de crypto-monnaie.

De nets gains d'efficacité, notamment en termes de bande passante, apparaissent en instanciant ces constructions à l'aide de groupes de classes. Mes travaux se positionnent d'ailleurs en première ligne du renouveau que connait, depuis quelques années, l'utilisation de ces objets en cryptographie.

Abstract

One of the current challenges in cryptographic research is the development of advanced cryptographic primitives ensuring a high level of confidence. In this thesis, we focus on their design, while proving their security under well-studied algorithmic assumptions.

My work grounds itself on the linearity of homomorphic encryption, which allows to perform linear operations on encrypted data. Precisely, I built upon the linearly homomorphic encryption scheme introduced by Castagnos and Laguillaumie at CT-RSA'15. Their scheme possesses the unusual property of having a prime order plaintext space, whose size can essentially be tailored to ones' needs. Aiming at a modular approach, I designed from their work technical tools (projective hash functions, zero-knowledge proofs of knowledge) which provide a rich framework lending itself to many applications.

This framework first allowed me to build functional encryption schemes; this highly expressive primitive allows a fine grained access to the information contained in e.g., an encrypted database. Then, in a different vein, but from these same tools, I designed threshold digital signatures, allowing a secret key to be shared among multiple users, so that the latter must collaborate in order to produce valid signatures. Such signatures can be used, among other applications, to secure crypto-currency wallets.

Significant efficiency gains, namely in terms of bandwidth, result from the instantiation of these constructions from class groups. This work is at the forefront of the revival these mathematical objects have seen in cryptography over the last few years.

Acknowledgments

Je ne peux pas assez remercier mes co-directeurs Guilhem Castagnos et Fabien Laguillaumie.[1] Le temps et la patience que vous m'avez accordé n'a jamais cessé de me surprendre. Et au delà du rôle académique d'encadrants que vous avez assuré à merveille – qu'il s'agisse d'une bande-son pour accompagner mon humeur, d'une BD pour alléger ma frustration, ou tout simplement vos blagues et votre bonne humeur – vous avez toujours su me faire rire quand je touchais le fond, me motiver quand j'étais à bout, me féliciter et m'encourager (même quand mes prestations étaient, disons le, médiocres).

I am extremely grateful to Michel Abdalla and Ivan Damgård for reviewing this thesis. I know it was a lot of work, and that I am fortunate they agreed to it. Thank you also Shweta Agrawal, Pierre-Alain Fouques, and Càrla Rafols for accepting to be part of my Jury.

Ai miei coautori Dario Catalano e Federico Savasta, grazie per le vostre idee perspicaci, la vostra pazienza e la vostra attenzione. Un ringraziamento speciale a Federico per aver corretto attentamente il mio italiano e per la sua positività.

A mi futuro co-autor (toco madera) Darío Fiore, gracias por recibirme en Madrid para mi Post-Doc ¡Estoy deseando que llegue! Y también por su flexibilidad en las condiciones de inicio, que (con suerte) hará que mi defensa sea un poco menos angustiante.

Thank you Dennis Hofheinz for accepting to have me as a summer intern; even though things didn't turn out as expected, the week I spent with the crypto group at ETH was *brilliant*, I regret not having been able to spend the intended months with you all, and I hope I will have more time to dedicate to our project in the short term future.

To the Crypto group of Aarhus University, and in particular Claudio Orlandi, thank you for your warm welcome, dynamism and enthusiasm. I also hope that I will have the opportunity to come back, and spend a decent amount of time working with the group.

Merci aux (ex-)membres de l'équipe AriC pour votre bonne humeur lors des pauses café (et le reste du temps aussi d'ailleurs). En particulier, merci Fabrice pour ton rire résonnant, ton sourire contagieux; Alice pour ton calme rassurant, ton humour pétillant et tranchant; et à vous deux pour vos nombreux conseils. Merci Weiqiang d'avoir été un super co-bureau, l'équilibre parfait entre discrétion et fou-rires. Merci Florent pour ton amitié tout simplement. Thank you Miruna and Radu for the energy and fun you bring to the team every time you come to Lyon, and for risking your life rock climbing with me. Thank you Huyen for sharing the office, and for introducing me to flavours I had never imagined existed. Merci Octavie pour ta tchatche, et la bonne humeur que tu transmet à tous ceux qui t'entourent. Merci Nathalie de m'aider à m'aérer l'esprit avec nos footings au parc Gerland, et pour tous les moments agréables passés avec toi; merci à toi et à Natacha pour l'organisation des repas des informaticiennes. J'en profite pour remercier Marie×2, Chiraz, Nelly, Kadiatou et Virginie d'avoir répondu à mes milles et unes questions et d'avoir géré avec patience et efficacité le tas de nœuds qu'est (à mes yeux) l'administration liée à ma thèse. Thank you Joris, Elena, Anastasia, Alexandre for always being up for a beer; Alonso for humoring my spanish, thank you all other *young* (current and

[1] L'ordre est alphabétique, pas de jaloux.

ex) AriC members for sharing drinks and culinary experiments (in pre-covid times of course) at the *gouters AriC*.[2] Un grand merci aux ‘~~ex~~-grimpeurs’ d’avoir animé bon nombre de mes vendredi midis, pour ces sorties en falaise, ces BBQ copieux, et pour mon premier aperçu des anneaux de Saturne, et tout particulièrement à Alain, pour m’avoir fait découvrir les petites cantines, et d’avoir accepté la tâche ingrate de relire mon introduction. Merci Nicolas Louvet pour tes blagues, tes potins, et pour tes passages réguliers m’offrant discussions et distraction.

Dans les contrées occidentales du LIP, j’aimerais aussi remercier Etienne, qui m’a fait découvrir le VTT et dont l’énergie débordante m’a bien aidée cette dernière année; ainsi que Alexandre qui m’a encouragé en les moments les plus difficiles de la rédaction de ma thèse. Plus loin encore, sur le territoire bordelais, merci Christine Bachoc pour tes mots encourageants qui ont résolument contribué à ma reprise d’études.

D’un point de vue plus personnel (mais tout aussi important à mon cheminement), sans Antoine, mon colocataire à la fois débordant d’énergie et sensible, ma santé mentale en aurait pris un coup: notre gourmandise partagée (à noter son hummus exquis) et sa simple présence a su me distraire de nombreuses préoccupations. Merci Fany, pour toutes les sorties au mur et en montagne mais aussi de ne jamais fatiguer et de danser jusqu’au bout de la nuit. Merci Lucas, Juanadil et Julien, pour la légèreté que vous avez apporté apporté à mes dernières semaines à Lyon. Merci Melissa et Carlos, pour votre folie, votre créativité débordante et pour l’aura de rire et de joie que vous portez autours de vous. Merci Alice d’être une des personnes les plus fortes que je connaisse, de croire qu’il y a en moi une super-héroïne.

Merci à la fanfare Marcel Frontale, et à l’atelier du Chat Perché. Je ne vais pas tous vous nommer, mais rarement ai-je rencontré de groupe de personnes aussi positives, altruistes, ouvertes, et fondamentalement belles que vous.

Grazie a te Fra, anche se niente è mai semplice, per le innumerevole volte che mi hai ascoltato, e mi hai consigliato. Per le serate a *la triche*, la giocoleria e il tuo contagioso spirito Peter Pan.

Finally, to end these acknowledgements with an original touch, thank you Jane and Tony for the huge freedom you have always granted me.

[2] Et oui, à 28 ans ça fait encore des goûters.

Contents

Résumé long en français

Dans un schéma de chiffrement à clé publique, chaque utilisateur a sa propre clé secrète qui, comme son nom l'indique, est maintenue secrète. Celle-ci est associée à une clé publique, mise à disposition de tous. N'importe qui, en utilisant une clé publique, peut chiffrer un message confidentiel et l'envoyer au propriétaire de la clé. Afin que le système de chiffrement soit considéré sûr, il est nécessaire que, sans la clé secrète associée qui permet de déchiffrer, aucune information ne puisse fuir sur le message clair sous-jacent hormis ce qui peut être appris sans même voir le texte chiffré (par exemple, la langue d'une communication).

Dans ce qui précède, le terme "sûr" a été employé de manière assez vague ; définir cette notion est en fait une tâche non triviale en cryptographie. Afin de garantir un niveau élevé de confiance dans la sécurité des systèmes cryptographiques, la sécurité dite *prouvée* c'est développée. Un des premiers exemples de sécurité prouvée fut présenté il y a près de quarante ans dans les travaux fondateurs de Goldwasser et Micali [GM84]. Ils se fondent sur le principe que la sécurité des systèmes cryptographiques repose sur des hypothèses mathématiques précises. Ces hypothèses peuvent être générales (comme l'existence de fonctions à sens unique) ou spécifiques (comme la difficulté de calculer des logarithmes discrets dans certaines familles de groupes, ou la factorisation des nombres entiers). L'argument de sécurité est une réduction, au sens de la théorie de la complexité, transformant tout adversaire contre un protocole cryptographique en un algorithme qui résout le problème mathématique sous-jacent. Afin de factoriser les efforts des cryptanalystes, et de se reposer sur des problèmes mathématiques étudiés depuis bien avant l'existence de la cryptographie moderne (voir même des ordinateurs), il est préférable de se réduire à un petit nombre de problèmes ciblés.

La Cryptographie Avancée

Le chiffrement à clé publique permet donc de communiquer de façon sûre et efficace via un canal non sécurisé. Il est clair qu'aujourd'hui, du fait de leur omniprésence dans la société moderne, nous utilisons les technologies de l'information pour bien plus que la simple communication. En effet, de nombreuses tâches, impliquant souvent un traitement de données confidentielles, sont actuellement réalisées en ligne. Il est essentiel que ce flux colossal d'information soit protégé, afin que nul individu, trop curieux ou mal intentionné, ne puisse accéder à des informations auxquelles il ne devrait pas avoir accès, ni abuser d'une quelconque manière de son pouvoir. Dans le même temps, si les solutions visant à protéger ces données entrainent un surcoût excessif en matière de temps de calcul ou de bande passante, ces solutions ne seront guère adoptées en pratique.

Ainsi, pour faire face à de nombreux problèmes du "monde réel", qui ne se limitent pas à sécuriser des communications, il faut définir des *primitives cryptographiques avancées.* Parmi ces problèmes du monde réel, on peut citer par exemple les systèmes de vote sécurisés, ou le fait de permettre à un appareil ayant des capacités de calcul limitées de déléguer ses calculs à un autre appareil, potentiellement malveillant, mais nettement plus puissant.

Les protocoles réalisant ces primitives avancées doivent être à la fois efficaces, de sorte qu'ils puissent remplacer sans encombre les protocoles traditionnels non sécurisés, et prouvés sûrs, afin de garantir un niveau de confiance élevé à l'égard de ces solutions. De plus, en raison de l'évolution constante des attaquants, et des tâches de plus en plus sensibles effectuées sur internet (prise de rendez-vous médicaux, remboursements d'actes de soin, paiements en ligne, élections etc.), la nécessité d'identifier et de mitiger les points individuels de défaillance est primordiale. Afin d'illustrer cela, considérons deux exemples concrets de tâches que le chiffrement traditionnel ne peut accomplir ; pour chacun d'entre eux, nous présentons une solution spécifique fournie par la cryptographie avancée.

Exemple 1 - Accès mesuré à l'information. Considérons des chercheurs ayant besoin d'effectuer une analyse sur les données médicales possédées par un hôpital. Supposons par ailleurs que l'hôpital veuille partager ces données, afin de faciliter les innovations et découvertes biomédicales ; ceci sans divulguer d'informations superflues sur l'identité des patients pour des raisons évidentes de confidentialité. On peut raisonnablement présumer que cet hôpital dispose d'une base de données chiffrée contenant les informations confidentielles des patients. Si la base de données est chiffrée en utilisant un chiffrement traditionnel, les chercheurs – ne disposant que de la base de données chiffrée – n'apprennent rien sur les données sous-jacentes ; à l'inverse, si l'hôpital leur accorde la clé de déchiffrement, ils apprennent tout le contenu de la base de données. Il est probable que cela représente bien plus d'informations que ne le requiert leur analyse. Une autre possibilité serait que l'hôpital lui-même effectue l'analyse sur la base de données, et partage les résultats. Pour des raisons évidentes (personnel hospitalier surchargé, non-qualifié pour l'exécution de ces calculs...), cette solution est insatisfaisante.

Une solution idéale serait de permettre aux chercheurs d'effectuer leur analyse statistique à partir d'une base de données chiffrée, de telle sorte qu'ils n'apprennent que le résultat de l'analyse, mais rien d'autre sur le contenu de la base de données. Si une autorité de confiance (ce pourrait être l'hôpital) approuve cette analyse avant de donner aux chercheurs les moyens de l'effectuer, alors cette solution protège la confidentialité des patients, tout en fournissant de précieuses ressources à la recherche biomédicale.

Le chiffrement fonctionnel, dont l'étude a été amorcée par O'Neill [O'N10] et indépendamment par Boneh et al. [BSW11], est une primitive cryptographique avancée résultant d'une série de raffinements du chiffrement à clé publique. Cette primitive permet de contrôler, à partir d'un seul chiffré, la portion des informations sous-jacentes que chaque utilisateur peut extraire. Plus précisément, le chiffrement fonctionnel permet à un utilisateur de récupérer une fonction $f(m)$ du message chiffré m, sans divulguer d'autres informations sur m.

La primitive permet de générer des clés de déchiffrement fonctionnelles sk_{f_i} – associées à des fonctionnalités spécifiques f_i – à partir d'une clé secrète maîtresse msk ; celles-ci sont attribuées aux bénéficiaires concernés. Un chiffré c, chiffrant le texte en clair m, est mis à disposition. À partir de c un utilisateur possédant sk_{f_i} peut récupérer $f_i(m)$ en déchiffrant c avec sk_{f_i}.

Pour reprendre l'exemple précédent, l'hôpital peut chiffrer sa base de données à l'aide d'un schéma de chiffrement fonctionnel et diffuser librement le chiffré qui en résulte. Ensuite, à la demande d'un groupe de chercheurs, souhaitant calculer une fonction f appliquée à la base de données, l'hôpital calcule une clé de déchiffrement fonctionnelle sk_f associée à f et la leur renvoie. Cette clé de déchiffrement leur permet de calculer la fonction demandée de la base de données, et même de toute base de données mise à jour, chiffrée avec la même clé publique.

Malgré l'attention considérable que la recherche scientifique consacre à cette question, la mise au point de systèmes de chiffrement fonctionnel efficaces permettant à la fois d'évaluer toute fonction et d'atteindre un niveau de sécurité satisfaisant reste un problème ouvert. Toutes les constructions offrant une telle fonctionnalité sont loin d'être utilisables en pratique. En outre, soit elles limitent le nombre de clés de déchiffrement que l'adversaire peut demander [SS10], soit elles s'appuient sur des hypothèses cryptographiques peu comprises et insuffisamment étudiées [GGHZ16]. De ce fait, la recherche s'est penchée sur des chiffrements fonctionnels se restreignant au calcul de classes de fonctions spécifiques, dans l'espoir que de telles primitives puissent être réalisées sous des hypothèses cryptographiques bien comprises, tout en étant suffisamment efficaces pour bénéficier à des applications concrètes.

Un exemple probant est l'étude du chiffrement fonctionnel calculant des produits scalaires, formalisé en premier lieu par Abdalla et al. dans [ABDP15]. Cette variante restreint la fonctionnalité calculée au produit scalaire de deux vecteurs (l'un résultant d'une clé de déchiffrement, l'autre d'un message). Une telle restriction permet non seulement de développer notre maîtrise du chiffrement fonctionnel, mais bénéficie également à diverses applications pratiques, allant de son utilisation immédiate, qui permet d'effectuer des opérations linéaires sur des données chiffrées, à la construction d'autres systèmes cryptographiques plus complexes [ALS16, ABP$^+$17, KY19].

Dans cette thèse, nous présentons des constructions génériques de chiffrement fonctionnel calculant des produits scalaires. Ceux-ci peuvent être instanciés à partir d"hypothèses algorithmiques diverses et sont suffisamment efficaces pour être utilisés dans des systèmes d'information modernes à grande échelle.

Exemple 2 - Le caractère secret des clés secrètes. Considérons à présent une problématique quelque peu orthogonale. Que l'on considère un chiffrement symétrique, un chiffrement à clé publique, ou bien n'importe quel schéma cryptographique visant à protéger la sécurité d'un système, si la clé secrète tombe entre de mauvaises mains, alors la sécurité de l'ensemble du système est compromise.

La nécessité que cette clé reste secrète pose plusieurs problèmes. Tout d'abord, le stockage de la clé secrète en un unique lieu, qu'il s'agisse d'un ordinateur à usage personnel, d'un serveur ou encore d'une carte à puce, réduit la sécurité du système cryptographique à la difficulté de pénétrer dans ce périphérique et de voler la clé. Cela peut être (c'est en fait souvent le cas) bien plus facile pour un attaquant que de casser le système cryptographique lui-même. Par ailleurs, l'absence de sauvegarde de la clé introduit le risque de sa perte si une défaillance logicielle ou matérielle survenait. Paradoxalement, le fait de conserver la clé sur plusieurs dispositifs ne fait que faciliter la tâche de l'attaquant, car les cibles qu'il peut choisir se multiplient, réduisant ainsi la sécurité du système à celle du périphérique le plus vulnérable.

Un exemple tangible où le vol de clés peut avoir des conséquences fâcheuses est dans le contexte des crypto-monnaies. Une manière répandue de valider les transactions de crypto-monnaies consiste à exiger du payeur qu'il authentifie sa transaction. Pour ce faire, il signe la transaction avec sa clé de signature *secrète* associée à un protocole de signature numérique. Par conséquent, toute infraction se traduisant par le vol de cette clé de signature peut entraîner de lourdes pertes financières.

L'idée du partage de secret fut introduite en 1979 par Blakley [Bla79] et Shamir [Sha79] afin de résoudre le problème ci-dessus. Un système de partage de secret permet de partager un secret parmi un groupe d'individus de manière à ce que seules des coalitions bien définies de ces individus puissent ensemble récupérer le secret, tandis qu'aucune autre coalition ne peut obtenir d'informations à son sujet. Un système à seuil (t, n) [Sha79] est un cas particulier de partage de secret. Dans ce cas, le secret est partagé entre n participants de sorte que tout sous-ensemble de strictement plus de t participants peut reconstruire le secret, tandis que s'ils ne sont que t ou moins, aucune information sur le secret n'est divulguée. Il est possible de renforcer plus encore la sécurité de ces deux concepts en utilisant la cryptographie à seuil, introduite par les travaux de Boyd [Boy86], Desmedt [Des88], et Desmedt et Frankel [DF90]. Comme pour le partage de secret à seuil mentionné ci-dessus, la cryptographie à seuil (t, n) permet à n utilisateurs de partager une clé secrète commune de sorte que tout sous-ensemble de $t+1$ d'entre eux puisse utiliser cette clé pour déchiffrer ou signer, tandis que toute coalition de t ou moins utilisateurs ne peut rien faire. La particularité de ce système est qu'il permet d'utiliser une clé partagée sans jamais la reconstruire explicitement. Cela implique qu'un sous-ensemble de strictement plus de t participants doit participer activement au protocole chaque fois que la clé secrète est utilisée. En particulier, cela signifie qu'un adversaire doit corrompre simultanément au moins $t + 1$ des participants (ou de manière équivalente pénétrer dans $t + 1$ périphériques) pour pouvoir effectuer l'opération sensible en question. De tels protocoles distribués relèvent du domaine du calcul multiparti sécurisé : ce sous-domaine de la cryptographie, qui date des articles de Yao [Yao82] et Goldreich et al. [GMW87], a pour but de développer des méthodes permettant à plusieurs utilisateurs de calculer collectivement une fonction à partir de leurs données respectives, tout en préservant la confidentialité de ces données.

Nous élaborons des variantes à seuil du schéma de signature utilisé pour valider les transactions Bitcoin : l'algorithme de signature numérique EC-DSA. Comme nous allons le voir, bien que pour d'autres protocoles de signature, des variantes à seuil efficaces existent depuis des décennies (c'est le cas des signatures RSA [GJKR96a] et Schnorr [Sch91,SS01]), la construction de variantes à seuil efficaces de EC-DSA s'est avérée nettement plus difficile.

La conception de systèmes cryptographiques avancés peut, à première vue, sembler une tâche ardue. En effet, les objectifs de tels systèmes sont bien plus complexes – à la fois en matière de fonctionnalité et de sécurité – que ceux de la cryptographie traditionnelle. Par conséquent, une approche logique pour concevoir de tels systèmes consiste à partir de briques élémentaires plus simples. Si ces entités élémentaires sont quelque peu malléables, dans le sens où l'on peut les combiner et les assembler, elles nous donnent les moyens de réaliser des fonctionnalités complexes. En outre, s'il est possible d'établir une interface claire définissant les propriétés de sécurité assurées par ces briques, il est alors bien plus simple de prouver la sécurité des constructions qui en résultent. Si l'on requiert de plus que ces briques (et par conséquent les constructions) puissent être instanciées à partir d'un large panel d'hypothèses cryptographiques – elles sont alors dites *génériques* – l'on évite de mettre tous nos œufs dans un même panier. En effet, concentrer ses efforts et ses ressources à l'élaboration de schémas à partir d'une seule hypothèse cryptographique présente un risque : si le problème sous-jacent se révèle moins difficile que prévu, on pourrait tout perdre. Enfin, si les constructions résultant de ces briques sont *fines*, c'est à dire soigneusement conçues, et que leur analyse de sécurité est précise, les systèmes cryptographiques obtenus peuvent être extrêmement flexibles et efficaces.

Cette approche modulaire, générique et fine a été celle de ma thèse.

Fonctions de Hachage Projectives

La principale brique de base que nous utilisons est la notion de *fonction de hachage projective*. Ce concept a été formalisé par Cramer et Shoup il y a une vingtaine d'années [CS02]. Les fonctions de hachage projectives ont été initialement introduites afin de permettre la réalisation de systèmes de chiffrement génériques sûrs face à des adversaires qui non seulement observent les communications chiffrées, mais tentent activement d'obtenir des informations en exécutant le protocole dans des conditions non conformes à sa spécification. Un tel comportement imprévu pourrait, entre autres, entraîner la fuite d'informations sur la clé secrète.

Dans le contexte du chiffrement à clé publique, pour faire face à des adversaires qui ne font qu'espionner (dits *passifs*), il suffit d'assurer la confidentialité des messages chiffrés. Afin de *prouver* qu'un chiffrement est sûr face à de tels adversaires, on donne à l'adversaire les paramètres publics du système, ainsi qu'un chiffré challenge c^* chiffrant soit m_0 soit m_1, où m_0 et m_1 sont des messages choisis par l'adversaire. On démontre alors que l'adversaire ne peut distinguer lequel des messages a été chiffré. Afin d'assurer une protection supplémentaire contre les adversaires *malicieux* (dits *actifs*), il faut également veiller à l'intégrité des textes chiffrés. Afin de modéliser cela, en plus des paramètres publics du système, on octroie à l'adversaire le résultat du déchiffrement de chiffrés de son choix (à l'exception du chiffré challenge).

En vue d'adopter une approche modulaire dans la conception de systèmes cryptographiques avancés, nous créons des fonctions de hachage projectives avec des propriétés *homomorphes.* Un cryptosystème est dit homomorphe si l'on peut publiquement manipuler des données confidentielles. Par exemple, un système de chiffrement est linéairement homomorphe si, connaissant les chiffrés c_1 et c_2 chiffrant les textes en clair m_1 et m_2, il est possible de produire un nouveau chiffré c qui, lors de son déchiffrement, retourne la somme $m_1 + m_2$. Paradoxalement, bien que les systèmes de chiffrement possédant de telles propriétés homomorphes sont extrêmement utiles pour la conception de cryptosystèmes avancés plus complexes, ceux-ci ne peuvent atteindre le niveau de sécurité maximal attendu d'un système de chiffrement, c'est-à-dire la sécurité face aux adversaires actifs[3].

Pour en revenir à nos fonctions de hachage projectives, nous définissons également de nouvelles propriétés caractérisant ces derniers, qui comprennent les critères essentiels nécessaires au bon fonctionnement et à la sécurité de nos constructions plus complexes. Précisément, à partir de la riche boîte à outils que constituent ces fonctions de hachage projectives, nous concevons des schémas de chiffrement linéairement homomorphes, des schémas de chiffrement fonctionnels calculant des produits scalaires, et des protocoles de signature EC-DSA à seuil. En s'inspirant des techniques de [CS02], nous sécurisons ces deux dernières constructions contre tout adversaire passif ou actif.

Chiffrement Linéairement Homomorphe et le Cadre CL

Comme mentionné précédemment, à partir de fonctions de hachage projectives ayant des propriétés homomorphes, il est possible de créer des schémas de chiffrement linéairement homomorphes. En fait, le chemin qui a conduit à la rédaction de cette thèse n'a pas commencé avec des fonctions de hachage projectives, mais plutôt par l'observation que l'on peut concevoir des systèmes de chiffrement avancés dont la sécurité est prouvée, en s'appuyant sur la malléabilité du chiffrement linéairement homomorphe.

Le premier schéma de chiffrement homomorphe (également le premier chiffrement probabiliste) fut suggéré par Goldwasser et Micali dans [GM84], tandis que l'un des schémas les plus aboutis, normalisé dans la norme ISO/IEC-18033-6, fût conçu par Paillier [Pai99].

Cette thèse se fonde sur un schéma de chiffrement bien plus récent, possédant des propriétés innovantes et séduisantes. Précisément, nous nous appuyons sur le cadre introduit par Castagnos et Laguillaumie à CT-RSA 2015 [CL15]. Ce cadre (le cadre CL) leur a permis de concevoir un schéma de chiffrement linéairement homomorphe bénéficiant de la propriété peu commune d'avoir un espace de texte clair d'ordre premier q, où q peut être choisi (avec certaines restrictions) indépendamment du paramètre de sécurité. Par ailleurs, leur schéma est le premier schéma de chiffrement linéairement homomorphe à la fois efficace, et dont la sécurité repose uniquement sur une hypothèse de type logarithme discret.

En s'appuyant sur le cadre CL, nous formalisons de nouvelles hypothèses algorithmiques à

[3] N'importe qui peut calculer un chiffré de zero, et peut donc – à partir du chiffré challenge c^* chiffrant m_b, pour un certain $b \in \{0, 1\}$ – calculer un chiffré de $m_b + 0 = m_b$ différent de c^*. L'adversaire actif peut alors demander le déchiffrement de ce dernier, découvrant ainsi quel message était chiffré.

partir desquelles nous élaborons de précieux outils ; en particulier, nous concevons des fonctions de hachage projectives ayant des propriétés homomorphes. Celles-ci permettent d'abstraire les caractéristiques du cadre, ce qui permet une plus grande généricité dans nos travaux ultérieurs.

Nous observons que, lors de l'instanciation de nos constructions génériques, les schémas les plus efficaces résultent soit du cadre CL, soit de celui dans lequel Paillier a conçu son cryptosystème [Pai99]. Puisqu'un certain nombre de nos travaux s'inspirent de constructions pré-existantes basées sur le chiffrement de Paillier, nous nous y comparons, et ferons souvent référence à son schéma. Qui plus est, de nombreuses similitudes peuvent être observées entre la structure des deux cadres : ils consistent tous deux en un groupe G (dans lequel un certain problème calculatoire est estimé difficile) contenant un sous-groupe F, généré par f; dans le sous-groupe F le calcul de logarithmes discrets peut être effectué efficacement. Les deux chiffrements encodent les messages clairs dans l'exposant de f, et masquent cet encodage avec un élément du groupe G. Cela permet d'effectuer des opérations linéairement homomorphes sur les données chiffrées (puisque $f^{m_1}f^{m_2} = f^{m_1+m_2}$), tout en admettant un déchiffrement efficace, car étant donné f^m, la valeur de m peut être calculée rapidement.

Instanciation du Cadre CL à partir de Groupes de Classes

Le cadre établi par Castagnos et Laguillaumie présente une caractéristique essentielle que nous exploitons tout au long de cette thèse. En effet, comme expliqué plus haut, il repose sur l'existence d'un groupe G dans lequel on suppose qu'un certain problème algorithmique est difficile (cette difficulté est essentielle à la sécurité des cryptosystèmes résultant du cadre), ainsi que d'un sous-groupe F de G, d'*ordre premier*, dans lequel il est facile de calculer des logarithmes discrets.

Afin de matérialiser cette propriété peu commune, ils utilisent une structure algébrique particulière : les groupes de classes d'idéaux dans un corps quadratique imaginaire. Ceux-ci possèdent certaines spécificités qui semblent difficiles à trouver dans d'autres groupes.

Les corps quadratiques imaginaires furent proposés comme base pour la conception de cryptosystèmes à clé publique à la fin des années 1980 par Buchmann et Williams [BW88]. Ils présentèrent une adaptation de l'échange de clés Diffie-Hellman dans les corps quadratiques imaginaires et décrivirent brièvement une adaptation du chiffrement Elgamal dans ce même cadre. De nombreux cryptosystèmes reposant sur cet objet algébrique ont depuis été conçus, pour lesquels des implémentations efficaces ont été discutées, entre autres, dans [JW09].

Il convient de noter que l'intérêt pour la cryptographie se fondant sur les groupes de classes a décliné après les attaques de Castagnos et al. [CL09, CJLN09] contre le cryptosystème NICE [HPT99, PT00] et sa variante réelle [JSW08]. Ces attaques ne s'appliquent cependant pas au cadre CL : elles récupèrent la clé secrète en exposant la factorisation du discriminant du corps, alors que dans le cadre CL cette factorisation est publique.

Cela dit, cette branche de la cryptographie connaît un regain d'intérêt du fait qu'elle offre des solutions flexibles et efficaces. On peut citer son rôle dans la construction d'accumulateurs ne nécessitant pas de configuration centralisée par une autorité de confiance [Lip12], de protocoles d'intervertion de chiffrement [CIL17], de fonctions à délai vérifiables [BBBF18, Wes18], d'arguments de connaissance à divulgation nulle de connaissance à la fois non-interactifs, succincts et ne nécessitant pas de configuration centralisée par une autorité de confiance [BFS20], et, comme nous le verrons au cours de cette thèse, de chiffrement fonctionnel calculant des produits scalaires et de signatures à seuil EC-DSA.

En ce qui concerne les hypothèses sous-jacentes, la sécurité des cryptosystèmes découlant du cadre CL est liée à la difficulté de calculer le nombre de classe. Les meilleurs algorithmes connus actuellement pour résoudre ce problème (cf. [BJS10]) ont une complexité asymptotique plus élevée que ceux résolvant le problème de la factorisation (ou le problème du logarithme discret dans les corps finis). Cela permet l'utilisation de clés plus courtes pour les schémas de chiffrement CL que pour les schémas de chiffrement reposant sur la difficulté de la factorisation (tels que celui de Paillier). À cela s'ajoute le fait que les opérations arithmétiques dans les groupes de classes sont quasi-linéaires en utilisant de l'arithmétique rapide. Ces deux faits contribuent à l'efficacité de nos protocoles résultant du cadre CL.

Contributions

Bien que la plupart des résultats présentés aient été publiés ou soient actuellement en cours de révision, nous avons réorganisé nos résultats dans un souci de clarté, afin de les présenter dans un ordre plus logique et modulaire, plutôt que de suivre l'ordre chronologique de nos publications ou de regrouper les chapitres par publication. L'objectif est bien sûr de favoriser la facilité de lecture.

Les résultats présentés ci-dessous sont principalement tirés d'une soumission actuelle (en collaboration avec Guilhem Castagnos et Fabien Laguillaumie), et des articles suivants :
[CLT18a] présenté à Asiacrypt 2018 (en collaboration avec Guilhem Castagnos et Fabien Laguillaumie); [CCL$^+$19] présenté à Crypto 2019 et [CCL$^+$20] présenté à PKC 2020 (tous deux en collaboration avec Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie et Federico Savasta).

Fonctions de Hachage Projectives à partir du Cadre CL

Dans cette thèse, nous formalisons de nouvelles hypothèses cryptographiques au sein du cadre CL. Nous argumentons la difficulté des problèmes associés lorsque le cadre est instancié avec des groupes de classes d'idéaux d'un corps quadratique imaginaire.

Nous définissons ensuite formellement la notion de fonction de hachage projective, ainsi que les diverses propriétés (dont un grand nombre sont propres à nos travaux) nécessaires au bon fonctionnement et à la sécurité de nos constructions. Nous illustrons toutes nos définitions et propriétés à l'aide de trois exemples récurrents. Le premier découle de l'hypothèse décisionelle bien connue de Diffie-Hellman dans un corps fini ; le but de cet exemple étant de fournir une instanciation avec laquelle la plupart des lecteurs seront familiers. Les deux autres exemples récurrents découlent des hypothèses susmentionnées définies dans le cadre CL, appelées respectivement HSM-CL et DDH-f.

Puisque les fonctions de hachage projectives sont utilisées comme briques de base pour nos constructions ultérieures, tous nos cryptosystèmes peuvent être instanciés par des hypothèses dans le cadre CL, et donc à partir de cryptographie fondée sur les groupes de classes. Toutes nos définitions tiennent compte du fait que les groupes dans lesquels nous travaillons peuvent être d'ordre inconnu. Ce phénomène est en fait inhérent au cadre CL. Nous concevons en premier lieu des schémas de chiffrement linéairement homomorphes à partir de ces fonctions de hachage projectives. Leur conception suit une modeste adaptation de la construction générique de Cramer-Shoup [CS02].

Preuves et Arguments à Divulgation Nulle de Connaissance Étoffant le Cadre

Pour nos protocoles de signature distribuée – puisque plusieurs partis effectuent conjointement un calcul – nous devons assurer que les partis se conduisent correctement et se conforment au protocole. Bien entendu, lorsqu'un parti prouve qu'il s'est comporté correctement, aucune information sur sa contribution confidentielle au calcul ne doit fuir. Une *preuve à divulgation nulle de connaissance* est un protocole interactif entre un prouveur et un vérificateur, visant à démontrer qu'une affirmation est vraie en ne révélant rien de plus que la véracité de l'affirmation.

Tandis qu'une *preuve* à divulgation nulle de connaissance convainc *statistiquement* le vérificateur de cette vérité, un *argument* à divulgation nulle de connaissance convainc le vérificateur sous des *hypothèses calculatoires*. Nous proposons des preuves et des arguments à divulgation nulle de connaissance pour le cadre CL, prouvant par exemple qu'un texte chiffré pour l'un des systèmes de chiffrement mentionnés plus haut est calculé de manière honnête.

Notre cadre se distingue des groupes d'ordre connu où il est possible de reconnaître efficacement les éléments du groupe, puisque dans le cadre CL, nous travaillons avec un sous-groupe cyclique $G^q := \langle g_q \rangle$ d'un groupe plus grand $\widehat{G}$, où g_q est d'ordre inconnu, et les éléments de G^q ne sont pas efficacement reconnaissables. Seuls les éléments résidant dans $\widehat{G}$ peuvent être efficacement reconnus. Il est impératif d'en tenir compte dans tous nos protocoles afin d'assurer une sécurité contre tout adversaire malicieux.

Du fait que nos systèmes de chiffrement suivent une structure semblable au chiffrement Elgamal, nous disposons de tout un arsenal de preuves et d'arguments à divulgation nulle de connaissance pour les chiffrés Elgamal ainsi que pour prouver la connaissance de logarithmes discrets dans des groupes d'ordre connu (par exemple, [Sch90, CP93]) sur lesquels nous pouvons nous appuyer. S'inspirant en outre des travaux de Dåmgard et Fujisaki sur les arguments de connaissance [DF02], et sur les travaux de Girault, Poupard et Stern sur les preuves masquant statistiquement les contributions secrètes des partis [GPS06], qui tous deux considèrent également des groupes d'ordre inconnu, nous concevons diverses techniques pour surmonter les complications susmentionnées, avec divers compromis entre sécurité et efficacité.

Preuves de Sécurité sans Pertes et Efficacité Accrue pour le Chiffrement Fonctionnel

Nous nous appuyons sur les idées sous-jacentes à la construction générique de [CS02] qui permet, à partir de fonctions de hachage projectives, de concevoir des schémas de chiffrement à clé publique sûrs contre tout adversaire actif. Nous adaptons ces propriétés au cas du chiffrement fonctionnel calculant des produits scalaires. Les messages sont alors des vecteurs, et les adversaires ont accès à des informations supplémentaires sur les paramètres secrets du système.

Précisément, nous définissons de nouvelles propriétés pour les fonctions de hachage projectives ; ces propriétés saisissent les critères essentiels permettant de construire des schémas de chiffrement fonctionnels calculant des produits scalaires. Afin d'isoler ces propriétés spécifiques, nous décortiquons les techniques de preuve utilisées par Agrawal et al. [ALS16] qui conçoivent des schémas de chiffrement fonctionnels calculant des produits scalaires, prouvés sûrs contre des adversaires passifs, mais à partir d'hypothèses algorithmiques spécifiques (et donc non-génériques). Cette décomposition nous permet d'identifier les étapes clés de leurs preuves, ce qui profite à notre approche modulaire et générique tout en préservant l'efficacité de leurs schémas.

Nous présentons ensuite des constructions génériques pour le chiffrement fonctionnel calculant des produits scalaires à partir de fonctions de hachage projectives possédant ces propriétés. Notre première construction est prouvée sûre contre tout adversaire passif, c'est-à-dire que lorsque le système est utilisé dans les conditions attendues, il ne fuit pas plus d'informations que prévu. En instanciant cette construction à partir de l'hypothèse décisionelle de Diffie Hellman, ou de l'hypothèse décisionelle de résiduosité composite (DCR), nous retrouvons les constructions de [ALS16, ABDP16], avec la même borne sur la sécurité ; lorsqu'elle est instanciée à partir de nos hypothèses dans le cadre CL, nous obtenons de nouvelles constructions. Les instanciations basées sur DCR et sur le cadre CL donnent les schémas les plus efficaces à ce jour, et calculent des produits scalaires dans $\mathbf{Z}$.

Délaissant quelque peu la construction générique, nous construisons également des schémas de chiffrement fonctionnel calculant des produits scalaires *modulo un nombre premier* à partir du cadre CL, et obtenons les schémas les plus efficaces à ce jour ; nous comparons la vitesse ainsi que la taille des chiffrés et des clés secrètes de notre schéma le plus efficace à ceux de [ALS16] afin d'illustrer les améliorations concrètes que nous réalisons.

Cette première construction générique sûre contre tout adversaire passif est en quelque sorte un tremplin pour construire un tel système sûr contre tout adversaire actif. En effet, nous étendons ensuite la construction générique mentionnée ci-dessus de manière à en assurer la sécurité contre tout adversaire actif, pouvant s'écarter arbitrairement du protocole. La construction peut être instanciée à partir de tous nos exemples récurrents. Nous observons que la construction obtenue est très proche de celle de Benhamouda et al. [BBL17], atteignant les mêmes objectifs en matière de sécurité et de fonctionnalité. Toutefois, nos techniques de preuves divergent fondamentalement, nous permettant d'améliorer nettement la réduction de sécurité. Cela justifie en particulier l'utilisation de clés secrètes et de chiffrés bien plus courts, tout en maintenant un niveau de sécurité équivalent.

Nos travaux sont les premiers à démontrer que le chiffrement fonctionnel calculant des produits scalaires et assurant la sécurité contre des adversaires actifs est utilisable en pratique. En effet, lorsque nous instancions notre construction générique, nous obtenons des schémas suffisamment efficaces pour être utilisés dans des systèmes d'information modernes à grande échelle. Afin d'illustrer cela, nous effectuons une comparaison théorique détaillée de l'efficacité de nos schémas, instanciés à partir de diverses hypothèses, à ceux de [BBL17].

Protocoles Distribués pour l'Algorithme de Signature Standardisé EC-DSA

Afin de concevoir des variantes à seuil de l'algorithme de signature EC-DSA, diverses solutions ont été proposées reposant sur les propriétés homomorphes du chiffrement de Paillier. Parmi ces solutions, certaines gèrent le cas biparti (c'est-à-dire $t = 1$ et $n = 2$) ; cette question a d'abord été traitée par Mackenzie et Reiter dans [MR01], puis plus efficacement par Lindell dans [Lin17a]. Tandis que d'autres travaux récents gèrent un seuil entièrement configurable (tout $t < n$), à commencer par le protocole de Gennaro, Goldfeder et Narayanan dans [GGN16].

En dépit de la popularité du système de chiffrement de Paillier pour la construction de protocoles EC-DSA à seuil, nous montrons que ce cryptosystème n'est en fait pas adapté à la tâche. En effet, pour des raisons qui seront discutées dans la suite de cette thèse, ce choix entraîne un surcoût en termes de bande passante, et, dans certains cas, l'utilisation d'une hypothèse interactive non standard portant sur le cryptosystème Paillier afin de prouver la sécurité du protocole. En revanche, si l'on utilise nos schémas de chiffrement linéairement homomorphes résultant du cadre CL, nous supprimons la nécessité d'un certain nombre de preuves à divulgation nulle de connaissance et évitons le recours à toute hypothèse interactive.

Nous nous appuyons sur les idées de Lindell dans [Lin17a] et concevons le premier protocole EC-DSA biparti générique à partir de fonctions de hachage projectives ayant des propriétés homomorphes. Une instanciation issue de l'hypothèse HSM-CL dans le cadre CL résulte en un protocole efficace (notamment en termes de bande passante) avec une preuve de sécurité sans pertes ni hypothèse interactive. Nous fusionnons ensuite les idées sous-jacentes à notre protocole biparti et la construction proposée par Gennaro et al. dans [GG18] afin d'obtenir un protocole EC-DSA à seuil entièrement configurable. Ces deux constructions font appel à nos preuves et arguments à divulgation nulle de connaissance dans le cadre CL évoqués plus haut, permettant d'assurer que les participants se conduisent comme prévu.

Nous comparons l'efficacité de nos protocoles aux protocoles préexistant les plus performants, utilisant des techniques de construction similaires et atteignant la même fonctionnalité. Ces comparaisons montrent que, pour tous les niveaux de sécurité considérés, nos protocoles de signature consomment nettement moins de bande passante. En matière de rapidité, bien que nos protocoles de signature soient légèrement plus lents pour les niveaux de sécurité standard, la tendance est inversée pour des niveaux de sécurité plus élevés (sécurité de 192-bit et plus).

Organisation de la Thèse

Le reste de cette thèse est organisé comme suit. Chapitre 2 introduit les notations et prérequis nécessaires pour comprendre cette thèse. Cela comprend des connaissances techniques de base sur les groupes de classes d'idéaux des ordres d'un corps quadratique imaginaire.

Chapitre 3 rappelle le cadre CL et explique comment il peut être instancié à partir de groupes de classes. Nous enrichissons ensuite ce cadre en formalisant de nouvelles hypothèses cryptographiques à partir desquelles nous construisons des fonctions de hachage projectives. Celles-ci donnent à leur tour naissance à trois schémas de chiffrement linéairement homomorphes. Toujours dans le but d'améliorer le cadre CL, nous concevons une série de preuves et d'arguments à divulgation nulle de connaissance adaptés au cadre et aux schémas de chiffrement susmentionnés.

Dans le Chapitre 4 sont définies un certain nombre de nouvelles propriétés que doivent satisfaire les fonctions de hachage projectives afin de permettre la construction de schémas de chiffrement fonctionnel calculant le produit scalaire qui soient sûrs contre tout adversaire passif ou actif. À partir de celles-ci nous présentons ensuite des constructions génériques, avec preuves de sécurité, pour ce chiffrement fonctionnel. Chaque nouvelle définition et construction est illustrée à l'aide d'exemples récurrents issus des fonctions de hachage projectives du Chapitre 3.

Dans le Chapitre 5 nous présentons notre construction générique pour EC-DSA biparti à base de fonctions de hachage projectives et notre protocole EC-DSA à seuil entièrement configurable issu du cadre CL.

Enfin, Chapitre 6 conclut brièvement la thèse et soulève quelques questions restées ouvertes.

Chapter 1

Introduction

In a public key encryption scheme, each user has their own secret key which, as the name suggests, is kept secret. This secret key is associated to a public key, which is made publicly available. Anyone, using a public key, can encrypt a confidential message, and send it to the key's owner. Security requires that without the associated secret key, which allows to decrypt the resulting ciphertext, no information should leak on the underlying plaintext message, other than what may already be known without seeing the ciphertext (e.g. the language of a communication).

In the above we used the term *security* quite broadly, in fact defining what *secure* means is a non-trivial task in cryptography. In order to guarantee a high level of trust in the security of cryptosystems, these must be *proven secure.* An early example of provable security was introduced almost fourty years ago in the pioneering work of Goldwasser and Micali [GM84]. It relies on the principle that the security of cryptographic schemes is based on mathematically precise assumptions. These assumptions can be general (such as the existence of one-way functions) or specific (such as the hardness of the discrete logarithm problem in specific group families, or integer factorisation). The security argument is a reduction, in the complexity theory meaning, transforming any adversary against a cryptographic protocol into an algorithm that solves the underlying mathematical problem.

1.1 Advanced Cryptography

As we have seen, public key encryption resolves the problem of securely and efficiently communicating over an insecure channel. Now clearly, due to it's omnipresence in modern society, we use information technology for *far more* than mere communication. Indeed many tasks which often involve some form of computation over *private information* are performed online. It is essential that the huge flow of information which occurs here is protected, so that no overly curious or ill-intentioned individual can access information it should not, or in any way abuse of its power. At the same time, if solutions to protect users hinder their experience, evidence tends to show that these solutions will see little practical implementation.

Thus to solve many 'real world' problems, which go beyond communication security, *advanced* cryptographic primitives must be defined. Such real world problems include e.g. secure voting systems, or allowing a computationally limited device to outsource computations to some other device, potentially malicious, but much more computationally powerful.

These advanced cryptographic primitives must be both practical, so as to seamlessly substitute traditional insecure protocols; and provably secure, to ensure a high level of trust in these solutions. What is more, due to ever evolving adversaries, and increasingly sensitive tasks being performed over the internet (e.g. large transfers of cryptocurrencies) the need to identify and mitigate single points of failure is prevalent. To illustrate this, we state two concrete examples of tasks which traditional encryption does not cater for, and specific solutions provided by advanced cryptography.

Fine grained access to information. Consider a hospital, willing to share medical data with researchers so as to facilitate medical innovations and discoveries, without disclosing unnecessary information on the patients' identities for obvious confidentiality reasons. We reasonably assume that this hospital has an *encrypted* database containing patients' confidential information. If the database is encrypted using traditional encryption, the researchers – given only the encrypted database – learn nothing about the underlying data; conversely, if the hospital grants them the decryption key, they learn *the whole contents* of the database. This may be far more information that what is required for their analysis. Alternatively, the hospital itself could perform the analysis on the database, and share the results. For obvious reasons (e.g. hospital staff are already overrun and are not trained to perform these computations) this solution is suboptimal.

An ideal solution would be enabling researchers to perform their statistical analysis from an encrypted database, in such a way that they *only* learn the output of the analysis, but *nothing else* about the contents of the database. If a trusted authority (this could be the hospital) authorises this analysis prior to giving researchers the means to run it, this solution protects patient privacy, while providing priceless resources to biomedical research.

Functional encryption, whose general study was initiated by O'Neill in [O'N10] and independently by Boneh et al. in [BSW11], is an advanced cryptographic primitive which emerged from a series of refinements of public key encryption, allowing to control, given a single ciphertext, how much of the underlying data each user can recover. Specifically, functional encryption allows for a receiver to recover a function $f(m)$ of the encrypted message m, without revealing anything else about m. The primitive derives functional decryption keys sk_{f_i} – associated to specific functionalities f_i – from a master secret key msk; these are delivered to the appropriate recipients. A single ciphertext c encrypting plaintext m is made available, from which a user possessing sk_{f_i} can recover $f_i(m)$ by decrypting c with sk_{f_i}.

Applied to our concrete example, the hospital can encrypt its database using a functional encryption scheme, and make the resulting ciphertext publicly available. Then upon request of a group of researchers, wishing to compute a function f applied to the database, the hospital computes a functional decryption key sk_f associated to f which it sends back to them. This decryption key allows them to compute the required function of the database, and in fact that of any updated database, encrypted under the same public key.

Despite the huge attention it has received from scientific research, devising efficient functional encryption schemes which support any function evaluation, and attaining a satisfying level of security, is still an open problem. All constructions achieving such a functionality are far from practical, and either bound the number of decryption keys the adversary can request (e.g. [SS10]), or rely on non-standard, ill-understood cryptographic assumptions (e.g., [GGHZ16]). Hence researchers started focussing on functional encryption restricted to the computation of specific classes of functions, in the hope that such primitives could be implemented efficiently under well understood cryptographic assumptions, while being efficient enough to benefit concrete practical applications.

One notable example is the study of inner-product functional encryption, as first formalised by Abdalla et al. in [ABDP15], which restricts the computed functionality to the dot product of two vectors (one resulting from a decryption key, the other from a message). Such a restriction not only develops our understanding of functional encryption, but also benefits diverse practical applications, from its direct use allowing to perform linear operations over encrypted data, to the construction of other more complex cryptosystems (*cf.* e.g. [ALS16, ABP$^+$17, KY19]). In this thesis we present generic constructions for inner-product functional encryption which can be instantiated from a range of algorithmic assumptions, and which are efficient enough to be used in large scale modern information systems.

Secrecy of secret keys. Let us now consider a somewhat orthogonal issue. Whether one considers symmetric encryption, public key encryption, or in fact any cryptosystem aiming at preserving information security, if the secrecy of the private key is compromised, then so is the security of the whole application.

The requirement of the key being secret brings several problems. First of all, storing a secret key in one location, be it one's personal computer, a server or a database, reduces the security of the cryptosystem to the difficulty of breaking into that device. This may be (and often is) considerably easier for an attacker than breaking the actual cryptosystem. In addition, not having a backup of the key introduces the risk of key loss if a software or hardware failure were to occur. On the other hand, storing the key on multiple devices just renders an attacker's task easier, as it multiplies the targets it can choose from, thereby reducing security to that of the least secure of these devices.

One tangible example where key theft can have devastating consequences is in the context of cryptocurrencies. A common approach to validate cryptocurrency transactions is to have the spender authenticate its transaction. The spender does so by signing the transaction with its *secret* signing key for a digital signature scheme. On account of this, a security break where one's signing key is stolen can result in concrete financial losses.

Secret sharing schemes were introduced in 1979 by Blakley [Bla79] and Shamir [Sha79] to solve the problems above. A secret sharing scheme makes it possible to share a secret among a group of people in such a way that only well-defined combinations of people can jointly recover the secret, while no other coalition can obtain any information about the secret. A (t, n)-threshold scheme [Sha79] is a particular case of a secret sharing scheme in which the secret is shared between n parties in such a way that any set of more than t participants can recover the secret, while any set of t or less participants gains no additional information. One can further strengthen the security of these two concepts by using the broader notion of threshold cryptography, which was introduced by the works of Boyd [Boy86], Desmedt [Des88], and Desmedt and Frankel [DF90]. Similarly to the aforementioned threshold secret sharing, (t, n)-threshold cryptography allows n users to share a common key in such a way that any subset of $t+1$ parties can use this key to decrypt or sign, while any coalition of less than t can do nothing. The key feature of this paradigm is that it allows to use the shared key without ever explicitly reconstructing it in the clear. This means a subset of t parties have to actively participate in the protocol whenever the secret key is used. In particular this means that an adversary must simultaneously corrupt at least $t+1$ parties (or equivalently break into $t+1$ devices) in order to perform sensitive operations. Such distributed protocols fall under the scope of *secure multi-party computation*: this subfield of cryptography, which dates back to papers by Yao [Yao82] and Goldreich et al. [GMW87], aims at creating methods for parties to jointly compute a function over their inputs while keeping those inputs private.

We devise threshold variants of the signature scheme used to validate Bitcoin transactions: the ellipic curve digital signature algorithm (EC-DSA). As we shall see, while for many other signature schemes fast threshold variants have been known for decades (e.g. RSA signing [GJKR96a] and Schnorr signatures [Sch91, SS01]) constructing efficient threshold variants of EC-DSA has proven to be much harder.

Building advanced cryptographic systems may at first sight seem a daunting task. Indeed the goals of such systems are much more complex – be it in terms of functionality or of security – than those of traditional cryptography. Hence a natural approach to devising such systems is to start from elementary building blocks. If these elementary objects are somewhat malleable, in that one can combine and assemble them, they give us the means to realise complex functionalities. If one can further write out a clear interface defining the security properties provided by theses building blocks, proving the security of the resulting constructions is highly simplified. Adding on the extra bonus of genericity, one can instantiate these constructions from a wide range of cryptographic assumptions, which avoids putting all our eggs in one basket. Indeed if we concentrate our efforts and resources on building schemes from one cryptographic assumption, in the event the underlying problem turns out to be less hard than expected, one could lose everything. Finally, if these constructions are carefully crafted, and their security analysis is precise, choosing the appropriate mathematical tools to instantiate them results in extremely flexible and efficient advanced cryptosystems.

This modular, generic and precise approach has been that of my thesis.

1.2 Projective Hash Functions

The main generic and elementary tool which we use as building block is a *projective hash function*. The concept of a projective hash function was formalised by Cramer and Shoup some 20 years ago [CS02]. They were initially introduced to generically build encryption schemes secure against adversaries which not only eavesdrop upon encrypted communications, but also actively attempt to gain information by running the cryptosystem in unprescribed conditions (e.g. requesting the decryption of malformed ciphertexts). Such unexpected behaviour could for instance cause the system to leak information on secret keys. In the context of public key encryption, to deal with eavesdropping (also referred to as *passive*) adversaries, one simply needs to ensure confidentiality of encrypted messages. To further protect against *malicious* (also referred to as *active*) adversaries, one must also ensure ciphertext integrity.

In view of adopting a modular approach in our design of advanced cryptosystems we build projective hash functions with *homomorphic* properties. A cryptosystem is said to be homomorphic if one can publicly manipulate secret data. For instance, an encryption scheme is linearly homomorphic if, knowing ciphertexts c_1 and c_2 which encrypt plaintexts m_1 and m_2, it is possible to produce a new ciphertext c that will decrypt to the sum $m_1 + m_2$. Paradoxically, though encryption schemes possessing such homomorphic properties are extremely useful for the design of more complex advanced cryptosystems, these can not attain the maximal level of security one could hope to get for an encryption scheme, i.e. security against active adversaries.

Returning to our projective hash functions, we also define new properties for these which encapsulate the essential features needed for both the correctness and security of our more complex constructions. Precisely, from the rich toolbox provided by these projective hash functions, we devise linearly homomorphic encryption schemes, functional encryption allowing the evaluation of linear functions and threshold EC-DSA protocols. Using similar techniques to [CS02], we are able to secure the latter two constructions against malicious adversaries (which deviate from the protocol).

1.3 Linearly Homomorphic Encryption and the CL Framework

As mentioned above, from projective hash functions with homomorphic properties, one can design linearly homomorphic encryption schemes. In fact, the path which led to the writing of this thesis did not start with projective hash functions, but rather from the observation that one can devise provably secure advanced cryptosystems, building upon the malleability of linearly homomorphic encryption.

The first homomorphic encryption scheme (which was also the first probabilistic encryption scheme) was put forth by Goldwasser and Micali in [GM84], while one of the most accomplished such schemes, standardised in ISO/IEC-18033-6, was designed by Paillier [Pai99].

This thesis grounds itself on a much more recent linearly homomorphic encryption scheme possessing interesting and novel properties. Specifically, we build upon the framework introduced by Castagnos and Laguillaumie at CT-RSA 2015 [CL15]. This framework (the CL framework) allowed them to devise a linearly homomorphic encryption scheme which benefits of the uncommon property of having a plaintext space of prime order q, where q can be chosen (with some restrictions) independently of the security parameter. Furthermore, their scheme is the first practical linearly homomorphic encryption scheme whose security relies solely on a discrete logarithm type assumption.

Building upon the CL framework, we formalise new hardness assumptions from which we build invaluable tools; in particular, we devise projective hash functions with homomorphic properties. These abstract away the framework's properties, thus allowing for more genericity in our subsequent work.

We note that, when instantiating our upcomming generic constructions, the most efficient schemes either result from the CL framework, or from that in which Paillier designed his cryptosystem [Pai99]. As we both build upon, and compare our work to Paillier-based constructions, we will often refer to his scheme. Moreover one may observe many similarities between the structure of both frameworks, as they both consist of a group G (in which some computational problem is hard) containing a subgroup F, generated by f, in which computing discrete logarithms can be done efficiently. In both cryptosystems one encodes plaintext messages in the exponent of f, and masks this encoding with some group element in G. This allows to perform linearly homomorphic operations over encrypted data (since $f^{m_1} f^{m_2} = f^{m_1+m_2}$), while allowing for efficient decryption, since one can efficiently recover m given f^m.

1.4 Instantiating the CL Framework from Class Groups

The framework introduced by Castagnos and Laguillaumie has a key feature which we will exploit throughout this thesis. It relies on the existence of a group G in which some algorithmic problem is assumed to be hard (the hardness of which underlies security of resulting cryptographic constructions), together with a *prime order* subgroup F of G in which computing discrete logarithms is easy.

In order to realise this uncommon property, they use the particular algebraic structure of class groups of imaginary quadratic fields, which possess some specificities that seem hard to find in other groups.

Imaginary quadratic fields were proposed as a setting for public-key cryptosystems in the late 1980s by Buchmann and Williams [BW88]. They proposed an adaptation of the Diffie-Hellman key exchange in imaginary quadratic fields and briefly described an adaptation of the Elgamal cryptosystem in the same setting. Many cryptosystems relying on this algebraic object have since been designed, for which efficient implementations have also been discussed in e.g. [JW09].

Notably, the interest in class group cryptography declined after critical attacks by Castagnos et al. [CL09, CJLN09] on the NICE cryptosystem [HPT99, PT00] and its real variant [JSW08]. However these attacks do not apply to the CL framework since they recover the secret key by exposing the factorisation of the discriminant of the field, whereas in the CL framework the factorisation of the discriminant is public.

This being said, class group cryptography is seeing renewed interest as it allows versatile and efficient solutions such as accumulators without trusted setup [Lip12], encryption switching protocols [CIL17], verifiable delay functions [BBBF18, Wes18], succinct non-interactive zero-knowledge arguments of knowledge without trusted setup [BFS20], and, as we shall see throughout this thesis, inner product functional encryption and threshold EC-DSA signatures.

Regarding underlying assumptions, the security of cryptosystems arising from the CL framework is related to the hardness of computing class numbers. The current best known algorithms to solve this problem (cf. [BJS10]) have a higher asymptotic complexity than those solving the factorisation problem (or the discrete logarithm problem in finite fields); this allows to use shorter keys for the CL cryptosystem than for linearly homomorphic encryption schemes relying on the hardness of factorisation (such as Paillier's encryption scheme). In addition, arithmetic operations in class groups are quasi linear using fast arithmetic, both these facts contribute to the efficiency of protocols resulting from this framework.

1.5 Contributions

We note that though most of the presented results have been published or are currently under review, for clarity of exposition we have restructured our results to show them in a more modular and logical order, rather than following the chronological order of our publications, or grouping chapters per publication. The goal is of course to benefit ease of readability.

The results discussed below have been mainly taken from a current submission (co-authored with Guilhem Castagnos and Fabien Laguillaumie), and the following papers: [CLT18a] presented at Asiacrypt 2018 (co-authored with Guilhem Castagnos and Fabien Laguillaumie), [CCL+19] presented at Crypto 2019 and [CCL+20] presented at PKC 2020 (both co-authored with Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie and Federico Savasta).

1.5.1 Projective Hash Functions from the CL Framework

In this thesis we formalise new cryptographic assumptions in the CL framework. We provide arguments backing their hardness when these are instantiated from class groups of an imaginary quadratic field. We then formally define projective hash functions, and the various properties (some of which are new to our work) required for both correctness and security of our constructions. We illustrate all of our definitions and properties with three running examples. The first arises from the traditional decision Diffie-Hellman assumption in finite fields, thereby providing an instantiation with which most readers will be familiar. The other two running examples arise from the aforementioned assumptions defined in the CL framework, called respectively HSM-CL and DDH-f.

As projective hash functions are used as building blocks for our subsequent constructions, all our constructions can be instantiated by assumptions in the CL framework, and hence from class group cryptography. We note that all our definitions cater for the fact the groups we work in may be of unknown order. In fact this is a feature which is inherent to the CL framework. As a first and simple application of these projective hash functions, we present linearly homomorphic encryption schemes, which can be seen as a slight adaptation of the Cramer-Shoup generic construction [CS02].

1.5.2 Zero-knowledge Proofs and Arguments for the CL Framework

For our distributed signature protocols, as multiple parties are jointly performing computations, we need to enforce parties behave correctly and follow protocol. Of course, when proving they behaved correctly, parties should not unwillingly leak information on their private inputs. A zero-knowledge proof is an interactive protocol between a prover and a verifier, which aims at demonstrating that some statement is true without revealing anything else but the truth of this statement. While a zero-knowledge proof convinces the verifier that *statistically* this truth must hold, zero-knowledge arguments convince the verifier under *computational* assumptions.

We provide zero-knowledge proofs and arguments for the CL framework, e.g. proving that a ciphertext for one of the aforementioned encryption schemes is honestly computed.

Our setting contrasts to groups of known order where one can efficiently recognise group elements, since in the CL framework, we deal with a cyclic subgroup $G^q := \langle g_q \rangle$ of some larger group $\widehat{G}$, where g_q is of unknown order, and elements of G^q are not efficiently recognisable. One can only efficiently check that elements live in $\widehat{G}$. It is crucial to take this into account in all of our protocols for security to hold against *malicious* adversaries. Since our encryption schemes follow an Elgamal like structure, we have at our disposal a whole arsenal of zero-knowledge proofs and arguments for Elgamal ciphertexts and for proving knowledge of discrete logarithms in groups of known order (e.g., [Sch90, CP93]) which we can build upon. Further inspired by Dåmgard and Fujisaki's work on arguments of knowledge [DF02], and Girault, Poupard and Stern's work on proofs which statistically hide secret inputs [GPS06], both of which also consider groups of unknown order, we devise various techniques to overcome the aforementioned complications, with varying trade-offs between security and efficiency.

1.5.3 Tighter Security and Improved Efficiency for Functional Encryption

We build upon the ideas underlying the generic construction of [CS02], allowing to devise public key encryption schemes which are secure against active adversaries from projective hash functions satisfying a number properties. We adapt these properties to the context of inner product functional encryption, where messages are vectors, and adversaries have access to additional information on the scheme's secret parameters.

Precisely, we define new properties for projective hash functions which capture the essential requirements to build inner-product functional encryption schemes. To isolate these specific properties, we break down the proof techniques used by Agrawal et al in [ALS16] in which they devise inner-product functional encryption schemes secure against passive adversaries from specific assumptions. This decomposition allows us to identify the essential steps in their proofs, which will benefit our modular and generic approach while preserving efficiency.

We then provide generic constructions for inner-product functional encryption from projective hash functions possessing such properties. Our first construction is proven secure against passive adversaries, i.e. when ran in expected conditions, the scheme leaks no further information than that intended.

When we instantiate this construction from the decision Diffie Hellman assumption, or from the decisional composite residuosity assumption, we retrieve the constructions of [ALS16, ABDP16], with the same security bound; and when instantiated from our assumptions in the CL framework, we obtain new constructions. Instantiations from DCR and from the CL framework yield the most efficient such schemes to date, and compute inner products in $\mathbf{Z}$.

Digressing slightly from the generic construction, we also to build functional encryption schemes computing inner products *modulo a prime* from the CL framework, and obtain the most efficient such schemes to date; we compare the speed as well as the sizes of ciphertexts and secret keys of our most efficient scheme to those of [ALS16] so as to illustrate the concrete improvements we achieve.

This first generic construction secure against passive adversaries is in some sense a stepping stone to build inner-product functional encryption secure against active adversaries. Indeed we next extend the aforementioned generic construction for security to hold against active adversaries, which may deviate arbitrarily from the protocol. The construction can be instantiated with all our running examples. We build upon the construction of Benhamouda et al. [BBL17], which attained the same security goals and functionalities. However inspired by the proof techniques of [ALS16], we hugely improve the security reduction. In particular this justifies using much shorter secret keys and ciphertexts, while maintaining an equivalent level of security. Our work is the first to demonstrates that inner product functional encryption which achieves security against active adversaries is usable in practice. Indeed, instantiations of our generic construction yield schemes which are efficient enough to be used in large-scale modern information systems. To illustrate this we provide a detailed theoretical comparison of the efficiency of our inner-product functional encryption schemes secure against active adversaries (instantiated from various assumptions) to those of [BBL17].

1.5.4 Distributing the Elliptic Curve Digital Signature Algorithm

In order to devise threshold variants of the EC-DSA signature algorithm, various solutions have been put forth which rely on the linearly homomorphic properties of the Paillier cryptosystem. Some of these are in the two party setting (i.e. $t = 1$ and $n = 2$), this was first addressed by Mackenzie and Reiter in [MR01], and then much more efficiently by Lindell in [Lin17a]. While recent works also consider the full threshold setting (any $t < n$), starting with Gennaro, Goldfeder and Narayanan's protocol in [GGN16].

We demonstrate that despite the widespread use of Paillier's encryption scheme to build threshold EC-DSA, this encryption scheme is in fact not adapted for the task. Indeed, for reasons which will be discussed later in this thesis, this choice entails an overhead in communication cost, and, in some cases, the introduction of a non standard interactive assumption on the Paillier cryptosystem so as to prove security. Conversely, if one uses our linearly homomorphic encryption schemes resulting from projective hash functions in the CL framework, we remove the need for a number of zero-knowledge proofs and need not resort to any interactive assumptions.

We build upon Lindell's ideas in [Lin17a] and devise the first generic two party EC-DSA protocol from projective hash functions with homomorphic properties. When instantiated from our HSM-CL based projective hash function in the CL framework, this construction yields an efficient protocol (especially in terms of bandwidth) with a tight security proof and without any interactive assumptions. We then merge the ideas underlying our two party protocol and the construction proposed by Gennaro et al. in [GG18] to obtain a full threshold EC-DSA protocol. Both constructions make use of our zero-knowledge proofs and arguments in the CL framework discussed earlier, so as to ensure that parties behave correctly.

We compare the speed and communication costs of our protocols to best performing preexisting protocols using similar construction techniques and which achieve the same functionality. Our comparisons show that for all considered security levels our signing protocols significantly reduce the bandwidth consumption. In terms of timings, though for standard levels of security our signing protocols are slightly slower, for higher levels of security (192-bit security and beyond) the trend is inverted.

1.6 Road Map

The rest of this thesis is organised as follows. In Chapter 2, we introduce necessary notations and preliminaries to understand this thesis. This includes basic technical background on ideal class groups of orders of an imaginary quadratic field.

In Chapter 3, we recall the CL framework and explain how it can be instantiated from class groups. We then enrich this framework by formalising new hardness assumptions from which we build families of projective hash functions. These in turn give rise to three linearly homomorphic encryption schemes. Still in view of enhancing the CL framework, we devise a range of zero-knowledge proofs and arguments tailored to the framework and to the aforementioned encryption schemes.

In Chapter 4 we define a number of new properties required of projective hash functions in order to build inner product functional encryption schemes which are secure against both passive and active adversaries. Then we present our generic constructions and proofs of security for inner-product functional encryption from projective hash functions. All new definitions and constructions are illustrated with running examples from the projective hash functions of Chapter 3.

In Chapter 5 we present our generic construction for two party EC-DSA from projective hash functions and our full threshold EC-DSA from the CL framework.

Finally Chapter 6 shortly concludes the thesis and raises some open questions.

Chapter 2

Preliminaries

In this chapter, we introduce the notations and basic notions that will be used throughout this thesis. In Sections 2.2 and 2.3 we introduce the main notions related to provable security, the different types of adversaries we consider, and, at a high level, the models in which we prove security. We do not provide formal definitions regarding these notions, but only enough intuition and information for the reader to understand our discussions and motivations in upcoming chapters. If formal definitions on these topics are required later in the thesis, they will be provided in the relevant chapter.

In Section 2.4 we recall definitions for discrete logarithm type assumptions and the decisional composite residuosity assumption. These are standard computational assumptions which will be used throughout our work. In Section 2.5 we give formal definitions for basic primitives which we either build, or build upon. Of particular interest to this thesis is the definition of linearly homomorphic public key encryption, as in Section 3.5 we build linearly homomorphic encryption schemes from the CL framework (defined in Section 3.1). We also recall Paillier's linearly homomoprhic cryptosystem [Pai99], as it will often be referred to throughout this thesis.

As our instantiation for the CL framework, detailed in Section 3.1.2, arises from ideal class groups of orders of an imaginary quadratic field, in Section 2.6 we give some background on these mathematical objects. Though this background is not exhaustive, we provide pointers for further reading, and present sufficient material to back our instantiation.

As we will often be working in groups of unknown order, in Section 2.7 we explain how to instantiate distributions, from which exponents will be sampled, so as to induce distributions close to uniform in these groups. We provide useful properties regarding such distributions, and – since we often instantiate these distributions with discrete Gaussians – we give minimal yet essential background on discrete Gaussian distributions.

Finally, in Section 2.8, we define zero knowledge proofs and arguments of knowledge as we will devise such protocols for the CL framework in Sections 3.6 and 3.7.

2.1 Notations

We denote sets by upper-case letters, matrices by bold upper-case letters, vectors by bold lower-case letters (which are by default column vectors), and for $\boldsymbol{a} \in A^\ell$, $\boldsymbol{a}^T = (a_1, \dots, a_\ell)$. For an integer x, we denote its size by $|x|$, and by $[x]$ the set of integers $\{1, \dots, x\}$. The coordinate-wise product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in A^ℓ is denoted:

$$\boldsymbol{x} \odot \boldsymbol{y} := (x_1 y_1, \dots, x_\ell y_\ell)^T \in A^\ell.$$

Given a ring $\mathscr{R}$, the dot product of $\boldsymbol{x} \in \mathscr{R}^\ell$ and $\boldsymbol{y} \in \mathscr{R}^\ell$ is denoted:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \sum_{i=1}^{\ell} x_i y_i \in \mathscr{R}.$$

Throughout this thesis we will abuse terminology and refer to the dot product as the *inner product*[1]. If $f : A \mapsto B$ is a function defined over A with co-domain B, and $\boldsymbol{a} \in A^{\ell}$, we denote $f(\boldsymbol{a}) \in B^{\ell}$ the vector satisfying $f(\boldsymbol{a})^T := (f(a_1), \ldots, f(a_{\ell}))$. For an element g of a group G, $\langle g \rangle$ is the subgroup generated by g. If $\mathcal{R}$ is either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q, for $\boldsymbol{m} \in \mathcal{R}^{\ell}$ we denote $\boldsymbol{m}^{\perp}$ the subset of $\mathcal{R}^{\ell}$ defined as $\boldsymbol{m}^{\perp} := \{\boldsymbol{k} \in \mathcal{R}^{\ell} \mid \langle \boldsymbol{m}, \boldsymbol{k} \rangle = 0\}$.

We denote $\mathcal{U}(B)$ the uniform distribution over the finite set B. For a distribution $\mathcal{D}$, we write $d \hookleftarrow \mathcal{D}$ to refer to d being sampled from $\mathcal{D}$ and $b \hookleftarrow B$ if b is sampled uniformly in the set B.

For any function in parameter λ, we denote by $f(\lambda) = \mathsf{poly}(\lambda)$ the fact f is a polynomial. We denote by $f(\lambda) = \mathsf{negl}(\lambda)$, if for all polynomial P, f is asymptotically dominated by $1/P$. An algorithm is said to be efficient or polynomial time (PT) if, on input a string of length λ, the number of steps of the algorithm is upper bounded by $\mathsf{poly}(\lambda)$. If the algorithm is further probabilistic, it is said to be probabilistic polynomial time (PPT). We say that a set (or a group) E is efficiently recognisable if its elements are uniquely encoded as bit strings of length bounded by $\mathsf{poly}(\lambda)$ (λ is the security parameter), and there exists a PT algorithm that determines if a bit string is a valid encoding of an element of E.

The L-notation, often called sub-exponential function, is useful to compare the asymptotic complexity of algorithms solving problems underlying cryptographic schemes. For a positive integer x, the L-notation is defined as:

$$L_x[\alpha, c] = \exp(c(\log(x))^{\alpha}(\log(\log(x)))^{1-\alpha}),$$

with $0 \leqslant \alpha \leqslant 1$ and $c > 0$. The parameter α measures the gap between polynomial time ($\alpha = 0$) and exponential time ($\alpha = 1$); we may at times omit c, which is an explicitly computable constant, and simply write $L_x[\alpha]$.

Let X and Y be random variables taking values in a finite set Ω. The statistical distance between X and Y is defined as:

$$\Delta(X, Y) \overset{\text{def}}{=} \frac{1}{2} \cdot \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]| = \max_{\Omega' \subset \Omega} |\Pr[X \in \Omega'] - \Pr[Y \in \Omega']|.$$

We shall say that X and Y are δ-close if $\Delta(X, Y) \leqslant \delta$.

Statistical and computational indistinguishability. Let $\{A_{\lambda}\}_{\lambda \in \mathbf{N}}$ and $\{B_{\lambda}\}_{\lambda \in \mathbf{N}}$ be ensembles of distributions over some set Ω, indexed by the security parameter λ (in the sequel the security parameter is often omitted for the sake of simplicity). Let $\mathcal{A}$ be an algorithm, called an adversary. The *advantage* of $\mathcal{A}$ in distinguishing $\{A_{\lambda}\}_{\lambda \in \mathbf{N}}$ and $\{B_{\lambda}\}_{\lambda \in \mathbf{N}}$ is defined as:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr_{x \hookleftarrow A_{\lambda}}[\mathcal{A}(x) = 1] - \Pr_{x \hookleftarrow B_{\lambda}}[\mathcal{A}(x) = 1] \right|$$

The distributions A and B are *computationally indistinguishable* if for any (probabilistic) polynomial time $\mathcal{A}$, it holds that $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$. They are *statistically indistinguishable* if this is true for any (not necessarily polynomial time) $\mathcal{A}$.

2.2 Secure Multi-Party Computation (MPC)

In Chapter 5 we build distributed signatures, which allow multiple parties to collaboratively compute digital signatures. These are a case of secure multi-party computation (MPC).

[1] This is due to the fact inner product functional encryption schemes (defined in Section 4.1), despite their misleading name, compute the dot product of two vectors.

Notation. In an interactive protocol IP, between parties $P_1, \ldots, P_n$ for some integer $n > 1$, we denote:

$$\mathsf{IP}\langle x_1; \ldots; x_n \rangle \rightarrow \langle y_1; \ldots; y_n \rangle$$

the joint execution of parties $\{P_i\}_{i \in [n]}$ in the protocol, with respective inputs x_i, and where P_i's private output at the end of the execution is y_i. If all parties use the same input x (resp. receive the same output y) we write $\mathsf{IP}\langle x \rangle \rightarrow \langle y_1; \ldots; y_n \rangle$ (resp. $\mathsf{IP}\langle x_1; \ldots; x_n \rangle \rightarrow \langle y \rangle$).

Multi-party protocol [GMW87]. Let $\{0,1\}^*$ denote the set of arbitrary length (but finite) bit strings. Consider a function $F : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$, and n parties $P_1, \ldots, P_n$ with respective secret inputs $x_1, \ldots, x_n$. Let $F(x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$. The goal of each P_i is to evaluate F on the inputs $x_1, \ldots, x_n$ receiving the i-th output without revealing its own input to the other parties. A multi-party protocol Π executed by $P_1, \ldots, P_n$ securely implements F if the following conditions hold:

Completeness: if all $P_1, \ldots, P_n$ honestly follow the protocol then

$$\Pi\langle x_1; \ldots; x_n \rangle \rightarrow \langle y_1; \ldots; y_n \rangle.$$

Privacy: any party behaving dishonestly in the protocol does not gain any information about the private inputs/outputs of the other parties (except what can be inferred by the output of the protocol and its own private input).

One may also require that other security properties be guaranteed, namely:

Independence of inputs: parties cannot choose their inputs as a function of other parties' inputs.

Fairness: all parties learn the output or no one does.

Robustness: output delivery is guaranteed for honest parties.

Secret Sharing

For many secure multi-party protocols, one needs to distribute a secret amongst the parties, each of whom is granted a share of the secret. The secret can be reconstructed only when a sufficient number of shares are combined together; individual shares should leak no information on the reconstructed secret. We here define the notion of threshold secret sharing, which allows to perform the aforementioned task, and verifiable secret sharing, which additionally allows to share the secret in a verifiable way. We will use Feldman's verifiable secret sharing for one of our distributed signature protocols of Chapter 5.

Threshold secret sharing. A (t, n) threshold secret sharing scheme allows to divide a secret s into shares $s_1, \ldots, s_n$, amongst a group of n parties, in such a way that knowledge of any $t + 1$ or more shares allows to compute s; whereas knowledge of any t or less shares reveals no information about s.

Feldman verifiable secret sharing. A verifiable secret sharing (VSS) protocol allows to share a secret between n parties $P_1, \ldots, P_n$ in a verifiable way. Feldmann's VSS [Fel87] relies on Shamir's secret sharing scheme [Sha79], but gives additional information allowing to check the sharing is done correctly.

Let G be a group of prime order q, g a generator of G, and suppose that one of the parties, that we call P, wants to share a secret $s \in \mathbf{Z}/q\mathbf{Z}$ with the others. To share the secret, P does the following steps:

1. P generates a random polynomial $Q \in \mathbf{Z}/q\mathbf{Z}[x]$ of degree t and with s as free term. We denote:
$$Q(x) = a_t x^t + a_{t-1}x^{t-1} + \ldots + a_2x^2 + a_1x + s \mod q,$$
where $s = Q(0) \bmod q$. The shares of s are $s_i = Q(i) \mod q$, for $1 \leqslant i \leqslant n$.

2. P sends s_i to P_i, for $1 \leqslant i \leqslant n$.

3. P publishes auxiliary information $\{v_i = g^{a_i} \in G\}_{i\in[t]}$ and $v_0 = g^s \in G$ allowing other players to check the shares are consistent and define a unique secret.

Each P_i, for $1 \leqslant i \leqslant n$, can check its share is consistent by verifying if the following condition holds:
$$g^{s_i} = \prod_{j=0}^{t} v_j^{i^j} \in G.$$

If one of the checks fails, then the protocol terminates. Furthermore, the only information that this VSS leaks about the secret s is $v_0 = g^s$.

2.3 Provable Security

We here provide a brief introduction to the setting in which we do our work. The goal of provable security is to provide a reduction from the security of the cryptosystem to some well-studied problem (the underlying problem). Alone the reduction says nothing about the security of the cryptosystem. But if one assumes there is no efficient algorithm solving the underlying problem with significant probability, the reduction guarantees some minimal cost for breaking the cryptosystem. In this section we explain what it means for an adversary to *break* a cryptosystem. We describe the considered adversaries, what kind of attacks we allow, and the different models in which we obtain our security proofs.

2.3.1 Adversary Model

Computational power. We distinguish between information and knowledge (following Goldreich [Gol01]). One has information about a value if it can be computed given unbounded computational resources. One has knowledge about a value if it can be computed given bounded computational resources. A computationally (un)bounded adversary is an adversary with (un)bounded computational resources. Unbounded adversaries can obtain any information fixed by the cryptosystem.

Active vs. Passive Adversaries. *Passive* adversaries attempt to obtain confidential information while running the cryptographic protocol as expected. This corresponds, for example, to a situation where the protocol is executed by honest parties, and the adversary watches their communications, but does not interfere. Security against passive adversaries does not however capture the scenario where an adversary coerces parties into running the protocol in unprescribed conditions. Such executions could leak further information on the secret parameters of the scheme, thereby compromising security. Such adversaries – which deviate from the protocol arbitrarily – are said to be *active*. Security against active adversaries is obviously the most desirable security that can be achieved, but it is also harder to realise.

In the context of public key encryption. Security against passive adversaries for public key encryption is called security against *chosen plaintext attacks*, formally defined in Section 2.5.1. This is because in a public key cryptosystem, the adversary knows the public key by definition. Consequently, any adversary can encrypt messages of its choice with the given public key. On the other hand, an active adversary may somehow convince the decryptor to run the decryption protocol on values which were not output by the encryption protocol, in the hope that, when ran on unexpected values, the algorithm leaks further information than the schemes' public parameters. For a public key encryption scheme, this translates as giving the adversary the output of the decryption algorithm ran with input ciphertexts of its choice (potentially malformed). The adversary is also given a *challenge* ciphertext, for which it can not request the decryption, and must determine if it encrypts a fixed value, or random garbage. This explains why, for public key encryption schemes, security against active adversaries is referred to as security against *chosen ciphertext attacks*.

In the context of multi-party computation [GMW87]. Secure multi-party computation aims at protecting the privacy of data even when a subset of the parties has been corrupted by an adversary. Here passive corruption refers to adversaries that will only get access to the view of the corrupted parties. This corresponds to a situation where the computation has been performed by honest parties, and the transcript of the computation (which has been recorded and stored by the parties) is later leaked through an adversarial breakthrough (say, a hack of a party's computer). Security with respect to passive corruption (also known as security against honest-but-curious adversaries, or semi-honest security) ensures that this transcript does not reveal unintended confidential information. On the other hand an active adversary is given full control of the parties it corrupts, and can arbitrarily modify their behaviour. This corresponds to a situation where some of the parties actively cheat during the protocol, in an attempt to gain information, or to alter the outcome of the protocol. This model is also known as security against malicious adversaries, or malicious security. Malicious adversaries for multi-party computation come in various flavours:

- **Adaptive/Static:** Adaptive adversaries decide which parties they corrupt throughout the protocol execution in an adaptive way (i.e. depending upon the transcript and the state of the parties corrupted so far). Static adversaries declare which parties are corrupted before the protocol execution starts.
- **Sequential/Concurrent Composability [GO94]:** One can require that the security of the protocol holds even when a dishonest party executes several instances of the same protocol sequentially (meaning that each execution concludes before the next execution begins). In this case we say that the protocol securely realises a functionality under sequential composition. Similarly, if security of the protocol holds even when a dishonest party executes several instances of the same protocol (resp. different protocols) concurrently, we say the protocol securely realises a functionality under concurrent (resp. general concurrent) composition.

2.3.2 Security Definitions

There are two main ways of defining security: the game-based definition and the simulation-based definition. The latter was born with the notion of semantic security for public key encryption [GM84], and is also sometimes called the real/ideal world paradigm.

Game-based definition

The security of a cryptographic algorithm is phrased as a game (or experiment) played between an adversary $\mathscr{A}$ and a hypothetical entity called the challenger $\mathcal{C}$. Both $\mathscr{A}$ and $\mathcal{C}$ are probabilistic processes that communicate with each other, and so the game is modelled as a probability space. Typically, the definition of security is tied to some particular event S. In this context, security means that for any PT adversary, the probability that event S occurs is negligibly close to some specified target value (either 0, 1/2, or the probability of some event in another game in which the same adversary is interacting with a different challenger).

Simulation-based definition

In the simulation based model one imagines what properties one would have in an ideal world: a protocol in the ideal world is called an *ideal functionality*, and is secure by definition. Then if a real world (constructed) protocol Π has similar properties to the ideal functionality $\mathcal{F}_\Pi$ then it is considered secure. In this case one says that Π securely implements $\mathcal{F}_\Pi$.

Security proofs for such definitions work by constructing a simulator that resides in the ideal world, which generates a view for the adversary in the real world. This simulated view must be computationally indistinguishable from the adversary's real view. The simulation-based proof technique is particularly useful for arguing security of multi-party computation protocols, as explained in Lindell's tutorial [Lin17b].

$\mathcal{F}$-hybrid model. We note that, in the simulation based model, proving protocols secure under composition[2] (*cf.* Section 2.3.1) is particularly useful for writing proofs of security. Specifically, assume one has proven that a protocol π securely implements some ideal functionality $\mathcal{F}_\pi$, and that security holds under the considered notion of composition. Then when building a more complex protocol Π which needs to use this functionality as a subroutine, one can prove the security of Π using as subroutine the ideal functionality $\mathcal{F}_\pi$, while in practice Π would be implemented using π. This allows for much simpler and modular proofs.

This way of designing protocols using calls to ideal functionalities (as sub-protocols), and analysing security in this partially ideal setting, is called the *hybrid* model. Specifically in the $\mathcal{F}$-hybrid model, parties can communicate as usual and in addition have ideal access to copies of the functionality $\mathcal{F}$. According to the considered composability model, parties may access copies of $\mathcal{F}$ sequentially, in parallel, or arbitrarily. This makes protocol design and analysis significantly more simple. Thus, composition theorems are one of the most important tools used to write simulation-based proofs of security.

Universal composability. A particularly interesting framework illustrating the above is that of *universal composability* (UC), introduced by Canetti in [Can01]. Canetti demonstrates that if a protocol Π securely implements an ideal functionality $\mathcal{F}_\Pi$ in the UC framework, then Π securely realises $\mathcal{F}_\Pi$ under general concurrent composition (*cf.* Section 2.3.1). In particular this means the protocol is secure in realistic settings where a protocol session may run concurrently with other protocols, thereby guaranteeing security in arbitrary protocol environments. This implies that a secure protocol for some task remains secure even if it is running in an arbitrary and unknown multi-party, multi-execution environment.

In particular, the following *universal composition theorem* is proven in [Can01]. Consider a protocol $\Pi_\mathcal{F}$ that operates in the $\mathcal{F}$-hybrid model, i.e. parties communicate as usual and also have ideal access to an unbounded number of copies of the functionality $\mathcal{F}$. Let π be a protocol that securely realises $\mathcal{F}$ in the UC framework, and let Π_π be identical to $\Pi_\mathcal{F}$, only the interaction with each copy of $\mathcal{F}$ is replaced with an interaction with a separate instance of π. Then Π_π and $\Pi_\mathcal{F}$ have essentially the same input/output behaviour. In particular, if $\Pi_\mathcal{F}$ securely realises some functionality $\tilde{\mathcal{F}}$ in the $\mathcal{F}$-hybrid model then Π_π securely realises $\tilde{\mathcal{F}}$ in the standard model (i.e., without access to any functionality).

[2] Be it sequential, concurrent, or general concurrent composability.

In the context of public key encryption

For traditional public key encryption the game based and simulation based security definitions were shown to be equivalent [GM84] in the sense that a public key encryption scheme meets the indistinguishability notion of security in the game based approach if and only if it is secure in the real/ideal paradigm, i.e. semantically secure (the definition of semantic security for public key encryption compares what can be learned by an adversary who receives a real ciphertext to what can be learned by an adversary who receives nothing).

We note that this is not the case however, for most cryptographic primitives. In particular, we shall see in Section 4.6.2 that both definitions are not equivalent for functional encryption.

2.4 Common Problems

In this section, we recall several problems that are common in cryptography and that are generally assumed to be hard. These will appear throughout this thesis, along with new cryptographic assumptions, which we introduce in Section 3.2, arising from the CL framework.

2.4.1 Discrete Logarithm Problems

We here describe two problems, both require a PPT algorithm group generator $\mathsf{Gen}_{\mathsf{DL}}$ which on input 1^λ returns a description (G, q, g) of a multiplicative cyclic group G of order q, generated by g.

The Discrete Logarithm Problem

The discrete logarithm (DL) problem is key to auxiliary structures we use, e.g. when building distributed versions of the EC-DSA signature scheme, computing discrete logarithms in the considered elliptic curve group must be hard.

Definition 2.1. Let λ be a positive integer. Let $\mathcal{A}$ be an adversary for the DL problem, its advantage is defined as:

$$\mathsf{Adv}^{\mathsf{dl}}_{\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \Pr\big[X = g^{x^*} : (G, g, q) \leftarrow \mathsf{Gen}_{\mathsf{DL}}(1^\lambda), x \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z}), \\ X \leftarrow g^x, x^\star \leftarrow \mathcal{A}(G, g, q, X)\big]$$

The DL problem is δ_{dl}-hard for $\mathsf{Gen}_{\mathsf{DL}}$ if for all PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{dl}}_{\mathcal{A}}(\lambda) \leqslant \delta_{\mathsf{dl}}(\lambda)$. We say the DL assumption holds for $\mathsf{Gen}_{\mathsf{DL}}$ (or the DL problem is hard for $\mathsf{Gen}_{\mathsf{DL}}$), if the DL problem is δ_{dl}-hard for $\mathsf{Gen}_{\mathsf{DL}}$ and $\delta_{\mathsf{dl}}(\lambda) = \mathsf{negl}(\lambda)$.

In the generic group model[3], Shoup [Sho97] showed that any generic algorithm solving the DL problem must perform $\Omega(\sqrt{p})$ group operations, where p is the largest prime dividing the order of the group. Generic algorithms are the best ones known for the DL problem on elliptic curves. These algorithms are hence exponential in the size of the group. However in finite fields there exist algorithms with sub-exponential complexity $L_q[1/3]$ solving the DL problem in a field with q elements.

[3] In *generic groups* it is not possible to exploit any special properties of the encodings and group elements can only be operated on using an oracle that provides access to the group operations.

The Decision Diffie-Hellman Problem

Since most readers are familiar with the Decision Diffie-Hellman (DDH) problem, the generic definitions and properties we introduce in Chapters 3 and 4 will be illustrated with running examples based on this problem.

Definition 2.2. Let λ be a positive integer. Let $\mathcal{A}$ be an adversary for the DDH problem, its advantage is defined as:

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Big| \Pr\big[b = b^\star : (G, g, q) \leftarrow \mathsf{Gen}_{\mathsf{DL}}(1^\lambda), \alpha, \beta, \gamma \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z}), X \leftarrow g^\alpha, Y \leftarrow g^\beta,$$
$$b \hookleftarrow \{0,1\}, Z_0 \leftarrow g^{\alpha\beta}, Z_1 \leftarrow g^\gamma, b^\star \leftarrow \mathcal{A}(G, g, q, X, Y, Z_b)\big] - 1/2 \Big|$$

The DDH problem is δ_{ddh}-hard for $\mathsf{Gen}_{\mathsf{DL}}$ if for all PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda) \leqslant \delta_{\mathsf{ddh}}(\lambda)$. We say the DDH assumption holds for $\mathsf{Gen}_{\mathsf{DL}}$ (or the DDH problem is hard for $\mathsf{Gen}_{\mathsf{DL}}$), if the DDH problem is δ_{ddh}-hard for $\mathsf{Gen}_{\mathsf{DL}}$ and $\delta_{\mathsf{ddh}}(\lambda) = \mathsf{negl}(\lambda)$.

2.4.2 The Decisional Composite Residuosity Problem

The Decisional Composite Residuosity (DCR) problem was introduced by Paillier, and underlies his linearly homomorphic public key encryption in [Pai99]. Much of our work improves cryptosystems based either on the Paillier cryptosystem, or directly on the DCR assumption. In order to compare our work to pre-existing protocols, we recall the definition of this problem. In Section 2.5.2 we also recall the Paillier cryptosystem.

Let $\mathsf{Gen}_{\mathsf{RSA}}$ be a PPT algorithm which on input 1^λ returns a description $(N, (P, Q))$ of an RSA group of order $N = PQ$ such that the best algorithm for factoring N takes time 2^λ. Essentially, the problem is to distinguish an N-th residue $r^N \bmod N^2$, where $r \hookleftarrow (\mathbf{Z}/N\mathbf{Z})^*$, from a random element of $(\mathbf{Z}/N^2\mathbf{Z})^*$.

Definition 2.3. Let λ be a positive integer. Let $\mathcal{A}$ be an adversary for the DCR problem, its advantage is defined as:

$$\mathsf{Adv}^{\mathsf{dcr}}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Big| \Pr\big[b = b^\star : (N, (P, Q)) \leftarrow \mathsf{Gen}_{\mathsf{RSA}}(1^\lambda), Y \hookleftarrow (\mathbf{Z}/N^2\mathbf{Z})^*,$$
$$b \hookleftarrow \{0,1\}, Z \leftarrow Y^{N^b} \bmod N^2, b^\star \leftarrow \mathcal{A}(N, Z)\big] - 1/2 \Big|$$

The DCR problem is δ_{dcr}-hard for $\mathsf{Gen}_{\mathsf{RSA}}$ if for all PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{dcr}}_{\mathcal{A}}(\lambda) \leqslant \delta_{\mathsf{dcr}}(\lambda)$. We say the DCR assumption holds for $\mathsf{Gen}_{\mathsf{RSA}}$ (or the DCR problem is hard for $\mathsf{Gen}_{\mathsf{RSA}}$), if the DCR problem is δ_{dcr}-hard for $\mathsf{Gen}_{\mathsf{RSA}}$ and $\delta_{\mathsf{dcr}}(\lambda) = \mathsf{negl}(\lambda)$.

The only known attacks against the DCR problem require factoring N. Hence the hardness of this problem relies on the hardness of factoring large integers. Best known algorithms for this problem use the general number field sieve algorithm which has a complexity in $L_N[1/3]$.

2.5 Basic Cryptographic Primitives

We here provide definitions of standard cryptographic primitives for which we either provide constructions in the upcoming chapters, or which we use as underlying building blocks.

2.5.1 Public Key Encryption

Symmetric or secret-key encryption enables two users, in possession of a common secret key sk, to communicate confidentially. Each user can encrypt a message using sk, send the resulting ciphertext to the other user, who decrypts using the same key sk, so as to recover the original message. The ciphertext alone (without sk) should leak no information on the original message. Bothersome aspects here are that users need to exchange the common secret key before communicating confidential information, and for each user with whom one wants to communicate, one needs to set up and store a secret key.

Public-key encryption schemes overcome these shortcomings. Each user has a public key, which is made available publicly to all other users, and a secret key which is kept secret. Anyone can encrypt a message for a given user, using their public key, while only the secret key associated to that public key will decrypt the ciphertext.

In this thesis we build public key encryption schemes which further allow to publicly manipulate private data, i.e. given a number of ciphertexts and the public key used for encryption, one can compute a new ciphertext encrypting a linear combination of the original messages. We first provide the definition of a public key encryption scheme, and the standard definition for security against passive adversaries.

Definition 2.4 (Public key encryption scheme)**.** Let λ be a positive integer. A *public key encryption* (PKE) scheme with plaintext space $\mathcal{M}$, and ciphertext space $\mathcal{C}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following specifications:

- $\mathsf{Setup}(1^\lambda)$ is a PPT algorithm which on input a security parameter 1^λ, outputs public parameters pp for the encryption scheme.
- $\mathsf{KeyGen}(\mathsf{pp})$ is a PPT algorithm which on input public parameters pp, outputs a public encryption key pk and a secret decryption key sk;
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m)$ is a PPT algorithm which on input parameters pp, a public key pk and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in \mathcal{C}$. To specify the random coins r used by this algorithm, we denote $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m; r)$.
- $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c)$ is a deterministic polynomial time (DPT) algorithm which on input public parameters pp, a secret key sk and a ciphertext c, outputs $m \in \mathcal{M} \cup \{\perp\}$, where $\perp$ is a special rejection symbol.

Correctness requires that for all $\lambda \in \mathbf{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and all messages m, if $c \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m)$, with overwhelming probability it holds that $m = \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c)$.

Remark. In order to avoid heavy notation, as public parameters pp are usually obvious from the context we sometimes omit the explicit description of the Setup algorithm. In this case the input pp to algorithms KeyGen, Enc and Dec is not written explicitly and we run KeyGen on input 1^λ.

We next define the standard security experiment against passive adversaries for PKE. We use the game based security model to formalise this notion, though as noted at the end of Section 2.3.2, game based security is equivalent to simulation based security for public key encryption.

The experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}$. Let $\Pi := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with plaintext space $\mathcal{M}$, and ciphertext space $\mathcal{C}$, and let $\mathcal{A}$ be a PPT adversary. For each $\lambda \in \mathbf{N}$ we denote by $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda)$ the random variable that is defined via the following experiment involving the scheme Π, the adversary $\mathcal{A}$, and a challenger:

1. **Setup phase:** the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\beta \hookleftarrow \{0, 1\}$.
2. **Challenge phase:** $\mathcal{A}$ on input $(1^\lambda, \mathsf{pk})$ outputs $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, then the challenger computes $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_\beta)$ and sends c^* to $\mathcal{A}$.
3. **Output phase:** $\mathcal{A}$ outputs β'. The output of the experiment is 1 if and only if $\beta = \beta'$.

Definition 2.5. A PKE scheme Π with plaintext space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ is indistinguishable under adaptive chosen plaintext attacks (ind-cpa-secure) if for any PPT adversary $\mathcal{A}$, and $\lambda \in \mathbf{N}$,

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}} = 1] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

2.5.2 Linearly Homomorphic Public Key Encryption

A PKE scheme is said to be linearly homomorphic if performing operations on ciphertexts translates into linear operations being performed on the underlying plaintexts. Such schemes thus allow to publicly manipulate confidential data. This malleability is paramount to many applications; take for instance e-voting: given ciphertexts each encrypting one ballot (consisting in 0 or 1 in the case of a 'yes' or 'no' referendum for instance), using additively homomorphic encryption one can efficiently compute an encryption of the sum of all the ballots, so that a single decryption reveals the result of the election [Ben87]. This saves considerable computational resources compared to decrypting each individual ciphertext. Beyond this application, linearly homomorphic encryption has attracted a lot of attention due to its innumerable applications in building more complex protocols (e.g. encryption switching protocols [CPP16], privacy-preserving biometric schemes [OPJM10], general multi party computation [BDOZ11], or general-purpose indistinguishability obfuscation [BDGM20]).

The first homomorphic encryption scheme (which was also the first probabilistic encryption scheme) was put forth by Goldwasser and Micali in [GM84]. This scheme was later improved by Benaloh in his thesis [Ben87], then by Naccache and Stern in [NS98], Okamoto and Uchiyama [OU98] and then further generalised by Joye and Libert in [JL13]. One of the most accomplished linearly homomorphic cryptosystems, standardised in ISO/IEC-18033-6, was designed by Paillier [Pai99] (described later in this section). Its semantic security relies on the DCR assumption. Paillier's scheme was then generalised by Damgård and Jurik [DJ01], allowing to encrypt larger messages. More recently, Castagnos and Laguillaumie [CL15] put forth the first practical linearly homomorphic encrytpion scheme relying solely on the DDH assumption. This latter encryption scheme will be detailed in Section 3.5.1 as it arises from the CL framework.

Besides devising linearly homomorphic encryption schemes with interesting new properties in Section 3.5, we also use these as building blocks to devise functional encryption schemes allowing for the computation of linear functions in Chapter 4, and distributed signatures in Chapter 5.

Definition 2.6 (Linearly homomorphic PKE)**.** Let λ be a positive integer. A linearly homomorphic public key encryption scheme with plaintext space a group $(\mathcal{M}, +)$, and ciphertext space $\mathcal{C}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EvalSum}, \mathsf{EvalScal})$, where $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a PKE scheme, and denoting $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, for any $m_0, m_1 \in \mathcal{M}$, $c_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0)$ and $c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1)$, the algorithms $\mathsf{EvalSum}$ and $\mathsf{EvalScal}$, which take as implicit input pp, must satisfy the following specifications:

- $\mathsf{EvalSum}(\mathsf{pk}, c_0, c_1)$ is a PPT algorithm which on input a public key pk and ciphertexts c_0, c_1, outputs a new ciphertext $c \in \mathcal{C}$ such that $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c) = m_0 + m_1$;
- $\mathsf{EvalScal}(\mathsf{pk}, c_0, \alpha)$ is a PPT algorithm which on input a public key pk a ciphertext c_0, and a $\alpha \in \mathbf{Z}$, outputs a new ciphertext $c \in \mathcal{C}$ such that $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c) = \alpha \cdot m_0$;

Paillier's Linearly Homomorphic Encryption Scheme from DCR

We here recall the linearly homomorphic encryption scheme of Paillier [Pai99], which is ind-cpa-secure under the DCR assumption, and has message space $\mathbf{Z}/N\mathbf{Z}$, where N is an RSA integer. This scheme has been extensively used to build inner product functional encryption schemes, and distributed EC-DSA signatures. Hence through Chapters 4 and 5 we often refer to this cryptosystem, as we both build upon, and compare our work to Paillier-based constructions. As will be detailed in Section 3.5, we note that Paillier's cryptosystem shares many similarities with the encryption scheme $\Pi_{\mathsf{hsm\text{-}cl}}$ we build in Fig. 3.5.

Paillier's encryption scheme is depicted in Fig. 2.1. The message space is $\mathbf{Z}/N\mathbf{Z}$, the ciphertext space is $(\mathbf{Z}/N^2\mathbf{Z})^*$ and the function L is defined as $L(x) = \frac{x-1}{N}$ for $x \in \mathbf{Z}$. Paillier's cryptosystem exploits the fact that in $\mathbf{Z}/N^2\mathbf{Z}$, and for $m \in \mathbf{Z}/N\mathbf{Z}$ it holds that $(1+N)^m = 1 + Nm \bmod N^2$. Consequently, as $L(1+Nm) = m \bmod N$, the group $\mathbf{Z}/N^2\mathbf{Z}$ contains a subgroup of order N, generated by $g = N+1$, in which discrete logarithms in base g can be computed efficiently. We will see parallels to this in our cryptosystems introduced in Section 3.5.

```
Algorithm KeyGen($1^\lambda$)
  1. $(N, (P, Q)) \leftarrow \mathsf{Gen}_{\mathsf{RSA}}(1^\lambda)$
  2. Let $\Lambda \leftarrow (P-1)(Q-1)$
  3. Let $g \leftarrow N + 1$
  4. Let $\mu \leftarrow \Lambda^{-1} \bmod N$. If the computation of $\mu$ fails go back to step 1.
  5. Set $\mathsf{pk} := (N, g)$ and $\mathsf{sk} := (N, \Lambda, \mu)$.
  6. Return $(\mathsf{pk}, \mathsf{sk})$

Algorithm Enc($\mathsf{pk}, m$)
  1. Sample $r \hookleftarrow (\mathbf{Z}/N\mathbf{Z})^*$
  2. Let $c \leftarrow g^m r^N \bmod N^2$
  3. Return $c$

Algorithm Dec($\mathsf{sk}, c$)
  1. Let $m \leftarrow L(c^\Lambda \bmod N^2) \cdot \mu \bmod N$
  2. Return $m$

Algorithm EvalSum($\mathsf{pk}, c_0, c_1$)
  1. Sample $r \hookleftarrow (\mathbf{Z}/N\mathbf{Z})^*$
  2. Let $c \leftarrow c_0 c_1 r^N \bmod N^2$
  3. Return $c$

Algorithm EvalScal($\mathsf{pk}, c_0, \alpha$)
  1. Sample $r \hookleftarrow (\mathbf{Z}/N\mathbf{Z})^*$
  2. Let $c \leftarrow c_0^\alpha r^N \bmod N^2$
  3. Return $c$
```

Figure 2.1: Paillier's linearly homomorphic encryption scheme

2.5.3 Collision Resistant Hashing

We will use collision resistant hash functions to build extended projective hash functions in Chapter 3, Section 3.4.3. They ensure that, though the hash functions are not injective, it is computationally infeasible to find two different input values which yield the same output value. A hash function generator is a PPT algorithm $\mathscr{H}$ that, on input 1^λ, outputs an efficiently computable function $\Gamma : \{0,1\}^* \mapsto \{0,1\}^\lambda$.

Definition 2.7. Let λ be a positive integer. Algorithm $\mathscr{H}$ is a δ_{cr}-hard collision resistant hash function (CRHF) generator if for any PPT adversary $\mathscr{A}$,

$$\mathsf{Adv}^{\mathsf{cr}}_{\mathscr{H},\mathscr{A}}(\lambda) \overset{\text{def}}{=} \Pr[\Gamma(x) = \Gamma(x') \wedge x \neq x' : \Gamma \leftarrow \mathscr{H}(1^\lambda), (x, x') \leftarrow \mathscr{A}(1^\lambda, \Gamma)] \leqslant \delta_{\mathsf{cr}}.$$

We say $\mathscr{H}$ is a collision resistant hash function if $\delta_{\mathsf{cr}}(\lambda) = \mathsf{negl}(\lambda)$.

2.5.4 Signature Schemes

In Chapter 5 we build distributed versions of existing digital signature algorithms. We here provide the definition of a (centralised) digital signature scheme, which is key to understanding Chapter 5. The specific signature scheme which we distribute in Chapter 5, called EC-DSA, will be presented in Section 5.1.2.

We also use one time signatures as a building block for our inner product functional encryption schemes secure against active adversaries in Chapter 4.

Definition 2.8 (Signature scheme)**.** Let λ be a positive integer. A *signature scheme* with message space $\mathcal{M}$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ with the following specifications:

- $\mathsf{Setup}(1^\lambda)$ is a PPT algorithm which on input 1^λ, outputs public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp})$ is a PPT algorithm which on input public parameters pp, outputs a public verification key vk and a secret signing key sk;
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$ is a PPT algorithm which on input public parameters pp, a secret signing key sk and a message $m \in \mathcal{M}$, outputs a signature σ.
- $\mathsf{Verif}(\mathsf{pp}, \mathsf{vk}, m, \sigma)$ is a DPT algorithm which on input public parameters pp, a verification key vk, a message $m \in \mathcal{M}$ and a signature σ, outputs either 0 or 1.

Correctness requires that for any $\lambda \in \mathbf{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, for all $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and all messages m, if $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$, then $\mathsf{Verif}(\mathsf{pp}, \mathsf{vk}, m, \sigma) = 1$.

Remark. In order to avoid heavy notation, the input pp to algorithms Sign and Verif will not be written explicitly as public parameters will be obvious from the context.

We next define standard security experiments for signature schemes.

The experiment $\mathsf{Exp}^{\mathsf{sec}}_{\Sigma,\mathscr{F}}$. Let $\Sigma := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ be a signature scheme with message space $\mathcal{M}$, and let $\mathscr{F}$ be a PPT forger. For each $\lambda \in \mathbf{N}$ we denote by $\mathsf{Exp}^{\mathsf{sec}}_{\Sigma,\mathscr{F}}(\lambda)$ the random variable that is defined via the following experiment involving the scheme Σ, the adversary $\mathscr{F}$, and a challenger $\mathcal{C}$:

1. **Setup phase:** $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and sends pp and vk to $\mathscr{F}$. It also initialises an empty list L_{sig}.
2. **Query phase:** $\mathscr{F}$ performs signature queries by sending messages $m \in \mathcal{M}$ of its choice to $\mathcal{C}$. Then $\mathcal{C}$ computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, adds (m, σ) to L_{sig} and sends σ to $\mathscr{F}$.
3. **Output phase:** $\mathscr{F}$ outputs (m', σ').

(a) If $\mathsf{sec} = \mathsf{euf\text{-}cma}$, the output of the experiment is 1 if and only if $\mathsf{Verif}(\mathsf{vk}, m', \sigma') = 1$ and $\mathcal{F}$ never queried m' to $\mathcal{C}$.

(b) If $\mathsf{sec} = \mathsf{suf\text{-}cma}$, the output of the experiment is 1 if and only if $\mathsf{Verif}(\mathsf{vk}, m', \sigma') = 1$ and $(m', \sigma') \notin L_{\mathsf{sig}}$.

The standard security notion required of digital signature schemes is existential unforgeability under chosen message attacks, we also consider some variants, definitions of which can be found in [BHJ$^+$13].

Definition 2.9 ([GMR88]). Consider a signature scheme Σ with message space $\mathcal{M}$, and a PPT forger $\mathcal{F}$ against security notion $\mathsf{sec} \in \{\mathsf{euf\text{-}cma}, \mathsf{suf\text{-}cma}\}$; for any positive integer λ, $\mathcal{F}$'s advantage is defined as:

$$\mathsf{Adv}^{\mathsf{sec}}_{\Sigma,\mathcal{F}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\Sigma,\mathcal{F}}(\lambda) = 1].$$

If $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{F}}(\lambda) = \mathsf{negl}(\lambda)$ then Σ is existentially unforgeable under chosen message attacks ($\mathsf{euf\text{-}cma}$). If $\mathsf{Adv}^{\mathsf{suf\text{-}cma}}_{\Sigma,\mathcal{F}}(\lambda) = \mathsf{negl}(\lambda)$ then Σ is strongly unforgeable under chosen message attacks ($\mathsf{suf\text{-}cma}$). A signature scheme Σ is called a *one-time signature* if $\mathcal{F}$ is only allowed to make a single signature query in $\mathsf{Exp}^{\mathsf{sec}}_{\Sigma,\mathcal{F}}$; we denote the corresponding security $\mathsf{euf\text{-}1\text{-}cma}$ (resp. $\mathsf{suf\text{-}1\text{-}cma}$). For $\mathsf{sec} \in \{\mathsf{euf\text{-}cma}, \mathsf{suf\text{-}cma}, \mathsf{euf\text{-}1\text{-}cma}, \mathsf{suf\text{-}1\text{-}cma}\}$ we say Σ is δ-sec if for all PPT forger $\mathcal{F}$, $\mathsf{Adv}^{\mathsf{sec}}_{\Sigma,\mathcal{F}}(\lambda) \leqslant \delta(\lambda)$.

2.5.5 Commitments

Equivocal commitment schemes, introduced by Beaver in [Bea96], allow a sender S to commit to a message m such that this message is perfectly hidden from the receiver R; however in the opening phase, when S reveals m to R, S cannot (under computational assumptions) reveal a different m than that committed. The scheme further allows for a trapdoor which enables the opening of commitments to arbitrary messages (this is called equivocating the commitment). The trapdoor should of course be hard to compute efficiently. If the commitment scheme is *non malleable*, one cannot use the commitments of other parties to construct a new commitment which is in some way related to the other parties' inputs.

We use non malleable and equivocal commitment schemes in our distributed signature schemes of Chapter 5. By requiring that all parties commit to their respective input values *before* any committed values are opened, one ensures[4] independence of inputs.

Definition 2.10 (Equivocal commitment scheme). Let λ be a positive integer. A (non-interactive) *equivocal commitment scheme* is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Equiv})$ with the following specifications:

- $\mathsf{Setup}(1^\lambda)$ is a PPT algorithm which on input 1^λ outputs public parameters pp and associated secret trapdoor key tk;
- $\mathsf{Com}(\mathsf{pp}, m; r)$ is a DPT algorithm which on input public parameters pp, a message m, and random coins r, outputs a commitment c and an opening value d (if a sender refuses to open a commitment $\mathsf{d} = \perp$);
- $\mathsf{Open}(\mathsf{pp}, \mathsf{c}, \mathsf{d})$ is a DPT algorithm which on input public parameters pp, a commitment c and an opening value d, outputs either a message m or an error symbol $\perp$;

[4] As long as the trapdoor is unknown to all parties.

- $\mathsf{Equiv}(\mathsf{pp}, \mathsf{tk}, m, r, m')$ is a PPT algorithm which on input public parameters pp, a trapdoor key tk, a message m, random coins r and another message m', outputs an equivocating opening value $\widehat{\mathsf{d}}$.

Correctness requires that $\forall \lambda \in \mathbf{N}$, any message m, and any random coins r, for any $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$ it holds that $\mathsf{Open}(\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m; r)) = m$. Equivocality requires that for all messages m and m', any random coins r, any $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$, denoting $(\mathsf{c}, \mathsf{d}) \leftarrow \mathsf{Com}(\mathsf{pp}, m; r)$ and $\widehat{\mathsf{d}} \leftarrow \mathsf{Equiv}(\mathsf{pp}, \mathsf{tk}, m, r, m')$ it holds that $\mathsf{Open}(\mathsf{pp}, \mathsf{c}, \widehat{\mathsf{d}}) = m'$.

Security properties. Let λ be a positive integer. The properties we will require of our commitment schemes are the following:

- *Perfect hiding:* for every message pair m, m' the distributions of the resulting commitments are statistically close.
- *Computational binding:* for any PPT adversary $\mathcal{A}$, and any $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$, the probability that $\mathcal{A}$, on input pp, outputs $(\mathsf{c}, \mathsf{d}_0, \mathsf{d}_1)$ satisfying $\mathsf{Open}(\mathsf{pp}, \mathsf{c}, \mathsf{d}_0) = m_0$; $\mathsf{Open}(\mathsf{pp}, \mathsf{c}, \mathsf{d}_1) = m_1$; $m_0 \neq \perp$; $m_1 \neq \perp$ and $m_0 \neq m_1$ is negligible in λ.
- *Indistinguishability of opening fake and real commitments:* for any PPT adversary $\mathcal{A}$, any $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$, any messages m, m' and randomness r, r', denoting $(\mathsf{c}, \mathsf{d}) \leftarrow \mathsf{Com}(\mathsf{pp}, m; r)$, $(\mathsf{c}', \mathsf{d}') \leftarrow \mathsf{Com}(\mathsf{pp}, m'; r')$ and $\widehat{\mathsf{d}} \leftarrow \mathsf{Equiv}(\mathsf{pp}, \mathsf{tk}, m', r', m)$, it holds that:

$$\Big|\Pr\big[b = b^* : b \hookleftarrow \{0,1\}, (\mathsf{c}_0, \mathsf{d}_0) \leftarrow (\mathsf{c}, \mathsf{d}),$$
$$(\mathsf{c}_1, \mathsf{d}_1) \leftarrow (\mathsf{c}', \widehat{\mathsf{d}}), b^* \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{c}_b, \mathsf{d}_b)\big]\Big| = \mathsf{negl}(\lambda).$$

- *Concurrent non-malleability:* a commitment scheme is non-malleable [DDN00] if no PPT adversary $\mathcal{A}$ can "maul" a commitment to a value m into a commitment to a related value $\overline{m}$. The notion of a concurrent non-malleable commitment [DDN00, PR05] further requires non-malleability to hold even if $\mathcal{A}$ receives many commitments and can itself produce many commitments. We refer the reader to [PR05] for a formal definition of concurrent non-malleability.

Concurrent non-malleability is achieved by the commitment schemes presented in [DG03, Gen04, MY04]. Any of these can be used in our full threshold signature protocol of Section 5.3. For the two party signing protocol of Section 5.2, we use an ideal commitment functionality, this implies that security of the commitment should be proven in the universally composable security framework (*cf.* Section 2.3.2). To implement the ideal commitment functionality used in our two party protocol, one can use e.g. [HMRT12, BCPV13, Fuj16].

2.6 Background on Class Groups

The framework adopted throughout this thesis – called the CL framework, introduced by Castagnos and Laguillaumie at [CL15], and presented in Chapter 3 – can be instantiated from ideal class groups of orders of an imaginary quadratic field. This is the main instantiation we will adopt to illustrate the feasibility and efficiency of all our constructions. We here provide some facts and notations from the theory of imaginary quadratic number fields, details can be found in [Cox89].

2.6.1 Imaginary Quadratic Fields and Class Groups

An imaginary quadratic field can be written uniquely in the form $K := Q(\sqrt{N})$, where N is a square free negative integer. The integer Δ_K, defined as $\Delta_K := N$ if $N = 1 \bmod 4$, and $\Delta_K := 4N$ otherwise, is the *fundamental discriminant* of K. The ring of integers $\mathcal{O}_{\Delta_K}$ of K is the free $\mathbf{Z}$-module of rank $2 = [K : \mathbf{Q}]$ defined as $\mathcal{O}_{\Delta_K} := \mathbf{Z} + \omega_K \mathbf{Z}$ where $\omega_K := \frac{\Delta_K + \sqrt{\Delta_K}}{2}$. An order $\mathcal{O}$ of a quadratic field K is a subring $\mathcal{O} \subset \mathcal{O}_{\Delta_K}$ that is also a free $\mathbf{Z}$-module of rank 2 so that it contains an integral basis of K; an order $\mathcal{O} = a\mathbf{Z} + b\mathbf{Z}$ can be succinctly represented by the basis $[a, b]$ (and so $\mathcal{O}_{\Delta_K} = [1, \omega_K]$). We call $\mathcal{O}_{\Delta_K}$ the *maximal order* since it contains every other order. Moreover, since both $\mathcal{O}$ and $\mathcal{O}_{\Delta_K}$ are free $\mathbf{Z}$-modules of the same rank, it holds that the index $f = [\mathcal{O}_{\Delta_K} : \mathcal{O}]$ is finite. The integer f is called the conductor of $\mathcal{O}$. Then as $\mathcal{O}_{\Delta_K} = [1, \omega_K]$, one can show that $\mathcal{O} = [1, f\omega_K]$ [Cox89, p.133]. Another important invariant of $\mathcal{O}$ is its discriminant, which is given by the formula $\Delta := f^2\Delta_K$.

We now give a little background on the theory of ideals in a non-maximal order. Throughout this section we closely follow the book [Cox89, p.132-150]. Consider an order $\mathcal{O}$ of discriminant Δ. An ideal $\mathfrak{a}$ of $\mathcal{O}$ is a sub-$\mathbf{Z}$-module of $\mathcal{O}$ such that for every $r \in \mathcal{O}$ and $a \in \mathfrak{a}$, we have $ra \in \mathfrak{a}$. If $\mathfrak{a}$ is a nonzero ideal of $\mathcal{O}$, then $\mathcal{O}/\mathfrak{a}$ is a finite ring, and one can thus define the *norm* of $\mathfrak{a}$ to be $N(\mathfrak{a}) = |\mathcal{O}/\mathfrak{a}|$.

A *fractional ideal* $\mathfrak{a}$ of $\mathcal{O}$ is a subset of K for which there exists a non-zero integer d such that $d\mathfrak{a}$ is an ideal of $\mathcal{O}$. An $\mathcal{O}$-ideal $\mathfrak{a}$ is *principal* if there exists $\alpha \in K^*$ such that $\mathfrak{a} = \alpha\mathcal{O}$. It is clear that if $\mathfrak{a}$ is a fractional ideal of $\mathcal{O}$, then $\mathfrak{a} \subset \mathcal{O}$ if and only if $\mathfrak{a}$ is an ideal of $\mathcal{O}$, in which case $\mathfrak{a}$ is said to be an *integral ideal*. Having defined fractional ideals, we can now introduce invertible ideals. A fractional $\mathcal{O}$-ideal $\mathfrak{a}$ is *invertible* if there is another fractional $\mathcal{O}$-ideal $\mathfrak{b}$ such that $\mathfrak{a}\mathfrak{b} = \mathcal{O}$. Clearly principal fractional ideals (which are of the form $\alpha\mathcal{O}$, $\alpha \in K^*$) are invertible. Given an order $\mathcal{O}$, let $I(\mathcal{O})$ denote the set of invertible fractional ideals. $I(\mathcal{O})$ is a group under multiplication. The principal $\mathcal{O}$-ideals give a subgroup $P(\mathcal{O}) \subset I(\mathcal{O})$. When $\mathcal{O}$ is the maximal order $\mathcal{O}_{\Delta_K}$, we will denote $I_{\Delta_K} := I(\mathcal{O}_{\Delta_K})$ and $P_{\Delta_K} := P(\mathcal{O}_{\Delta_K})$. We can now define the *ideal class group* $Cl(\mathcal{O})$ of the order $\mathcal{O}$, which is the quotient

$$Cl(\mathcal{O}) := I(\mathcal{O})/P(\mathcal{O}).$$

The order of the group $Cl(\mathcal{O})$, which is finite, is called the class number of $Cl(\mathcal{O})$, and denoted $h(\mathcal{O})$. The class number is not efficiently computable if the discriminant is fundamental; in fact the determination of $h(\mathcal{O}_{\Delta_K})$ is one of the main problems in algorithmic algebraic number theory.

Unless $K = \mathbf{Q}(\sqrt{-3})$ or $\mathbf{Q}(i)$, i.e. $\Delta = -3$ or 4, for all imaginary quadratic fields it holds that the group of units of $\mathcal{O}$, denoted $\mathcal{O}^*$ is $\{\pm 1\}$. Given an order $\mathcal{O}$ of conductor f, a non-zero $\mathcal{O}$-ideal $\mathfrak{a}$ is prime to f if $\mathfrak{a} + f\mathcal{O} = \mathcal{O}$. It holds that $\mathcal{O}$-ideals prime to f lie in $I(\mathcal{O})$ and are closed under multiplication. They generate a subgroup of fractional ideals denoted $I(\mathcal{O}, f) \subset I(\mathcal{O})$.

Representing classes

We will be computing with equivalence classes of class groups, and therefore need to select representatives from each class. Each fractional ideal of an imaginary quadratic order can be written $\mathfrak{a} = r(a\mathbf{Z} + (-b + \sqrt{\Delta})/2\mathbf{Z})$ where $a, b \in \mathbf{Z}$, $r \in \mathbf{Q}$, $a > 0$ and $4a|(b^2 - \Delta)$. Therefore a fractional ideal can be represented by a triple (r, a, b). An ideal (r, a, b) is *integral* if $r \in \mathbf{Z}$, and if $r = 1$ the ideal is *primitive*, and is denoted (a, b) for short. It then holds that the norm of $\mathfrak{a}$ is $N(\mathfrak{a}) = a$. This notation also represents the binary quadratic form $ax^2 + bxy + cy^2$ with $b^2 - 4ac = \Delta$. If the form is normal (i.e. $-a < b \leqslant a$) then this representation is unique.

Two ideals $\mathfrak{a}_1, \mathfrak{a}_2$ are said to be *equivalent* if there is a non zero number $\alpha \in \mathbf{Q}(\sqrt{\Delta})$ such that $\mathfrak{a}_2 = \alpha\mathfrak{a}_1$. As our class representative, we can thus chose a primitive ideal. Moreover, in class groups of imaginary quadratic orders each equivalence class of ideals has exactly one *reduced* primitive ideal. An ideal is reduced if it is normal, $a \leqslant c$ and if $a = c$, then $b > 0$. We represent each class of ideals by the unique reduced ideal in the class.

To compute this reduced ideal from any given ideal in the class, one can use the efficient algorithm presented in [Coh00, Algo. 5.4.2, p.243], which is a variant of Euclid's algorithm. This reduction procedure allows us to work with reduced ideals, instead of classes. The product of two reduced ideals can be computed efficiently using an algorithm which composes quadratic forms, due to Shanks, and presented in [Coh00, Algo. 5.4.7, p.247]. Both the aforementioned algorithms have quadratic complexity (even quasi linear using fast arithmetic). A useful result for our purpose is that any normal ideal $\mathfrak{a} = (a, b)$ with $|a| < \sqrt{|\Delta|/4}$ is reduced [Coh00, Lem. 5.3.4, p.232].

Relations between classes

For any order $\mathcal{O}$, ideals prime to the conductor relate nicely to the ideals of the maximal order $\mathcal{O}_{\Delta_K}$. Given any positive integer m, an $\mathcal{O}_{\Delta_K}$-ideal $\mathfrak{a}$ is said to be prime to m if $\mathfrak{a} + m\mathcal{O}_{\Delta_K} = \mathcal{O}_{\Delta_K}$; and so the subgroup $I_{\Delta_K}(m) \subset I_{\Delta_K}$ is the subgroup generated by $\mathcal{O}_{\Delta_K}$-ideals prime to m. Let us consider $\mathcal{O}_{\Delta_f}$, an order of conductor f with $\Delta_f = f^2\Delta_K$.

- If $\mathfrak{A}$ is an $\mathcal{O}_{\Delta_K}$-ideal prime to f, then $\mathfrak{A} \cap \mathcal{O}_{\Delta_f}$ is an $\mathcal{O}_{\Delta_f}$-ideal prime to f of the same norm.
- If $\mathfrak{a}$ is an $\mathcal{O}_{\Delta_f}$-ideal prime to f, then $\mathfrak{a}\mathcal{O}_{\Delta_K}$ is an $\mathcal{O}_{\Delta_K}$-ideal prime to f of the same norm.
- The map $\mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_K}$ is an isomorphism $\phi_f : I(\mathcal{O}_{\Delta_f}, f) \xrightarrow{\sim} I_{\Delta_K}(f)$.
- The inverse of ϕ_f maps $\mathfrak{a}$ to $\mathfrak{a} \cap \mathcal{O}_{\Delta_f}$.

The map ϕ_f induces a surjection

$$\bar{\phi}_f : Cl(\mathcal{O}_{\Delta_f}) \twoheadrightarrow Cl(\mathcal{O}_{\Delta_K}).$$

The maps ϕ_f, its inverse ϕ_f^{-1} and $\bar{\phi}_f$ can be efficiently computed knowing the conductor f (*cf.* [PT00]). Moreover the following formula relates the class numbers of both orders, and gives the order of the kernel of $\bar{\phi}_f$ (*cf.* [Cox89, Theorem 7.24]):

$$h(\mathcal{O}_{\Delta_f}) = \frac{h(\mathcal{O}_{\Delta_K})f}{[\mathcal{O}^*_{\Delta_K} : \mathcal{O}^*_{\Delta_f}]} \prod_{p|f} (1 - (\frac{\Delta_K}{p})\frac{1}{p}).$$

Note that $h(\mathcal{O}_{\Delta_f})$ is always an integer multiple of $h(\mathcal{O}_{\Delta_K})$.

Throughout this thesis we will consider a prime conductor $f = q$ and consider $\Delta_q := q^2\Delta_K$, for a large fundamental discriminant $\Delta_K < -4$ divisible by q. Using the above formula, one can show that in this case the order of the kernel of $\bar{\phi}_q$ is

$$\frac{h(\mathcal{O}_{\Delta_q})}{h(\mathcal{O}_{\Delta_K})} = q.$$

Structure of the kernel of $\bar{\phi}_q$

We here explain how one can exhibit, in the class group of an imaginary quadratic field, the existence of a subgroup in which the DL problem can be efficiently solved. For more details, we refer the reader to [CL15, Prop. 1] and [Cas19, Section 1.3]. Recall that we consider the case where $f = q$ is a prime conductor, and consider $\Delta_q := q^2\Delta_K$, for a large fundamental discriminant $\Delta_K < -4$ divisible by q. From the previous paragraph we know that the order of the kernel of $\bar{\phi}_q$ is q. One can demonstrate this by proving there is an effective isomorphism (still in [Cox89, Theorem 7.24]) between this kernel and

$$(\mathcal{O}_{\Delta_K}/q\mathcal{O}_{\Delta_K})^\times/(\mathbf{Z}/q\mathbf{Z})^\times.$$

Now as $q|\Delta_K$ the above quotient is cyclic of order q, and denoting $\mathfrak{f}$ the ideal (q^2, q) of $\mathcal{O}_{\Delta_q}$, and by $f := [\mathfrak{f}]$ the class of $\mathfrak{f}$ in $Cl(\mathcal{O}_{\Delta_q})$, Castagnos and Laguillaumie [CL15, Prop. 1] demonstrate that f generates the kernel of $\bar{\phi}_q$. Hence f is of order q. For $m \in \{1, \ldots, q-1\}$ let us denote $L(m)$ the odd integer in $[-q, q]$ such that $L(m) = 1/m \bmod q$; they also show that the reduced ideal equivalent to f^m is equal to $(q^2, L(m)q)$. Moreover, if we impose $q < \sqrt{|\Delta_K|/4}$, then all these ideals of norm q^2 will be reduced. As a result the kernel of $\bar{\phi}_q$ is a cyclic subgroup of order q of $Cl(\mathcal{O}_{\Delta_q})$ where the DL problem is easy. Indeed any element h of this subgroup $\langle f \rangle$ can be represented by a reduced ideal of the form (q^2, kq) for some integer k, and the discrete logarithm of h in base f is $k^{-1} \bmod q$.

The existence of this prime order subgroup in which the DL problem is easy is fundamental to the design of the CL framework. It is analogue to the fact that the DL problem is easy in the subgroup of order N of $(\mathbf{Z}/N^2\mathbf{Z})^\times$. This latter property is essential to the Paillier cryptosystem (*cf.* Section 2.5.2). Moreover just as the Paillier cryptosystem's message space is $\mathbf{Z}/N\mathbf{Z}$, the message space of cryptosystems arising from the CL framework is $\mathbf{Z}/q\mathbf{Z}$ (as detailed in Chapter 3).

2.6.2 The Discrete Logarithm Problem and Computing the Class Number

For most cryptosystems built in ideal class groups of an imaginary quadratic order, the computational problem on which the security is based is the discrete logarithm (DL) problem. Solving instances of the DL problem in imaginary quadratic orders is closely related to computing the class number $h(\Delta_K)$ of the ideal class group $Cl(\mathcal{O}_{\Delta_K})$. The fastest algorithms for solving these problems in imaginary quadratic fields are based on an improved version of Hafner and McCurley's index-calculus algorithm [HM89] due to Jacobson [Jac99]. Biasse et al. in [BJS10], conjectured that the best known algorithms to solve these problems have complexity $L_{|\Delta_K|}[1/2, o(1)]$.

In [HM00], Hamdy and Möller give recommendations for securely choosing the discriminant Δ_K for use in imaginary quadratic field cryptography, and in particular, such that the DL problem in $Cl(\mathcal{O}_{\Delta_K})$ is as hard as in finite fields. It is advised to construct a fundamental discriminant Δ_K and to minimise the 2-Sylow subgroup of the class group. In our case, by construction Δ_K will be the product of two odd primes. By choosing $\Delta_K = -\tilde{q}q$ with $\tilde{q}$ and q such that $q \equiv -\tilde{q} \bmod 4$, it holds that Δ_K is a fundamental discriminant. Moreover the 2-Sylow subgroup is isomorphic to $\mathbf{Z}/2\mathbf{Z}$ if we choose $\tilde{q}$ and q such that $\left(\frac{q}{\tilde{q}}\right) = \left(\frac{\tilde{q}}{q}\right) = -1$ (*cf.* [Kap73, p. 598]). We will thus work with the odd part, which is the group of squares of $Cl(\mathcal{O}_{\Delta_K})$. Following the Cohen-Lenstra heuristics, *cf.* [Coh00, Chapter 5.10.1], the probability that the odd part of the class group is cyclic is 97.75%; and extending their heuristics, it is conjectured in [HS06] that the probability that an odd prime d divides $h(\mathcal{O}_{\Delta_K})$ is less than $1/d + 1/(d \log d)$. As a result, we can not guarantee that the order of the odd part is not divisible by small primes. Nevertheless, as indicated in [HM00], if one chooses Δ_K large enough, the tendency of the class number towards having small factors does not mean it will be smooth with non-negligible probability, hence an adaptation of the Pohlig-Hellman algorithm is not possible. On average, $h(\mathcal{O}_{\Delta_K})$ behaves roughly as $\sqrt{|\Delta_K|}$, see [Coh00, Chapter 4.9.15] (Brauer-Siegel). Moreover (*cf.* [Coh00, p. 295]),

$$h(\mathcal{O}_{\Delta_K}) \leqslant \frac{1}{\pi}\log(|\Delta_K|)\sqrt{|\Delta_K|}. \tag{2.1}$$

To conclude, following [HM00], if Δ_K is taken large enough, generic methods to compute discrete logarithms such as Pollard ρ-method are slower than the index-calculus algorithms. Thus, since index-calculus algorithms for solving the DL problem are asymptotically much slower than index-calculus algorithms to solve the integer factorisation problem, the discriminant can be taken smaller than the corresponding RSA modulus. This is illustrated in Table 2.1 where we give, for a given security parameter λ, the size in bits of RSA moduli N and discriminants Δ_K of imaginary quadratic fields for which factoring and the quadratic field DL problem have the same estimated running time (figures are taken from [BJS10]).

2.6.3 Key Sizes and Timings

When instantiating our cryptosystems of Chapters 3 to 5 in the CL framework from class groups, group elements are reduced ideals in the class group of discriminant $-q^3\tilde{q}$, and can thus be represented by two integers smaller than $q\sqrt{|\Delta_K/3|}$. Conversely, for constructions relying on Paillier's cryptosystem (or equivalently on the DCR assumption), group elements are elements of $\mathbf{Z}/N^2\mathbf{Z}$. Hence in constructions arising from the CL framework, group elements are considerably smaller than for those arising from Paillier. This is illustrated in Table 2.1, where λ denotes the security parameter.

For our cryptosystems arising from class groups, the underlying message space will be $\mathbf{Z}/q\mathbf{Z}$. This is much smaller than the Paillier message space $\mathbf{Z}/N\mathbf{Z}$. In practice, a 128 bits message space is large enough, if for instance, one needs to perform computations with double or quadruple precision.

	$\lambda = 112$		$\lambda = 128$		$\lambda = 192$		$\lambda = 256$	
size	CL	DCR	CL	DCR	CL	DCR	CL	DCR
N	-	2048	-	3072	-	7680	-	15360
q	112	-	128	-	192	-	256	-
Δ_K	1348	-	1827	-	3598	-	5971	-
group element	1572	4096	2083	6144	3982	15360	6483	30720

Table 2.1: Sizes in bits required to build cryptosystems arising from DCR and from the CL framework

In terms of timings, a fair comparison is difficult since to our knowledge, no library for the arithmetic of quadratic forms is as optimised as a standard library for the arithmetic of modular integers.

In Table 2.2 we measure timings for a multiplication in the class group versus a multiplication in $\mathbf{Z}/N^2\mathbf{Z}$ for different security parameters λ. These timings were performed with the

Pari C Library [PAR20], as this library handles arithmetic in class groups and in $\mathbf{Z}/N^2\mathbf{Z}$. Timings are measured on a desktop computer (with Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz processor).

	$\lambda = 112$		$\lambda = 128$		$\lambda = 192$		$\lambda = 256$	
size	CL	DCR	CL	DCR	CL	DCR	CL	DCR
Multiplication (ms.)	0.037	0.003	0.048	0.007	0.098	0.030	0.176	0.087

Table 2.2: Comparing timings for one multiplication in class groups and in $\mathbf{Z}/N^2\mathbf{Z}$

Throughout this thesis the values in Tables 2.1 and 2.2 allow us to estimate the sizes of elements relevant to the efficiency of our cryptosystems, their theoretical computational complexity, and extrapolate expected running times.

Precisely we assume the standard square-and-multiply algorithm for exponentiation, with uniformly distributed exponents – which has an average complexity of about ℓ squarings and $\ell/2$ multiplications, where ℓ is the bit-size of the exponent. In class groups, our measures indicate that the cost of a squaring is essentially equal to the cost of multiplication, hence we multiply the exponent by 1.5 times the cost of a multiplication. For squarings in $\mathbf{Z}/N^2\mathbf{Z}$, the cost is 0.9 times the cost of multiplication in $\mathbf{Z}/N^2\mathbf{Z}$. Hence we multiply the exponent by 1.4 times the cost of a multiplication. Finally for arithmetic in $\mathbf{Z}/N^2\mathbf{Z}$, we also distinguish exponentiations modulo N, as we measure that a multiplication mod N costs 1/3 of the cost of a multiplication mod N^2.

As a final note, in our tables, we adopt the following color-code:

Green values highlight best performing figures.

Orange values highlight figures which are not the best, but not the worst either.

Red values highlight figures which are the most expensive.

2.7 Distributions

As previously noted, the order $h(\Delta_K)$ of the ideal class group $Cl(\mathcal{O}_{\Delta_K})$ is unknown, we only have an upper bound for $h(\Delta_K)$ (given by Eq. (2.1)). Hence in our forthcoming applications, the order will be unknown. This implies that exponents are sampled over the integers, and not modulo the order of the group, and that we can only sample elements *close to uniform* in the considered groups. We first provide lemmas describing distributions from which one can sample exponents in order to induce distributions close to uniform in these groups, while minimising the size of exponents so as to optimise efficiency. Next, we prove some properties of these distributions which will be essential for the analysis of information leakage in our security proofs.

When possible in our applications we sample exponents from folded Gaussians instead of folded uniforms to induce distributions close to uniform. This is because Gaussian sampling allows us to have shorter exponents, which, in our cryptosystems, translates as shorter keys and randomness, and hence better overall efficiency (the use of Gaussian sampling instead of uniform has been suggested for instance in [ALS16, CIL17]). In Section 2.7.3 we thus present definitions and basic results about Gaussian distributions which will be useful for our security proofs.

2.7.1 Sampling Close to the Uniform Distribution

The following lemmas explain how one can sample exponents over the integers which induce distributions close to uniform in groups of unknown order (provided one has an upper bound for this order). Lemma 2.11, proven in [CL15, Appendix C, Lemma 4], explains how to sample exponents from folded uniforms, while Lemma 2.12, proven in [CIL17, Appendix C, Lemma 1], explains how to sample exponents from folded discrete Gaussian distributions (defined in Definition 2.17).

Lemma 2.11 ([CL15])**.** Let G be a cyclic group of order n, generated by g. Consider the random variable X sampled uniformly from G; as such it satisfies $\Pr[X = h] = \frac{1}{n}$ for all $h \in G$. Now consider the random variable Y with values in G defined as follows: draw y uniformly from $\{0, \ldots, B-1\}$, with $B \geqslant n$, and set $Y := g^y$. Then, denoting $r := B \bmod n$, it holds that:

$$\Delta(X, Y) = \frac{r(n-r)}{nB} \leqslant \frac{n}{4B}.$$

Lemma 2.12 ([CIL17])**.** Let G be a cyclic group of order n, generated by g. Consider the random variable X sampled uniformly from G; as such it satisfies $\Pr[X = h] = \frac{1}{n}$ for all $h \in G$. Now consider the random variable Y with values in G defined as follows: draw y from the discrete Gaussian distribution $\mathcal{D}_{\mathbf{Z},\sigma}$, with $\sigma \geqslant n\sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}}$, and set $Y := g^y$. Then it holds that:

$$\Delta(X, Y) \leqslant 2\epsilon.$$

2.7.2 Properties of Almost Uniform Distributions

We here prove some useful results on distributions close to uniform modulo $n = ab$ where a and b are co-prime. In particular, on the independence of their distribution taken mod a and taken mod b. It is well know that if exponents are sampled *uniformly* mod n, then they are uniformly distributed modulo any divisor of n, and hence mod a and b; moreover, since a is co-prime with b, the value of these exponents taken mod a is independent of their value mod b.

As mentioned previously, in our work exponents are often sampled from distributions close to uniform. We must therefore adapt the aforementioned results to this *approximate* setting. We note that though these properties may be folklore, we did not find explicit proofs for them, which is why we prove them here.

Lemma 2.13. If k is sampled from a distribution $\mathcal{D}_a$, which is ϵ-close to $\mathcal{U}(\mathbf{Z}/a\mathbf{Z})$ for $a \in \mathbf{Z}$ then for all $b \in \mathbf{Z}$ such that b divides a it holds that ($k \bmod b$) follows a distribution ϵ-close to $\mathcal{U}(\mathbf{Z}/b\mathbf{Z})$.

Proof. Let X_a be a random variable sampled from $\mathcal{D}_a$, and let X_b be the random variable defined as $X_b = X_a \bmod b$ for b dividing a. Then:

$$\begin{aligned}\sum_{k \in \mathbf{Z}/b\mathbf{Z}} \Big|\Pr[X_b = k] - \frac{1}{b}\Big| &= \sum_{k \in \mathbf{Z}/b\mathbf{Z}} \Big| \sum_{i=0}^{a/b-1} \Big(\Pr[X_a = k + ib]\Big) - \frac{1}{b}\Big| \\ &= \sum_{k \in \mathbf{Z}/b\mathbf{Z}} \Big| \sum_{i=0}^{a/b-1} \Big(\Pr[X_a = k + ib] - \frac{1}{a}\Big)\Big|\end{aligned}$$

$$\leqslant \sum_{k\in\mathbf{Z}/b\mathbf{Z}} \sum_{i=0}^{a/b-1} \Big|\Pr[X_a = k+ib] - \frac{1}{a}\Big| = \sum_{k\in\mathbf{Z}/a\mathbf{Z}} \Big|\Pr[X_a = k] - \frac{1}{a}\Big| = \epsilon.$$

Thus X_b follows a distribution at distance $\leqslant \epsilon$ from the uniform distribution modulo b.

□

Lemma 2.14. Consider positive integers a and b such that $\gcd(a,b)=1$. For $x_a \in \mathbf{Z}/a\mathbf{Z}$ and $x_b \in \mathbf{Z}/b\mathbf{Z}$, let use denote $\mathsf{crt}(x_a, x_b, a, b) := x_a \cdot b \cdot (b^{-1} \bmod a) + x_b \cdot a \cdot (a^{-1} \bmod b) \bmod ab$. And let $\mathcal{D}_a$ (resp. $\mathcal{D}_b$) be a distribution δ_a-close to $\mathcal{U}(\mathbf{Z}/a\mathbf{Z})$ (resp. δ_b-close to $\mathcal{U}(\mathbf{Z}/b\mathbf{Z})$). Then the distribution $\{\mathsf{crt}(X_a, X_b, a, b), X_a \hookleftarrow \mathcal{D}_a, X_b \hookleftarrow \mathcal{D}_b\}$ is δ-close to $\mathcal{U}(\mathbf{Z}/ab\mathbf{Z})$, where $\delta \leqslant \delta_a\delta_b + \delta_a + \delta_b$.

Proof. As $\gcd(a,b) = 1$, one has $\mathbf{Z}/ab\mathbf{Z} \simeq \mathbf{Z}/a\mathbf{Z} \times \mathbf{Z}/b\mathbf{Z}$. The induced isomorphism from $\mathbf{Z}/ab\mathbf{Z}$ to $\mathbf{Z}/a\mathbf{Z} \times \mathbf{Z}/b\mathbf{Z}$ maps c to $(c \bmod a, c \bmod b)$. The probability that a random variable $C := \mathsf{crt}(X_a, X_b, a, b)$ for $X_a \hookleftarrow \mathcal{D}_a, X_b \hookleftarrow \mathcal{D}_b$ is equal to a fixed element c of $\mathbf{Z}/ab\mathbf{Z}$, is $\Pr[X_a = c \bmod a] \cdot \Pr[X_b = c \bmod b]$. Then:

$$\begin{aligned}
&\sum_{c\in\mathbf{Z}/ab\mathbf{Z}} \left|\frac{1}{ab} - \Pr[X_a = c \bmod a]\Pr[X_b = c \bmod b]\right| \\
&= \sum_{\substack{c_1\in\mathbf{Z}/a\mathbf{Z}\\ c_2\in\mathbf{Z}/b\mathbf{Z}}} \left|\frac{1}{ab} - \Pr[X_a = c_1]\Pr[X_b = c_2]\right| \\
&= \sum_{\substack{c_1\in\mathbf{Z}/a\mathbf{Z}\\ c_2\in\mathbf{Z}/b\mathbf{Z}}} \Big|\Big(\frac{1}{a} - \Pr[X_a = c_1]\Big)\Big(\frac{1}{b} - \Pr[X_b = c_2]\Big) \\
&\qquad\qquad - \frac{1}{a}\Big(\frac{1}{b} - \Pr[X_b = c_2]\Big) - \frac{1}{b}\Big(\frac{1}{a} - \Pr[X_a = c_1]\Big)\Big| \\
&\leqslant \sum_{\substack{c_1\in\mathbf{Z}/a\mathbf{Z}\\ c_2\in\mathbf{Z}/b\mathbf{Z}}} \Big|\Big(\frac{1}{a} - \Pr[X_a = c_1]\Big)\Big(\frac{1}{b} - \Pr[X_b = c_2]\Big)\Big| \\
&\qquad\qquad + \frac{1}{a}\left|\frac{1}{b} - \Pr[X_b = c_2]\right| + \frac{1}{b}\left|\frac{1}{a} - \Pr[X_a = c_1]\right|.
\end{aligned}$$

As a result, the statistical distance δ to $\mathcal{U}(\mathbf{Z}/ab\mathbf{Z})$ satisfies $\delta \leqslant \delta_a\delta_b + \delta_b + \delta_a$. □

Lemma 2.15. Consider positive integers a and b such that $\gcd(a,b) = 1$, and let $\mathcal{D}$ be a distribution ϵ-close to $\mathcal{U}(\mathbf{Z}/ab\mathbf{Z})$. Consider the random variable X sampled from $\mathcal{D}$, and let X_a (resp. X_b) be the random variable defined as $X_a = X \bmod a$ (resp. $X_b = X \bmod b$). Then the random variables X_a and X_b follow distributions ϵ-close to $\mathcal{U}(\mathbf{Z}/a\mathbf{Z})$ and $\mathcal{U}(\mathbf{Z}/b\mathbf{Z})$ respectively. Moreover even knowing X_b, X_a remains 2ϵ-close to $\mathcal{U}(\mathbf{Z}/a\mathbf{Z})$ (and vice versa).

Proof. Let $\mathcal{C}$ be an unbounded algorithm which takes as input a tuple $(a,b,x) \in \mathbf{N}^2 \times \mathbf{Z}/ab\mathbf{Z}$, which is either a sample of $\mathcal{U}$ or $\mathcal{V}$, before outputting a bit, where:

$\mathcal{U} := \{(a,b,x) | \gcd(a,b) = 1 \ \wedge \ x \hookleftarrow \mathbf{Z}/ab\mathbf{Z}\}$ and $\mathcal{V} := \{(a,b,x) | \gcd(a,b) = 1 \ \wedge \ x \hookleftarrow \mathcal{D}\}$.

Since distributions $\mathcal{U}$ and $\mathcal{V}$ are ϵ-close, for any such algorithm $\mathcal{C}$, it holds that:

$$|\Pr[\mathcal{C}(\mathcal{U}) \to 1] - \Pr[\mathcal{C}(\mathcal{V}) \to 1]| \leqslant \epsilon.$$

We further denote $\mathcal{U}_{\mathcal{A}} := \{(a,b,x_b,x_a) | (a,b,x) \hookleftarrow \mathcal{V};\ x_b \leftarrow x \bmod b;\ x_a \hookleftarrow \mathbf{Z}/a\mathbf{Z}\}$ and $\mathcal{V}_{\mathcal{A}} := \{(a,b,x_b,x_a)\ | (a,b,x) \hookleftarrow \mathcal{V};\ x_b \leftarrow x \bmod b;\ x_a \leftarrow x \bmod a\}$.

Consider an unbounded algorithm $\mathcal{A}$ which takes as input a sample (a, b, x_b, x_a^*) of either $\mathcal{U}_{\mathcal{A}}$ or $\mathcal{V}_A$, and outputs a bit β'. $\mathcal{A}$'s goal is to tell whether x_a^* is sampled uniformly at random from $\mathbf{Z}/a\mathbf{Z}$ or if $x_a^* \leftarrow x \bmod a$. We demonstrate that if $\mathcal{A}$ has significant probability in distinguishing both input types, then $\mathcal{C}$ can use $\mathcal{A}$ to distinguish distributions $\mathcal{U}$ and $\mathcal{V}$.

On input (a, b, x), algorithm $\mathcal{C}$ proceeds as follows; it first sets $x_b \leftarrow x \bmod b$; then it samples a bit $\beta^* \hookleftarrow \{0,1\}$, if $\beta^* = 0$, then $\mathcal{C}$ samples $x_a^* \hookleftarrow \mathbf{Z}/a\mathbf{Z}$, else if $\beta^* = 1$, then it sets $x_a^* \leftarrow x$ mod a. Next $\mathcal{C}$ sends (a, b, x_b, x_a^*) to $\mathcal{A}$, which outputs a bit β'. If $\beta = \beta'$, $\mathcal{C}$ outputs 1, else it outputs 0.

If $\mathcal{C}$ gets as input an tuple of $\mathcal{U}$, whatever the value of β^*, x_a^* follows the uniform distribution mod a and is independent of x_b. So $\mathcal{A}$'s success probability in outputting β' equal to β^* is 1/2:

$$\Pr[\mathcal{A}(a, b, x_b, x_a^*) \to \beta^* | (a, b, x) \hookleftarrow \mathcal{U}] = 1/2 \quad \text{and so} \quad \Pr[\mathcal{C}(\mathcal{U}) \to 1] = 1/2.$$

On the other hand if $(a, b, x) \hookleftarrow \mathcal{V}$, then $\mathcal{C}$ outputs 1 if $\mathcal{A}$ guesses the correct bit β^* (when its inputs are either in $\mathcal{U}_{\mathcal{A}}$ or $\mathcal{V}_{\mathcal{A}}$ as expected), i.e.

$$\Pr[\mathcal{C}(\mathcal{V}) \to 1] = \Pr[\mathcal{A} \to \beta^* | (a, b, x) \hookleftarrow \mathcal{V}].$$

And so

$$\begin{aligned} |\Pr[\mathcal{C}(\mathcal{U}) \to 1] - \Pr[\mathcal{C}(\mathcal{V}) \to 1]| &= |\Pr[\mathcal{A} \to \beta^* | (a, b, x) \hookleftarrow \mathcal{V}] - 1/2| \\ &= 1/2 \cdot |\Pr[\mathcal{A}(\mathcal{U}_{\mathcal{A}}) \to 1] - \Pr[\mathcal{A}(\mathcal{V}_A) \to 1]|. \end{aligned}$$

Since $\mathcal{U}$ and $\mathcal{V}$ are ϵ-close, it holds that $|\Pr[\mathcal{C}(\mathcal{U}) \to 1] - \Pr[\mathcal{C}(\mathcal{V}) \to 1]| \leqslant \epsilon$, and so for any $\mathcal{A}$ as above:

$$|\Pr[\mathcal{A}(\mathcal{U}_{\mathcal{A}}) \to 1] - \Pr[\mathcal{A}(\mathcal{V}_A) \to 1]| \leqslant 2\epsilon.$$

Thus the statistical distance between $\mathcal{U}_{\mathcal{A}}$ and $\mathcal{V}_{\mathcal{A}}$ is smaller than 2ϵ, which implies that even given (x mod b), the value of (x mod a) remains at statistical distance 2ϵ of the uniform distribution modulo a, which concludes the proof. □

2.7.3 Technical Tools on Discrete Gaussian Distributions

Though the cryptosystems we present in this thesis are not based on lattice-based cryptography, we will use folded discrete Gaussian distributions in order to sample secret exponents. To ensure these sampled exponents statistically mask confidential information, we use a result from [GPV08] which explains the conditions for a discrete Gaussian distribution over a lattice which is reduced modulo a sublattice to be close to a uniform distribution. We will also need to evaluate the distribution of an inner product when one of the two vectors follows a discrete Gaussian distribution. The following definitions and basic results about Gaussian distributions will thus be useful for our security proofs.

A lattice Λ in dimension n is the set of all integer linear combinations of a set of linearly independent vectors in a Euclidean space of dimension n. This set is called a basis of the lattice. In some of our security proofs we also refer to the minimum $\lambda_1(\Lambda)$ which is the (Euclidean) norm of a shortest non-zero vector of Λ. For a comprehensive background on Euclidean lattices in cryptography the reader may refer to [Ste11], though a strong background on the subject is not required to understand this thesis.

Definition 2.16 (Gaussian Function)**.** For any $\sigma > 0$ define the Gaussian function on $\mathbf{R}^\ell$ centred at $\boldsymbol{c}$ with parameter σ:

$$\forall \boldsymbol{x} \in \mathbf{R}^\ell, \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi||\boldsymbol{x} - \boldsymbol{c}||^2/\sigma^2).$$

If $\sigma = 1$ (resp. $\boldsymbol{c} = \mathbf{0}$), then the subscript σ (resp. $\boldsymbol{c}$) is omitted.

Definition 2.17 (Discrete Gaussians)**.** For any $\boldsymbol{c} \in \mathbf{R}^\ell$, real $\sigma > 0$, and ℓ-dimensional lattice Λ, define the discrete Gaussian distribution of support Λ, standard deviation parameter σ and center $\boldsymbol{c}$ as:

$$\forall \boldsymbol{x} \in \Lambda, \quad \mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})/\rho_{\sigma,\boldsymbol{c}}(\Lambda),$$

where $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$.

Lemma 2.18 is a supporting lemma for proof of Lemma 2.19, which allows to evaluate the distribution of the inner product resulting from a constant vector $\boldsymbol{x}$, and a vector with coordinates sampled from a Gaussian distribution over the lattice $\boldsymbol{x} \cdot \mathbf{Z}$.

Lemma 2.18. Let $\boldsymbol{x} \in \mathbf{R}^\ell \setminus \{\mathbf{0}\}$, $\boldsymbol{c} \in \mathbf{R}^\ell$, $\sigma \in \mathbf{R}$ with $\sigma > 0$ and $\sigma' = \sigma/||\boldsymbol{x}||_2$, $c' = \frac{\langle \boldsymbol{c},\boldsymbol{x}\rangle}{\langle \boldsymbol{x},\boldsymbol{x}\rangle}$. A random variable K is distributed according to $\mathcal{D}_{\mathbf{Z},\sigma',c'}$ if and only if $V := K\boldsymbol{x}$ is distributed according to $\mathcal{D}_{\boldsymbol{x}\mathbf{Z},\sigma,\boldsymbol{c}}$.

Proof. Let $k \in \mathbf{Z}$, and $\boldsymbol{v} := k\boldsymbol{x} \in \boldsymbol{x}\mathbf{Z}$, then

$$\Pr[V = \boldsymbol{v}] = \Pr[V = k\boldsymbol{x}] = \Pr[K = k] = \frac{\rho_{\sigma',c'}(k)}{\rho_{\sigma',c'}(\mathbf{Z})}.$$

As in the proof of [GPV08, Lemma 4.5], one can compute $\rho_{\sigma',c'}(k) = \rho_\sigma((k - c')||\boldsymbol{x}||_2) = \rho_\sigma((k - c')\boldsymbol{x}) = \rho_\sigma(\boldsymbol{v} - c'\boldsymbol{x})$. It holds that $\boldsymbol{u} := c'\boldsymbol{x}$ is the orthogonal projection of $\boldsymbol{c}$ on $\boldsymbol{x}\mathbf{R}$. By Pythagoras' Theorem,

$$||\boldsymbol{v} - \boldsymbol{c}||_2^2 = ||\boldsymbol{v} - \boldsymbol{u}||_2^2 + ||\boldsymbol{c} - \boldsymbol{u}||_2^2.$$

Thus $\rho_\sigma(\boldsymbol{v} - c'\boldsymbol{x}) = \rho_\sigma(||\boldsymbol{v} - \boldsymbol{u}||_2) = \rho_\sigma(||\boldsymbol{v} - \boldsymbol{c}||_2) \times C$ where $C = \exp(\frac{\pi||\boldsymbol{c}-\boldsymbol{u}||_2^2}{\sigma^2})$ is a constant. Therefore we have demonstrated that for $k \in \mathbf{Z}$, $\boldsymbol{v} = k\boldsymbol{x}$, $\rho_{\sigma',c'}(k) = \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{v}) \times C$. And so:

$$\Pr[V = \boldsymbol{v}] = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{v}) \times C}{\sum_{z\in\mathbf{Z}} \rho_{\sigma,\boldsymbol{c}}(z\boldsymbol{x}) \times C} = \mathcal{D}_{\boldsymbol{x}\mathbf{Z},\sigma,\boldsymbol{c}}.$$

□

Lemma 2.19. Let $\boldsymbol{x} \in \mathbf{R}^\ell$ with $\boldsymbol{x} \neq \mathbf{0}$, $\boldsymbol{c} \in \mathbf{R}^\ell$, $\sigma \in \mathbf{R}$ with $\sigma > 0$. Let V be a random variable distributed according to $\mathcal{D}_{\boldsymbol{x}\cdot\mathbf{Z},\sigma,\boldsymbol{c}}$. Then the random variable S defined as $S = \langle \boldsymbol{x}, V\rangle$ is distributed according to $\mathcal{D}_{||\boldsymbol{x}||_2^2\cdot\mathbf{Z},\sigma\cdot||\boldsymbol{x}||_2,\langle \boldsymbol{c},\boldsymbol{x}\rangle}$.

Proof. As V is distributed according to $\mathcal{D}_{\boldsymbol{x}\cdot\mathbf{Z},\sigma,\boldsymbol{c}}$, we have $V = K\boldsymbol{x}$ where K is sampled from $\mathcal{D}_{\mathbf{Z},\sigma/||x||_2,c'}$ where $c' = \frac{\langle \boldsymbol{c},\boldsymbol{x}\rangle}{\langle \boldsymbol{x},\boldsymbol{x}\rangle}$ (*cf.* Lemma 2.18). As a result, one can write $S = K\langle \boldsymbol{x}, \boldsymbol{x}\rangle$, and applying the previous lemma another time in dimension 1, we get that S is distributed according to $\mathcal{D}_{||x||_2^2\cdot\mathbf{Z},\sigma\cdot||\boldsymbol{x}||_2,\langle \boldsymbol{c},\boldsymbol{x}\rangle}$. □

Lemma 2.21 gives sufficient conditions on the standard deviation σ for a Gaussian sample over a lattice Λ_0 to be distributed almost-uniformly modulo a sublattice Λ_0'. To state Lemma 2.21, we first need to define the *smoothing parameter* of a lattice, as defined in [MR04b].

Definition 2.20 ([MR04b])**.** For an n-dimensional lattice Λ, and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ of Λ is defined to be the smallest s such that $\rho_{1/s}(\Lambda^*\backslash\{\mathbf{0}\}) \leqslant \epsilon$, where $\Lambda^* := \{x : \forall \boldsymbol{y} \in \Lambda, \langle \boldsymbol{x}, \boldsymbol{y}\rangle \in \mathbf{Z}\}$ is the dual of Λ.

Lemma 2.21 ([GPV08])**.** Let $\Lambda_0' \subset \Lambda_0 \subset \mathbf{R}^\ell$ be two lattices with the same dimension. Let $0 < \epsilon < 1/2$. Then for any $c \in \mathbf{R}^\ell$ and any $\sigma \geqslant \eta_\epsilon(\Lambda_0')$, the distribution $D_{\Lambda_0,\sigma,c} \mod \Lambda_0'$ is within statistical distance 2ϵ from the uniform distribution over Λ_0/Λ_0'.

2.8 Zero-Knowledge Proofs and Arguments

A zero-knowledge (ZK) proof [GMR85] is an interactive protocol between a prover and a verifier. At the end of the protocol the verifier should be convinced of the truth of a statement (within some probability, called soundness error), while the prover is guaranteed that the verifier learns nothing more than the fact the statement is true. These proofs are widespread in privacy-preserving protocols, in particular in multi-party computation. Zero-knowledge proofs are a particularly useful tool allowing parties to *prove* to each other that they behave honestly, thereby enforcing them to follow the prescribed protocol. In this section we give some high level background on zero-knowledge proofs and arguments as we will devise, in Sections 3.6 and 3.7, zero-knowledge proofs and arguments resulting from the CL framework.

In the following, a *relation generator* $\mathcal{R}$ is an algorithm which on input 1^λ for some positive integer λ, outputs the description of a binary relation R, a language $\mathcal{L}$ and a set of witnesses $\mathcal{W}$. A statement x is in $\mathcal{L}$ if and only if there exists a witness w such that $(x, w) \in \mathsf{R}$. We say that $x \in \mathcal{L}$ is a *true* statement while $x \notin \mathcal{L}$ is a *false* statement. We at times use R to denote both the description $(\mathsf{R}, \mathcal{L}, \mathcal{W})$, and the relation itself.

2.8.1 Zero-Knowledge Proofs

Consider $(\mathsf{R}, \mathcal{L}, \mathcal{W}) \leftarrow \mathcal{R}(1^\lambda)$ for a security parameter $\lambda \in \mathbf{N}$. An (interactive) zero-knowledge proof system (P, V) for language $\mathcal{L}$ is an interactive protocol between two probabilistic algorithms: a prover P and a PT verifier V. Informally P — who detains a witness for a given statement — must convince V that the statement is true without revealing anything other than the truth of this statement to V. Specifically if $(P, V)(x)$ is a random variable representing the output of V at the end of an interaction with P, and $(P, V)(x) = 1$ (resp. $(P, V)(x) = 0$) indicates V accepts (resp. rejects) the proof, the following properties must hold:

- *Completeness*: for any $x \in \mathcal{L}$, it holds that $\Pr[(P, V)(x) = 1] \geqslant 2/3$.
- *Soundness*: for any prover P^* and for any $x \notin \mathcal{L}$, it holds that $\Pr[(P^*, V)(x) = 1] \leqslant 1/3$.
- *Zero-knowledge*: for every PPT verifier V^*, there exists a probabilistic simulator $\mathcal{S}$ running in expected PT such that the following two probability ensembles are indistinguishable (either perfectly, computationally or statistically):

  1. $\{(P, V^*)(x)\}_{x \in \mathcal{L}} \stackrel{\text{def}}{=}$ the output of V^* after interacting with P on common input $x \in \mathcal{L}$, and
  2. $\{\mathcal{S}(x)\}_{x \in \mathcal{L}} \stackrel{\text{def}}{=}$ the output of $\mathcal{S}$ on input $x \in \mathcal{L}$

  The simulator's purpose is to efficiently produce a view of what V^* would see if it were to interact with the prover P. The existence of $\mathcal{S}$ demonstrates clearly that V^* did not need P to gain whatever information it obtained, and thus that V^* learns nothing from the interaction with P. Usually, $\mathcal{S}$ achieves this goal by using V^* as a black-box over which it has complete control. Also, $\mathcal{S}$ can interact with V^*, making V^* believe that it is interacting with P.

The *soundness error* of the protocol is the probability that a cheating prover P^* succeeds in giving a valid proof for some statement $x \notin \mathcal{L}$.

Proving knowledge. While a zero-knowledge proof suffices to convince V of the *existence* of a witness w for the statement, a zero knowledge *proof of knowledge* (ZKPoK) additionally convinces V that the prover *knows* such a witness. This is formalised by the additional requirement that there exists a *knowledge extractor* M, which can extract a valid witness from any prover who succeeds in giving a correct proof with high-enough probability. Specifically, the extractor can interact with P, making P believe that it is interacting with V. The extractor's purpose is to extract the "secret knowledge" that P claims to have. The existence of such an extractor demonstrates P must be honest about knowing the secret.

We use the notation introduced by Camenisch-Stadler [CS97], which conveniently expresses the goals of a ZKPoK:

$$\mathsf{ZKPoK}_x\{(w) : (x, w) \in \mathsf{R}\}.$$

For a full explanation on this model see [Gol01].

Sigma protocols. There exists a general 3-move framework for interactive zero knowledge proofs, called Σ-protocols [Cra97, CD98]. For such protocols there is a clear and fairly straightforward method of proving that the soundness and zero-knowledge properties hold. A Σ protocol is of the following form, where for $(x, w) \in \mathsf{R}$, the statement x is common input to P, V and the witness w for x is private input to P:

1. P sends a message t to V (often referred to as a commitment)
2. V sends a random integer k to P (k is the challenge)
3. P answers with some value u, and V accepts or rejects according to everything it has seen so far (its *view*), i.e. x, t, k, u.

A Σ-protocol satisfies the following properties:

Special soundness Given x and the transcript of any two accepting interactions with input x, denoted t, k, u and t, k', u' where $k \neq k'$, one can efficiently compute w such that $(x, w) \in \mathsf{R}$. Note that this implies soundness of the protocol.

Special honest verifier zero-knowledge There exists a PT simulator $\mathcal{S}$, which on input x and a random challenge k outputs a transcript (t, k, u) such that (1) this transcript would lead V to accept and (2) (t, k, u) follows the same probability distribution as transcripts produced by real executions of the protocol between P and V. This implies the protocol is zero-knowledge with respect to verifiers which follow the protocol, called *honest* verifiers.

2.8.2 Zero-Knowledge Arguments

In the formulation of an interactive zero knowledge proof, the soundness property must hold for any unbounded prover P^*, it hence refers to *all possible ways* of trying to fool a verifier. A fundamental variant of the notion of interactive proofs was introduced by Brassard et al. [BCC88] who relaxed soundness so that it only refers to *feasible ways* of trying to fool the verifier, i.e. the prover P^* must also run in polynomial time. Such protocols are called *zero-knowledge arguments of knowledge* (also referred to as *computationally convincing proofs of knowledge*). Hence a zero-knowledge argument should satisfy properties of *completeness* and *zero-knowledge* as in Section 2.8.1, however the soundness property is replaced by *computational soundness.*

As in the case of proofs, a zero-knowledge argument convinces the verifier V of the existence of a witness for a given statement, whereas a *zero-knowledge argument of knowledge* (ZKAoK) additionally convinces V that the prover *knows* such a witness. We will also use the notation of [CS97] to express the goals of a zero knowledge argument of knowledge:

$$\mathsf{ZKAoK}_x\{(w) : (x, w) \in \mathsf{R}\}.$$

Since we now consider *polynomial time* provers, the knowledge extractor M must now also run in polynomial time. Indeed recall that M interacts with P so as to extract P's "secret knowledge". The existence of such an extractor, *running in polynomial time*, demonstrates that a *polynomial time* P must be honest about knowing the secret.

As the notion of zero-knowledge for arguments is identical to that of zero-knowledge proofs, let us, for the time being forget this property, and focus on arguments of knowledge, so as to lighten terminology.

Framework adopted to prove soundness. For the ZKAoK which we build in Section 3.7, we reduce an argument's soundness to the hardness of some computational problem. As observed by Damgård and Fujisaki [DF02], Bellare and Goldreich's definition of soundness for computationally convincing proofs of knowledge [BG93] requires that, for a given relation R, no cheating prover P^* can falsely prove that it knows a witness w for some statement x; this must hold for *all* large enough instances x, which are chosen by the prover.

Now since soundness only holds under *computational* assumptions, prior to its interaction with V, P^* could, with significant probability (but in arbitrary time), compute some trapdoor information associated to the language $\mathscr{L}$, the set of witnesses $\mathscr{W}$ and the relation R. This would allow it to successfully answer V's challenge, even without knowing the witness for its input statement (jumping ahead, in our protocols of Section 3.7, P^* could compute the structure of the class group, given which computing roots would no longer be hard).

This is why we use a relation generator $\mathscr{R}$ (as introduced in [DF02]). This relation generator produces the challenge of the underlying computational problem which will be given as input to P^*. Then P^* produces a statement x, used as input (along with the relation R, and the public parameters received from $\mathscr{R}$) for the interactive proof it conducts with V (this interactive proof is ran by the machine called P_{view} below). Now P^* wins if, for such an x, the standard soundness requirement fails. The protocol will be considered computationally sound if any polynomial time P^*'s probability of winning is upper bound by some small function of the security parameter.

Let us now formally provide terminology and definitions relating to arguments of knowledge as defined in [DF02]. Consider $(\mathsf{R}, \mathscr{L}, \mathscr{W}) \leftarrow \mathscr{R}(1^\lambda)$ for a security parameter $\lambda \in \mathbf{N}$. The prover P gets as input the relation R, outputs a statement x and runs the interactive proof with a verifier V using (R, x) as common input.

Consider P's view view after outputting x. This view contains all inputs, messages exchanged and random coins; consequently the statement x is determined by view. From P, one can define a machine P_{view} which starts in the state P is in after having seen view view and having produced x. P_{view} then conducts the protocol with V following P's algorithm. We denote:

$\mathsf{acc}_{\mathsf{view},P}$ the probability that P makes V accept, conditioned on view.

Intuitively the *knowledge error function* κ is:

κ is the probability that P can make V accept without knowing a witness w s.t. $(x, w) \in \mathsf{R}$.

An *extractor* is a machine M that gets the relation R and a statement x as an input, has black-box access to P_{view} for some view consistent with x and computes a witness w satisfying $(x, w) \in \mathsf{R}$.

Intuitively, if a cheating prover P^*, which does not know the witness w for x, is capable of making V accept with probability greater than κ, then the extractor M should be able to *efficiently* extract a witness from P^*. If so M is *successful* in its extraction task. However, if M cannot extract a witness in reasonable time, it *fails*. We formally define the notion of failure in Definition 2.22.

Definition 2.22 ([DF02])**.** Consider a positive integer λ, a knowledge error function $\kappa : \mathbf{N} \to [0, 1]$, a PT cheating prover P^*, an extractor M and a polynomial p. We say that M fails on view view if $\mathsf{acc}_{\mathsf{view},P^*}(\lambda) > \kappa(\lambda)$, and the expected running time of M using P^*_{view} as oracle, is greater than $\frac{p(\lambda)}{\mathsf{acc}_{\mathsf{view},P^*}(\lambda)-\kappa(\lambda)}$.

Definition 2.23. Let λ be a positive integer. Let $\mathscr{R}$ be a PPT relation generator, and consider a proof system (P, V), a cheating prover P^*, a knowledge extractor M, a polynomial p and a knowledge error function $\kappa : \mathbf{N} \to [0, 1]$. Consider the following experiment with input λ:

1. Sample $(\mathsf{R}, \mathscr{L}, \mathscr{W}) \leftarrow \mathscr{R}(1^\lambda)$.
2. Give the relation R as input to P^*; which outputs a statement x.
3. From steps 1 and 2, define the view view.
4. Run the extractor M on input the relation R and the statement x, with black-box access to $\mathsf{acc}_{\mathsf{view},P^*}$ for view view.

The advantage of P^*, denoted $\mathsf{Adv}^{\kappa,M,p}_{P^*}(\lambda)$, is the probability (taken over the random coins of $\mathscr{R}$, P^*) that M fails in this experiment.

Definition 2.24. Let λ be a positive integer and let $(\mathsf{R}, \mathscr{L}, \mathscr{W}) \leftarrow \mathscr{R}(1^\lambda)$. The proof system (P, V) is an *argument of knowledge* for $\mathscr{R}$, with knowledge error κ, if the following hold:

- *Completeness:* If an honest prover P, on input R, produces $(x, w) \in \mathsf{R}$, sends x to V and conducts the protocol with V, then V accepts with overwhelming probability in λ.
- *Computational Soundness:* There exists a polynomial p and an extractor M, such that for any PT prover P^*, $\mathsf{Adv}^{\kappa,M,p}_{P^*}(\lambda) = \mathsf{negl}(\lambda)$.

If the zero-knowledge property of Section 2.8.1 also holds for (P, V), then it is said to be a *zero-knowledge* argument of knowledge.

2.8.3 Groups of Unknown Order

To understand the issues which arise in groups of unknown order, let us first consider the well known Σ-protocol due to Schnorr [Sch90]. This protocol is a proof of knowledge of a discrete logarithm in groups of known prime order. Precisely, consider a group G generated by g of prime order q, $w \in \mathbf{Z}/q\mathbf{Z}$ is a witness for $x \in G$ if $x = g^w$. The prover P knows a secret $w \in \mathbf{Z}/q\mathbf{Z}$ and wants to convince V that it knows the DL of $x := g^w \in G$. To prove knowledge of w, P performs with V the protocol described Fig. 2.2.

Input : w and (x, G, g, q)		Input : (x, G, g, q)
`Repeat` ℓ `times`		
$r \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$t \leftarrow g^r$	$\xrightarrow{t}$	
	$\xleftarrow{k}$	$k \hookleftarrow \mathcal{C}$
$u \leftarrow r + kw \bmod q$	$\xrightarrow{u}$	`If` $x^k t = g^u \in G$ `accept, else reject.`

Figure 2.2: The Schnorr ZKPoK for knowldge of w such that $x = g^w$.

Elementary rounds of a Σ-protocol can be repeated sequentially; we denote by ℓ the number of repetitions. The security analysis of the scheme shows that even a dishonest V^* cannot learn any additional information about w, since r perfectly masks the secret value, as long as $|\mathcal{C}|$ and ℓ are polynomial in the security parameter; so the proof is perfectly zero-knowledge. Furthermore, if V accepts with probability substantially greater than $1/|\mathcal{C}|$, P must know the discrete logarithm w of x; this yields a knowledge error of $(1/|\mathcal{C}|)^\ell$, which proves soundness. Indeed, assume P is accepted with probability significantly greater than $1/|\mathcal{C}|$, then there must exist, for a given commitment t, two different challenges k and k' for which P can get V to accept. Denoting u and u' the corresponding answers of P, since both proofs were accepted, it holds that $t = g^u x^{-k} = g^{u'} x^{-k'}$, such that P could compute $w = (u' - u)(k' - k)^{-1} \bmod q$; this proves special soundness.

Now observe that if q were not prime, one has no guarantee that $(k' - k)$ is invertible mod q unless $\mathcal{C} = \{0, 1\}$. Thus if one adapts Σ-protocols to groups of unknown order in a straightforward way, one must use binary challenges in order to be able to extract w from two accepting transcripts. This implies that one round of the protocol has a knowledge error of $1/2$ and must be repeated sequentially sufficiently many times to achieve a reasonably small knowledge error. These repetitions result in protocols which are an order of magnitude less efficient. It is thus desirable to design (computationally) convincing proofs of knowledge for groups of unknown order with *large challenge spaces*.

Dealing with groups of unknown order makes such proofs typically harder to analyse, one can use techniques developed for zero-knowledge proofs over the integers [Lip03]. The first efficient solution for proofs in groups of unknown order was given in [FO97] and was later corrected by Damgård and Fujisaki [DF02]. Since their work, other variants overcoming some of their restrictions have been put forth. In particular, Camenisch et al. [CKY09] provided a framework to identify input distributions, and initial constraints under which protocols can be employed and satisfy the standard properties of ZK proofs. We note that the CL framework (defined in Section 3.1) is not compatible with the framework of [CKY09], since group elements are not efficiently recognisable (a property required for their notion of a safeguard group). Consequently in our protocols of Section 3.7 we build upon the techniques of [DF02] for our arguments of knowledge.

The high level idea is the following: assume an extractor M running P has obtained transcripts (t, k, u) and (t, k', u') as above (this occurs independently of whether the group order is known or not). We say that the pair of transcripts is *good*, if M can extract w from them; *i.e.,* for groups of unknown order if $(k' - k)$ divides $(u' - u)$ in $\mathbf{Z}$. Conversely, a set of values which does not allow to extract w is *bad*, but will allow to break some computational assumption with significant probability. Then one demonstrates that if there exists a prover P^* which can cheat with significant probability, but that M cannot extract a witness from P^*, then M must obtain a set of bad values with significant probability, which contradicts the hardness of the underlying assumption.

Chapter 3

Enriching the CL framework

The framework adopted throughout this thesis – called the CL framework – was introduced by Castagnos and Laguillaumie in [CL15]. As we shall see, its homomorphic properties, and its unusual ability to encode confidential information in a prime order group, render it particularly interesting. In this chapter, we devise simple building blocks from the CL framework. These tools will be the foundations providing support for the more complex cryptosystems of Chapters 4 and 5. In Section 3.1 we first define the CL framework, and describe how one instantiates it from ideal class groups of an imaginary quadratic field. Next in Section 3.2, we present the mathematical problems (arising from this framework) which will underlie the security of our cryptosystems. Two of these assumptions, which we call HSM-CL and DDH-f, are new to our work, while the DDH-CL assumption is the original assumption used in [CL15] (the difficulty of breaking these problems is compared in Section 3.5.4). At this point, we will be ready to devise, in Section 3.3, projective hash functions from the CL framework.

Projective hash functions (PHF) were first introduced by Cramer and Shoup in [CS02], we build upon their framework, and consider the more general setting where group elements may not be efficiently recognisable. We define a number of properties for PHFs, namely homomorphic properties, essential for the correctness of our upcoming constructions, as well as properties required to ensure security. In particular, we recall the standard smoothness property for PHFs, but we also define a new property, called decomposability, which will be extremely useful in *all our applications*, as it allows a clear separation between the information which can be leaked to an adversary without having any harmful consequences on the security of the cryptosystem, and that which *must stay hidden.* These PHFs will be the main building blocks used in Chapters 4 and 5.

Running examples. To help understand technical notions, all of the definitions and properties regarding PHFs will be illustrated with three running examples. These will also be used through Chapter 4.

- The first arises from traditional DDH in finite fields. This allows us to illustrate all our concepts in a framework with which most readers are familiar. Moreover, as we shall see in Chapter 4, many of our generic constructions, when instantiated from DDH, result in DDH based schemes which pre-date our work (e.g. the IPFE schemes from DDH of [ALS16]), while our proofs yield the best known security bounds. This demonstrates that our approach truly provides a unified view of a range of cryptographic protocols without sacrificing efficiency.
- Our other two running examples result from our previously defined assumptions in the CL framework: HSM-CL and DDH-f. We note that these examples are notably less intuitive, and proving they attain the aforementioned properties tends to be more involved than for the DDH assumption. These complications are mainly due to the fact that in the CL

framework, the group order is unknown, and elements may not be efficiently recognisable.

We note that one can also build PHFs from the DCR assumption, which could be made to satisfy all our properties. Such an instantiation would very much resemble our HSM-CL based PHFs, the main difference being that confidential information would be encoded in a subgroup of order an RSA integer N, whereas in the CL framework confidential information is encoded modulo a prime q. Hence when we use the primality of q, one would rely on the hardness of factoring N. Instantiating constructions from DCR based PHFs would lead to existing DCR based cryptosystems (e.g. the (simplified ind-cpa-secure) encryption scheme of [CS03], or the Paillier-based IPFE schemes of [ALS16]).

In Section 3.4 we recall a folklore generic construction allowing to build linearly homomorphic ind-cpa-secure public key encryption from PHFs which satisfy some homomorphic properties. This generic construction can be seen as a simpler version of that of [CS02], which allows to build ind-cca-secure public key encryption from PHFs. Then in Section 3.5, we present three linearly homomorphic encryption schemes from the CL framework, two of which are new schemes we introduce, and are direct applications of the aforementioned generic construction with the HSM-CL and DDH-f based PHFs of our running examples.

Finally, in a different vein, in Sections 3.6 and 3.7, we devise zero knowledge proofs and arguments of knowledge for the CL framework. These will be essential to attain security against malicious adversaries for our distributed signatures of Chapter 5, as they are a means of forcing parties to follow the prescribed protocol. In particular, we provide proofs that ciphertexts for the aforementioned encryption schemes are well formed and variants thereof.

Summary of contributions. To sum up, our contributions in this chapter are new assumptions for the CL framework, corresponding projective hash functions, two new linearly homomorphic encryption schemes with prime order message space and zero knowledge proofs and arguments for these schemes.

Related publications and submissions. Most of my personal contributions in this chapter can be found in:

- [CLT18a] For the introduction of the hardness assumptions HSM-CL and DDH-f in the CL framework, along with resulting linearly homomorphic public key encryption schemes.
- [CCL$^+$19] For the HSM-CL based projective hash function and the zero-knowledge proof of knowledge for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ of Section 3.6.
- [CCL$^+$20] For the lcm trick of Section 3.6, and the zero-knowledge arguments of knowledge of Section 3.7.
- [CLT20] (submission) For the definition of new properties for projective hash functions, along with running examples 1 and 2 from DDH and HSM-CL.

3.1 The CL Framework

In [CL15], Castagnos and Laguillaumie introduced the framework of a *group with an easy DL subgroup*: a cyclic group G where the DDH assumption holds together with a subgroup F of G where the discrete logarithm problem is easy. Within this framework, they designed a linearly homomorphic variant of ElGamal (*cf.* Section 3.5.1). Moreover, they gave an instantiation in class groups of an imaginary quadratic field which allows to perform linear operations over encrypted data modulo a prime q. In this section, we recall the framework of [CL15], denoted the CL framework throughout the rest of this thesis. To start with, we explicitly define the generator $\mathsf{Gen}_{\mathsf{CL}}$ used in the framework of a group with an easy DL subgroup introduced in [CL15].

3.1.1 Definition of the CL Framework

Definition 3.1. Consider a positive integer λ. Let $\mathsf{Gen}_{\mathsf{CL}}$ be a pair of algorithms $(\mathsf{Gen}, \mathsf{Solve})$. The Gen algorithm is a group generator which takes as inputs the parameter λ and a prime q and outputs a tuple $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$. The set

$$(\widehat{G}, \cdot) \text{ is a finite abelian group of order } \widehat{n} := q \cdot \widehat{s}$$

where the bitsize of $\widehat{s}$ is a function of λ and $\gcd(q, \widehat{s}) = 1$. Algorithm Gen only outputs an upper bound $\tilde{s}$ of $\widehat{s}$. It is also required that one can efficiently recognise valid encodings of elements in $\widehat{G}$. The set

$$(F, \cdot) \text{ is the unique cyclic subgroup of } \widehat{G} \text{ of order } q \text{ , generated by } f.$$

The set

$$(G, \cdot) \text{ is a cyclic subgroup of } \widehat{G} \text{ of order } n := q \cdot s \text{ where } s \text{ divides } \widehat{s}.$$

By construction $F \subset G$, and, denoting:

$$G^q := \{x^q, x \in G\}$$

the subgroup of order s of G, it holds that:

$$G \simeq G^q \times F.$$

The algorithm Gen outputs f, g_q and $g := f \cdot g_q$ which are respective generators of F, G^q and G. Moreover, the DL problem is easy in F, which means that the Solve algorithm is a DPT algorithm that solves the discrete logarithm problem in F:

$$\begin{aligned}\Pr\big[x = x^\star : (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q), x \hookleftarrow \mathbf{Z}/q\mathbf{Z}, X \leftarrow f^x,& \\ x^\star \leftarrow \mathsf{Solve}(q, \tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q, X)\big] = 1.&\end{aligned}$$

Remark. We note that in all our applications it is essential that the order of the group G^q is unknown for security to hold. Indeed, if one knew the order s of G^q, given an element $u \in G$, one could raise u to the power s, thereby obtaining $u^s \in F$ of which one can efficiently compute the discrete logarithm in base f. Knowing s breaks all of our assumptions and hence our resulting applications (*cf.* [CL15, Lemmas 1 and 2].

Remark. In Definition 3.1 there are a few modifications compared to the original definition of [CL15]. Namely we take as input the prime q instead of having Gen generate it, and we output the group $\widehat{G}$ from which the group G with an easy DL subgroup F is produced. In practice, with the concrete instantiation from class groups described in Section 3.1.2, this is a just a matter of rewriting. We note that it is easy to recognise valid encodings of $\widehat{G}$ while it will be not so for elements of $G \subset \widehat{G}$. This is an important difference with Paillier's encryption, where one can efficiently tell if an element is in $\mathbf{Z}/N^2\mathbf{Z}$.

Notation 3.2. We denote $\widehat{G}^q$ the subgroup of all q-th powers in $\widehat{G}$. Since q and $\widehat{s}$ are co-prime it holds that:

$$\widehat{G} \simeq \widehat{G}^q \times F.$$

The exponent of a finite Abelian group is the least common multiple of the orders of its elements. We denote ϖ the group exponent of $\widehat{G}^q$. As such, the order of any $x \in \widehat{G}^q$ divides ϖ.

3.1.2 Instantiation from Class Group Cryptography

Our instantiation of the CL framework results from class groups of orders of imaginary quadratic fields. We describe this instantiation of algorithm Gen from Definition 3.1, as detailed in Fig. 3.1 (adapted from [CL15]). We reuse the notations and concepts presented in Section 2.6.

Consider positive integers λ and μ such that $\mu \geqslant \lambda$. According to Table 2.1, and given the security parameter λ, we know the required size of a discriminant Δ_K so that the problem of computing the class number (or equivalently computing discrete logarithms in the class group) is intractable; let us denote $\eta(\lambda)$ this size of Δ_K. We further require $2\mu + 2 < \eta(\lambda)$, the motivation for this will become clear when we define the subgroup F. Now given λ and a μ-bit prime q, we sample an $\eta(\lambda) - \mu$ bit prime $\tilde{q}$, such that $q \cdot \tilde{q} \equiv -1 \pmod 4$ and $(q/\tilde{q}) = -1$. We then construct a fundamental discriminant $\Delta_K := -q \cdot \tilde{q}$ and associated class group $Cl(\mathcal{O}_{\Delta_K})$. As noted in Section 2.6.2, this ensures that the even part of the class group $Cl(\mathcal{O}_{\Delta_K})$ is isomorphic to $\mathbf{Z}/2\mathbf{Z}$, and that the odd part is the group of squares of $Cl(\mathcal{O}_{\Delta_K})$. We then consider the non-maximal order of discriminant $\Delta_q := q^2 \cdot \Delta_K$ and its class group $Cl(\mathcal{O}_{\Delta_q})$ of order $h(\mathcal{O}_{\Delta_q}) = q \cdot h(\mathcal{O}_{\Delta_K})$. Given the factorisation of the discriminant, recognising squares in $Cl(\mathcal{O}_{\Delta_q})$ can be done efficiently (*cf.* [Lag80]). This allows us to define the efficiently recognisable group:

$$(\widehat{G}, \cdot) \text{ as the subgroup of squares of } Cl(\mathcal{O}_{\Delta_q}).$$

This implies that $\widehat{n} := h(\mathcal{O}_{\Delta_q})/2$ and $\widehat{s} := h(\mathcal{O}_{\Delta_K})/2$. For the upper bound $\tilde{s}$ of $\widehat{s}$ we use the upper bound on the class number of $\mathcal{O}_{\Delta_K}$ (Eq. (2.1)) so that

$$\tilde{s} = \left\lceil \frac{1}{2\pi} \log |\Delta_K| \sqrt{|\Delta_K|} \right\rceil.$$

For the subgroup F, we use the kernel of the surjection $\bar{\phi}_q : Cl(\mathcal{O}_{\Delta_q}) \twoheadrightarrow Cl(\mathcal{O}_{\Delta_K})$, $[\mathfrak{a}] \mapsto [\mathfrak{a}\mathcal{O}_{\Delta_K}]$. This kernel is generated by f, where f is the class of (q^2, q), which is a reduced ideal since we required $2\mu + 2 < \eta(\lambda)$. In Section 2.6.1 on page 49 we detailed how one can efficiently compute discrete logarithms in base f, which explains how we instantiate the algorithm Solve.

As the bit size of q has at least λ bits, where λ is the security parameter, q is prime to $\widehat{s}$ except with negligible probability (following the Cohen-Lenstra heuristics).

Now in order to build the deterministic generator g_q, we first construct a small prime r such that Δ_K is a square modulo r. We then consider $\mathfrak{r}$ an ideal lying above r (as in [HJPT98, Section 3.1]) and the class $[\mathfrak{r}^2] \in Cl(\mathcal{O}_{\Delta_K})$. We assume that this class will be of order s, an integer of the same order of magnitude as $\widehat{s}$. We then use the lifting map $Cl(\mathcal{O}_{\Delta_K}) \to Cl(\mathcal{O}_{\Delta_q})$, $[\mathfrak{x}] \mapsto [\phi_q^{-1}(\mathfrak{x})]^q$ where $\mathfrak{x}$ is a representative ideal prime to q and ϕ_q^{-1} is defined in Section 2.6.1, page 48. One can show that this map is well defined, as the kernel of $\bar{\phi}_q$ has order q, and is injective, as $\gcd(q, s) = 1$. Then g_q has order s, where:

$$g_q := [\phi_q^{-1}(\mathfrak{r}^2)]^q.$$

Finally we let $g := g_q f$, and denote G the subgroup generated by g of order $n := s \cdot q$.

3.1.3 Instantiating Distributions

For the assumptions introduced in Section 3.2, we will need to sample elements from distributions $\mathcal{D}_q$ and $\mathcal{D}$ such that the distributions $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$ and $\{g^x, x \hookleftarrow \mathcal{D}\}$ are statistically close to the uniform distribution in G^q and G respectively. For security to hold against active adversaries in our upcoming constructions, we will further need to sample elements from distributions $\widehat{\mathcal{D}}_q$ and $\widehat{\mathcal{D}}$, inducing distributions $\{x \bmod \varpi, x \hookleftarrow \widehat{\mathcal{D}}_q\}$ and $\{x \bmod q\varpi, x \hookleftarrow \widehat{\mathcal{D}}\}$

```
$\mathsf{Gen}(1^\lambda, q)$

1. Let $\mu$ be the bit size of $q$. Pick $\tilde{q}$ a $\eta(\lambda) - \mu$ bits prime
   such that $q\tilde{q} \equiv -1 \pmod 4$ and $(q/\tilde{q}) = -1$.
2. $\Delta_K \leftarrow -q\tilde{q}$, $\Delta_q \leftarrow q^2\Delta_K$ and $\widehat{G} \leftarrow Cl(\mathcal{O}_{\Delta_q})$
3. $f \leftarrow [(q^2, q)]$ in $Cl(\mathcal{O}_{\Delta_q})$ and $F := \langle f \rangle$
4. $\tilde{s} \leftarrow \lceil \frac{1}{2\pi} \log |\Delta_K| \sqrt{|\Delta_K|} \rceil$
5. Let $r$ be a small prime, with $r \neq q$ and $\left(\frac{\Delta_K}{r}\right) = 1$, set $\mathfrak{r}$
   an ideal lying above $r$.
6. Set $g_q \leftarrow [\phi_q^{-1}(\mathfrak{r}^2)]^q$ in $Cl(\mathcal{O}_{\Delta_q})$ and $G^q \leftarrow \langle g_q \rangle$
7. Set $g \leftarrow g_q \cdot f$ and $G \leftarrow \langle g \rangle$
8. Return $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$
```

Figure 3.1: Group generator Gen

which are statistically close to the uniform distribution in $\mathbf{Z}/\varpi\mathbf{Z}$ and $\mathbf{Z}/q\varpi\mathbf{Z}$ respectively. Indeed, since one can only efficiently recognise valid encodings of elements in $\widehat{G}$ (but not those of G), a malicious adversary $\mathcal{A}$ could run any of our schemes inputting elements in $\widehat{G}^q$ when they should be in G^q. Such malicious behaviour cannot be efficiently detected, so $\mathcal{A}$ could learn information on the sampled exponents modulo the group exponent ϖ of $\widehat{G}^q$.

Requiring that the induced distributions be *statistically close* to uniform in the aforementioned groups allows for more flexibility in our upcoming proofs, which is of interest, since the assumptions we define in Section 3.2 do not depend on the choice of the distribution.

Since the order s of G^q, and the group exponent ϖ of $\widehat{G}^q$ are unknown, we use the upper bound $\tilde{s}$ output by Gen, and the lemmas of Section 2.7.1, in order to instantiate the distributions $\mathcal{D}_q$, $\mathcal{D}$, $\widehat{\mathcal{D}}_q$ and $\widehat{\mathcal{D}}$. In fact, since $\tilde{s}$ is an upper bound for both s and $\widehat{s}$ (where ϖ divides $\widehat{s}$), we simply let $\widehat{\mathcal{D}} := \mathcal{D}$ and $\widehat{\mathcal{D}}_q := \mathcal{D}_q$. We instantiate $\mathcal{D}$ and $\mathcal{D}_q$ thanks to the following lemma:

Lemma 3.3. Consider distributions $\mathcal{D}_q$, $\mathcal{D}$, $\widehat{\mathcal{D}}_q$ and $\widehat{\mathcal{D}}$ such that the distributions $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$ and $\{g^x, x \hookleftarrow \mathcal{D}\}$ are δ-close to the uniform distribution in G^q and G respectively; and distributions $\{x \bmod \varpi, x \hookleftarrow \widehat{\mathcal{D}}_q\}$ and $\{x \bmod q\varpi, x \hookleftarrow \widehat{\mathcal{D}}\}$ are δ-close to the uniform distribution in $\mathbf{Z}/\varpi\mathbf{Z}$ and $\mathbf{Z}/q\varpi\mathbf{Z}$ respectively. These distributions can be implemented from the output of Gen as follows:

1. One can set $\mathcal{D}_q := \widehat{\mathcal{D}}_q$ and $\mathcal{D} := \widehat{\mathcal{D}}$.
2. One can choose $\widehat{\mathcal{D}}$ to be the uniform distribution on $\{0, \ldots, \tilde{s}q/(4\delta) - 1\}$.
3. Alternatively, choosing $\widehat{\mathcal{D}} = \mathcal{D}_{\mathbf{Z},\sigma}$ with $\sigma = \tilde{s}q\sqrt{|\log_2(\delta)|}$ allows for more efficient constructions as the sampled elements will tend to be smaller.
4. Likewise, one can choose $\widehat{\mathcal{D}}_q = \mathcal{D}_{\mathbf{Z},\sigma'}$ with $\sigma' = \tilde{s}\sqrt{|\log_2(\delta)|}$.
5. One can also, less efficiently, define $\widehat{\mathcal{D}}_q = \widehat{\mathcal{D}}$.

6. One can induce a distribution δ-close to uniform in G from $\mathcal{D}_q$ and the uniform distribution modulo q: the distribution $\{g_q^x \cdot f^a, x \hookleftarrow \mathcal{D}_q, a \hookleftarrow \mathbf{Z}/q\mathbf{Z}\}$ is statistically close to the uniform distribution in G.

Proof. The first item is a consequence of Lemma 2.13, since the order s of G^q divides ϖ, and the order n of G divides $q\varpi$.
Item 2 is a consequence of Lemma 2.11, which tells us that $\{x \bmod q\varpi, x \hookleftarrow \widehat{\mathcal{D}}\}$ is at distance less than $4\delta\varpi q/(4\tilde{s}q) \leqslant \delta$ from $\mathcal{U}(\mathbf{Z}/\varpi q\mathbf{Z})$.
Item 3 follows from Lemma 2.12: the choice of $\mathcal{D}_{\mathbf{Z},\sigma}$ with

$$\sigma > \varpi q\sqrt{|\log_2(\delta)|} > \varpi q\sqrt{\ln(2(1+2\delta^{-1}))\pi^{-1}}$$

induces a distribution δ-close to $\mathcal{U}(\mathbf{Z}/\varpi q\mathbf{Z})$. This choice therefore trades a factor $1/2 \cdot \delta^{-1}$ for a factor $\sqrt{|\log_2(\delta)|}$ compared to the previous choice. This also proves Item 4.

Since $q\varpi$ divides ϖ, $\widehat{\mathcal{D}}_q$ can be defined from $\widehat{\mathcal{D}}$ as in Item 5: the distribution $\{x \bmod \varpi, x \hookleftarrow \widehat{\mathcal{D}}_q\}$ is statistically close to $\mathcal{U}(\mathbf{Z}/\varpi\mathbf{Z})$. This choice makes sense if e.g. one does not know if a given element u is in G or in G^q. Then if one samples exponents from $\widehat{\mathcal{D}}$, the induced distribution will be uniform, be it over G^q or over G.

Item 6 follows from the fact $G \simeq F \times G^q$ and Lemma 2.14. □

3.2 Hard Problems in the CL Framework

In this section, we first generalise the DCR problem to fit the CL framework with a hard subgroup membership problem. We also present an adaptation of the standard DDH problem to the CL framework before introducing a weaker DDH-like problem, which will be better suited for our constructions.

3.2.1 Hard Subgroup Membership Problem

We here introduce the definition of a hard subgroup membership problem within a group with an easy DL subgroup (HSM-CL). The HSM-CL assumption is closely related to Paillier's DCR assumption. Such hard subgroup membership problems are based on a long line of assumptions on the hardness of distinguishing powers in groups. In short, DCR and HSM-CL are essentially the same assumption but in different groups, hence there is no direct reduction between them.

We are the first to consider such an assumption within class groups. Recall that, as per Definition 3.1, one has $G \simeq F \times G^q$. The assumption is that it is hard to distinguish the elements of G^q in G.

Definition 3.4 (HSM-CL assumption)**.** Let λ be a positive integer and $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator for a group with an easy DL subgroup. Let $\mathcal{D}$ (resp. $\mathcal{D}_q$) be a distribution over the integers such that the distribution $\{g^x, x \hookleftarrow \mathcal{D}\}$ (resp. $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$) is at distance less than $\delta(\lambda)$ from the uniform distribution in G (resp. in G^q), for some $\delta(\lambda) = \mathsf{negl}(\lambda)$. Let $\mathcal{A}$ be an adversary for the HSM-CL problem, its advantage is defined as:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{hsm\text{-}cl}}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Big| \Pr\big[b = b^\star : \mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q),\ |q| \geqslant \lambda, \\ x \hookleftarrow \mathcal{D}, x' \hookleftarrow \mathcal{D}_q, b \hookleftarrow \{0,1\}, Z_0 \leftarrow g^x, Z_1 \leftarrow g_q^{x'}, \\ b^\star \leftarrow \mathcal{A}(q, \mathsf{pp}_{\mathsf{CL}}, Z_b, \mathsf{Solve}(.))\big] - 1/2 \Big|.\end{aligned}$$

The HSM-CL problem is $\delta_{\mathsf{hsm\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{hsm\text{-}cl}}(\lambda) \leqslant \delta_{\mathsf{hsm\text{-}cl}}(\lambda)$. We say the HSM-CL assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ (or the HSM-CL problem is hard for $\mathsf{Gen}_{\mathsf{CL}}$), if the HSM-CL problem is $\delta_{\mathsf{hsm\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\delta_{\mathsf{hsm\text{-}cl}}(\lambda) = \mathsf{negl}(\lambda)$.

3.2.2 Decision Diffie Hellman

We here present the DDH-CL problem, which is the original problem used in [CL15] to underlie security of their linearly homomorphic PKE scheme (recalled in Section 3.5.1). The DDH-CL assumption is essentially the DDH assumption in the group G output by Gen.

Definition 3.5 (DDH-CL assumption)**.** Let λ be a positive integer and $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator for a group with an easy DL subgroup. Let $\mathcal{D}$ be a distribution over the integers such that the distribution $\{g^x, x \hookleftarrow \mathcal{D}\}$ is at distance less than $\delta(\lambda)$ from the uniform distribution in G, for some $\delta(\lambda) = \mathsf{negl}(\lambda)$. Let $\mathcal{A}$ be an adversary for the DDH-CL problem, its advantage is defined as:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh\text{-}cl}}(\lambda) \overset{\text{def}}{=} \Big| \Pr\big[b = b^\star : \mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q),\ |q| \geqslant \lambda,& \\ x, y, z \hookleftarrow \mathcal{D}, X = g^x, Y = g^y, b \hookleftarrow \{0,1\}, Z_0 = g^{xy}, Z_1 = g^z,& \\ b^\star \leftarrow \mathcal{A}(q, \mathsf{pp}_{\mathsf{CL}}, X, Y, Z_b, \mathsf{Solve}(.))\big] - 1/2 \Big|.&\end{aligned}$$

The DDH-CL problem is $\delta_{\mathsf{ddh\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh\text{-}cl}}(\lambda) \leqslant \delta_{\mathsf{ddh\text{-}cl}}(\lambda)$. We say the DDH-CL assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ (or the DDH-CL problem is hard for $\mathsf{Gen}_{\mathsf{CL}}$), if the DDH-CL problem is $\delta_{\mathsf{ddh\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\delta_{\mathsf{ddh\text{-}cl}}(\lambda) = \mathsf{negl}(\lambda)$.

3.2.3 Extended Decision Diffie Hellman in F

Finally, we introduce a new assumption called DDH-f. Roughly speaking, DDH-f states that it is hard to distinguish the distributions

$$\{(g^x, g^y, g^{xy}), x, y \hookleftarrow \mathcal{D}\} \text{ and } \{(g^x, g^y, g^{xy} f^u), x, y \hookleftarrow \mathcal{D}, u \hookleftarrow \mathbf{Z}/q\mathbf{Z}\},$$

where $\mathcal{D}$ induces distributions statistically close to the uniform in G and F. In other words, as $g = f \cdot g_q$, we have on the left, a Diffie-Hellman triplet in G, and on the right, a triplet whose components in G^q form a Diffie-Hellman triplet, and whose components in F form a random triplet: (f^x, f^y, f^{xy+u}). We note that DDH-f can be seen as an instance of the Extended-DDH (EDDH) problem defined by Hemenway and Ostrovsky in [HO12]. They demonstrate that the hardness of the quadratic residuosity (QR) problem (which is to decide, given integers a and N, if a is a quadratic residue modulo N), and that of the DCR problem, imply the hardness of two different instantiations of EDDH. Our implication from HSM-CL to DDH-f somewhat generalises their proof since DDH-f is more generic than either of the hardness assumptions obtained from their reductions.

We will see in Section 3.5.1 that the security of the DDH-CL-based encryption scheme of [CL15] is *equivalent* to the hardness of DDH-f. We further demonstrate in Section 3.5.4 that the DDH-f assumption is *weaker* than both the DDH-CL assumption and the HSM-CL assumption.

Definition 3.6 (DDH-f assumption)**.** Let λ be a positive integer and let $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator for a group with an easy DL subgroup. Let $\mathcal{D}$ be a distribution over the integers such that the distribution $\{g^x, x \hookleftarrow \mathcal{D}\}$ is at distance less than $\delta(\lambda)$ from the uniform distribution in G, for some $\delta(\lambda) = \mathsf{negl}(\lambda)$. Let $\mathcal{A}$ be an adversary for the DDH-f problem, its advantage is defined as:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh\text{-}f}}(\lambda) \overset{\text{def}}{=} \Big| \Pr\big[b = b^\star : \mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q),\ |q| \geqslant \lambda,$$
$$x, y \hookleftarrow \mathcal{D}, u \hookleftarrow \mathbf{Z}/q\mathbf{Z}, X = g^x, Y = g^y, b \hookleftarrow \{0,1\}, Z_0 = g^{xy}, Z_1 = g^{xy} f^u,$$
$$b^\star \leftarrow \mathcal{A}(q, \mathsf{pp}_{\mathsf{CL}}, X, Y, Z_b, \mathsf{Solve}(.))\big] - 1/2 \Big|.$$

The DDH-f problem is $\delta_{\mathsf{ddh\text{-}f}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for all PPT attacker $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh\text{-}f}}(\lambda) \leqslant \delta_{\mathsf{ddh\text{-}f}}(\lambda)$. We say the DDH-$f$ assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ (or the DDH-f problem is hard for $\mathsf{Gen}_{\mathsf{CL}}$), if the DDH-$f$ problem is $\delta_{\mathsf{ddh\text{-}f}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\delta_{\mathsf{ddh\text{-}f}}(\lambda) = \mathsf{negl}(\lambda)$.

3.2.4 Low Order & Strong Root Assumptions

Since the order of $\widehat{G}$ is unknown, some of our upcoming proofs use additional assumptions. The first assumption, called the LO assumption, states that it is hard to find low order elements in $\widehat{G}$. The second, called the SR assumption, states that it is hard to find roots in $\widehat{G}$ of random elements of the subgroup G^q. These assumptions are not *necessary* for security to go through, however in our distributed protocols, a party P must provide *proofs of knowledge* to other parties, so as to convince them of its' honest behaviour. The combination of the LO and SR assumptions allows to significantly improve the efficiency of protocols implementing these proofs of knowledge. Indeed, as was explained in Section 2.8.3, the fact the order of $\widehat{G}$ is unknown is typically a bad thing when performing proofs of knowledge. This is because, unless P's proof is repeated many times, it is not immediate how P can convince other parties that it is not cheating. In Section 3.7, we provide arguments of knowledge (*cf.* Definition 2.24) which need not be repeated, and demonstrate that if P were able to cheat with significant probability, then one could use P to find a root for some given (random) element of the group, thus violating the strong root assumption. We also need the low order assumption to make sure that no undetected low order elements are maliciously injected in the protocols (e.g. to extract unauthorised information).

Definition 3.7 (Low order assumption)**.** Let $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator for a group with an easy DL subgroup. Consider positive integers $\lambda \in \mathbf{N}$, and $\gamma \in \mathbf{N}$. Let $\mathcal{A}$ be an adversary for the γ-*low order* (LO$_\gamma$) *problem*, its advantage is defined as:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{lo}_\gamma}(\lambda) \overset{\text{def}}{=} \Pr\big[\mu^d = 1, 1 \neq \mu \in \widehat{G}, 1 < d < \gamma \ :$$
$$\mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \hookleftarrow \mathsf{Gen}(1^\lambda, q),\ |q| \geqslant \lambda\,, (\mu, d) \hookleftarrow \mathcal{A}(q, \mathsf{pp}_{\mathsf{CL}}, \mathsf{Solve}(.))\big].$$

The LO$_\gamma$ problem is ϵ_{lo}-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for all PPT attacker $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{lo}_\gamma}(\lambda) \leqslant \epsilon_{\mathsf{lo}}(\lambda)$. We say the LO$_\gamma$ assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ (or the LO$_\gamma$ problem is hard for $\mathsf{Gen}_{\mathsf{CL}}$), if the LO$_\gamma$ problem is ϵ_{lo}-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\epsilon_{\mathsf{lo}}(\lambda) = \mathsf{negl}(\lambda)$.

We now define a strong root assumption for class groups. This can be seen as a generalisation of the strong RSA assumption, introduced by Baric et al. in [BP97], which states that it is hard, given an RSA integer N and a random $s \in (\mathbf{Z}/N\mathbf{Z})^*$, to find $a, b \in \mathbf{Z}$, $b \neq \pm 1$ satisfying $a^b = s \bmod N$. In more general terms, the assumption states that it is hard to take *arbitrary* roots of *random* elements. We specialise this assumption for class groups where computing square roots is easy knowing the factorisation of the discriminant [Lag80], and tailor it to our needs by considering challenges in a subgroup.

Definition 3.8 (Strong root assumption for class groups)**.** Let λ be a positive integer and let $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator for a group with an easy DL subgroup. Let $\mathcal{A}$ be an adversary for the *strong root* (SR) *problem in class groups*, its goal is to output a positive integer $e \neq 2^k$, $k \in \mathbf{N}$, and $X \in \widehat{G}$, s.t. $Y = X^e$, given a random Y in G^q, the output of Gen and access to $\mathsf{Solve}(.)$. Precisely, let $\mathcal{D}_q$ be a distribution over the integers such that the distribution $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$ is at distance less than $\delta(\lambda)$ from the uniform distribution in G^q, for some $\delta(\lambda) = \mathsf{negl}(\lambda)$. $\mathcal{A}$'s advantage is defined as:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sr}}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Pr[Y = X^e, X \in \widehat{G}, e \neq 2^k; e, k \in \mathbf{N} :& \\ \mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \hookleftarrow \mathsf{Gen}(1^\lambda, q),& \\ |q| \geqslant \lambda, y \hookleftarrow \mathcal{D}_q; Y \leftarrow g_q^y, (e, X) \hookleftarrow \mathcal{A}(q, \mathsf{pp}_{\mathsf{CL}}, Y, \mathsf{Solve}(.))]&.\end{aligned}$$

The SR problem for class groups is ϵ_{sr}-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for any PPT attacker $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{sr}}_{\mathcal{A}}(\lambda) \leqslant \epsilon_{\mathsf{sr}}(\lambda)$. We say the SR assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ (or the SR problem is hard for $\mathsf{Gen}_{\mathsf{CL}}$), if the SR problem is ϵ_{sr}-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\epsilon_{\mathsf{sr}}(\lambda) = \mathsf{negl}(\lambda)$.

Hardness of the LO and SR Assumptions in Class Groups

Recent uses of similar assumptions. We first note that in the generic group model (for groups of unknown order), both assumptions were proven to be hard in [BBF19]. For our applications, we will use the SR assumption and the LO assumption in the context of class groups. These assumptions are not completely novel in this setting: Damgård and Fujisaki [DF02] explicitly consider variants of these assumptions in this context. Then, Lipmaa used a strong root assumption in class groups to build accumulators without trusted setup in [Lip12]. Recently, an interactive variant of the SR assumption was used, still in the context of class groups, by Wesolowski to build verifiable delay functions without trusted setup [Wes19].

Both the LO and SR assumptions were also used by Bünz et al. in the context of class groups [BFS20]. They call these assumptions the order assumption and the 2-strong RSA assumption, where the '2' refers to the fact computing e-th roots, where e is a power 2, is easy, and hence dos not break the assumption. They use these assumptions to devise a polynomial commitment scheme in groups of unknown order with efficient verifier time and small proof sizes. The assumptions serve similar purposes to our use of them, ensuring that if a prover can cheat with significant probability, then one can leaverage the prover to break one of the assumptions. Bünz et al. [BFS20] also use an *adaptive root assumption*, this assumption was introduced and used in the context of class groups by Wesolowski [Wes19] to build verify delay functions, then re-formulated by Boneh et al. [BBF18], and has since been used by e.g. [LM19] to build succinct non-interactive arguments of knowledge (SNARK) and [BBF18,Pie19] to build verifiable delay functions. The adaptive root assumption is *dual* to the strong root assumption, as it states that it is hard to take *random* roots of *arbitrary* group elements; *i.e.,* an adversary, having chosen a group element w, is given a random prime ℓ and must output a group element u satisfying $u^\ell = w \neq 1$. Bünz et al. [BFS20] further demonstrate that the adaptive root assumption implies the low order assumption. As a final note Lai et al. [LM19] suggested a public coin setup for class groups so that the adaptive root assumption is conjectured to hold in the resulting group.

On the hardness of these assumptions. In the following, we advocate the hardness of these assumptions in the context of class groups.

The root problem and its hardness was considered in [BH01, BBHM02] in the context of class groups to design signature schemes. It is similar to the RSA problem: the adversary is not allowed to choose the exponent e. These works compare the hardness of this problem with the problem of computing the group order and conclude that there is no better known method to compute a solution to the root problem than to compute the order of the group.

As mentioned previously the strong root assumption is a generalisation of the strong RSA assumption. Again, the best known algorithm to solve this problem is to compute the order of the group to be able to invert exponents. For strong RSA this means factoring the modulus. For the SR problem in class groups, this means computing the class number. Best known algorithms for this problem have worse complexity than those for factoring integers (*cf.* Sections 2.4.2 and 2.6), and in our applications, discriminants are chosen such that this problem is intractable.

Concerning the LO assumption, we will need the LO_γ problem to be hard in $\widehat{G}$, where γ can be as big as 2^{128}. Note that in our instantiation, the discriminant is chosen such that the $2-$Sylow subgroup is isomorphic to $\mathbf{Z}/2\mathbf{Z}$. It is well known that elements of order 2 can be computed from the (known) factorisation of Δ_q. However, we work with the odd part, which is the group of squares in this context, so we do not take this element into account.

Let us see that the proportion of such elements of low order is very low in the odd part. As noted in Section 2.6, from the Cohen Lenstra heuristics [CL84] the odd part of a class group $Cl(\mathcal{O}_\Delta)$ of an imaginary quadratic field is cyclic with probability 97.75%; and it is conjectured [HS06] that the probability an integer d divides $h(\mathcal{O}_\Delta)$ is less than:

$$\frac{1}{d}+\frac{1}{d\log d}.$$

As a consequence, if the odd part of $Cl(\mathcal{O}_\Delta)$ is cyclic then, denoting φ Euler's totient function, the expected number of elements of order less than γ is less than:

$$\sum_{d\leqslant\gamma}\left(\frac{1}{d}+\frac{1}{d\log d}\right)\varphi(d),$$

which can be upper bounded by 2γ. For 128 bits of security, our class number will have around 913 bits, so the proportion of elements of order less than 2^{128} is less than 2^{-784}.

Moreover, if the odd part of the class group is non cyclic, it is very likely that it is of the form $\mathbf{Z}/n_1\mathbf{Z}\oplus\mathbf{Z}/n_2\mathbf{Z}$ where $n_2|n_1$ and n_2 is very small, hence it should behave as in the case where it were cyclic.

There have been intense efforts on the construction of families of discriminants such that there exist elements of a given small order p or with a given $p-$rank (number of cyclic factors in the $p-$Sylow subgroup). However, these families are very sparse and will be reached by our generation algorithm of the discriminant only with negligible probability. The basic idea of these constructions is to build a discriminant Δ in order to obtain solutions of a Diophantine equation that gives m and the representation of a non principal ideal I of norm m such that I^p is principal, and I has order p in $Cl(\mathcal{O}_\Delta)$ (see eg [Bue76] or [Bel04] for more references). Solving such a norm equation for a fixed discriminant was mentioned as a starting point for an attack in [BBF18] combined with the Coppersmith method, but no concrete advances on the problem have been proposed.

3.2.5 Summary of Assumptions in the CL Framework

In Table 3.1 we provide a summary of the assumptions from the CL framework presented in this section. Algorithm $\mathsf{Gen}_{\mathsf{CL}}=(\mathsf{Gen},\mathsf{Solve})$ is a generator for a group with an easy DL subgroup; λ and γ are positive integers, and we let $\mathsf{pp}_{\mathsf{CL}}:=(\tilde{s},g,f,g_q,\widehat{G},G,F,G^q)\leftarrow\mathsf{Gen}(1^\lambda,q)$ for some prime $q\geqslant\lambda$. Furthermore $\mathcal{A}$ denotes a PPT adversary for the considered problem. We assume $\mathcal{A}$ is always given as input $\mathsf{pp_{CL}}$, distributions $\mathcal{D}$ and $\mathcal{D}_q$, and has access to the Solve algorithm.

	HSM-CL	DDH-CL	DDH-f
Challenge set up	$x \hookleftarrow \mathcal{D}, x' \hookleftarrow \mathcal{D}_q$ $b \hookleftarrow \{0,1\}$, $Z_0 \leftarrow g^x$, $Z_1 \leftarrow g_q^{x'}$	$x, y, z \hookleftarrow \mathcal{D}$, $X = g^x, Y = g^y$, $b \hookleftarrow \{0,1\}$, $Z_0 \leftarrow g^{xy}$, $Z_1 \leftarrow g^z$	$x, y \hookleftarrow \mathcal{D}$, $u \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, $X = g^x, Y = g^y$, $b \hookleftarrow \{0,1\}$, $Z_0 = g^{xy}, Z_1 = g^{xy} f^u$
$\mathcal{A}$'s input	Z_b	X, Y, Z_b	X, Y, Z_b
$\mathcal{A}$'s winning output	b^* such that $b = b^*$	b^* such that $b = b^*$	b^* such that $b = b^*$

	LO$_\gamma$	SR
Challenge set up	–	$y \hookleftarrow \mathcal{D}_q, Y \leftarrow g_q^y$
$\mathcal{A}$'s input	–	Y
$\mathcal{A}$'s winning output	$\mu \in \widehat{G}, \mu \neq 1$ and d such that $1 < d < \gamma$ and $\mu^d = 1$	$e \neq 2^k \wedge e, k \in \mathbf{N}$ and $X \in \widehat{G}$ such that $Y = X^e$

Table 3.1: Summary of assumptions in the CL framework

3.3 Projective Hash Functions from the CL Framework

Using the formalism of [CS02], we can now build projective hash functions from the CL framework. We define a number of both new and existing properties for PHFs. To help understand these technical notions, we illustrate them with running examples arising from the traditional DDH assumption in finite fields, as well as from our HSM-CL and DDH-f assumptions of Section 3.2. As noted in the chapter introduction an instantiation from the DCR assumption could also be made to satisfy all our properties. Such an instantiation would very much resemble our HSM-CL based PHFs.

The definitions we provide are for a general class of group-theoretic language membership problems. Moreover we consider the more general case where elements of the considered groups may not be efficiently recognisable, thus allowing for a wider range of instantiations.

To build PHFs one starts with an instance of a subgroup membership problem.

3.3.1 Subgroup Membership Problems

Definition 3.9 ([CS02])**.** Let λ be a positive integer. A generator for a $\delta_{\mathcal{L}}$-hard subgroup membership problem is a PPT algorithm $\mathsf{Gen}_{\mathcal{SM}}$ which on input 1^λ returns the description of a subgroup membership problem:

$$\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R}), \quad \text{where}$$

- $\widehat{\mathcal{X}}$ is a finite Abelian group whose elements are recognisable;
- $\mathcal{X} \subseteq \widehat{\mathcal{X}}$ is a subgroup of $\widehat{\mathcal{X}}$ (whose elements may not be recognisable);
- $\widehat{\mathcal{L}} \subset \widehat{\mathcal{X}}$ is a subgroup of $\widehat{\mathcal{X}}$, and $\mathcal{L} := \mathcal{X} \cap \widehat{\mathcal{L}}$;

- $\mathsf{R} \subset \mathcal{X} \times \mathcal{W}$ is a binary relation. For $x \in \mathcal{L}$ and $w \in \mathcal{W}$, w is a witness for x if $(x, w) \in \mathsf{R}$. The relation R is efficiently samplable: one samples a random $x \in \mathcal{L}$ along with a witness $w \in \mathcal{W}$ for x, this implicitly defines a way to sample random elements of $\mathcal{L}$. We denote this sampling $(x, w) \leftarrow \mathsf{R}$;
- It is hard to distinguish random elements of $\mathcal{L}$ from those of $\mathcal{X}$. Precisely $\delta_{\mathcal{L}}$ is the maximal advantage of any PPT adversary in solving this problem.

If $\widehat{\mathcal{X}} = \mathcal{X}$ (and hence $\widehat{\mathcal{L}} = \mathcal{L}$) we simply denote $\mathcal{SM} := (\mathcal{X}, \mathcal{L}, \mathcal{W}, \mathsf{R})$.

Remark. In this definition "random" and "samplable" mean for a distribution that is to be defined when instantiating the subgroup membership problem.

Remark. In the following running examples, in particular those arising from HSM-CL and DDH-f where $\widehat{\mathcal{X}} \neq \mathcal{X}$, our choices for $\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{L}$ are not intuitive. These choices will become clearer latter in our work, namely when we introduce *decomposability* (*cf.* Definition 3.18). Indeed in our upcoming proofs we will need $\widehat{\mathcal{X}}$ (resp. $\mathcal{X}$) to be isomorphic to the direct product of $\widehat{\mathcal{L}}$ (resp. $\mathcal{L}$) and the subgroup generated by some element $\widehat{\Upsilon} \in \widehat{\mathcal{X}}$ (resp. $\Upsilon \in \mathcal{X}$). This decomposability allows a clear separation between the information that must remain hidden from adversaries in order to ensure security, and that which the cryptosystem may leak without incurring significant harm to the security of the scheme. The values of Υ and $\widehat{\Upsilon}$ depend on the PHFs which will be defined in Sections 3.3.2 and 3.3.3.

Running Example 1 – DDH

Let $(G, g, q) \leftarrow \mathsf{Gen}_{\mathsf{DL}}(1^\lambda)$ as described in Section 2.4.1, and $g_0, g_1 \hookleftarrow G$. Here $\widehat{\mathcal{X}} = \mathcal{X}$ and we set

$$\mathcal{X} := G^2.$$

The subgroup $\widehat{\mathcal{L}} = \mathcal{L}$ of $\mathcal{X}$ is that generated by (g_0, g_1), i.e.

$$\mathcal{L} := \langle (g_0, g_1) \rangle.$$

A witness for $(x_0, x_1) \in \mathcal{L}$ is $w \in \mathbf{Z}/q\mathbf{Z}$ such that $(x_0, x_1) = (g_0^w, g_1^w)$; we denote

$$\mathsf{R}_{\mathsf{ddh}} := \{((x_0, x_1), w) \in ((\mathcal{L} \times \mathbf{Z}/q\mathbf{Z}) \mid (x_0, x_1) = (g_0^w, g_1^w)\}.$$

Thus $\mathcal{SM}_{\mathsf{ddh}} := (G^2, \langle (g_0, g_1) \rangle, \mathbf{Z}/q\mathbf{Z}, \mathsf{R}_{\mathsf{ddh}})$. One samples random elements of $\mathcal{L}$, together with the corresponding witness, by sampling $w \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, and computing $(x_0, x_1) := (g_0^w, g_1^w)$. Sampling on G^2 is done[1] by sampling $r, r' \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, and outputting $(g_0^r, g_1^r) \odot (1, g_1^{r'})$.

It is clear that this defines a subset membership problem, and that the hardness of this subset membership problem is implied by the hardness of the DDH problem in G. If the DDH problem is δ_{ddh}-hard for $\mathsf{Gen}_{\mathsf{DL}}$, then $\mathcal{SM}_{\mathsf{ddh}}$ is δ_{ddh}-hard.

Running Example 2 – HSM-CL

Consider a generator $\mathsf{Gen}_{\mathsf{CL}}$ of a HSM-CL group with an easy DL subgroup (*cf.* Definition 3.4), which, on input 1^λ and a μ-bit prime q (for $\mu \geqslant \lambda$), outputs $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$. We set

$$\widehat{\mathcal{X}} := \widehat{G}, \quad \mathcal{X} := G = \langle g \rangle \quad \text{and} \quad \widehat{\mathcal{L}} := \widehat{G}^q.$$

[1] This choice, which may seem convoluted when one could output $(g_0^r, g_1^{r'})$, is for consistency with the notion of decomposability introduced in Section 3.3.5.

Then

$$\mathcal{L} := \mathcal{X} \cap \widehat{\mathcal{L}} = G^q = \langle g_q \rangle.$$

A witness for $x \in \mathcal{L}$ is $w \in \mathbf{Z}$ such that $x = g_q^w$; we denote

$$\mathsf{R}_{\mathsf{hsm\text{-}cl}} := \{(x, w) \in G^q \times \mathbf{Z} \mid x = g_q^w\}.$$

Thus $\mathcal{SM}_{\mathsf{hsm\text{-}cl}} := (\widehat{G}, G, \widehat{G}^q, \mathbf{Z}, \mathsf{R}_{\mathsf{hsm\text{-}cl}})$.

Witnesses are sampled from $\mathcal{D}_q$ chosen as per Lemma 3.3. This induces a distribution δ-close to uniform on G^q. Sampling in G is done by sampling $w \hookleftarrow \mathcal{D}_q$, $u \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, and outputting $g_q^w f^u$, this induces a distribution δ-close to uniform on G (*cf.* Lemma 3.3). It is clear from Definition 3.4 that if the HSM-CL problem is $\delta_{\mathsf{hsm\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ then $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is $\delta_{\mathsf{hsm\text{-}cl}}$-hard.

Comparison with DCR. Though we do not detail a running example arising from the DCR assumption, such an instantiation would share many similarities with our running example from HSM-CL. Indeed, if we denote $\mathcal{SM}_{\mathsf{dcr}}$ the subgroup membership problem arising from DCR, using the notations of Fig. 2.1, one essentially substitutes f in $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ for the generator $g = (1 + N)$ of the subgroup of order N in $\mathcal{SM}_{\mathsf{dcr}}$; and sets $\widehat{\mathcal{L}}$ to be the subgroup of N-th powers in $\mathbf{Z}/N^2\mathbf{Z}$.

Running Example 3 – DDH-f

Consider a generator $\mathsf{Gen}_{\mathsf{CL}}$ of a DDH-f group with an easy DL subgroup (*cf.* Definition 3.6), which on input 1^λ and a μ-bit prime q (for $\mu \geqslant \lambda$), outputs $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$. Sample $\alpha \hookleftarrow \mathcal{D}$ satisfying $\alpha \neq 1$, and set $h := g^\alpha$. Let

$$\widehat{\mathcal{X}} := \widehat{G}^2 \quad \text{and} \quad \mathcal{X} := \langle (g, h), (1, f) \rangle.$$

Let us denote $F_\alpha := \langle (f, f^\alpha) \rangle$. Then we set

$$\widehat{\mathcal{L}} := \langle (\widehat{G}^q)^2, F_\alpha \rangle = \{(z_0 f^r, z_1 f^{\alpha r}) | z_0, z_1 \in \widehat{G}^q \,; r \in \mathbf{Z}/q\mathbf{Z}\}.$$

Then, by intersecting $\mathcal{X}$ and $\widehat{\mathcal{L}}$, we obtain $\mathcal{L}$. Observe that letting $z_0' \leftarrow z_0 g_q^{-r}, z_1' \leftarrow z_1 g_q^{-\alpha r} \in \widehat{G}^q$ one can rewrite $\widehat{\mathcal{L}} = \{(g^r, h^r) \odot (z_0', z_1') | z_0', z_1' \in \widehat{G}^q \,; r \in \mathbf{Z}/q\mathbf{Z}\}$. Now since $\widehat{G} \simeq \widehat{G}^q \times F$, it is clear that:

$$\mathcal{L} := \langle (g, h) \rangle.$$

A witness for $(x_0, x_1) \in \mathcal{L}$ is $w \in \mathbf{Z}$ s.t. $(x_0, x_1) = (g^w, h^w)$; we denote

$$\mathsf{R}_{\mathsf{ddh\text{-}f}} := \{((x_0, x_1), w) \in (G^2 \times \mathbf{Z}) \mid (x_0, x_1) = (g^w, h^w)\}.$$

Thus $\mathcal{SM}_{\mathsf{ddh\text{-}f}} := (\widehat{G}^2, \langle (g, h), (1, f) \rangle, \langle (\widehat{G}^q)^2, F_\alpha \rangle, \mathbf{Z}, \mathsf{R}_{\mathsf{ddh\text{-}f}})$.

Witnesses are sampled from a distribution $\mathcal{D}$ chosen as per Lemma 3.3. This induces a distribution δ-close to $\mathcal{U}(G)$. One similarly samples random elements of $\mathcal{X}$ by sampling $w \hookleftarrow \mathcal{D}$, $u \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, and outputting $(g^w, h^w f^u)$. It is clear from Definition 3.6 that if the DDH-f problem is $\delta_{\mathsf{ddh\text{-}f}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ then $\mathcal{SM}_{\mathsf{ddh\text{-}f}}$ is $\delta_{\mathsf{ddh\text{-}f}}$-hard.

3.3.2 Projective Hash Functions

We here recall the definition of *projective hash functions*, these where first introduced in [CS02] to build efficient ind-cpa and ind-cca-secure public key encryption schemes. Indeed, as we will see in Section 3.5, PHFs possessing a specific property called smoothness allow for the construction of $\mathsf{ind\text{-}cpa}$-secure public key encryption schemes. Using a second *extended* projective hash function (*cf.* Definition 3.22), one can further guarantee ciphertext integrity, and thereby attain $\mathsf{ind\text{-}cca}$-security. We build upon the idea mentioned in [CS02], of a projective hash function *for a generalised subset membership problem*, since this definition takes account of instantiations where $\mathcal{X} \subsetneq \widehat{\mathcal{X}}$, and elements of $\mathcal{X}$ are not efficiently recognisable. We formally define this intuition by introducing two new algorithms $\widehat{\mathsf{projkg}}$ and $\widehat{\mathsf{projhash}}$. These algorithms will only be used to prove security of our constructions, so as to quantify the maximum information an adversary can learn. As such they need not be efficiently computable.

Definition 3.10. Let λ be a positive integer. Consider a generator $\mathsf{Gen}_{\mathcal{SM}}$ for a subgroup membership problem, and an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R}) \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$. A projective hash function (PHF) for the subgroup membership problem $\mathcal{SM}$ is a tuple of algorithms $\mathsf{H} := (\mathsf{hashkg}, \widehat{\mathsf{projkg}}, \mathsf{projkg}, \mathsf{hash}, \widehat{\mathsf{projhash}}, \mathsf{projhash})$, where all algorithms have as implicit input the description $\mathcal{SM}$ of the subset membership problem, and:

- $\mathsf{hashkg}(\mathcal{SM})$ is a PPT algorithm which on input the description of $\mathcal{SM}$, outputs a *hashing key* hk in some set K_{hk};

- $\widehat{\mathsf{projkg}}(\mathsf{hk})$ is a deterministic algorithm which on input $\mathsf{hk} \in K_{\mathsf{hk}}$ outputs a *projection key* $\widehat{\mathsf{hp}}$. The image of K_{hk} through $\widehat{\mathsf{projkg}}$ is denoted $K_{\widehat{\mathsf{hp}}}$;

- $\mathsf{projkg}(\mathsf{hk})$ is a DPT algorithm which on input $\mathsf{hk} \in K_{\mathsf{hk}}$ outputs a *public projection key* hp, such that for $\mathsf{hk} \in K_{\mathsf{hk}}$, hp is a fixed deterministic function of the output of $\widehat{\mathsf{projkg}}(\mathsf{hk})$. The image of K_{hk} through projkg is denoted K_{hp};

- $\mathsf{hash}(\mathsf{hk}, x)$ is a DPT algorithm which on input $\mathsf{hk} \in K_{\mathsf{hk}}$, $x \in \widehat{\mathcal{X}}$ outputs the *hash value* $\mathsf{hash}(\mathsf{hk}, x)$. The image of $\widehat{\mathcal{X}}$ through hash is a finite Abelian group called the *set of hash values* and is denoted Π;

- $\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}, x)$ is a deterministic algorithm which on input $\widehat{\mathsf{hp}} \in K_{\widehat{\mathsf{hp}}}$ and $x \in \widehat{\mathcal{L}}$, outputs the hash value $\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}, x)$ in Π;

- $\mathsf{projhash}(\mathsf{hp}, x, w)$ is a DPT algorithm which on input $\mathsf{hp} \in K_{\mathsf{hp}}$, $x \in \mathcal{L}$ and the corresponding witness $w \in \mathcal{W}$, outputs the hash value $\mathsf{projhash}(\mathsf{hp}, x, w)$ in Π.

Correctness requires that for all $\lambda \in \mathbf{N}$, any $\mathcal{SM} \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$, any $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM})$, $\widehat{\mathsf{hp}} \leftarrow \widehat{\mathsf{projkg}}(\mathsf{hk})$ and $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$, it holds that (1) for any $x \in \widehat{\mathcal{L}}$, $\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}, x) = \mathsf{hash}(\mathsf{hk}, x)$; and (2) for any $(x, w) \in \mathsf{R}$, $\mathsf{projhash}(\mathsf{hp}, x, w) = \mathsf{hash}(\mathsf{hk}, x)$.
If $\widehat{\mathcal{X}} = \mathcal{X}$, then $\widehat{\mathsf{projkg}} = \mathsf{projkg}$ and $\widehat{\mathsf{projhash}} = \mathsf{projhash}$, so we denote $\mathsf{H} := (\mathsf{hashkg}$, projkg, hash, $\mathsf{projhash})$.

Remark. In upcoming running examples (*cf.* Section 3.3.3) we will see that the choices of $\widehat{\mathcal{L}}$ and $\mathcal{L}$ are tightly linked to the definitions for algorithms $\widehat{\mathsf{projkg}}$, projkg, $\widehat{\mathsf{projhash}}$ and $\mathsf{projhash}$, since given the output of $\widehat{\mathsf{projkg}}$ (resp. projkg) one must be able to evaluate algorithm $\widehat{\mathsf{projhash}}$ (resp. $\mathsf{projhash}$) over any $(x, w) \in \mathsf{R}$ (resp. $x \in \widehat{\mathcal{L}}$).

3.3.3 Homomorphic Properties

If PHFs – derived from Abelian groups – satisfy some natural homomorphic properties, they allow for the construction of advanced cryptographic primitives. In particular we will need Definitions 3.11 and 3.13, for correctness of our constructions.

Definition 3.11 ([HO09])**.** Recall that Π is the set of hash values, and that $(\widehat{\mathcal{X}}, \cdot)$ and $(\Pi, \cdot)$ are Abelian groups. A PHF H is *homomorphic* if for all $\mathsf{hk} \in K_{\mathsf{hk}}$, and $u_1, u_2 \in \widehat{\mathcal{X}}$, one has $\mathsf{hash}(\mathsf{hk}, u_1) \cdot \mathsf{hash}(\mathsf{hk}, u_2) = \mathsf{hash}(\mathsf{hk}, u_1 \cdot u_2)$. That is to say $\mathsf{hash}(\mathsf{hk}, \cdot)$ is a homomorphism for each hk.

If H is homomorphic and correct then clearly for $\widehat{\mathsf{hp}} \leftarrow \widehat{\mathsf{projkg}}(\mathsf{hk})$ (resp. $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$) the function $\widehat{\mathsf{projhash}}$ (resp. $\mathsf{projhash}$) is linearly homomorphic with respect to elements $u_1, u_2 \in \widehat{\mathcal{L}}$ (resp. $u_1, u_2 \in \mathcal{L}$). Note however that this holds for $\mathsf{hp} \in K_{\mathsf{hp}}$, where K_{hp} may not be efficiently recognisable. We define the notion of a *homomorphically-extended* PHF which ensures the homomorphic properties of the PHF hold for any public projection key sampled from an efficiently recognisable set. This property will be of particular interest for our multi-party protocols, as it allows us to avoid expensive zero-knowledge proofs that a public key is well formed.

Definition 3.12 (Homomorphically extended PHF)**.** Consider a PHF H with set of public projection keys K_{hp}. We say H is *homomorphically extended* if H is homomorphic and there exists an efficiently recognisable space $K'_{\mathsf{hp}} \supseteq K_{\mathsf{hp}}$ such that for any $\mathsf{hp}' \in K'_{\mathsf{hp}}$, $\mathsf{projhash}(\mathsf{hp}', \cdot)$ is a homomorphism (respectively to its inputs in $\mathcal{L}$).

Definition 3.13 ([BBL17])**.** A PHF H is *key homomorphic* if $(K_{\mathsf{hk}}, +)$ and $(\Pi, \cdot)$ are Abelian groups, and for all $x \in \widehat{\mathcal{X}}$ and $\mathsf{hk}_0, \mathsf{hk}_1 \in K_{\mathsf{hk}}$, one has $\mathsf{hash}(\mathsf{hk}_0, x) \cdot \mathsf{hash}(\mathsf{hk}_1, x) = \mathsf{hash}(\mathsf{hk}_0 + \mathsf{hk}_1, x)$. That is to say $\mathsf{hash}(\cdot, x)$ is a homomorphism for each x.

Running Example 1 – DDH-based PHF

Recall that for $\mathcal{SM}_{\mathsf{ddh}}$ it holds that $\widehat{\mathcal{X}} = \mathcal{X}$, $\widehat{\mathcal{L}} = \mathcal{L}$ and consequently $\widehat{\mathsf{projkg}} = \mathsf{projkg}$ and $\widehat{\mathsf{projhash}} = \mathsf{projhash}$. One defines $\mathsf{H}_{\mathsf{ddh}}$ from $\mathcal{SM}_{\mathsf{ddh}}$ as follows. The hash key space is $K_{\mathsf{hk}} := (\mathbf{Z}/q\mathbf{Z})^2$. The hashkg algorithm samples keys uniformly from K_{hk}. The projective hash function $\mathsf{H}_{\mathsf{ddh}}$ defines keyed hash functions with co-domain $\Pi := G$, such that:

$$\begin{array}{rccc} \mathsf{hash}: & \mathbf{Z}/q\mathbf{Z}^2 \times G^2 & \to & G. \\ & ((\kappa_0, \kappa_1), (x_0, x_1)) & \mapsto & x_0^{\kappa_0} x_1^{\kappa_1} \end{array}$$

Algorithm projkg takes values in $K_{\mathsf{hp}} := G$ and is defined as:

$$\begin{array}{rccc} \mathsf{projkg}: & (\mathbf{Z}/q\mathbf{Z})^2 & \to & G. \\ & (\kappa_0, \kappa_1) & \mapsto & g_0^{\kappa_0} g_1^{\kappa_1} \end{array}$$

For $\mathsf{hp} \leftarrow g_0^{\kappa_0} g_1^{\kappa_1} \in G$, and $((x_0, x_1), w) \in \mathsf{R}_{\mathsf{ddh}}$, one defines:

$$\mathsf{projhash}(\mathsf{hp}, (x_0, x_1), w) := \mathsf{hp}^w,$$

which is equal to $g_0^{w \cdot \kappa_0} g_1^{w \cdot \kappa_1} = x_0^{\kappa_0} x_1^{\kappa_1} = \mathsf{hash}(\mathsf{hk}, (x_0, x_1))$; hence correctness holds.

Homomorphic properties: let the group operation for G^2 be coordinate-wise multiplication. Then $\forall (\kappa_0, \kappa_1) \in \mathbf{Z}/q\mathbf{Z}^2$ and $x_0, x_1, x_2, x_3 \in G$, $\mathsf{hash}((\kappa_0, \kappa_1), (x_0, x_1)) \cdot \mathsf{hash}((\kappa_0, \kappa_1), (x_2, x_3)) = (x_0 \cdot x_2)^{\kappa_0} \cdot (x_1 \cdot x_3)^{\kappa_1} = \mathsf{hash}((\kappa_0, \kappa_1), (x_0 \cdot x_2, x_1 \cdot x_3)) = \mathsf{hash}((\kappa_0, \kappa_1), (x_0, x_1) \odot (x_2, x_3))$. Thus $\mathsf{H}_{\mathsf{ddh}}$ is homomorphic. The homomorphic properties of G readily imply that $\mathsf{H}_{\mathsf{ddh}}$ is also key homomorphic.

Running Example 2 – HSM-CL-based PHF

We define $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ from $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ as follows. The hash key space is $K_{\mathsf{hk}} := \mathbf{Z}$. The hashkg algorithm samples hash keys from the distribution $\widehat{\mathcal{D}} = \mathcal{D}$ as per Lemma 3.3. $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ defines keyed hash functions with co-domain $\Pi := \widehat{G}$, such that:

$$\begin{array}{rccc} \mathsf{hash}: & \mathbf{Z} \times \widehat{G} & \to & \widehat{G}. \\ & (\mathsf{hk}, x) & \mapsto & x^{\mathsf{hk}} \end{array}$$

Functions $\widehat{\mathsf{projkg}}$ and projkg, which output values in $K_{\widehat{\mathsf{hp}}} := \mathbf{Z}/\varpi\mathbf{Z}$ and $K_{\mathsf{hp}} := G^q$ are defined as:

$$\begin{array}{rccc} \widehat{\mathsf{projkg}}: & \mathbf{Z} & \to & \mathbf{Z}/\varpi\mathbf{Z} \\ & \mathsf{hk} & \mapsto & \mathsf{hk} \bmod \varpi \end{array} \quad \text{and} \quad \begin{array}{rccc} \mathsf{projkg}: & \mathbf{Z} & \to & G^q. \\ & \mathsf{hk} & \mapsto & g_q^{\mathsf{hk}} \end{array}$$

Consider any $\mathsf{hk} \in \mathbf{Z}$, and $\widehat{\mathsf{hp}} \leftarrow \widehat{\mathsf{projkg}}(\mathsf{hk})$. For any $z \in \widehat{G}^q$ (note that the order of z divides the group exponent ϖ), we define:

$$\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}, z) := z^{\widehat{\mathsf{hp}}},$$

which is equal to $\mathsf{hash}(\mathsf{hk}, z)$. For $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$, $x \in G^q$ with witness $w \in \mathbf{Z}$ we define:

$$\mathsf{projhash}(\mathsf{hp}, x, w) := \mathsf{hp}^w,$$

which is equal to $\mathsf{hash}(\mathsf{hk}, x)$, hence correctness holds.

Homomorphic properties: $\mathbf{Z}$ is an additive Abelian group, $\widehat{G}$ is a multiplicative finite Abelian group. This readily implies that $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is homomorphic and key homomorphic. Furthermore, setting $K'_{\mathsf{hp}} := \widehat{G}$ which contains G, $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is also homomorphically extended.

Running Example 3 – DDH-f-based PHF

We define $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ from $\mathcal{SM}_{\mathsf{ddh\text{-}f}}$ as follows. The hash key space is $K_{\mathsf{hk}} := \mathbf{Z}^2$. The hashkg algorithm samples two elements from distribution $\widehat{\mathcal{D}} = \mathcal{D}$ as per Lemma 3.3. The hashing algorithm has co-domain $\Pi := \widehat{G}$, and is defined as:

$$\begin{array}{rccc} \mathsf{hash}: & \mathbf{Z}^2 \times \widehat{G}^2 & \to & \widehat{G}. \\ & ((\kappa_0, \kappa_1), (x_0, x_1)) & \mapsto & x_0^{\kappa_0} x_1^{\kappa_1} \end{array}$$

Recall that in $\mathcal{SM}_{\mathsf{ddh\text{-}f}}$ we defined $h := g^\alpha$. Functions $\widehat{\mathsf{projkg}}$ and projkg, which take values in $K_{\widehat{\mathsf{hp}}} := (\mathbf{Z}/\varpi\mathbf{Z})^2 \times \mathbf{Z}/q\mathbf{Z}$ and $K_{\mathsf{hp}} := G$ are defined as:

$$\begin{array}{rccc} \widehat{\mathsf{projkg}}: & \mathbf{Z}^2 & \to & (\mathbf{Z}/\varpi\mathbf{Z})^2 \times \mathbf{Z}/q\mathbf{Z} \\ & (\kappa_0, \kappa_1) & \mapsto & (\kappa_0 \bmod \varpi, \kappa_1 \bmod \varpi, \kappa_0 + \alpha\kappa_1 \bmod q) \end{array}$$

and

$$\begin{array}{rccc} \mathsf{projkg}: & \mathbf{Z}^2 & \to & G. \\ & (\kappa_0, \kappa_1) & \mapsto & g^{\kappa_0} h^{\kappa_1} \end{array}$$

Consider any $\mathsf{hk} := (\kappa_0, \kappa_1) \in \mathbf{Z}^2$. Let $\hat{\kappa}_0 := \kappa_0 \bmod \varpi$; $\hat{\kappa}_1 := \kappa_1 \bmod \varpi$; $\iota := \kappa_0 + \alpha\kappa_1 \bmod q$ and $\widehat{\mathsf{hp}} := (\hat{\kappa}_0, \hat{\kappa}_1, \iota) \in K_{\widehat{\mathsf{hp}}}$. By definition of $\widehat{\mathcal{L}} = \{(z_0 f^r, z_1 f^{\alpha r}) | z_0, z_1 \in \widehat{G}^q\,; r \in \mathbf{Z}/q\mathbf{Z}\}$, for any $(y_0, y_1) \in \widehat{\mathcal{L}}$ there exist unique $z_0, z_1 \in \widehat{G}^q$ and $r \in \mathbf{Z}/q\mathbf{Z}$ such that $(y_0, y_1) = (z_0, z_1) \odot (f^r, f^{\alpha r})$. We define

$$\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}, (y_0, y_1)) := z_0^{\hat{\kappa}_0} z_1^{\hat{\kappa}_1} (f^r)^\iota,$$

and insist on the fact that $\widehat{\mathsf{projhash}}$ is not (and need not be) efficiently computable. Since the order of z_0 and z_1 divide the group exponent ϖ of $\widehat{G}^q$, it holds that $\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}, (y_0, y_1)) = \mathsf{hash}(\mathsf{hk}, (y_0, y_1))$. For $\mathsf{hp} \in K_{\mathsf{hp}}$ and $((x_0, x_1), r) \in \mathsf{R}_{\mathsf{ddh\text{-}f}}$ we define:

$$\mathsf{projhash}(\mathsf{hp}, (x_0, x_1), r) := \mathsf{hp}^r,$$

which is equal to $\mathsf{hash}(\mathsf{hk}, (x_0, x_1))$, hence correctness holds.

Homomorphic properties: Let the group operation for $\widehat{G}^2$ be coordinate-wise multiplication. Then $\forall(\kappa_0, \kappa_1) \in \mathbf{Z}^2$, and $y_0, y_1, y_2, y_3 \in \widehat{G}$, $\mathsf{hash}((\kappa_0, \kappa_1), (y_0, y_1)) \cdot \mathsf{hash}((\kappa_0, \kappa_1), (y_2, y_3)) = \mathsf{hash}((\kappa_0, \kappa_1), (y_0, y_1) \odot (y_2, y_3))$. Thus $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is homomorphic, and setting $K'_{\mathsf{hp}} := \widehat{G}$, it is also homomorphically extended. Moreover $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is key homomorphic.

3.3.4 Smoothness

In our upcoming construction for ind-cpa-secure PKE from PHFs (*cf.* Fig. 3.2) one masks the sensitive information (encoded in a subgroup $F \subseteq \Pi$) with an evaluation of the hash function in a random point $x \in \mathcal{L}$. By the hardness of the subgroup membership problem, the distributions induced by sampling a random element of $\mathcal{X}\backslash\mathcal{L}$ or a random element of $\mathcal{L}$ and then computing $\mathsf{hash}(\mathsf{hk}, x)$ are computationally indistinguishable, so replacing the mask with the hash of some random point $x \in \mathcal{X}\backslash\mathcal{L}$ should go unnoticed. To ensure this hash value indeed masks the message, thus ensuring confidentiality, it must hold that for a randomly sampled $x \in \mathcal{X}\backslash\mathcal{L}$, the projection of $\mathsf{hash}(\mathsf{hk}, x)$ onto F be uniformly distributed over F knowing $\widehat{\mathsf{projkg}}(\mathsf{hk})$. A PHF satisfying this property is *smooth over $\mathcal{X}$ on F*.

Definition 3.14 ([CS02])**.** Let λ be a positive integer. Consider a generator $\mathsf{Gen}_{\mathcal{SM}}$ for a subgroup membership problem, and $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R}) \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$. Consider the associated PHF H, whose hash values belong to a finite Abelian group Π. Let F be a subgroup of Π. We say that H is δ_s*-smooth* over $\mathcal{X}$ on F if the following distributions are δ_s-close

$$\{(x, \widehat{\mathsf{projkg}}(\mathsf{hk}), \mathsf{hash}(\mathsf{hk}, x)) | \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM}), x \hookleftarrow \mathcal{X}\backslash\mathcal{L}\}$$

and

$$\{(x, \widehat{\mathsf{projkg}}(\mathsf{hk}), \pi' \cdot \mathsf{hash}(\mathsf{hk}, x) | \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM}), x \hookleftarrow \mathcal{X}\backslash\mathcal{L}, \pi' \hookleftarrow \mathcal{U}(F)\}.$$

If $\delta_s(\lambda) = \mathsf{negl}(\lambda)$, then H is said to be *smooth* over $\mathcal{X}$ on F.

Running Example 1 – DDH

As demonstrated in [CS02, Section 8.1.1] the projective hash function $\mathsf{H}_{\mathsf{ddh}}$ is 0-smooth.

Lemma 3.15. $\mathsf{H}_{\mathsf{ddh}}$ is 0-smooth (over G on G), i.e. for $\kappa_0, \kappa_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, $\alpha \in \mathbf{Z}/q\mathbf{Z}$, and $\beta \in \mathbf{Z}/q\mathbf{Z}^*$, the following distributions are identical:

$$\mathcal{U} = \left\{(g_0^\alpha, g_1^{\alpha+\beta}), g_0^{\kappa_0} g_1^{\kappa_1}, g_1^\gamma \mid \gamma \hookleftarrow \mathbf{Z}/q\mathbf{Z}\right\} \text{ and } \mathcal{V} = \left\{(g_0^\alpha, g_1^{\alpha+\beta}), g_0^{\kappa_0} g_1^{\kappa_1}, g_0^{\alpha\kappa_0} g_1^{(\alpha+\beta)\kappa_1}\right\}.$$

Running Example 2 – HSM-CL

Recall that $F = \langle f \rangle$ is the subgroup of G of order q in which the discrete logarithm problem is easy; that $G \simeq G^q \times F$ where $G^q = \langle g_q \rangle$ is of order s; and that ϖ is the group exponent of $\widehat{G}^q$, as such, ϖ divides s.

In Lemma 3.16 we demonstrate that for an appropriate choice of $\widehat{\mathcal{D}}$, from which hashing keys are sampled, the projective hash function $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is smooth over G on F.

Lemma 3.16 (smoothness)**.** If $\widehat{\mathcal{D}}$ is a distribution δ-close to $\mathcal{U}(\mathbf{Z}/q\varpi\mathbf{Z})$, then $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is δ_s-smooth over G on F, with $\delta_s \leqslant 2\delta$.

Proof. For $x \in G\backslash G^q$, there exist $a \in \mathbf{Z}/s\mathbf{Z}$ and $b \in (\mathbf{Z}/q\mathbf{Z})^*$ s.t. $x = g_q^a f^b$. Thus, for $\mathsf{hk} \hookleftarrow \widehat{\mathcal{D}}$, the task is to evaluate the statistical distance between distributions $\{g_q^a f^b$, $\mathsf{hk} \bmod \varpi, g_q^{a\cdot\mathsf{hk}} f^{b\cdot\mathsf{hk}+\gamma} | \gamma \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})\}$ and $\{g_q^a f^b, \mathsf{hk} \bmod \varpi, g_q^{a\cdot\mathsf{hk}} f^{b\cdot\mathsf{hk}}\}$. Clearly it suffices to study the distance between the third coordinates of both distributions, given the two first coordinates, *i.e*, knowing $a \bmod s$, $b \bmod q$, and $\mathsf{hk} \bmod \varpi$.

Given this information the value of $g_q^{a\cdot\mathsf{hk}}$ is fixed, since the order s of g_q divides ϖ. Hence we evaluate the statistical distance between the distribution followed by $Y := b \cdot \mathsf{hk} \bmod q$ and $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, conditioned on the knowledge of $a \bmod s$, $b \bmod q$, and $\mathsf{hk} \bmod \varpi$. As hk is sampled from $\widehat{\mathcal{D}}$, which is δ-close to $\mathcal{U}(\mathbf{Z}/q\varpi\mathbf{Z})$, and since $\gcd(q, \varpi) = 1$, from Lemma 2.15 it holds that even knowing $\mathsf{hk} \bmod \varpi$ the distribution followed by $\mathsf{hk} \bmod q$ is 2δ-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Thus the distance between $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$ and the distribution followed by Y is upper bounded by 2δ, which concludes the proof. □

Running Example 3 – DDH-f

In Lemma 3.17 we demonstrate that for an appropriate choice of $\widehat{\mathcal{D}}$, from which hashing key components are sampled, the projective hash function $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is smooth over G on F.

Lemma 3.17 (smoothness)**.** If $\widehat{\mathcal{D}}$ is a distribution δ-close to $\mathcal{U}(\mathbf{Z}/q\varpi\mathbf{Z})$, then $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is δ_s-smooth over G on F, with $\delta_s \leqslant 2\delta$.

Proof. For $(y_0, y_1) \in (\langle(g, h), (1, f)\rangle)\backslash\langle(g, h)\rangle$, there exist $a \in \mathbf{Z}$ and $b \in (\mathbf{Z}/q\mathbf{Z})^*$ s.t. $(y_0, y_1) = (g^a, h^a f^b)$. Thus, for $\kappa_0, \kappa_1 \hookleftarrow \widehat{\mathcal{D}}$, the task is to evaluate the statistical distance between distributions $\{(g^a, h^a f^b)$, $(\kappa_0 \bmod \varpi, \kappa_1 \bmod \varpi, \kappa_0 + \alpha\kappa_1 \bmod q), g^{a\cdot\kappa_0} h^{a\cdot\kappa_1} f^{b\cdot\kappa_1+\gamma} | \gamma \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})\}$ and $\{(g^a, h^a f^b)$, $(\kappa_0 \bmod \varpi, \kappa_1 \bmod \varpi, \kappa_0 + \alpha\kappa_1 \bmod q), g^{a\cdot\kappa_0} h^{a\cdot\kappa_1} f^{b\cdot\kappa_1}\}$. It suffices to study the distance between the third coordinates of both distributions, given the two first coordinates, *i.e* knowing $a \bmod n$, $b \bmod q$, $\kappa_0 \bmod \varpi$, $\kappa_1 \bmod \varpi$ and $\kappa_0 + \alpha\kappa_1 \bmod q$. Given this information, since s divides ϖ and $n = qs$, the value $\kappa_0 + \alpha\kappa_1 \bmod n$ is also known. Hence $g^{a\cdot\kappa_0} h^{a\cdot\kappa_1} = g^{a(\kappa_0+\alpha\kappa_1)}$ is fixed information theoretically, and so it suffices to compare the distribution followed by the random variable $Y := b \cdot \kappa_1 \bmod q$ and the uniform distribution modulo q, conditioned on the knowledge of $a \bmod n$, $b \bmod q$, $\kappa_0 \bmod \varpi$, $\kappa_1 \bmod \varpi$ and $\kappa_0 + \alpha\kappa_1 \bmod q$.

Let us denote $\pi_0 := \kappa_0 + \alpha\kappa_1 \bmod q$, which is fixed given by the second coordinate. Since κ_0, κ_1 are sampled from $\widehat{\mathcal{D}}$, knowing π_0 the joint distribution of $(\kappa_0 \bmod q, \kappa_1 \bmod q)$ is:

$$\{(\pi_0 - \alpha k, k) | k \hookleftarrow \widehat{\mathcal{D}}\}.$$

Thus the value of $\kappa_0 \bmod q$ is fixed by that of $\kappa_1 \bmod q$. Now let us consider the additional information on $(\kappa_0 \bmod q, \kappa_1 \bmod q)$ fixed by the knowledge of $\kappa_0 \bmod \varpi$, $\kappa_1 \bmod \varpi$. As $\widehat{\mathcal{D}}$ is at statistical distance δ from $\mathcal{U}(\mathbf{Z}/q\varpi\mathbf{Z})$, and $\gcd(q, \varpi) = 1$, from Lemma 2.15 it holds that even knowing $\kappa_1 \bmod \varpi$ the distribution followed by $\kappa_1 \bmod q$ is 2δ-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Finally since $b \neq 0 \bmod q$, Y follows a distribution 2δ-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, which concludes the proof. □

3.3.5 Decomposability

We introduce the notion of a *decomposable* projective hash function, this property states that the domain of hash is the direct product of the language and some cyclic subgroup. Since –

given the projection key – one can publicly compute hash values over elements in the language, decomposability allows to have a clear separation between the part of a given hash value which is predictable (whose pre-image is in the language), and the part which must appear random for security to hold.

Definition 3.18. Let $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be a subgroup membership problem, and consider the associated PHF H. We say that H is $(\widehat{\Upsilon}, \Upsilon, F)$*-decomposable* if the co-domain Π of hash is a finite Abelian group which contains a cyclic subgroup F, and there exist $\widehat{\Upsilon} \in \widehat{\mathcal{X}}$ and $\Upsilon \in \mathcal{X}$ s.t.:

- $\mathcal{X} \simeq \mathcal{L} \times \langle \Upsilon \rangle$;
- $\widehat{\mathcal{X}} \simeq \widehat{\mathcal{L}} \times \langle \widehat{\Upsilon} \rangle$;
- $\forall \mathsf{hk} \in K_{\mathsf{hk}}$ it holds that $\mathsf{hash}(\mathsf{hk}, \Upsilon) \in F$ and $\mathsf{hash}(\mathsf{hk}, \widehat{\Upsilon}) \in F$.

Notation. If $\widehat{\mathcal{X}} = \mathcal{X}$ or $\widehat{\Upsilon} = \Upsilon$, we simply say H is (Υ, F)-decomposable.

Remark. By sampling $(\tilde{x}, \tilde{w}) \leftarrow \mathsf{R}$, $y \hookleftarrow \langle \Upsilon \rangle \setminus \{1\}$, and outputting $x := \tilde{x} \cdot y$ one induces a sampling on $\mathcal{X} \backslash \mathcal{L}$. We denote this sampling $x \hookleftarrow \mathcal{X} \backslash \mathcal{L}$.

For a homomorphic and $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable PHF, we have another formulation for smoothness which is easier to handle, and more precise in the sense that we need only consider the distribution of elements in the subgroup $\langle \Upsilon \rangle$ and their image by hash, rather than that of elements in the group $\mathcal{X}$.

Lemma 3.19. A homomorphic and $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable PHF is δ_s-smooth over $\mathcal{X}$ on F if the following distributions are δ_s-close

$$\{(y, \widehat{\mathsf{projkg}}(\mathsf{hk}), \mathsf{hash}(\mathsf{hk}, y)) | \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM}), y \hookleftarrow \langle \Upsilon \rangle\}$$

and

$$\{(y, \widehat{\mathsf{projkg}}(\mathsf{hk}), \pi' \cdot \mathsf{hash}(\mathsf{hk}, y)) | \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM}), y \hookleftarrow \langle \Upsilon \rangle, \pi' \hookleftarrow \mathcal{U}(F)\}.$$

Running Example 1 – DDH

The group G is cyclic, and a trivial subgroup of itself, so we take $F := G$, and set:

$$\Upsilon := (1, g_1),$$

such that $G^2 = \langle (1, g_1), (g_0, g_1) \rangle$, and it holds that:

$$G^2 \simeq \langle (1, g_1) \rangle \times \langle (g_0, g_1) \rangle.$$

Clearly $\forall \mathsf{hk} \in (\mathbf{Z}/q\mathbf{Z})^2$, $\mathsf{hash}(\mathsf{hk}, (1, g_1)) \in G$. For any $(y_0, y_1) \in G^2$, we have $y_0 = g_0^{w_0}$ and $y_1 = g_1^{w_1}$. Denoting $x_0 := y_0 = g_0^{w_0}$, $x_1 := g_1^{w_0}$, and $\upsilon := w_1 - w_0 \in \mathbf{Z}/q\mathbf{Z}$ it holds that $((x_0, x_1), w_0) \in \mathsf{R}_{\mathsf{DDH}}$; $(y_0, y_1) = (1, g_1)^\upsilon \odot (x_0, x_1)$; and given (y_0, y_1) such values of $(x_0, x_1), w_0, \upsilon$ are unique modulo q. Hence:

$$\mathsf{H}_{\mathsf{ddh}} \text{ is } ((1, g_1), G)\text{-decomposable.}$$

Running Example 2 – HSM-CL

Recall that F denotes the cyclic subgroup of $\widehat{G}$ generated by f, and by definition:

$$\widehat{G} \simeq \widehat{G}^q \times F \quad \text{and} \quad G \simeq G^q \times F.$$

Moreover $\forall \mathsf{hk} \in \mathbf{Z},\ \mathsf{hash}(\mathsf{hk}, f) = f^{\mathsf{hk}} \in F$. Thus we set:

$$\widehat{\Upsilon} = \Upsilon := f,$$

and consequently:

$$\mathsf{H}_{\mathsf{hsm\text{-}cl}} \text{ is } (f, F)\text{-decomposable.}$$

Running Example 3 – DDH-f

Recall that F denotes the cyclic subgroup of $\widehat{G}$ generated by f, and by definition:

$$\widehat{G} \simeq \widehat{G}^q \times F \quad \text{and} \quad G \simeq G^q \times F.$$

Further recall that $\widehat{\mathcal{X}} = \widehat{G}^2$ and that $\widehat{\mathcal{L}} := \{(z_0 f^r, z_1 f^{\alpha r}) | z_0, z_1 \in \widehat{G}^q\,; r \in \mathbf{Z}/q\mathbf{Z}\}$. It holds that:

$$\widehat{\mathcal{X}} \simeq \widehat{\mathcal{L}} \times \langle (1, f) \rangle.$$

Similarly $\mathcal{X} = \langle (g, h), (1, f) \rangle$ and $\mathcal{L} = \langle (g, h) \rangle$, so:

$$\mathcal{X} \simeq \langle (g, h) \rangle \times \langle (1, f) \rangle.$$

Moreover for all $(\kappa_0, \kappa_1) \in \mathbf{Z}^2$, $\mathsf{hash}((\kappa_0, \kappa_1), (1, f)) = f^{\kappa_1} \in F$. Thus we set:

$$\widehat{\Upsilon} = \Upsilon := (1, f),$$

and consequently:

$$\mathsf{H}_{\mathsf{ddh\text{-}f}} \text{ is } ((1, f), F)\text{-decomposable.}$$

3.4 Public Key Encryption from Projective Hash Functions

In this section we recall generic constructions for building a public key encryption schemes secure against passive and active adversaries from projective hash functions. The generic construction to attain ind-cca-security is an adaptation of the Cramer-Shoup [CS02] generic construction, while that attaining ind-cpa-security is a folklore simplification thereof. We have adapted these constructions so that plaintext messages are encoded in the exponent of f, a generator for the subgroup F on which the considered PHF is assumed to be smooth. If the PHF also satisfies some of the homomorphic properties of Section 3.3.3 we obtain ind-cpa-secure schemes which are linearly homomorphic.

In order to attain security against active adversaries, we recall the definition of *extended projective hash functions* (in Section 3.4.3), as these are required to ensure ciphertext integrity. Finally we provide instantiations for extended projective hash functions from our running examples.

3.4.1 Security against Passive Adversaries

In Fig. 3.2 we provide a folklore generic construction for $\mathsf{ind\text{-}cpa}$-secure PKE from projective hash functions. This construction is a simplification of the $\mathsf{ind\text{-}cca}$-secure generic construction put forth in [CS02] (*cf.* Section 3.4.2). The high level idea is to mask the confidential message m with the hash of a random value for some projective hash function H. The hardness of the subset membership problem underlying H and the smoothness of H ensure that the hash acts as a one time pad and statistically hides the message.

Requirements for the projective hash function. Let $\mathcal{SM} = (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be a subgroup membership problem. Consider the associated PHF H, whose hash values belong to a finite Abelian group Π. We suppose there exists a cyclic subgroup F of Π generated by f, of order q, in which computing discrete logarithms can be done efficiently. We adapt the traditional construction to our setting, and encode m in the exponent of f (the generator of the subgroup F) before masking f^m with the hash of a random value. This ensures that if H is homomorphic, the resulting encryption scheme will be linearly homomorphic. To build a PKE scheme $\mathcal{E}$ with plaintext space $\mathbf{Z}/q\mathbf{Z}$ from H, one defines $\mathcal{E} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ as described in Fig. 3.2. Correctness of $\mathcal{E}$ follows from that of H.

```
Setup($1^\lambda$)
  1. $\mathcal{SM} \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$
  2. Return $\mathsf{pp} := \mathcal{SM}$

KeyGen(pp)
  1. $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM})$
  2. $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$
  3. Set $\mathsf{pk} := \mathsf{hp}$ and $\mathsf{sk} := \mathsf{hk}$
  4. Return $(\mathsf{pk}, \mathsf{sk})$

Enc(hp, $m$)
  1. Pick $(x, w) \leftarrow \mathsf{R}$
  2. Let $e \leftarrow \mathsf{projhash}(\mathsf{hp}, x, w) \cdot f^m$
  3. Return $(x, e)$
```

```
Dec(hk, $(x, e)$)
  1. Compute $M \leftarrow e \cdot \mathsf{hash}(\mathsf{hk}, x)^{-1}$
  2. If $M \notin F$ output $\perp$
  3. Return $\log_f(M)$

EvalSum(hp, $(x, e)$, $(x', e')$)
  1. Pick $(x'', w'') \leftarrow \mathsf{R}$
  2. Let $x'' \leftarrow x \cdot x' \cdot x''$
  3. Let $e'' \leftarrow e \cdot e' \cdot \mathsf{projhash}(\mathsf{hp}, x'', w'')$
  4. Return $(x'', e'')$

EvalScal(hp, $(x, e)$, $\alpha$)
  1. Pick$(x', w') \leftarrow \mathsf{R}$
  2. Let $x'' := x^\alpha \cdot x'$
  3. Let $e'' := e^\alpha \cdot \mathsf{projhash}(\mathsf{hp}, x', w')$
  4. Return $(x'', e'')$
```

Figure 3.2: Linearly homomorphic encryption scheme $\mathcal{E}$ from a homomorphic PHF

Homomorphic properties. If H is homomorphic (*cf.* Definition 3.11), then algorithms $\mathsf{EvalSum}$ and $\mathsf{EvalScal}$ of Fig. 3.2 correctly implement those of Definition 2.6, and so $\mathcal{E}$ is linearly homomorphic.

Security. The following theorem states the requirements on $\mathcal{SM}$ and H for $\mathcal{E}$ to be $\mathsf{ind\text{-}cpa}$-secure. This is a well known result and indeed there is nothing new about the proof provided. However as many of our more complex upcoming proofs of Chapter 4 follow a similar structure, we provide details here, mainly for readers who may not be familiar with PHFs.

Theorem 3.20. Let $\mathcal{E}$ be the public key encryption scheme depicted in Fig. 3.2. If $\mathcal{SM}$ is $\delta_{\mathcal{L}}$-hard and H is δ_s-smooth over $\mathcal{X}$ on F then for any PPT adversary $\mathcal{A}$ it holds that $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E},\mathcal{A}} \leqslant \delta_{\mathcal{L}} + \delta_s$. Consequently, if $\mathcal{SM}$ is hard and H is smooth, then $\mathcal{E}$ is $\mathsf{ind\text{-}cpa}$-secure.

Proof. The proof proceeds as a sequence of games, starting in Game_0 which is the $\mathsf{ind\text{-}cpa}$-security experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E},\mathcal{A}}$ (*cf.* Section 2.5.1), and ending in a game where the challenge bit β is statistically hidden from $\mathcal{A}$'s view. By demonstrating each game step is indistinguishable from $\mathcal{A}$'s view, we prove that $\mathcal{A}$'s probability of guessing the correct bit β in $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E},\mathcal{A}}$ is negligibly close to 1/2, which concludes the proof. Let S_i denote the event $\mathcal{A}$ outputs β in Game_i. By definition $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E},\mathcal{A}} = |\Pr[S_0] - 1/2|$.
In Game_1, the challenger $\mathcal{C}$ uses hk instead of hp and the witness to compute the ciphertext, *i.e.* after sampling $(x, w) \leftarrow \mathsf{R}$ it computes:

$$c \leftarrow (x, \mathsf{hash}(\mathsf{hk}, x) f^{m_\beta}).$$

Other than this change Game_1 is identical to Game_0, and so by correctness of H the view of $\mathcal{A}$ does not change. Hence:

$$|\Pr[S_1] - \Pr[S_0]| = 0.$$

In Game_2, $\mathcal{C}$ samples $x \hookleftarrow \mathcal{X} \backslash \mathcal{L}$ instead of sampling x from $\mathcal{L}$. Other than this change Game_2 is identical to Game_1, and so by the $\delta_{\mathcal{L}}$-hardness of $\mathcal{SM}$, adversary $\mathcal{A}$ cannot distinguish both games with probability greater than $\delta_{\mathcal{L}}$. Hence:

$$|\Pr[S_2] - \Pr[S_1]| \leqslant \delta_{\mathcal{L}}.$$

In Game_3, $\mathcal{C}$ samples $\gamma \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and multiplies the second element of the ciphertext by f^γ. By the smoothness over $\mathcal{X}$ on F, the distribution of $\mathcal{A}$'s view in Game_2 and Game_3 is δ_s close, so :

$$|\Pr[S_3] - \Pr[S_2]| \leqslant \delta_s.$$

Finally observe that in Game_3, the adversary receives an encryption of $\gamma + m_\beta$, where γ is sampled uniformly from $\mathbf{Z}/q\mathbf{Z}$, and so is a perfect one time pad for the message m_β. Hence $\mathcal{A}$'s probability of guessing β is exactly 1/2, and:

$$Pr[S_3] = 1/2.$$

Combining the above probability equations concludes the proof. □

Remark. We emphasize that in Cramer-Shoup like encryption schemes resulting from PHFs (as described above), the smoothness of the PHF allows for the challenger to know hk without compromising the hardness of the underlying subset membership problem. This property will be particularly useful for building secure distributed signature protocols in Chapter 5.

Invalid ciphertexts. We define invalid ciphertexts as these will be useful in our upcoming proofs.

Definition 3.21. Let $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be a subgroup membership problem. Consider the associated PHF H, whose hash values belong to a finite Abelian group Π. We suppose there exists a cyclic subgroup F of Π generated by f, of order q, in which computing discrete logarithms can be done efficiently. Consider the resulting PKE scheme $\mathcal{E}$ as described in Fig. 3.2. A pair (x, e) is said to be an *invalid* ciphertext for $\mathcal{E}$ if it is of the form $(x, e) = (x, \mathsf{hash}(\mathsf{hk}, x) \cdot f^m)$ where $x \in \widehat{\mathcal{X}} \backslash \widehat{\mathcal{L}}$.

Note that one can compute such a ciphertext using the secret hashing key hk, but not the public projection key hp; that the decryption algorithm applied to (x, e) with secret key hk recovers m; and that an invalid ciphertext is indistinguishable of a real ciphertext if $\mathcal{SM}$ is hard.

Homomorphic properties over invalid ciphertexts. Since the homomorphic properties of H hold over the whole group $\widehat{\mathcal{X}}$, if H is homomorphic, then homomorphic operations hold even if a ciphertext is invalid, whether this be between two invalid ciphertexts or between a valid and invalid ciphertext.

3.4.2 Security against Active Adversaries

As explained in Section 3.4.1, smooth PHFs ensure confidentiality for public key encryption protocols. They thereby guarantee security against passive adversaries. In order to further ensure that active adversaries cannot learn sensitive information by running algorithms on unexpected inputs, one must enforce ciphertext integrity. To this end a second PHF is used, called an extended projective hash function (as first introduced by [CS02]), which allows to perform a sanity check on the ciphertext, prior to revealing any confidential information. An extended projective hash function (EPHF) for an instance of a subgroup membership problem $\mathcal{SM}$ is defined as a PHF, only the hashing and projective hashing algorithms take an additional input from some efficiently recognisable finite set E (we will formally define EPHFs in Section 3.4.3).

We here recall the ideas underlying the generic construction of [CS02] allowing build ind-cca-secure PKE. We do not formally define this generic construction, as we will not be detailing ind-cca-secure encryption schemes in this work. We could build such schemes from our running examples, however one then looses the homomorphic properties of the encryption scheme, which are our main focus of interest to build more advanced protocols in upcoming chapters.

We note however that the intuition underlying the security of the generic [CS02] construction for ind-cca-secure PKE, and in particular the distinction between valid and invalid ciphertexts as explained hereafter, is similar to that used in Chapter 4 to build functional encryption schemes secure against active adversaries. Furthermore, the extended projective hash functions defined in Section 3.4.3 will be needed to build these functional encryption schemes.

Consider an ind-cpa-secure PKE built from a projective hash function H, as described in Fig. 3.2. Recall that ciphertexts are of the form $c = (x, e)$. One augments the decryption key sk of the PKE with a private hashing key ehk for an extended projective hash function eH, and one adds eH's public projection key ehp to the public encryption key pk. To further secure the scheme against active adversaries, upon encryption, one computes a hash of (x, e) using the the witness w associated to x and eH's public projective hash function to obtain a hash value π. The resulting ciphertext for the ind-cca-secure scheme is (x, e, π). Intuitively, it should be infeasible, without ehk, to compute hash values for eH if $x \notin \widehat{\mathcal{L}}$. Hence if the ciphertext (x, e) were invalid (*cf.* Definition 3.21), one should not be able to guess π without the knowledge of sk[2]. Furthermore invalid ciphertexts are the *only* ciphertexts which, if decrypted, can leak more (relevant) information than what an adversary learns in an $\mathsf{ind\text{-}cpa}$ attack (and hence may leak harmful information to the scheme's security).

Now upon receiving a ciphertext (x, e, π) (which may be invalid as we are now dealing with active adversaries), the decryptor first computes the hash of (x, e) using ehk and the private hashing algorithm of eH to obtain π', and checks that $\pi = \pi'$. If equality holds, it pursues decryption as in the $\mathsf{ind\text{-}cpa}$-protocol, if not it aborts the decryption protocol, returning the error symbol $\perp$, so that no information is leaked other than the fact the hash value π was wrong. In particular no information is leaked on the hashing key hk used to mask confidential information, or on the encrypted message m.

Finally since all invalid ciphertexts are rejected, the adversary learns no more (significant) information than it would in an $\mathsf{ind\text{-}cpa}$ attack. And hence the $\mathsf{ind\text{-}cca}$-security of the scheme follows from Theorem 3.20.

3.4.3 Extended Projective Hash Functions

We here formally define extended projective hash functions, and provide concrete instances from our running examples. Definition 3.22 is an adaptation of the definition which was first provided in [CS02], as once again we introduce additional algorithms to deal with groups of unknown order and unrecognisable elements.

Definition 3.22. Let λ be a positive integer, and let E be an efficiently recognisable finite set. Consider a generator $\mathsf{Gen}_{\mathcal{SM}}$ for a subgroup membership problem, and an instance $\mathcal{SM} \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$. An extended projective hash function (EPHF) for the subgroup membership problem $\mathcal{SM}$ and auxiliary input space E is a tuple of algorithms $\mathsf{eH} := (\mathsf{ehashkg}, \widehat{\mathsf{eprojkg}}, \mathsf{eprojkg}, \mathsf{ehash}, \widehat{\mathsf{eprojhash}}, \mathsf{eprojhash})$, where all algorithms have as implicit input the description $\mathcal{SM}$ of the subset membership problem and:

- $\mathsf{ehashkg}(\mathcal{SM})$ is a PPT algorithm which on input the description of $\mathcal{SM}$, outputs a hashing key ehk in some set K_{ehk};

- $\widehat{\mathsf{eprojkg}}(\mathsf{ehk})$ is a deterministic algorithm which on input $\mathsf{ehk} \in K_{\mathsf{ehk}}$ outputs a projection key $\widehat{\mathsf{ehp}}$. The image of K_{ehk} through $\widehat{\mathsf{eprojkg}}$ is denoted $K_{\widehat{\mathsf{ehp}}}$;

- $\mathsf{eprojkg}(\mathsf{hk})$ is a DPT algorithm which on input $\mathsf{ehk} \in K_{\mathsf{ehk}}$ outputs a public projection key ehp, such that for $\mathsf{ehk} \in K_{\mathsf{ehk}}$, ehp is a fixed deterministic function of the output of $\widehat{\mathsf{eprojkg}}(\mathsf{ehk})$. The image of K_{ehk} through $\mathsf{eprojkg}$ is denoted K_{ehp}.

- $\mathsf{ehash}(\mathsf{hk}, x, e)$ is a DPT algorithm which on input $\mathsf{ehk} \in K_{\mathsf{ehk}}$, $(x, e) \in \widehat{\mathcal{X}} \times E$ outputs the hash value $\mathsf{ehash}(\mathsf{hk}, x, e)$. The image of $\widehat{\mathcal{X}} \times E$ through ehash is called the set of hash values and is denoted Σ;

- $\widehat{\mathsf{eprojhash}}(\widehat{\mathsf{ehp}}, x)$ is a deterministic algorithm which on input $\widehat{\mathsf{ehp}} \in K_{\widehat{\mathsf{ehp}}}$, $x \in \widehat{\mathcal{L}}$, and $e \in E$ outputs the hash value $\widehat{\mathsf{eprojhash}}(\widehat{\mathsf{ehp}}, x, e)$ in Σ;

- $\mathsf{eprojhash}(\mathsf{ehp}, x, w, e)$ is a DPT algorithm which on input $\mathsf{ehp} \in K_{\mathsf{ehp}}$, $x \in \mathcal{L}$, the corresponding witness $w \in \mathcal{W}$ and $e \in E$, outputs the hash value $\mathsf{projhash}(\mathsf{hp}, x, w, e)$ in Σ.

[2] This property for eH is called 'universal$_2$' in [CS02], it can easily be seen that it is implied by our notion of vector-universality, defined in Chapter 4, allowing to build functional encryption secure against active adversaries

Correctness holds if for all $\lambda \in \mathbf{N}$, any $\mathcal{SM} \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$, any $\mathsf{ehk} \leftarrow \mathsf{ehashkg}(\mathcal{SM})$, $\widehat{\mathsf{ehp}} \leftarrow \widehat{\mathsf{eprojkg}}(\mathsf{ehk})$ and $\mathsf{ehp} \leftarrow \mathsf{eprojkg}(\mathsf{ehk})$ it holds that (1) $\forall (x, e) \in \widehat{\mathcal{L}} \times E$, $\widehat{\mathsf{eprojhash}}(\widehat{\mathsf{ehp}}, x, e) = \mathsf{ehash}(\mathsf{ehk}, x, e)$; and (2) for any $(x, w) \in \mathsf{R}$ and $e \in E$, $\mathsf{eprojhash}(\mathsf{ehp}, x, w, e) = \mathsf{ehash}(\mathsf{ehk}, x, e)$.

Remark. The notions of homomorphism and key homomorphism can be adapted to EPHFs in a straightforward way and must hold for any $e \in E$.

Generic construction for building EPHF

We use the generic construction of [CS02, Sec. 7.2] to build EPHFs, which we recall here for completeness. Let λ be a positive integer, $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R}) \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$ be a subgroup membership problem, and consider the associated PHF $\mathsf{H} := (\mathsf{hashkg}, \widehat{\mathsf{projkg}}, \mathsf{projkg}, \mathsf{hash}, \widehat{\mathsf{projhash}}, \mathsf{projhash})$. Let $\tilde{p}$ be the smallest prime dividing $|\widehat{\mathcal{X}}/\widehat{\mathcal{L}}|$, and further consider $\Gamma : \widehat{\mathcal{X}} \times E \mapsto \{0, \dots, \tilde{p}-1\}$, sampled from a collision resistant hash function generator $\mathcal{H}$. The generic construction to build an EPHF eH with auxiliary input space E from H and Γ works as follows:

- $\mathsf{ehashkg}(\mathcal{SM})$: Run $\mathsf{hk}_0 \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$; $\mathsf{hk}_1 \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$. Output $\mathsf{ehk} := (\mathsf{hk}_0, \mathsf{hk}_1)$.
- $\mathsf{ehash}(\mathsf{ehk}, x, e)$: Let $\gamma \leftarrow \Gamma(x, e)$. Output $\mathsf{hash}(\mathsf{hk}_0, x) \cdot \mathsf{hash}(\mathsf{hk}_1, x)^\gamma$.
- $\widehat{\mathsf{eprojkg}}(\mathsf{ehk})$: Let $\widehat{\mathsf{hp}}_0 \leftarrow \widehat{\mathsf{projkg}}(\mathsf{hk}_0)$; $\widehat{\mathsf{hp}}_1 \leftarrow \widehat{\mathsf{projkg}}(\mathsf{hk}_1)$. Output $\widehat{\mathsf{ehp}} := (\widehat{\mathsf{hp}}_0, \widehat{\mathsf{hp}}_1)$.
- $\mathsf{eprojkg}(\mathsf{ehk})$: Let $\mathsf{hp}_0 \leftarrow \mathsf{projkg}(\mathsf{hk}_0)$; $\mathsf{hp}_1 \leftarrow \mathsf{projkg}(\mathsf{hk}_1)$. Output $\mathsf{ehp} := (\mathsf{hp}_0, \mathsf{hp}_1)$.
- $\widehat{\mathsf{eprojhash}}(\widehat{\mathsf{ehp}}, \widehat{x}, e)$: Let $\gamma \leftarrow \Gamma(\widehat{x}, e)$. Output $\widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}_0, \widehat{x}) \cdot \widehat{\mathsf{projhash}}(\widehat{\mathsf{hp}}_1, \widehat{x})^\gamma$.
- $\mathsf{eprojhash}(\mathsf{ehp}, x, w, e)$: Let $\gamma \leftarrow \Gamma(x, e)$. Output $\mathsf{projhash}(\mathsf{hp}_0, x, w) \cdot \mathsf{projhash}(\mathsf{hp}_1, x, w)^\gamma$.

The hash key space of eH is $K_{\mathsf{ehk}} = K_{\mathsf{hk}}^2$, the set of projection keys is $K_{\widehat{\mathsf{ehp}}} = K_{\widehat{\mathsf{hp}}}^2$, the set of public projection keys is $K_{\mathsf{ehp}} = K_{\mathsf{hp}}^2$ and the set of hash values is $\Sigma = \Pi$.

Lemmas 3.23 and 3.24 follow immediately from the above construction and the definitions of decomposability (*cf.* Definition 3.18) and key homomorphism (*cf.* Definition 3.13).

Lemma 3.23. An EPHF built from a $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable PHF via the above generic construction is also $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable. Furthermore, if H is $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable then $\tilde{p}$, the smallest prime dividing $|\widehat{\mathcal{X}}/\widehat{\mathcal{L}}|$, which parametrises Γ, is the smallest prime dividing the order of $\widehat{\Upsilon}$.

Lemma 3.24. Assuming $(K_{\mathsf{hk}}, +)$ is an Abelian group, let the group operation over K_{hk}^2 be coordinate-wise addition. Then an EPHF eH built from a key homomorphic PHF H via the above generic construction is also key homomorphic.

Remark. Note that due to the action of the collision resistant hash function Γ, which does *not* possess any homomorphic properties, and which takes as input (x, e), even if H is homomorphic, eH is not.

Running Example 1 – DDH

We set $E := G$ and sample $\Gamma : G^3 \mapsto \{0, \dots, q-1\}$ from a collision resistant hash function generator $\mathscr{H}$. The extended projective hash function $\mathsf{eH}_{\mathsf{ddh}}$ has hash key space $K_{\mathsf{ehk}} := \mathbf{Z}/q\mathbf{Z}^4$; set of projection keys $K_{\mathsf{ehp}} := G^2$; set of hash values $\Sigma = G$; and is defined from $\mathsf{H}_{\mathsf{ddh}}$ as:

- $\mathsf{ehashkg}(\mathcal{SM})$ samples $\kappa_0, \kappa_1, \kappa_2, \kappa_3 \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, and outputs $\mathsf{ehk} = (\kappa_0, \kappa_1, \kappa_2, \kappa_3)$.
- $\mathsf{ehash}(\mathsf{ehk}, (x_0, x_1), e)$ computes $\gamma \leftarrow \Gamma(x_0, x_1, e)$ and outputs $x_0^{\kappa_0+\gamma\kappa_2} x_1^{\kappa_1+\gamma\kappa_3}$.
- $\mathsf{eprojkg}(\mathsf{ehk})$ computes $\mathsf{hp}_0 := g_0^{\kappa_0} g_1^{\kappa_1}$; $\mathsf{hp}_1 := g_0^{\kappa_2} g_1^{\kappa_3}$, and outputs $\mathsf{ehp} := (\mathsf{hp}_0, \mathsf{hp}_1)$.
- $\mathsf{eprojhash}(\mathsf{hp}_0, \mathsf{hp}_1, (x_0, x_1), w, e)$ where $(x_0, x_1) = (g_0^w, g_1^w)$ computes $\gamma \leftarrow \Gamma(x_0, x_1, e)$ and outputs $(\mathsf{hp}_0 \cdot \mathsf{hp}_1^{\gamma})^w$.

Running Example 2 – HSM-CL

Let $E := \widehat{G}$ and sample $\Gamma : \widehat{G}^2 \mapsto \{0, \dots, q-1\}$ from a collision resistant hash function generator $\mathscr{H}$. The extended projective hash function $\mathsf{eH}_{\mathsf{hsm\text{-}cl}}$ has hash key space $K_{\mathsf{ehk}} := \mathbf{Z}^2$; set of projection keys $K_{\widehat{\mathsf{ehp}}} := (\mathbf{Z}/\varpi\mathbf{Z})^2$; set of public projection keys $K_{\mathsf{ehp}} := (G^q)^2$; set of hash values $\Sigma = \widehat{G}$; and is defined from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ as:

- $\mathsf{ehashkg}(\mathcal{SM})$ samples $\mathsf{hk}_0 \hookleftarrow \widehat{\mathscr{D}}$; $\mathsf{hk}_1 \hookleftarrow \widehat{\mathscr{D}}$; and outputs $\mathsf{ehk} := (\mathsf{hk}_0, \mathsf{hk}_1)$.
- $\mathsf{ehash}(\mathsf{ehk}, x, e)$, computes $\gamma \leftarrow \Gamma(x, e)$ and outputs $x^{\mathsf{hk}_0+\gamma\mathsf{hk}_1}$.
- $\widehat{\mathsf{eprojkg}}(\mathsf{ehk})$, computes $\widehat{\mathsf{hp}}_0 := \mathsf{hk}_0 \bmod \varpi$, $\widehat{\mathsf{hp}}_1 := \mathsf{hk}_1 \bmod \varpi$, and outputs $\widehat{\mathsf{ehp}} := (\widehat{\mathsf{hp}}_0, \widehat{\mathsf{hp}}_1)$.
- $\mathsf{eprojkg}(\mathsf{ehk})$, computes $\mathsf{hp}_0 := g_q^{\mathsf{hk}_0}$ and $\mathsf{hp}_1 := g_q^{\mathsf{hk}_1}$, and outputs $\mathsf{ehp} := (\mathsf{hp}_0, \mathsf{hp}_1)$.
- $\widehat{\mathsf{eprojhash}}(\widehat{\mathsf{ehp}}, x, e)$, where $x \in \widehat{G}^q$, computes $\gamma \leftarrow \Gamma(\widehat{x}, e)$ and outputs $x^{\widehat{\mathsf{hp}}_0+\gamma\widehat{\mathsf{hp}}_1}$.
- $\mathsf{eprojhash}(\mathsf{ehp}, x, w, e)$, where $x = g_q^w$, computes $\gamma \leftarrow \Gamma(x, e)$ and outputs $(\mathsf{hp}_0 \cdot \mathsf{hp}_1^{\gamma})^w$.

Running Example 3 – DDH-f

We set $E := \widehat{G}$ and sample $\Gamma : \widehat{G}^3 \mapsto \{0, \dots, q-1\}$ from a collision resistant hash function generator $\mathscr{H}$. The extended projective hash function $\mathsf{eH}_{\mathsf{ddh\text{-}f}}$ has hash key space $K_{\mathsf{ehk}} := \mathbf{Z}^4$; set of projection keys $K_{\widehat{\mathsf{ehp}}} := (\mathbf{Z}/\varpi\mathbf{Z})^4 \times (\mathbf{Z}/q\mathbf{Z})^2$; set of public projection keys $K_{\mathsf{ehp}} := G^2$; set of hash values $\Sigma = \widehat{G}$; and is defined from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ as:

- $\mathsf{ehashkg}(\mathcal{SM})$ samples $\kappa_0, \kappa_1, \kappa_2, \kappa_3 \hookleftarrow \widehat{\mathscr{D}}$, and outputs $\mathsf{ehk} = (\kappa_0, \kappa_1, \kappa_2, \kappa_3)$.
- $\mathsf{ehash}(\mathsf{ehk}, (x_0, x_1), e)$ computes $\gamma \leftarrow \Gamma(x_0, x_1, e)$ and outputs $x_0^{\kappa_0+\gamma\kappa_2} x_1^{\kappa_1+\gamma\kappa_3}$.
- $\widehat{\mathsf{eprojkg}}(\mathsf{ehk})$ computes $\hat{\kappa}_i := \kappa_i \bmod \varpi$ for $i \in \{0, \dots 4\}$; $\iota_0 := \kappa_0 + \alpha\kappa_1 \bmod q$; $\iota_1 := \kappa_2 + \alpha\kappa_3 \bmod q$ and outputs $\widehat{\mathsf{ehp}} := ((\hat{\kappa}_0, \hat{\kappa}_2), (\hat{\kappa}_1, \hat{\kappa}_3), (\iota_0, \iota_1))$.
- $\mathsf{eprojkg}(\mathsf{ehk})$ computes $\mathsf{hp}_0 := g^{\kappa_0} h^{\kappa_1}$; $\mathsf{hp}_1 := g^{\kappa_2} h^{\kappa_3}$, and outputs $\mathsf{ehp} := (\mathsf{hp}_0, \mathsf{hp}_1)$.
- $\widehat{\mathsf{eprojhash}}(\widehat{\mathsf{ehp}}, (y_0, y_1), e)$ where $(y_0, y_1) = (z_0, z_1) \odot (f^r, f^{\alpha r})$ with $z_0, z_1 \in \widehat{G}^q$, computes $\gamma \leftarrow \Gamma(y_0,\, y_1, e)$ and outputs $z_0^{\hat{\kappa}_0+\gamma\hat{\kappa}_2} z_1^{\hat{\kappa}_1\gamma\hat{\kappa}_3} (f^r)^{\iota_0+\gamma\iota_1}$.
- $\mathsf{eprojhash}(\mathsf{ehp}, (x_0, x_1), r, e)$ where $(x_0, x_1) = (g^r, h^r)$ computes $\gamma \leftarrow \Gamma(x_0, x_1, e)$ and outputs
  $(\mathsf{hp}_0 \cdot \mathsf{hp}_1^{\gamma})^r = x_0^{\kappa_0+\gamma\kappa_2} x_1^{\kappa_1+\gamma\kappa_3}$.

3.5 Linearly Homomorphic PKE from the CL Framework

As mentioned in Section 2.5.2, one of the most accomplished linearly homomorphic encryption schemes is that of Paillier [Pai99] and generalisations thereof (e.g. [DJ01]). The security of these schemes is based on the problem of factoring RSA integers (including the elliptic curve variant of Paillier [Gal02]). The problem of devising a linearly homomorphic encryption based on the discrete logarithm problem was left open until quite recently, with the work of Castagnos and Laguillaumie [CL15]. Previous existing solutions either only supported a limited number of additions [CPP06, CC07], or relied not only on the discrete logarithm problem but also on the factorisation problem [BCP03].

In [CL15], Castagnos and Laguillaumie thus devised both the CL framework, and from this framework the first linearly homomorphic public key encryption scheme whose security relies solely on a discrete logarithm type assumption. What is more, their scheme, which we call Π_{cl}, benefits of the uncommon property of having a plaintext space of prime order q, where q can be chosen (with some restrictions) independently of the security parameter. This notable feature contrasts with the Paillier cryptosystem, in which the message space is of composite order N, where N is an RSA integer and is thus much larger than what is required for many practical applications.

We note that Π_{cl} does not arise from projective hash functions. Consequently one cannot isolate the different properties underlying its security as well as with PHF-based encryption schemes (resulting from the generic construction of Fig. 3.2). To address this, while preserving the unusual properties of Π_{cl}, we devise two new linearly homomorphic encryption schemes in the CL framework. Both of these are direct applications of the generic construction of Fig. 3.2 applied to the projective hash functions $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ and $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ of running examples 2 and 3. Applying Theorem 3.20, we thus obtain two ind-cpa-secure linearly homomorphic encryption schemes, which, just as Π_{cl}, have a message space of prime order q, where q can be chosen according to ones' needs. As we shall see in Chapters 4 and 5, when one uses linearly homomorphic encryption schemes or, somewhat equivalently, projective hash functions to realise advanced cryptographic primitives, this unusual property of being able to choose ones' message space has a strong impact on efficiency. We note that our encryption scheme relying on $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ has better efficiency than Π_{cl} (and than the $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ based scheme) as it allows to sample shorter exponents without compromising security. Hence in our constructions of Chapters 4 and 5, instantiations from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ yield the most efficient protocols.

In this section we first present the original encryption scheme of [CL15] before presenting the two PHF based schemes arising from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ and $\mathsf{H}_{\mathsf{ddh\text{-}f}}$, whose security rely on the HSM-CL and the DDH-f assumption respectively.

3.5.1 Original Castagnos-Laguillaumie PKE Secure under DDH-f

Castagnos and Laguillaumie put forth in [CL15] a generic construction for a linearly homomorphic encryption scheme over $\mathbf{Z}/q\mathbf{Z}$ based on a DDH-CL group with an easy DL subgroup. Their protocol is similar to the one of Bresson *et. al.* [BCP03] whose ind-cpa-security relies on the DDH assumption in $(\mathbf{Z}/N^2\mathbf{Z})^\times$, where $N = PQ$, using the arithmetic ideas of Paillier's encryption (*cf.* Fig. 2.1).

Castagnos and Laguillaumie prove their scheme is ind-cpa-secure under the DDH-CL assumption (*cf.* Definition 3.5). We demonstrate below that one can be more precise and that the security of this scheme is equivalent to the DDH-f assumption (*cf.* Definition 3.6). Note that as this encryption scheme does not result from a projective hash function, one cannot simply apply Theorem 3.20 to prove security.

The original [CL15] encryption scheme is an Elgamal type scheme in G, with plaintext message $m \in \mathbf{Z}/q\mathbf{Z}$ mapped to $f^m \in F$. Thanks to the Solve algorithm, decryption does not need a complex discrete logarithm computation.

Setting the parameters. Secret keys and encryption randomness are sampled from $\mathcal{D} := \mathcal{D}_{\mathbf{Z},\sigma}$ for $\sigma > q\tilde{s}\sqrt{\lambda}$ to ensure that the distribution $\{g^x, x \hookleftarrow \mathcal{D}\}$ is $2^{-\lambda}$-close to the uniform distribution in G (*cf.* Lemma 3.3). The plaintext space is $\mathbf{Z}/q\mathbf{Z}$, where q is a μ-bit prime, with $\mu \geqslant \lambda$. We depict their scheme in Fig. 3.3, where the Gen and Solve algorithms are those of Definition 3.1. Verifying correctness and homomorphic properties is straightforward.

```
Setup($1^\lambda$)
  1. Sample a $\mu$-bit prime $q$
  2. $\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Gen}(1^\lambda, q)$
  3. Return $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CL}}, q)$

KeyGen(pp)
  1. Pick $\alpha \hookleftarrow \mathcal{D}$ and set $h \leftarrow g^\alpha$
  2. Set $\mathsf{pk} := h$ and $\mathsf{sk} := \alpha$.
  3. Return $(\mathsf{pk}, \mathsf{sk})$

Enc(pk, $m$)
  1. Sample $r \hookleftarrow \mathcal{D}$
  2. Compute $c_1 \leftarrow g^r$, $c_2 \leftarrow f^m h^r$
  3. Return $(c_1, c_2)$

Dec(sk, $(c_1, c_2)$)
  1. Compute $M \leftarrow c_2/c_1^\alpha$
  2. $m \leftarrow \mathsf{Solve}(q, \mathsf{pp}_{\mathsf{CL}}, M)$
  3. Return $m$

EvalSum(pk, $(c_1, c_2), (c_1', c_2')$)
  1. Pick $r \hookleftarrow \mathcal{D}$
  2. Compute $c_1'' \leftarrow c_1 c_1' g^r$, $c_2'' = c_2 c_2' h^r$
  3. Return $(c_1'', c_2'')$

EvalScal(pk, $(c_1, c_2), \alpha$)
  1. Pick $r \hookleftarrow \mathcal{D}$
  2. Compute $c_1' \leftarrow c_1^\alpha g^r$, $c_2' \leftarrow c_2^\alpha h^r$
  3. Return $(c_1', c_2')$
```

Figure 3.3: Linearly homomorphic encryption scheme Π_{cl} from [CL15]

Theorem 3.25. The encryption scheme Π_{cl} of Fig. 3.3 is ind-cpa-secure if and only if the DDH-f problem is hard in G.

Proof. Let us consider the ind-cpa-security experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi_{\mathsf{cl}},\mathcal{A}}$ against adversary $\mathcal{A}$, with public key $h = g^\alpha$, where $\alpha \hookleftarrow \mathcal{D}$, and a challenge ciphertext $(c_1, c_2) = (g^r, f^{m_\beta} h^r)$ with $r \hookleftarrow \mathcal{D}$ and $\beta \hookleftarrow \{0,1\}$, $m_0, m_1 \in \mathbf{Z}/q\mathbf{Z}$. Assuming the DDH-$f$ problem is $\delta_{\mathsf{ddh\text{-}f}}$-hard, one can replace $(h, c_1, h^r) = (g^\alpha, g^r, g^{\alpha r})$ by $(g^\alpha, g^r, g^{\alpha r} f^u) = (g^\alpha, g^r, h^r f^u)$ for $u \hookleftarrow \mathbf{Z}/q\mathbf{Z}$; and $\mathcal{A}$ cannot tell the difference with probability greater than $\delta_{\mathsf{ddh\text{-}f}}$. Hence we substitute the original c_2 for $c_2' := h^r f^{u+m_\beta}$. For $\mathcal{A}$, the value of (r mod n) is fixed by $c_1 = g^r$ as g is a generator, so the value of h^r is fixed. As a result from c_2' an unbounded $\mathcal{A}$ can infer $u + m_\beta \bmod q$ but as u is uniformly distributed in $\mathbf{Z}/q\mathbf{Z}$, $\mathcal{A}$ gains no information on β.

Conversely, we construct an ind-cpa adversary from a distinguisher for the DDH-f problem. Choose $m_0 \in \mathbf{Z}/q\mathbf{Z}$ and $m_1 := m_0 + u$ with $u \hookleftarrow \mathbf{Z}/q\mathbf{Z}$. From the public key and the challenge ciphertext, construct the triplet

$$(h, c_1, c_2/f^{m_0}) = (g^\alpha, g^r, g^{\alpha r} f^{m_\beta - m_0}).$$

This gives a DH triplet if and only if $\beta = 0$ and the exponent of f is uniformly distributed

```
Setup(1^λ)
    1. Sample a $\mu$-bit prime $q$
    2. $\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Gen}(1^\lambda, q)$
    3. Return $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CL}}, q)$

KeyGen(pp)
    1. Sample $x, y, \alpha \hookleftarrow \mathcal{D}$
    2. Compute $h \leftarrow g^\alpha$, $\eta \leftarrow g^x h^y$
    3. Set $\mathsf{pk} := (h, \eta)$ and $\mathsf{sk} := (x, y)$
    4. Return $(\mathsf{pk}, \mathsf{sk})$

Enc(pk, m)
    1. Sample $r \hookleftarrow \mathcal{D}$
    2. Return $(g^r, h^r, \eta^r f^m)$

Dec(sk, (c_1, c_2, c_3))
    1. Compute $M \leftarrow c_3/(c_1^x c_2^y)$
    2. Return $\mathsf{Solve}(q, \mathsf{pp}_{\mathsf{CL}}, M)$

EvalSum(pk, (c_1, c_2, c_3), (c'_1, c'_2, c'_3))
    1. Sample $r \hookleftarrow \mathcal{D}$
    2. Let $c''_1 \leftarrow c_1 c'_1 g^r$, $c''_2 \leftarrow c_2 c'_2 h^r$, $c''_3 \leftarrow c_3 c'_3 \eta^r$
    3. Return $(c''_1, c''_2, c''_3)$

EvalScal(pk, (c_1, c_2, c_3), α)
    1. Sample $r \hookleftarrow \mathcal{D}$
    2. Let $c'_1 \leftarrow c_1^\alpha g^r$, $c'_2 \leftarrow c_2^\alpha h^r$ , $c'_3 \leftarrow c_3^\alpha \eta^r$
    3. Return $(c'_1, c'_2, c'_3)$
```

Figure 3.4: Enhanced linearly homomorphic encryption scheme $\Pi_{\mathsf{ddh\text{-}f}}$ from DDH-f

in $\mathbf{Z}/q\mathbf{Z}$ if and only if $\beta = 1$. As a result, one can use the output of a distinguisher for the DDH-f problem to win the ind-cpa experiment. □

3.5.2 Enhanced Variant Secure under DDH-f

We here modify the original PKE scheme of Fig. 3.3 by adding a key *à la* Cramer-Shoup (*cf.* [CS98]). The security of this scheme also relies on the DDH-f assumption.

This scheme, which we call $\Pi_{\mathsf{ddh\text{-}f}}$, turns out to be a direct application of the generic construction of Section 3.4.1 for building PKE from PHF, resulting from $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ of running example 3. Consequently, the ind-cpa-security of $\Pi_{\mathsf{ddh\text{-}f}}$ follows immediately from Theorem 3.20.

Setting the parameters. We use the projective hash function $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ of running example 3. Hashing keys are sampled from $\mathcal{D} := \mathcal{D}_{\mathbf{Z},\sigma}$ for $\sigma > q\tilde{s}\sqrt{\lambda}$ to ensure that $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is $2^{-\lambda}$-smooth over G on F (*cf.* Lemmas 3.3 and 3.17). The plaintext space is $\mathbf{Z}/q\mathbf{Z}$, where q is a μ-bit prime, with $\mu \geqslant \lambda$. The scheme is depicted in Fig. 3.4.

Corollary 3.26 (of Theorem 3.20)**.** Let $\Pi_{\mathsf{ddh\text{-}f}}$ denote the encryption scheme described in Fig. 3.4. If $\mathcal{SM}_{\mathsf{ddh\text{-}f}}$ is $\delta_{\mathcal{L}}$-hard, and $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is δ_s-smooth then:

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi_{\mathsf{ddh\text{-}f}},\mathcal{A}}(\lambda, \mu) \leqslant \delta_{\mathcal{L}} + \delta_s.$$

Thus if $\mathcal{SM}_{\mathsf{ddh\text{-}f}}$ is hard, and $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is smooth then $\Pi_{\mathsf{ddh\text{-}f}}$ is ind-cpa-secure.

3.5.3 Linearly Homomorphic PKE Secure under HSM-CL

The original PKE scheme of Section 3.5.1 was inspired by the scheme of [BCP03]. Another encryption scheme based on Elgamal over $(\mathbf{Z}/N^2\mathbf{Z})^\times$ was proposed by Camenisch and Shoup in [CS03]. Its ind-cpa-security relies on the DCR assumption (*cf.* Definition 2.3). In Fig. 3.5

```
Setup(1^λ)
  1. Sample a $\mu$-bit prime $q$
  2. $\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Gen}(1^\lambda, q)$
  3. Return $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CL}}, q)$

KeyGen(pp)
  1. Sample $\alpha \hookleftarrow \mathcal{D}_q$ and $h \leftarrow g_q^\alpha$
  2. Set $\mathsf{pk} := h$ and $\mathsf{sk} := \alpha$
  3. Return $(\mathsf{pk}, \mathsf{sk})$

Enc(pk, m)
  1. Sample $r \hookleftarrow \mathcal{D}_q$
  2. Return $(g_q^r, f^m h^r)$

Dec(sk, (c_1, c_2))
  1. Compute $M \leftarrow c_2 / c_1^\alpha$
  2. Return $\mathsf{Solve}(q, \mathsf{pp}_{\mathsf{CL}}, M)$

EvalSum(pk, (c_1, c_2), (c'_1, c'_2))
  1. Sample $r \hookleftarrow \mathcal{D}_q$
  2. Compute $c''_1 \leftarrow c_1 c'_1 g_q^r$, $c''_2 \leftarrow c_2 c'_2 h^r$
  3. Return $(c''_1, c''_2)$

EvalScal(pk, (c_1, c_2), α)
  1. Sample $r \hookleftarrow \mathcal{D}_q$
  2. Compute $c'_1 \leftarrow c_1^\alpha g_q^r$, $c'_2 \leftarrow c_2^\alpha h^r$
  3. Return $(c'_1, c'_2)$
```

Figure 3.5: linearly homomorphic encryption scheme $\Pi_{\mathsf{hsm\text{-}cl}}$ from HSM-CL

we present a slight modification to this scheme so that it relies on the HSM-CL assumption of Definition 3.4 in order to somewhat generalise the Camenisch-Shoup approach. The resulting scheme, which we call $\Pi_{\mathsf{hsm\text{-}cl}}$, is in fact a direct application of the generic construction of Section 3.4.1 for building PKE from PHF, resulting from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ of running example 2. Consequently, the ind-cpa-security of $\Pi_{\mathsf{hsm\text{-}cl}}$ follows immediately from Theorem 3.20.

Setting the parameters. We use the PHF $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ of running example 2. The plaintext space is $\mathbf{Z}/q\mathbf{Z}$, where q is a μ-bit prime, with $\mu \geqslant \lambda$. Note that here the secret key x (and the randomness r) is drawn from a distribution $\mathcal{D}_q$ such that $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$ is at distance less than δ_s from the uniform distribution in G^q, this does not change the view of the adversary $\mathcal{A}$. Indeed, following the same steps as in proof of Theorem 3.20, one can add an intermediate $\mathsf{Game_0}'$ between Game_0 (the real ind-cpa experiment) and Game_1, in which one samples the secret key x from $\mathcal{D}$ instead of $\mathcal{D}_q$. Since at this stage, the only view $\mathcal{A}$ has of x is in the exponent of g_q, which is of order s (where s and q are co-prime, and $n = qs$), this change is unnoticeable to $\mathcal{A}$. We can then proceed as in proof of Theorem 3.20, using the hardness of $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ and the smoothness of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ to prove that the challenge message is statistically hidden. The scheme is depicted in Fig. 3.5. Correctness follows from that of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$.

Corollary 3.27 (of Theorem 3.20)**.** Let $\Pi_{\mathsf{hsm\text{-}cl}}$ denote the encryption scheme described in Fig. 3.5. If $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is $\delta_{\mathcal{L}}$-hard, and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is δ_s-smooth over $\widehat{G}$ on F then:

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi_{\mathsf{hsm\text{-}cl}},\mathcal{A}}(\lambda, \mu) \leqslant \delta_{\mathcal{L}} + \delta_s.$$

Thus if $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is hard, and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is smooth then $\Pi_{\mathsf{hsm\text{-}cl}}$ is ind-cpa-secure.

A note on efficiency. We emphasize that in $\Pi_{\mathsf{hsm\text{-}cl}}$, we can sample shorter exponents than in the Π_{cl} and $\Pi_{\mathsf{ddh\text{-}f}}$ encryption schemes. This results in faster encryption, decryption, and homomorphic operations. Hence in our constructions of Chapter 4 instantiations from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ yield more efficient protocols than instantiations from $\mathsf{H}_{\mathsf{ddh\text{-}f}}$. This explains why, in Chapter 5, we only provide instantiations of our constructions from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$.

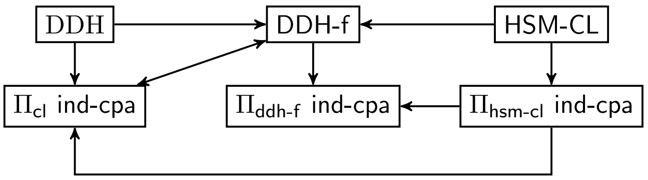


Figure 3.6: Reductions between assumptions and ind-cpa security of CL variants

3.5.4 Relations between Assumptions DDH-CL, DDH-f and HSM-CL

One can establish direct reductions between the problems underlying the DDH-CL, DDH-f and HSM-CL assumptions. However it is somewhat easier to use intermediate results on the ind-cpa-security of the schemes defined in the previous subsection to demonstrate these reductions.

We proved in Theorem 3.25 that the original CL cryptosystem is ind-cpa if and only if the DDH-f assumption holds. In [CL15], it was proven that this scheme is ind-cpa under the DDH-CL assumption. As a result, DDH-f is a weaker assumption than DDH-CL. Furthermore, it is easy to see that if $\Pi_{\mathsf{hsm\text{-}cl}}$ of Fig. 3.5 is ind-cpa then the original Π_{cl} is ind-cpa: from a public key $h = g_q^x$, $x \hookleftarrow \widehat{\mathcal{D}}_q$ and a ciphertext $c = (c_1, c_2) = (g_q^r, f^m \cdot h^r)$, $r \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma'}$ for $\Pi_{\mathsf{hsm\text{-}cl}}$, one can chose $a, b \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and construct $h' = h \cdot f^a$, and the ciphertext $c' = (c_1', c_2') = (c_1 \cdot f^b, c_2 \cdot f^{ab})$. According to Lemma 3.3, Item 6, h' and c_1' are statistically indistinguishable from the uniform distribution in G. Furthermore, $h' = g_q^x f^a = g^\alpha$ where α is defined modulo n from the Chinese remainder theorem, such that $\alpha \equiv x \pmod s$ and $\alpha \equiv a \pmod q$. Likewise, $c_1' = g_q^r f^b = g^\beta$ for some β defined equivalently. Finally, one has $c_2'/f^m = g_q^{xr} f^{ab} = g_q^{\alpha\beta \bmod s} f^{\alpha\beta \bmod q} = g^{\alpha\beta}$. As a result, $(h', c_1', c_2'/f^m)$ is a DH triplet in G, so h', c' are a public key and a ciphertext for m for Π_{cl}. As a result, an ind-cpa attacker against $\Pi_{\mathsf{hsm\text{-}cl}}$ can be built from an ind-cpa attacker against Π_{cl}. Now, if the HSM-CL assumption holds, from Corollary 3.27, $\Pi_{\mathsf{hsm\text{-}cl}}$ is ind-cpa, so Π_{cl} is also ind-cpa and the DDH-f assumption holds.

We can sum up these results with the following theorem (see also Fig. 3.6).

Theorem 3.28. The DDH-CL assumption implies the DDH-f assumption. Furthermore, the HSM-CL assumption implies the DDH-f assumption.

3.6 Zero-Knowledge Proofs for the CL Framework

Our setting contrasts to groups of known order where one can efficiently recognise group elements, such as the group of points of an elliptic curve $\mathbf{G}$, generated by P of order q, or subgroups of order q of $(\mathbf{Z}/p\mathbf{Z})^*$, where q is known. Indeed in the CL framework, we deal with a cyclic subgroup $G^q := \langle g_q \rangle$ of $\widehat{G}$, where g_q is of unknown order, and elements of G^q are not efficiently recognisable. One can only efficiently check that elements live in $\widehat{G}$. It is crucial to take this into account in all of our protocols for security to hold against *malicious* adversaries.

Since our encryption schemes presented in Section 3.5 follow an Elgamal like structure, we have at our disposal a whole arsenal of zero-knowledge proofs and arguments for Elgamal ciphertexts and for proving knowledge of discrete logarithms (e.g., [Sch90, CP93]) which we can build upon. Of course, when adapting these proofs to the CL framework, one must be careful not to sacrifice efficiency.

We thus develop proofs and arguments of knowledge for various relations in the CL framework. Among other uses, these allow to prove knowledge of an encrypted value, as well as the encryption randomness, thereby ensuring a ciphertext is honestly computed. The relations come in various flavours, depending on the requirements of our applications, as well as trade-offs between efficiency and security.

Throughout the rest of the chapter $\lambda \in \mathbf{N}$ refers to the security parameter; q is a μ-bit prime, where $\mu \geqslant \lambda$; we consider $\mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$, where $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ is the generator of a HSM-CL group with an easy DL subgroup; and $\mathbf{G}$ is an (additive) cyclic group of prime order q generated by P. We consider the subset membership problem $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$, associated projective hash function $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ of running example 2; when referring to the HSM-CL encryption scheme, we mean the resulting $\mathsf{ind\text{-}cpa}$-secure public key encryption scheme $\Pi_{\mathsf{hsm\text{-}cl}}$ of Fig. 3.5. The secret key of the encryption scheme is denoted hk, while the public key is $\mathsf{hp} \leftarrow g_q^{\mathsf{hk}}$.

For our zero knowledge *proofs*, we need exponents for which we are proving knowledge to be sampled uniformly from some bounded set $\{0, \dots, S\}$. As noted in Lemma 3.3, to instantiate $\mathcal{D}_q$, from which secret keys and encryption randomness are sampled in the HSM-CL encryption scheme, one can use the uniform distribution on $\{0, \dots, 2^{\lambda-2}\tilde{s}\}$. We thus let $S := 2^{\lambda-2}\tilde{s}$.

3.6.1 A Zero-Knowledge Proof of Knowledge for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$

In this section, we provide a ZKPoK for the relation $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ (defined hereafter), which proves knowledge of the randomness used for encryption, and of the value x which is both encrypted in a given ciphertext, and the discrete logarithm of some $Q \in \mathbf{G}$. This proof will be essential for our two party signing protocol in Section 5.2, to ensure correct behaviour of party P_1.

Recall that if a ciphertext $ct := (c_1, c_2)$ is an honestly generated[3] encryption of x under public key hp it holds that $ct = (g_q^r, f^x \mathsf{hp}^r)$ for some $r \in \{0, \dots, S\}$. The relation is hence defined as follows, where the description of the group $(\mathbf{G}, P, q)$ and $\mathsf{pp}_{\mathsf{CL}}$ are implicit public inputs:

$$\mathsf{R}_{\mathsf{cl\text{-}dl}} := \{(\mathsf{hp}, (c_1, c_2), Q); (x, r) \ \mid \ c_1 = g_q^r \wedge c_2 = f^x \mathsf{hp}^r \wedge Q = xG\}.$$

As our ZKPoK is partly performed in a group of unknown order, we rely on the Schnorr-like Girault-Poupard-Stern statistically ZK identification scheme [GPS06], which we turn into a ZKPoK of the randomness used for encryption and of the discrete logarithm of an element of $\mathbf{G}$, using binary challenges. The resulting protocol, called $\Sigma_{\mathsf{cl\text{-}dl}}$ and presented in Fig. 3.7, provides a statistical zero knowledge proof of knowledge for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$.

Though it provides strong security guarantees, we note that the protocol of Fig. 3.7 is quite inefficient, as it must be repeated many times to achieve a satisfying soundness error. In Section 3.6.2 we describe a trick which allows to increase the challenge space, and thereby reduce the number of rounds, while maintaining statistical soundness. In Section 3.7 we drastically reduce the number of rounds by relying on computational assumptions introduced in Section 3.2.

The following theorem states the security of the ZKPoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$.

Theorem 3.29. The protocol $\Sigma_{\mathsf{cl\text{-}dl}}$ described in Fig. 3.7 is a statistical zero knowledge proof of knowledge with soundness $2^{-\ell}$, as long as ℓ is polynomial and $\ell S/A$ is negligible, where A is a positive integer.

Proof. We prove completeness, soundness and zero-knowledge.
Completeness. This follows easily by observing that when $((\mathsf{hp}, (c_1, c_2), Q); (x, r)) \in \mathsf{R}_{\mathsf{cl\text{-}dl}}$, for

[3] By *honestly generated* we mean computed as prescribed by the encryption algorithm.

Public parameters: $A \in \mathbf{N}$, $S := 2^{\lambda-2}\tilde{s}$, $(\mathbf{G}, P, q)$ and $\mathsf{pp}_{\mathsf{CL}}$.		
Input : (r, x) and $(\mathsf{hp}, c_1, c_2, Q, P)$		Input : $(\mathsf{hp}, c_1, c_2, Q, P)$
`Repeat` ℓ `times`		
$r_1 \hookleftarrow \{0, \dots, A-1\}$; $r_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$t_1 \leftarrow \mathsf{hp}^{r_1} f^{r_2}$; $T_2 \leftarrow r_2 P$; $t_3 \leftarrow g_q^{r_1}$	$\xrightarrow{t_1, T_2, t_3}$	
	$\xleftarrow{k}$	$k \hookleftarrow \{0, 1\}$
$u_1 \leftarrow r_1 + kr$ in $\mathbf{Z}$		
$u_2 \leftarrow r_2 + kx \bmod q$	$\xrightarrow{u_1, u_2}$	`Check` $u_1 \in \{0, \dots, A+S\}$ and $t_1 \cdot c_2^k = \mathsf{hp}^{u_1} \cdot f^{u_2}$ and $T_2 + k \cdot Q = u_2 \cdot P$ and $t_3 \cdot c_1^k = g_q^{u_1}$

Figure 3.7: The ZKPoK $\Sigma_{\mathsf{cl\text{-}dl}}$ for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$

any $k \in \{0, 1\}$ the values computed by an honest prover will indeed verify the four relations checked by the verifier.
Soundness. For soundness, the protocol is in fact special sound. Indeed notice that for committed values t_1, T_2, t_3, if a prover P^* can answer correctly for two different values of k, he must be able to answer to challenges 0 and 1 with u_1, u_2 and u_1', u_2', where u_1 and u_1' are smaller than $A + S - 1$, such that $u_2 P = u_2' P - Q$, $\mathsf{hp}^{u_1} f^{u_2} c_2 = \mathsf{hp}^{u_1'} f^{u_2'}$ and $g_q^{u_1} c_1 = g_q^{u_1'}$. Let $\sigma_1 \leftarrow u_1' - u_1$, $\sigma_2 \leftarrow u_2' - u_2 \bmod q$; we obtain $g_q^{\sigma_1} = c_1$, $\sigma_2 P = Q$ and $\mathsf{hp}^{\sigma_1} f^{\sigma_2'} = c_2$. Thus P^* can easily compute $x \leftarrow \sigma_2 \bmod q$ and $r \leftarrow \sigma_1$ in $\mathbf{Z}$. While this gives a knowledge error of 1/2, the soundness is amplified to $2^{-\ell}$ by repeating the protocol sequentially ℓ times.
Zero-knowledge against malicious verifiers. We must exhibit a simulator $\mathcal{S}$ which, given the code of some verifier V^*, produces a transcript indistinguishable from that which would be produced between V^* and an honest prover P (proving the knowledge of a tuple in $\mathsf{R}_{\mathsf{cl\text{-}dl}}$) without knowing the witnesses (x, r) for $(\mathsf{hp}, (c_1, c_2), Q)$ in the relation $\mathsf{R}_{\mathsf{cl\text{-}dl}}$.

The potentially malicious verifier may use an adaptive strategy to bias the choice of the challenges to learn information about (r, x). This implies that challenges may not be randomly chosen, which must be taken into account in the security proof.

We describe an expected PT simulation of the communication between a prover P and a malicious verifier V^* for one round of the proof. Since the simulated round may not be the first round, we assume V^* has already obtained data, denoted by hist, from previous interactions with P. Then P sends the commitments t_1, T_2, t_3 and V^* chooses – possibly using hist and t_1, T_2, t_3 – the challenge $k(t_1, T_2, t_3, \mathsf{hist})$.

Description of the simulator: Consider $\mathcal{S}$ simulating a given round of identification as follows:

1. $\mathcal{S}$ chooses random values $\bar{k} \in \{0, 1\}$, $\bar{u}_1 \in \{S-1, \dots, A-1\}$ and $\bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}$.

2. $\mathcal{S}$ computes $\bar{t}_1 \leftarrow \mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2} / c_2^{\bar{k}}$; $\bar{T}_2 \leftarrow \bar{u}_2 \cdot P - \bar{k} \cdot Q$ and $\bar{t}_3 \leftarrow g_q^{\bar{u}_1} / c_1^{\bar{k}}$, and sends $\bar{t}_1$, $\bar{T}_2$ and $\bar{t}_3$ to V^*.

3. $\mathcal{S}$ receives $k(\bar{t}_1, \bar{T}_2, \bar{t}_3, \mathsf{hist})$ from V^*.

4. If $k(\bar{t}_1, \bar{T}_2, \bar{t}_3, \mathsf{hist}) \neq \bar{k}$ then return to step 1, else return $(\bar{t}_1, \bar{T}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2)$.

To demonstrate that $\Sigma_{\mathsf{cl\text{-}dl}}$ is indeed ZK, one justifies that the distribution output by $\mathcal{S}$ is statistically close to that output in a real execution of the protocol, and that the simulation runs in expected polynomial time.

Intuitively, sampling the randomness r_1 from a large enough distribution (i.e. $S \ll A$) ensures that the distribution of t_1, T_2, t_3 in a real execution is statistically close to that in a simulated execution.

The analysis of this statistical distance, denoted ϵ, between the distribution of tuples output by $\mathcal{S}$ and that of tuples output by a real execution of the protocol is quite technical and similar to that of [GPS06]. We do not provide the details here, though the interested reader can refer to Appendix B. Applying their analysis to our setting allows us obtain the following bound:

$$\epsilon < \frac{8S}{A}.$$

Thus the real and simulated distributions are statistically indistinguishable if S/A is negligible. Therefore for the simulation of all ℓ rounds to be indistinguishable from ℓ rounds of a real execution, $\ell S/A$ must be negligible.

Running time of the Simulator: To see that the simulator runs in expected PT, notice that step 3 outputs a tuple $(\bar{t}_1, \bar{T}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2)$ if $k(\bar{t}_1, \bar{T}_2, \bar{t}_3, \mathsf{hist}) = \bar{k}$. Since $\bar{k}$ is sampled at random from $\{0, 1\}$, the expected number of iterations of the loop is 2. Therefore the complexity of the simulation of all ℓ rounds is $O(\ell)$. Thus if $\ell S/A$ is negligible and ℓ is polynomial, the protocol is statistically ZK. □

3.6.2 A trick to improve efficiency.

In this section we introduce the *lowest common multiple* (lcm) *trick*. This trick allows us to significantly increase the size of the challenge set compared to that of Section 3.6.1, and thereby reduce the number of repetitions required of the proof. We explain the trick via a ZKPoK for a simple relation: that of discrete logarithms in a group of unknown order, we call the resulting proof $\Sigma_{\mathsf{lcm\text{-}dl}}$. This allows us to focus on the essential idea of the trick. However, as will be discussed at the end of this subsection, the trick can also be applied to more complex relations (e.g. to improve $\Sigma_{\mathsf{cl\text{-}dl}}$ of Section 3.6.1, or for the interactive setup protocol of Section 5.3.2).

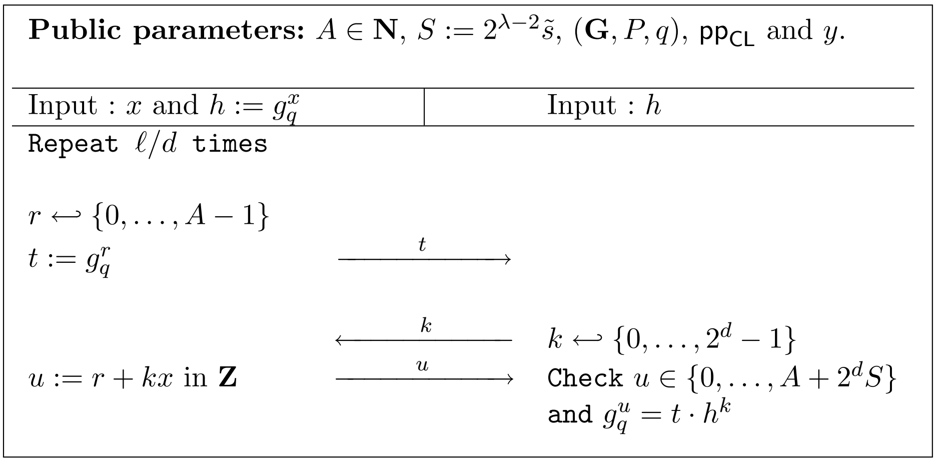
Throughout this section we denote $y := \mathsf{lcm}(1, 2, 3, \dots, 2^d - 1)$ and $\mathcal{C} = \{0, \dots, 2^d - 1\}$.

The lowest common multiple trick. For a given statement h, the proof $\Sigma_{\mathsf{lcm\text{-}dl}}$ does not actually prove knowledge of the discrete logarithm of h, but rather of h^y. Precisely, the protocol $\Sigma_{\mathsf{lcm\text{-}dl}}$ of Fig. 3.8 is a ZKPoK for the following relation:

$$\mathsf{R}_{\mathsf{lcm\text{-}dl}} := \{(h, g_q); z \mid h^y = g_q^z;\ y = \mathsf{lcm}(1, 2, 3, \dots, 2^d)\}.$$

Theorem 3.30. The protocol $\Sigma_{\mathsf{lcm\text{-}dl}}$ described in Fig. 3.8 is a statistical zero knowledge proof of knowledge with soundness $2^{-\ell}$, as long as ℓ/d and 2^d are polynomial and $\frac{2^d \ell S}{Ad}$ is negligible, where A is a positive integer.

Proof. We prove correctness, soundness and zero-knowledge.
Correctness. If $h = g_q^x$, then $g_q^u = g_q^{r+kx} = g_q^r \cdot (g_q^x)^k = t \cdot h^k$ and V accepts.
Special soundness. Suppose that for a committed value t, prover P^* can answer correctly for two different challenges k_1 and k_2. We call u_1 and u_2 the two answers. Let $k := k_1 - k_2$ and $u := u_1 - u_2$, then since $g_q^{u_1} = t \cdot h^{k_1}$ and $g_q^{u_2} = t \cdot h^{k_2}$, it holds that $g_q^u = h^k$. By the choice of the challenge set, $k < 2^d$, and since $y = \mathsf{lcm}(1, 2, 3, \dots, 2^d - 1)$, it holds that y/k is an integer.



Public parameters: $A \in \mathbf{N}$, $S := 2^{\lambda-2}\tilde{s}$, $(\mathbf{G}, P, q)$, $\mathsf{pp}_{\mathsf{CL}}$ and y.

Input : x and $h := g_q^x$		Input : h
`Repeat` ℓ/d `times`		
$r \hookleftarrow \{0, \ldots, A-1\}$		
$t := g_q^r$	$\xrightarrow{t}$	
	$\xleftarrow{k}$	$k \hookleftarrow \{0, \ldots, 2^d - 1\}$
$u := r + kx$ in $\mathbf{Z}$	$\xrightarrow{u}$	`Check` $u \in \{0, \ldots, A + 2^d S\}$ `and` $g_q^u = t \cdot h^k$

Figure 3.8: The ZKPoK $\Sigma_{\mathsf{lcm\text{-}dl}}$ for $\mathsf{R}_{\mathsf{lcm\text{-}dl}}$ where $y = \mathsf{lcm}(1, 2, 3, \ldots, 2^d - 1)$

Consequently $(g_q^u)^{y/k} = (h^k)^{y/k} = h^y$. Denoting $z := uy/k$, P^* can compute z satisfying $g_q^z = h^y$, so if P can convince V for two different challenge values, then P^* can compute a witness z for (h, g_q). Hence one execution of the elementary protocol gives a soundness error of $1/|\mathcal{C}| = 2^{-d}$. The soundness is amplified to $2^{-\ell}$ by repeating the protocol sequentially ℓ/d times.

Zero knowledge. We here only sketch the simulator, and the intuition backing the zero-knowledge property of the protocol, as the proof is very similar to that of Theorem 3.29.

Given h the simulator samples $k \hookleftarrow \mathcal{C}$ and $u \hookleftarrow \{0, \ldots, A+kS-1\}$, computes $t \leftarrow g_q^u \cdot h^{-k}$. The distribution followed by the transcript (h, t, k, u) is statistically close to that produced by real executions of the protocol. This holds since an honest prover samples x from $\{0, \ldots S-1\}$, the challenge space is of size $|\mathcal{C}| = 2^d$ and r is sampled $\{0, \ldots, A-1\}$, where $A \gg S \cdot |\mathcal{C}| = S \cdot 2^d$. As the simulator must simulate ℓ/d rounds of the protocol, we need, as in proof of Theorem 3.29, $(\ell/d)S2^d/A$ to be negligible (for the distribution of all ℓ/d real and simulated transcripts to be indistinguishable), and ℓ/d to be polynomial (for the simulator to run in polynomial time).

□

Applying the lcm trick to $\Sigma_{\mathsf{cl\text{-}dl}}$ and implications. In our distributed signature protocols of Chapter 5 we use the homomorphic properties of the $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme to allow parties to combine their private inputs while keeping these inputs secret. To ensure that no information leaks from the homomorphic operations performed on ciphertexts, one must enforce that ciphertexts are computed honestly as per the encryption algorithm. To this end, we will require that parties prove their ciphertexts are well formed. One can use the ZKPoK $\Sigma_{\mathsf{cl\text{-}dl}}$ of Fig. 3.7, which proves (among other things) that a ciphertext is well formed, and ensures that statistically no parties can cheat. This choice is somewhat unsatisfactory in terms of efficiency, since in order to attain a satisfying soundness error of $2^{-\lambda}$, the elementary proof of Fig. 3.7 must be repeated λ times, as it uses binary challenges. If we use the above technique (the lcm trick) applied to $\Sigma_{\mathsf{cl\text{-}dl}}$, with $y := \mathsf{lcm}(1, 2, 3, \ldots, 2^d - 1)$, one can divide by d the number of repetitions. However one obtains a ZKPoK for the following relation:

$$\mathsf{R}_{\mathsf{lcm\text{-}cl\text{-}dl}} := \{(\mathsf{hp}, (c_1, c_2), Q); (x, z) \mid c_1^y = g_q^z \wedge c_2^y = f^{x \cdot y}\mathsf{hp}^z \wedge Q = xP\}.$$

This entails a few modification for the overall protocol (*cf.* Section 5.2.6, page 179), since at the end of this proof, parties are only convinced that ciphertexts to the power y are well formed.

On the choice of d. The size of the challenge set $\mathcal{C}$ from which k is sampled determines the number of times the protocol needs to be repeated in order to achieve a reasonable soundness error. Consequently it is desirable to take $\mathcal{C}$ as large as possible. However, when used as part of a larger protocol, parties will need to raise ciphertexts to the power y prior to performing homomorphic operations. Consequently $|\mathcal{C}|$ must be chosen small enough for this exponentiation to take reasonable time. Hence setting $\mathcal{C} := \{0,1\}^{10}$, and $y = \mathsf{lcm}(1, \dots, 2^{10} - 1)$, which is a 1479 bits integer, ensures exponentiating to the power y remains efficient. To achieve a soundness error of $2^{-\ell}$ the protocol must then repeated $\ell/10$ times.

3.7 Zero-Knowledge Arguments for the CL Framework

For the following protocols, we use the LO assumption and the SR assumption for class groups. We show that whatever the challenge space, if one cannot extract a witness, then one can break at least one of these two assumptions. This allows to significantly increase the challenge space of our proofs, and reduce the number of rounds needed to achieve a satisfying soundness, which yields improvements both in terms of bandwidth and of computational complexity. Using such assumptions in the context of generalised Schnorr proofs in groups of unknown order is not novel (e.g. [DF02, CKY09]). We adapt these techniques to our framework. Accordingly, when referring the HSM-CL encryption scheme, we mean $\Pi_{\mathsf{hsm\text{-}cl}}$ of Fig. 3.5, with the modification that we do not use the somewhat deterministic generator g_q output by Gen, but rather a random power $\hat{g}_q$ of g_q. Thus the secret key of the encryption scheme is hk, while the public key is now $\mathsf{hp} \leftarrow \hat{g}_q^{\mathsf{hk}}$. This modification is necessary, since in our arguments of knowledge, soundness reduces to the difficulty of computing roots of $\hat{g}_q$.

The first ZKAoK we present ensures a ciphertext for the HSM-CL encryption scheme is well formed. The second extends this to deal with the $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ relation presented in Section 3.6.1. Settling for computationally convincing arguments of knowledge for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ allows us to hugely improve efficiency compared to the protocols of Section 3.6. Indeed, our arguments of knowledge allow us to chose a challenge space of size 2^{λ}, and hence one execution of the elementary proof suffices to obtain a satisfying knowledge error (under computational assumptions) of $\approx 2^{-\lambda}$. Our results are quite general and can have useful applications even beyond the specific threshold setting which we use it for in Chapter 5.

A note on the zero-knowledge property. Due to the fact the challenge space is no longer polynomial in the security parameter, the simulator used in proofs of Theorems 3.29 and 3.30, allowing us to back the statistical zero-knowledge property of $\Sigma_{\mathsf{cl\text{-}dl}}$ and $\Sigma_{\mathsf{lcm\text{-}dl}}$ no longer runs in polynomial time. Indeed, its expected running time (for one execution of the protocol) is $O(\ell \cdot C)$, where ℓ is the number of iterations of an elementary proof, and C is the size of the challenge set. Clearly if we chose $C = 2^{\lambda}$ as suggested above, the simulator is not efficient.

Thus we only prove our ZKAoK to be special honest-verifier zero-knowledge, i.e., the zero-knowledge property holds as long as the verifier samples the challenge from the description prescribed by the protocol; this must of course be taken into account when these arguments are used to secure more complex protocols. Indeed, a malicious verifier could, in a real life application, choose the challenge arbitrarily (i.e. not sampled from the prescribed distribution), in which case special honest-verifier zero-knowledge is no longer sufficient to ensure there is no leakage of information. However there exist standard techniques [Gro04, GMY06] allowing to convert special honest-verifier zero-knowledge arguments into full zero-knowledge arguments secure against arbitrary verifiers in the common reference string model which can be very efficient, and only cost a small additive overhead.

We note that when using these arguments of knowledge in our full threshold protocol of Section 5.3, honest verifier zero knowledge is sufficient for our purposes. Indeed, when simulating the view an adversary in the overall protocol, the ZKAoK do not need to be simulated (the simulator does know the values for which it is proving knowledge), hence our proof to goes through with arguments which are only honest verifier zero knowledge. This is not the case for our two-party protocol of Section 5.2, in which the simulator (simulating party P_1) does *not know* the value for which it is proving knowledge.

The fact we only consider honest verifiers also allows us to sample exponents from Gaussian distributions, as simulating distributions when dealing with honest verifiers is much simpler. We do not rule out the possibility of sampling exponents from Gaussian distributions for the proofs $\Sigma_{\mathsf{cl\text{-}dl}}$ and $\Sigma_{\mathsf{lcm\text{-}dl}}$ of Section 3.6, though we have not yet performed the analysis in this setting, we believe it would be an interesting point to investigate.

Setting the parameters. For both our ZKAoK we need to bound the range in which sampled exponents live (those of the group of unknown order generated by g_q for which we are proving knowledge). As noted in Lemma 3.3, to instantiate $\mathcal{D}_q$, from which secret keys and encryption randomness are sampled, one can use the Gaussian distribution $\mathcal{D}_q = \mathcal{D}_{\mathbf{Z},\sigma'}$ where $\sigma' = \tilde{s}\sqrt{\lambda}$. For such a choice of $\mathcal{D}_q$, exponents lie in the set $\{-10\sigma', \dots, 10\sigma'\}$ with probability $> 1 - 2^{-80}$. We thus let $S := 10\tilde{s}\sqrt{\lambda}$ and assume $r \hookleftarrow \mathcal{D}_q$ satisfies $r \in \{-S, \dots, S\}$. The fact we use Gaussian distributions does not come into play in the upcoming proofs (all we need is the aforementioned bound on the size of exponents), however it will have an impact on the efficiency of our protocols of Chapter 5, since as mentioned in Section 2.7, sampling from discrete Gaussian distributions allows us to have shorter exponents than folded uniforms.

Proving a ciphertext for the HSM-CL encryption scheme is well formed

Recall that a ciphertext encrypting $a \in \mathbf{Z}/q\mathbf{Z}$ for $\Pi_{\mathsf{hsm\text{-}cl}}$ is of the form $ct := (c_1, c_2)$ with $c_1 = \hat{g}_q^r$, $c_2 = \mathsf{hp}^r f^a$ and $r \in \{-S, \dots, S\}$. For some $C \in \mathbf{N}$, consider the relation:

$$\mathsf{R}_{\mathsf{Enc}} := \{(\mathsf{hp}, ct); (a, r) \mid \mathsf{hp} \in \widehat{G};\ \ c_1 = \hat{g}_q^r \wedge c_2 = \mathsf{hp}^r f^a\}.$$

It is easy to see that $\mathsf{R}_{\mathsf{Enc}}$ is essentially the $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ relation, without the proof of knowledge of the discrete logarithm of $Q \in \mathbf{G}$. For our full threshold signatures of Section 5.3 we only need parties to prove ciphertexts are correct, so relation $\mathsf{R}_{\mathsf{Enc}}$ is sufficient. We shall provide a ZKAoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ in Fig. 3.10, however we note that the resulting protocol is very similar to that for $\mathsf{R}_{\mathsf{Enc}}$ presented here. Moreover the main difficulties in proving the soundness of these ZKAoK are addressed in proof of Theorem 3.31, and are the same for both protocols.

We here present a ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$, where the challenge set is $\{0, \dots, C-1\}$. The protocol is given in Fig. 3.9, the only constraint on C is that the LO_C assumption holds (*cf.* Definition 3.7).

Theorem 3.31. Let $\{0, \dots, C-1\}$ be the challenge set for the protocol of Fig. 3.9. Suppose the SR assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ and the LO_C assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$. Then the interactive protocol of Fig. 3.9 is an argument of knowledge for $\mathsf{R}_{\mathsf{Enc}}$ with knowledge error $\kappa = 4/C$. If $r \in \{-S, \dots, S\}$, and SC/A is negligible, where $A \in \mathbf{N}$ then the protocol is honest verifier statistical zero-knowledge.

Proof. We prove the properties of completeness, honest verifier zero-knowledge and soundness. *Completeness.* If P knows $r \in \{-S, \dots, S\}$ and $a \in \mathbf{Z}/q\mathbf{Z}$ s.t. $(\mathsf{hp}, ct); (a, r) \in \mathsf{R}_{\mathsf{Enc}}$, and both parties follow the protocol, all of V's check pass.
Special honest verifier zero-knowledge. Given hp, $ct = (c_1, c_2)$, and a random challenge $k \in \{0, \dots, C-1\}$, a simulator can sample $u_1 \hookleftarrow \{-CS, \dots, SC + A - 1\}$ and $u_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$,

The relation generator $\mathscr{R}_{\mathsf{Enc}}$:

1. Run $\mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$.
2. Let $S := 10 \cdot \tilde{s} \cdot \sqrt{\lambda}$, and set $A \in \mathbf{N}$. Sample $t \hookleftarrow \mathscr{D}_q$ and let $\hat{g}_q := g_q^t$.
3. Output $\mathsf{R}_{\mathsf{Enc}}$, S, A and $(\mathsf{pp}_{\mathsf{CL}}, \hat{g}_q, \mathsf{Solve}(\cdot))$.

Input: (a, r) and $(\mathsf{hp}, ct))$		Input: (hp, ct)
$r_1 \hookleftarrow \{0, \dots, A-1\}$ $r_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $t_1 := \hat{g}_q^{r_1}$ $t_2 := \mathsf{hp}^{r_1} f^{r_2}$	$\xrightarrow{t_1, t_2}$	
		$k \hookleftarrow \{0, \dots, C-1\}$
	$\xleftarrow{k}$	
$u_1 := r_1 + kr \in \mathbf{Z}$ $u_2 := r_2 + ka \in \mathbf{Z}/q\mathbf{Z}$	$\xrightarrow{u_1, u_2}$	`Check` $u_1 \in \{-SC, \dots, SC + A - 1\}$ `and` $\mathsf{hp}, c_1, c_2, t_1, t_2 \in \widehat{G}$ `and` $u_2 \in \mathbf{Z}/q\mathbf{Z}$ and $\hat{g}_q^{u_1} = t_1 c_1^k$ `and` $\mathsf{hp}^{u_1} f^{u_2} = t_2 c_2^k$

Figure 3.9: ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$.

compute $t_2 := \mathsf{hp}^{u_1} f^{u_2} c_2^{-k}$ and $t_1 := \hat{g}_q^{u_1} c_1^{-k}$ such that the transcript $(\mathsf{hp}, ct, t_2, t_1, k, u_1, u_2)$ is indistinguishable from a transcript produced by a real execution of the protocol.
Soundness. Consider a PT prover P^* and let view be any view P^* may have after having produced (hp, ct). Recall that $\mathsf{acc}_{\mathsf{view}, P^*}$ denotes P^*'s probability of making V accept, conditioned on view view, and that κ is the probability that P^* can make V accept without knowing a witness.

Since there are C different challenges, if $\mathsf{acc}_{\mathsf{view}, P^*} > \kappa(\lambda) = 4/C$, standard rewinding techniques allow to obtain in expected PT a situation where, for a commitment (t_1, t_2), P^* has correctly answered two different values k and k'. We call u_1, u_2 and u_1', u_2' the corresponding answers, such that:

- $\hat{g}_q^{u_1} = t_1 c_1^k$ and $\hat{g}_q^{u_1'} = t_1 c_1^{k'}$ s.t. $\hat{g}_q^{u_1 - u_1'} = c_1^{k-k'}$,
- $\mathsf{hp}^{u_1} f^{u_2} = t_2 c_2^k$ and $\mathsf{hp}^{u_1'} f^{u_2'} = t_2 c_2^{k'}$ s.t. $\mathsf{hp}^{u_1 - u_1'} f^{u_2 - u_2'} = c_2^{k-k'}$.

Since $k \neq k'$ and q is prime, with overwhelming probability it holds that $k - k'$ is invertible mod q. In the following we assume this is the case[4].

Let Rewind be a (probabilistic) procedure that creates $k, k', u_1, u_2, u_1', u_2'$ in this way. A concrete algorithm for Rewind is given in [DF02, Appendix A]. It runs in expected time $56/\mathsf{acc}_{\mathsf{view}, P^*}$, counting the time to do the protocol once with P^* as one step. Assume without loss of generality that $k > k'$ and suppose that $(k - k')$ divides $(u_1 - u_1')$ in $\mathbf{Z}$. We denote:

$$\nu_1 := \hat{g}_q^{(u_1 - u_1')/(k-k')} c_1^{-1} \quad \text{and} \quad \nu_2 := \mathsf{hp}^{(u_1 - u_1')/(k-k')} f^{(u_2 - u_2') \cdot (k-k')^{-1}} c_2^{-1}.$$

[4] In fact in our applications $C < q$, so this will hold with probability 1.

If $(k-k')$ divides (u_1-u_1') in $\mathbf{Z}$, and $\nu_1=\nu_2=1$, we say (k,k',u_1,u_2,u_1',u_2') is a set of *good* values. Suppose the set of values is good. Then $V's$ check on the size of u_1,u_1' implies that $(u_1-u_1')/(k-k')$ is in the required interval, and one can now easily verify that P^* knows $((\mathsf{hp},ct);((u_2-u_2')\cdot(k-k')^{-1} \bmod q,(u_1-u_1')/(k-k')))\in \mathsf{R}_{\mathsf{Enc}}$. Consequently:

from a set of *good values* one can extract a witness for (hp,ct).

A set of values (k,k',u_1,u_2,u_1',u_2') is said to be *bad* if $(k-k')$ divides (u_1-u_1') but $\nu_1\neq 1$ or $\nu_2\neq 1$; or if $(k-k')$ does not divide (u_1-u_1').

Given the above, we can define our extractor M:

M repeatedly calls Rewind (for the same (hp,ct)) until it gets a set of good values.

We now consider the probability M fails on view view, setting the polynomial p from Definition 2.24 to the constant 112. This choice for p is due to the running time of Rewind, which we recall is $56/\mathsf{acc}_{\mathsf{view},P^*}$, counting the time to do the protocol once with P^* as one step. By choosing $p=112$, we ensure that for any view on which M fails, the values produced by Rewind are bad with probability at least $1/2$, since otherwise M could expect to find a witness for (hp,ct) after calling Rewind twice, which takes expected time $\frac{112}{\mathsf{acc}_{\mathsf{view},P^*}}\leqslant\frac{p(\lambda)}{\mathsf{acc}_{\mathsf{view},P^*}-\kappa(\lambda)}$; this would contradict the fact M fails on view.

Let us now assume M fails on view view output by P^*. This occurs with probability $\mathsf{Adv}_{P^*}^{\kappa,M,p}$. In this case, we build an algorithm $\mathscr{A}$ which uses Rewind to break either the SR problem for class groups with input $(\mathsf{pp}_{\mathsf{CL}},\hat{g}_q,\mathsf{Solve}(\cdot))$, or the LO problem with input $(\mathsf{pp}_{\mathsf{CL}},\mathsf{Solve}(\cdot))$. Specifically, $\mathscr{A}$ forwards $(\mathsf{pp}_{\mathsf{CL}},\hat{g}_q,\mathsf{Solve}(\cdot))$ to P^*, as P^* expects to receive from the relation generator; calls Rewind and hopes to get a set of bad values. However, if Rewind runs more than $448/\kappa(\lambda)$ times with P^*, $\mathscr{A}$ aborts and stops.

Observe that since the Rewind procedure runs the protocol with P^* at most $56/\mathsf{acc}_{\mathsf{view},P^*}\leqslant 56/\kappa(\lambda)$ times, Rewind is allowed to run for at least 8 times its expected running time. Moreover since we assume M fails on view view, the values produced by Rewind are bad with probability at least $1/2$. Hence, given M fails, with significant probability $\mathscr{A}$ gets a set of bad values. So let E be the event that M fails on view and Rewind has returned a set of bad values.

Claim. If E occurs, $\mathscr{A}$ solves either the SR problem, or the LO_C problem with probability $p^*\geqslant 1/2$.

Proof of Claim Since E occurs, $\mathscr{A}$ has obtained from Rewind a set of bad values k,k',u_1,u_2,u_1',u_2' s.t. $\hat{g}_q^{u_1-u_1'}=c_1^{k-k'}$ and $\mathsf{hp}^{u_1-u_1'}f^{u_2-u_2'}=c_2^{k-k'}$. If $(k-k')$ divides (u_1-u_1') then $\nu_1\neq 1$ or $\nu_2\neq 1$ where ν_1 and ν_2 are defined as above. Clearly $\nu_1^{k-k'}=\nu_2^{k-k'}=1$. And since $k-k'<C$, and $\nu_1,\nu_2\in\widehat{G}$ this is a solution for the LO_C problem.

Now consider the case where $(k-k')$ does not divide (u_1-u_1'). We denote $d:=\gcd(k-k',u_1-u_1')$ and $e:=(k-k')/d$; and split in two cases:

Case 1: If $e=2^m$ for some positive integer m (in this case solving the root problem is easy). Our goal here is to show that since P^* does not have control over k,k' this case happens with probability $\leqslant 1/2$, given that E occurs. Hence the Case 2 – where we solve either the SR problem or the LO_C problem – happens with significant probability, given E. Indeed, observe that for e to be a power of 2, it must hold that $(k-k')$ is a multiple of 2^m, and in particular a multiple of 2. However since k and k' are chosen uniformly at random from $\{0,\dots,C-1\}$ by the honest V, with probability $1/2$, $(k-k')$ is an odd integer.

Case 2: If e is not a power of 2. It holds that $d < |k - k'| < C$. Choosing γ and δ s.t. $\gamma(k-k') + \delta(u_1 - u_1') = d$ one has $\hat{g}_q^d = \hat{g}_q^{\gamma(k-k')+\delta(u_1-u_1')} = (\hat{g}_q^\gamma c_1^\delta)^{k-k'}$. Now let:

$$\mu := (\hat{g}_q^\gamma c_1^\delta)^{\frac{(k-k')}{d}} \hat{g}_q^{-1}.$$

Clearly $\mu^d = 1$, so since $d < C$, if $\mu \neq 1$, $\mathcal{A}$ has a solution to the LO_C problem in $\widehat{G}$. Now suppose that $\mu = 1$. We can rewrite the above as:

$$\hat{g}_q = (\hat{g}_q^\gamma c_1^\delta)^{(k-k')/d},$$

which gives a solution for the SR problem with input $\hat{g}_q$, which is $e = (k-k')/d$ and $X := \hat{g}_q^\gamma c_1^\delta$, s.t. $\hat{g}_q = X^e$, $e > 1$ and e is not a power of 2. The claim above now follows.

To summarize, for every view view where M fails, running Rewind will fail to solve either the SR problem or the LO problem with probability at most $1 - p^*/2 \leqslant 3/4$. Hence if $\mathsf{Adv}_{P^*}^{\kappa,M,p}$ is a non negligible function of the security parameter, then the resulting probability that $\mathcal{A}$ can break either the SR problem or the LO problem is non negligible, thereby contradicting the assumption these problems are hard. Thus $\mathsf{Adv}_{P^*}^{\kappa,M,p}(\lambda) = \mathsf{negl}(\lambda)$ and the protocol of Fig. 3.9 is an argument of knowledge for $\mathsf{R}_{\mathsf{Enc}}$ with soundness error $C/4$.

□

Efficient ZKAoK for $\mathsf{R}_{\text{cl-dl}}$

The interactive proof provided for $\mathsf{R}_{\text{cl-dl}}$ in Section 3.6.1 uses binary challenges, consequently in order to achieve a satisfying soundness error of $2^{-\lambda}$, the proof must be repeated λ times. Moreover the proof $\Sigma_{\text{lcm-dl}}$ for $\mathsf{R}_{\text{lcm-dl}}$, though more efficient, does not prove exactly the same relation, and hence requires parties to perform large exponentiations on the received ciphertexts. Though for some protocols, this may be acceptable (e.g. depending on whether parties need to be online when they perform this exponentiation), often the cost of this exponentiation is prohibitive. Relying on the SR and LO assumptions for $\mathsf{Gen}_{\mathsf{CL}}$, we describe in Fig. 3.10 a much more efficient ZKAoK for the relation:

$$\mathsf{R}_{\text{cl-dl}} := \{(\mathsf{hp}, (c_1, c_2), Q); (x, r) \;\mid\; \mathsf{hp} \in \widehat{G}; c_1 = \hat{g}_q^r \wedge\; c_2 = f^x \mathsf{hp}^r \wedge Q = xP\},$$

where C denotes the size of the challenge set, and $A \in \mathbf{N}$. Theorem 3.32 states the security of the protocol of Fig. 3.10.

Theorem 3.32. Let $\{0, \dots, C-1\}$ be the challenge set for the interactive protocol of Fig. 3.10. Suppose the SR assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ and the LO_C assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$. Then the interactive protocol of Fig. 3.10 is an argument of knowledge for $\mathsf{R}_{\text{cl-dl}}$ with knowledge error $\kappa = 4/C$. If $r \in \{-S, \dots, S\}$, and SC/A is negligible, where $A \in \mathbf{N}$, then the protocol is special honest verifier statistical zero-knowledge.

Proof. Completeness. If P knows $r \in \{-S, \dots, S\}$ and $x \in \mathbf{Z}/q\mathbf{Z}$ s.t. $(\mathsf{hp}, (c_1, c_2), Q); (x, r) \in \mathsf{R}_{\text{cl-dl}}$, and if both parties follow the protocol then V accepts.

Special honest verifier zero-knowledge. Given hp, $ct = (c_1, c_2)$, Q and $k \hookleftarrow \{0, \dots, C-1\}$, a simulator can sample $u_1 \hookleftarrow \{-CS, \dots, SC+A-1\}$ and $u_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, compute $t_1 := \hat{g}_q^{u_1} \cdot (c_1)^{-k}$, $t_2 := \mathsf{hp}^{u_1} \cdot f^{u_2} \cdot (c_2)^{-k}$ and $T := u_2 \cdot P - k \cdot Q$ such that the transcript $(t_1, t_2, T, k, u_1, u_2)$ is indistinguishable from a transcript produced by a real execution of the protocol where V runs on input $(\mathsf{hp}, c_1, c_2, Q, P)$.

The relation generator $\mathscr{R}_{\mathsf{cl\text{-}dl}}$:

1. Run $\mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$.
2. Let $S := 10 \cdot \tilde{s} \cdot \sqrt{\lambda}$, and set $A \in \mathbf{N}$. Sample $t \hookleftarrow \mathscr{D}_q$ and let $\hat{g}_q := g_q^t$.
3. Output $\mathsf{R}_{\mathsf{cl\text{-}dl}}$; $(\mathsf{pp}_{\mathsf{CL}}, \hat{g}_q, \mathsf{Solve}(\cdot))$; and the description of an additive prime order group $(\mathbf{G}, P, q)$.

Input : (r, x) and $(\mathsf{hp}, c_1, c_2, Q, P)$		Input : $(\mathsf{hp}, c_1, c_2, Q, P)$
$r_1 \hookleftarrow \{0, \ldots, A-1\}$ $r_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $t_1 := \hat{g}_q^{r_1}$ $t_2 := \mathsf{hp}^{r_1} f^{r_2}$		
$T := r_2 P$	$\xrightarrow{t_1, t_2, T}$	$k \hookleftarrow \{0, \ldots, C-1\}$
	$\xleftarrow{k}$	
$u_1 := r_1 + k \cdot r \in \mathbf{Z}$		
$u_2 := r_2 + k \cdot x \in \mathbf{Z}/q\mathbf{Z}$	$\xrightarrow{u_1, u_2}$	$\texttt{Check}$ $u_1 \in \{-SC, \ldots, SC + A - 1\}$ and $u_2 \in \mathbf{Z}/q\mathbf{Z}$ $\textsf{and}$ $\mathsf{hp}, c_1, c_2, t_1, t_2 \in \widehat{G}$ $\textsf{and}$ $\hat{g}_q^{u_1} = t_1 \cdot (c_1)^k$ $\textsf{and}$ $T + k \cdot Q = u_2 \cdot P$ $\textsf{and}$ $\mathsf{hp}^{u_1} f^{u_2} = t_2 \cdot (c_2)^k$

Figure 3.10: ZKAoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$.

Computational soundness. The proof proceeds exactly as that of Theorem 3.31. The only difference being that for a set $(k, k', u_1, u_2, u'_1, u'_2)$ of good values (defined in exactly the same way), it holds that $((\mathsf{hp}, ct, Q); ((u_2 - u'_2) \cdot (k - k')^{-1} \bmod q, (u_1 - u'_1)/(k - k'))) \in \mathsf{R}_{\mathsf{cl\text{-}dl}}$, such that one can thus extract a witness for a statement (hp, ct, Q) of relation $\mathsf{R}_{\mathsf{cl\text{-}dl}}$. □

Chapter 4

Functional Encryption for Computing Inner Products

In this chapter we devise generic constructions for building inner product functional encryption schemes from projective hash functions, proven secure against passive and active adversaries respectively. This involves defining new properties for projective hash functions. Our proofs notably improve the tightness of security reductions for generic inner product functional encryption, thereby allowing to implement more efficient schemes for a given security level. We instantiate our constructions from the running examples of Chapter 3, and retrieve pre-existing schemes, along with new schemes from class group cryptography, which are among the most efficient inner product functional encryption schemes to date.

State of the Art

From functional encryption to inner product functional encryption. Traditional public key encryption provides an all-or-nothing approach to data access: given a ciphertext encrypting m, a receiver either decrypts and recovers the entire message m, or learns nothing about the encrypted message. Since many real life applications call for a more subtle disclosure of information, according to a receiver's privileges, functional encryption (FE) [SW05, BSW11, O'N10] emerged from a series of refinements of PKE, allowing to control how much of the data each user can recover. Specifically, FE allows for a receiver to recover a function $f(m)$ of the encrypted message m, without revealing anything else about m. The primitive derives functional decryption keys sk_{f_i} – associated to specific functionalities f_i – from a master secret key msk; these are delivered to the appropriate recipients. A single ciphertext c encrypting plaintext m is made available, from which a user possessing sk_{f_i} can recover $f_i(m) = \mathsf{Dec}(\mathsf{sk}_i, c)$.

Realising functional encryption, which attains a satisfying level of security, for any computation has proven difficult: all such existing constructions are far from practical, and either bound the number of decryption keys the adversary can request (e.g. [SS10, GVW12, GKP$^+$13]), or rely on strong cryptographic assumptions such as the existence of multilinear maps [GGH13a, GGHZ16] or indistinguishability obfuscation [GGH$^+$13b, BKS18]. Hence researchers started focussing on functional encryption restricted to the computation of specific classes of functions, in the hope that such primitives could be implemented efficiently under well understood cryptographic assumptions, while being efficient enough to benefit concrete practical applications.

One notable example is the study of inner-product functional encryption (IPFE), as first formalised by Abdalla et al. in [ABDP15], which restricts the computed functionality to the inner product of two vectors (one resulting from a decryption key, the other from a message). This restriction has two benefits: developing our understanding of FE, and a range of practical applications. These applications include the computation of weighted averages and sums for statistical analysis on encrypted data, where the statistical analysis itself has sensitive information; the evaluation of polynomials over encrypted data [KSW08] and [ALS16, Appx. B]; the construction of bounded collusion FE for all circuits [ALS16]; the construction of trace-and-revoke systems [ABP+17]; and more recently the construction of non zero inner product encryption (NIPE) schemes [KY19], which themselves are used to build identity-based revocation (IBR) schemes (a type of broadcast encryption scheme allowing for efficient revocation of users' ability to decrypt).

Increasing security of IPFE. The first IPFE schemes relying on standard assumptions were put forth by Abdalla et al. in 2015 [ABDP15]. They provided a generic construction to devise IPFE schemes which are secure against passive adversaries in the *selective* model. Selective security requires that adversaries commit to challenge messages *before* seeing the public parameters of the scheme, or being able to perform any key derivation queries. Though of great theoretical interest, such a notion of security does not reflect the power of an attacker in the real world, hence such schemes are not sufficiently secure for practical applications. In the public key setting, the first *adaptively* secure (ind-fe-cpa) schemes were put forth by Agrawal et al. [ALS16] under the learning with errors (LWE), DDH and DCR assumptions. Conversely to [ABDP15], the schemes of [ALS16] are *not* generic. Shortly afterwards (and independently), Abdalla et al. [ABDP16] provided a generic construction for ind-fe-cpa-secure IPFE, though when instantiated from concrete assumptions, the security reductions are less tight than those obtained in [ALS16].

Zhang et al. [ZMY17] and then independently, and more formally Benhamouda et al. [BBL17], where the first to consider security against active adversaries for IPFE (ind-fe-cca-security). Benhamouda et al. provide generic constructions from projective hash functions – as defined in Chapter 3 – satisfying a number of properties which they introduce, to both ind-fe-cpa and ind-fe-cca-secure IPFE schemes. They instantiate their generic construction from the DDH, the DCR, and the matrix DDH assumptions [EHK+13].

Regarding simulation based security, Agrawal et al. [ALMT20a], building upon the work of [Wee17] showed that variants of the schemes in [ALS16] can be proven adaptively secure in the simulation based security model, these are the first IPFE schemes which are proven adaptively secure in this model.

Increasing expressiveness of IPFE. Independently, many concurrent works have aimed at increasing the expressiveness of IPFE. Function privacy for IPFE was first realised by Bishop et al. using bilinear maps [BJK15,DDM16]; intuitively, this means that a secret key sk_f should not reveal any more information on the function f it encodes than what is implicitly leaked by $f(x)$. Then IPFE was extended to the multi-input setting: multi-input FE, introduced by Goldwasser et al. in [GGG+14], is a generalisation of FE to the setting of multi-input functions, so that the function can be computed over several different inputs that can be encrypted independently. The first construction for multi-input IPFE, relying on standard assumptions (the k-Lin assumption in prime-order bilinear groups), was put forth by Abdalla et al. in [AGRW17]. Datta et al. [DOT18] were the first to provide a function-hiding multi-input IPFE scheme (still from pairings). Then in [ACF+18], Abdalla et al. provide new techniques converting single-input IPFE into multi-input schemes for the same functionality. This leads to multi-input IPFE schemes from a range of assumptions e.g. DDH, LWE, and DCR. In [Tom19], Tomida presents a generic conversion from function-hiding IPFE to function-hiding multi-input IPFE. He also improves the state of the art for security of function hiding IPFE by providing the first tightly secure schemes (security does not degrade with the number of ciphertexts from different users granted to the adversary); these rely on a generalisation of the DDH assumption. The multi-client setting (similar to multi-input FE, but where each input is generated by a different party) was first adressed by Chotard et al. in [CDG$^+$18a] (which they later improve in [CDG$^+$18b]). They provide such a scheme from the DDH assumption, along with a decentralised scheme relying on pairings, where one no longer needs a trusted authority to produce decryption keys. Abdalla et al. [ABKW19] provide a compiler transforming any multi-client IPFE scheme (satisfying some properties) into a decentralised scheme. They thereby obtain decentralised schemes in the standard model from LWE, DDH and DCR. Shortly afterwards, Abdalla et al. [ABG19] presented a generic construction for multi-client IPFE from single-input IPFE in the standard model (they also adapt the compiler of [ABKW19] to obtain a decentralised version of their scheme). On a different note, Do et al. [DPP20] introduce traceability, which prevents users (called traitors) from leaking or selling their functional decryption key; they also provide a solution from DDH with the help of pairings. As a final example, in [ACGU20], Abdalla et al. combine an access control functionality with the possibility of performing linear operations on encrypted data provided by IPFE.

Our Contributions

The aforementioned studies demonstrate the huge attention IPFE has received, and the effort which has been put into diversifying the provided functionalities. However much less effort has been applied to reinforcing security. We focus on providing efficient and generic solutions which strengthen the security of standard IPFE; an interesting line for future work would be to consider how the aforementioned extensions in terms of functionality would articulate with our generic constructions. In this work, we provide some of the most efficient IPFE schemes to date from class group cryptography, and we hugely improve the tightness of security proofs for generic IPFE, thereby allowing to implement more efficient schemes for a given security level. Precisely, we devise generic constructions for building ind-fe-cpa and ind-fe-cca-secure IPFE schemes from projective hash functions. Though these generic schemes may appear similar to those of [BBL17], we require starkly different properties of our underlying PHFs, while our proofs, which are notably different to theirs, are much tighter.

We observe that, as in many proposals for IPFE schemes (e.g. [ALS16,CLT18a,ALMT20b]), the constructions provided by [BBL17] follow the lines of the generic construction of [ABDP15]. They also use similar proof techniques to [ABDP15] even though, as previously mentioned, these proofs are for selective security, and so the challenger sees the challenge messages chosen by the adversary *before* setting the scheme's public parameters and choosing the master key pair. This somewhat explains why, so as to attain adaptive security (both against passive and active adversaries), [BBL17] have their challenger guess the difference between the challenge messages. Precisely denoting msk the master secret key sampled by the challenger, using the difference in the challenge messages $\boldsymbol{m}_0 - \boldsymbol{m}_1$, they build another master secret key msk', such that using msk' in place of msk is unnoticeable to the adversary. However, if the initial guess for $\boldsymbol{m}_0 - \boldsymbol{m}_1$ is wrong, the challenger aborts. This results in an exponential loss in a term of the security reduction, which appears in both their ind-fe-cpa and their ind-fe-cca-secure schemes. For their only scheme able to decrypt inner products of any size (based on DCR), parameters must be well chosen to compensate for the security loss. This multiplies key sizes by a factor which grows (at least) linearly in the dimension and in the security parameter λ. When concretely instantiated, this results in schemes with large ciphertexts, decryption keys, and timings for encryption and decryption which are prohibitive for practical use.

Though their constructions are similar, [ALS16] use a very different proof technique. For each of their considered assumptions, they provide a specific proof, all of which follow a similar structure: they carefully evaluate the maximum information leaked to the adversary by the public key, decryption key queries and by the challenge ciphertext, and analyse the distribution of keys used to mask confidential information conditioned on the view of the adversary. Using conditional probabilities allows one to carry out the analysis *a posteriori*, while ensuring security against adversaries which are adaptive with regard to key queries and the choice of the challenge messages.

This technique resembles that used in [CS02] where Cramer and Shoup's definition of smoothness allows to do exactly the above, only in the context of PKE instead of IPFE.

Inspired by the proof techniques of [ALS16] for *non generic* ind-fe-cpa-secure IPFE, we define the notion of *vector-smoothness* (*cf.* Section 4.2.3) for PHFs, which extends smoothness to the IPFE setting. This property allows us to devise a *generic* construction for ind-fe-cpa-secure IPFE from PHFs. Our requirements on the underlying PHFs differ notably from those of [BBL17], and we obtain a tighter security proof. We instantiate this generic construction from the running examples of Chapter 3. When instantiated from DDH or DCR, we obtain the schemes of [ALS16], while instantiations from HSM-CL and DDH-f yield the FE schemes computing inner products in $\mathbf{Z}$ which we published in [CLT18a]. These are the most efficient ind-fe-cpa-secure IPFE schemes to date. We thereby provide a unified view of all these schemes, extracting the essence of their multiple proofs. This explains why, in this thesis, we choose to present our schemes in a different light to that done in the work we published in [CLT18a], since we are able to retrieve them from a generic approach.

Further extending the ideas of Cramer and Shoup, who use universal$_2$ PHFs to build ind-cca-secure PKE, we also define a new property for PHFs called *vector-universality* (*cf.* Section 4.2.4). This allows us to devise a generic construction for ind-fe-cca-secure IPFE from PHFs. Again, the resulting scheme is essentially that of [BBL17], but with notably different properties required of the underlying PHFs, and a *much tighter* security proof. We further note that though the comparison is quite involved (*cf.* Appendix A), for equivalent security goals, the properties we require of our PHFs to build IPFE are implied by those required in [BBL17]. We thereby provide the first generic constructions for ind-fe-cpa and ind-fe-cca secure IPFE which do not suffer an exponential loss in a term of the security reduction. Instantiations of our constructions demonstrate that ind-fe-cca-secure IPFE is practical.

Related publications and submissions. Most of the work in this chapter can be found in:

- [CLT18a] For instantiations from HSM-CL and DDH-f of our ind-fe-cpa-secure IPFE schemes in $\mathbf{Z}$ and modulo a prime, along with efficiency comparisons to [ALS16].
- [CLT20] (submission) For the generic constructions, both ind-fe-cpa and ind-fe-cca-secure, their security proofs and efficiency comparisons to [BBL17].

Road map

In Section 4.1 we formally define FE, the security models adopted for our constructions, and the restriction of FE to the inner product functionality. As mentioned previously, we need to evaluate the maximum information leaked to the adversary by the public key, decryption key queries and by the challenge ciphertext. In order to capture this *maximum information* in a *generic* way, in Section 4.2 we define compatibility and security properties required of the underlying PHFs. To help the reader assimilate technical concepts, all definitions and properties are illustrated with the running examples from DDH, HSM-CL and DDH-f studied in Chapter 3. Next in Section 4.3 we provide a generic construction secure against passive adversaries, which we also instantiate from our running examples, thereby retrieving the DDH scheme of [ALS16], and our schemes of [CLT18a]. In Section 4.3.2 we consider modular IPFE, i.e. functional encryption computing inner products modulo an integer p, where p may or may not be prime. We discuss how modular IPFE schemes fit into out framework, so as to retrieve the stateful IPFE scheme from DCR of [ALS16]; we also describe stateful modular IPFE schemes arising from DDH-f and HSM-CL. In Section 4.4 we further extend the former generic construction to deal with stronger active adversaries, and detail instantiations from our running examples. As detailed in Section 4.5, the impact of our results on the practical efficiency of protocols secure against active adversaries is staggering.

Finally, in Section 4.6 we discuss future perspectives. Namely we detail an interesting application of our work: using the generic construction of [KY19] one can build IBR schemes where the collusion and misbehaviour of users does not compromise the scheme's security. In this same section, building upon the techniques of [Wee17, ALMT20a], we discuss which properties would be required of the underlying PHFs to further attain security in the simulation based model.

4.1 Inner Product Functional Encryption

We here first define general functional encryption and the functionality considered in this thesis, i.e. the inner product functionality. We next provide security definitions against active and passive adversaries, in both the indistinguishability game based model, and the stronger simulation model.

4.1.1 Inner Product Functional Encryption

Inner Product Functional Encryption (IPFE) is a special case of functional encryption, as first formalised by Boneh, Sahai and Waters in [BSW11]. Let us first provide the definition of a *functionality.*

Definition 4.1 (Functionality)**.** A functionality $F = \{F_\lambda\}_{\lambda\in\mathbf{N}}$ defined over $(\mathcal{K}, \mathcal{M})$ is a function $F : \mathcal{K} \times \mathcal{M} \to \Sigma \cup \{\perp\}$, where $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbf{N}}$ is a message space, $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbf{N}}$ is a key space and Σ is an output space, which does not contain the special symbol $\perp$.

Definition 4.2 (Functional encryption scheme)**.** A functional encryption (FE) scheme for a functionality $F \in \{F_\lambda\}_{\lambda\in\mathbf{N}}$ over $(\mathcal{K}, \mathcal{M})$ is a tuple $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms with the following specifications:

- $\mathsf{Setup}(1^\lambda)$ is a PPT algorithm which on input 1^λ, outputs a master public key mpk and a master secret key msk;
- $\mathsf{KeyDer}(\mathsf{msk}, k)$ is a PT algorithm which on input msk and a key $k \in \mathcal{K}$, outputs a functional decryption key sk_k;
- $\mathsf{Enc}(\mathsf{mpk}, m)$ is a PPT algorithm which on input mpk and a message $m \in \mathcal{M}$, outputs a ciphertext c;
- $\mathsf{Dec}(\mathsf{mpk}, sk_k, c)$ is a DPT algorithm which on input mpk, a functional decryption key sk_k and a ciphertext $c \in \mathcal{C}$, outputs $v \in \Sigma \cup \{\perp\}$.

Correctness requires that for all $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, all keys $k \in \mathcal{K}$ and all messages $m \in \mathcal{M}$, if $sk_k \leftarrow \mathsf{KeyDer}(\mathsf{msk}, k)$ and $c \leftarrow \mathsf{Enc}(\mathsf{mpk}, m)$, with overwhelming probability it holds that, if $v \leftarrow \mathsf{Dec}(\mathsf{mpk}, sk_k, c)$ then $v = F(k, m)$ whenever $F(k, m) \neq \perp$.

The Inner Product Functionality

We consider the inner product functionality, i.e. given a ring $\mathcal{R}$ and two efficiently recognisable subsets $\mathcal{M}$ and $\mathcal{K}$ of $\mathcal{R}^\ell$, the functionality is $F : \mathcal{K} \times \mathcal{M} \to \mathcal{R} \cup \{\perp\}$ such that $F(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{R}$.

4.1.2 Security

Intuitively security for FE states that given the ciphertext of a message m, the only information obtained from the decryption key sk_k is the evaluation of the function $F(k, m)$. There exist two main security definitions for FE, indistinguishability-based and a stronger simulation-based security. The former – which is the model we mainly consider – asks that no PT adversary can distinguish ciphertexts of any two plaintexts m_0 and m_1 of its' choice. One can grant various degrees of power to the adversary, thereby defining different levels of security [BSW11, O'N10]. *Security against chosen plaintext attacks* captures the idea that a user, who is granted specific decryption keys, learns nothing more than the information these keys are intended to reveal. It does not however capture the scenario where an adversary (said to be active) additionally coerces honest users to run the decryption protocol (with decryption keys unknown to the adversary) on potentially malformed ciphertexts. To deal with such *active* adversaries, one must ensure *security against chosen ciphertext attacks* [NP15, BBL17].

The definitions we provide are for *adaptive* security, meaning the adversary has access to the systems' public parameters, and can perform a series of key derivation requests *before* choosing the challenge messages. The weaker *selective* security model requires the adversary commits to challenge messages before seeing the public key (or performing any queries). We first present the existing game-based definition of security against passive adversaries as provided in [BSW11]. Then we consider a natural extension which deals with *active adversaries.*

In this section we consider an FE scheme $\mathsf{FE} := (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ for functionality F over message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbf{N}}$, and key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbf{N}}$, and let $\mathcal{A}$ be a PPT adversary.

Indistinguishability against Chosen Plaintext Attacks

We define standard security experiment against adaptive passive adversaries for FE.

The experiment $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda)$. For $\lambda \in \mathbf{N}$ we denote by $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda)$ the random variable that is defined via the following experiment involving the scheme FE, an adversary $\mathcal{A}$, and a challenger $\mathcal{C}$:

1. **Setup phase:** $\mathcal{C}$ samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$; and $\beta \hookleftarrow \{0, 1\}$.

2. **Pre-challenge phase:** $\mathcal{A}$ on input $(1^\lambda, \mathsf{mpk})$ adaptively issues queries (key, k) where $k \in \mathcal{K}_\lambda$. Upon receiving the i-th query (key, k_i), $\mathcal{C}$ computes $sk_{k_i} \leftarrow \mathsf{KeyDer}(\mathsf{msk}, k_i)$; and sends sk_{k_i} to $\mathcal{A}$.

3. **Challenge phase:** $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}$; $\mathcal{C}$ computes $c^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_\beta)$ and sends it to $\mathcal{A}$.

4. **Post-challenge phase:** $\mathcal{A}$ adaptively issues queries as in the pre-challenge phase.

5. **Output phase:** $\mathcal{A}$ outputs β'. The output of the experiment is 1 if and only if $\beta = \beta'$.

Valid adversaries. As standard in FE, we rule out adversaries that can easily distinguish between the challenge messages m_0 and m_1 using their queries. Specifically, an $\mathsf{ind\text{-}fe\text{-}cpa}$ adversary is *valid* if all key queries (key, k) satisfy $F(k, m_0) = F(k, m_1)$.

Having defined the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda)$ and the notion of a valid adversary, we are now ready to present the notion of adaptive security against chosen plaintext attacks for FE schemes.

Definition 4.3. An FE scheme FE for functionality F over a message space $\mathcal{M}$, and a key space $\mathcal{K}$ is adaptively secure against chosen plaintext attacks ($\mathsf{ind\text{-}fe\text{-}cpa}$) if for any PPT valid adversary $\mathcal{A}$, it holds that:

$$\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

Indistinguishability against Chosen Ciphertext Attacks

We here provide a definition which additionally deals with *active adversaries*. Since such adversaries may corrupt data, we allow them to perform decryption queries for ciphertexts of their choice.

The experiment $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}}(\lambda)$. For $\lambda \in \mathbf{N}$ we denote by $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}}(\lambda)$ the random variable that is defined via the following experiment involving the scheme FE, the adversary $\mathcal{A}$, and a challenger $\mathcal{C}$:

1. **Setup phase:** $\mathcal{C}$ samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$; and $\beta \hookleftarrow \{0, 1\}$.
2. **Pre-challenge phase:** $\mathcal{A}$ on input $(1^\lambda, \mathsf{mpk})$ adaptively issues queries:
   - (key, k) where $k \in \mathcal{K}$. Upon receiving the i-th query (key, k_i), $\mathcal{C}$ computes $sk_{k_i} \leftarrow \mathsf{KeyDer}(\mathsf{msk}, k_i)$; and sends sk_{k_i} to $\mathcal{A}$.
   - $(\mathsf{decrypt}, c, k)$ where $k \in \mathcal{K}$ and c is a ciphertext. Upon receiving the j-th query $(\mathsf{decrypt}, c_j, k_j)$, $\mathcal{C}$ computes $sk_{k_j} \leftarrow \mathsf{KeyDer}(\mathsf{msk}, k_j)$; $\mathsf{res}_j \leftarrow \mathsf{Dec}(\mathsf{mpk}, sk_{k_j}, c_j)$; and sends res_j to $\mathcal{A}$.
3. **Challenge phase:** $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}$; $\mathcal{C}$ computes $c^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_\beta)$ and sends c^* to $\mathcal{A}$.
4. **Post-challenge phase:** $\mathcal{A}$ adaptively issues queries as in the pre-challenge phase.
5. **Output phase:** $\mathcal{A}$ outputs β'. The output of the experiment is 1 if and only if $\beta = \beta'$.

Valid adversaries. An $\mathsf{ind\text{-}fe\text{-}cca}$ adversary is *valid* if every key query (key, k) satisfies $F(k, m_0) = F(k, m_1)$, and every decryption query $(\mathsf{decrypt}, c, k)$ satisfies $c \neq c^*$.

Having defined the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}}$ and the notion of a valid adversary, we are ready to present our notion of adaptive security against chosen ciphertext attacks for FE schemes.

Definition 4.4. An FE scheme FE for functionality F over a message space $\mathcal{M}$, and a key space $\mathcal{K}$ is adaptively secure against chosen ciphertext attacks ($\mathsf{ind\text{-}fe\text{-}cca}$) if for any PPT valid adversary $\mathcal{A}$, it holds that:

$$\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}} = 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

Remark. In the specific case where the considered functionality is the inner product, such a level of security was already considered in [ZMY17,BBL17]. The above definition of ind-fe-cca-security is equivalent to that used in [BBL17]. The work of Zhang et al. [ZMY17] does not provide a formal definition for ind-fe-cca-security. However, the model they use is very restrictive as they bound the number of key derivation queries allowed by the adversary. Indeed, in the protocol they provide, computing inner products for vectors of length ℓ, if an adversary obtains ℓ decryption keys (potentially for the same key $\boldsymbol{k} \in \mathcal{K}$), it can then trivially break ciphertext integrity, and thereby force the protocol to run in unprescribed conditions.

4.2 Building IPFE from PHF

In Sections 4.3 and 4.4 we provide generic constructions for ind-fe-cpa and ind-fe-cca-secure inner product functional encryption from PHFs. For correctness of these constructions, we first introduce some compatibility properties which the PHF must satisfy. For security we define two new properties: *vector-smoothness* and *vector-universality.* If the PHF used for confidentiality is *vector-smooth*, one can build ind-fe-cpa-secure inner product functional encryption schemes. To further attain ind-fe-cca-security, the PHF used to ensure ciphertext integrity must be *vector-universal.*

4.2.1 Compatibility Properties for PHFs

To build inner product functional encryption from a projective hash function, one needs the PHF to be *compatible* with the ring in which inner products are computed; one also needs to impose restrictions on the message space $\mathcal{M}$ and the space $\mathcal{K}$ from which decryption keys are derived. Throughout this chapter, we restrict ourselves to inner products computed in the ring $\mathcal{R} := \mathbf{Z}$ or $\mathcal{R} := \mathbf{Z}/q\mathbf{Z}$ for some prime q.

Definition 4.5 (ipfe-compatibility)**.** Let $\mathcal{R}$ be a ring, either $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q. Let $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be an SMP, and consider $\mathsf{H} := (\mathsf{hashkg}, \widehat{\mathsf{projkg}}, \mathsf{projkg}, \mathsf{hash}, \widehat{\mathsf{projhash}}, \mathsf{projhash})$ the associated PHF. One says H is $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible if:

- the hash key space is $K_{\mathsf{hk}} := \mathcal{R}^a$ for some positive integer a;
- H is key homomorphic, where the (additive) group operation associated to K_{hk} is the addition of $\mathcal{R}$ performed point-wise;
- the co-domain Π of hash is a finite Abelian group which contains a cyclic subgroup F, generated by f, of order n_f;
- if $\mathcal{R} = \mathbf{Z}/q\mathbf{Z}$ then $F = \Pi$ is of prime order $n_f = q$;
- $\mathcal{M}$ and $\mathcal{K}$ are efficiently recognisable subsets of $\mathcal{R}^\ell$, for some positive integer ℓ;
- there exists an efficient algorithm $\log_f$ which, for all $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{k} \in \mathcal{K}$, computes $\log_f(f^{\langle \boldsymbol{m}, \boldsymbol{k} \rangle}) = \langle \boldsymbol{m}, \boldsymbol{k} \rangle \in \mathcal{R}$.

Remark. An EPHF eH built from an $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible PHF H via the generic construction of Section 3.4.3 is $(\mathcal{R}, 2a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible.

Notation 4.6. Let $\mathcal{R}$ be a ring, either $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q, and $\mathcal{M}$ a subset of $\mathcal{R}$. We denote:

$$\Delta\mathcal{M} := \{\boldsymbol{x}_0 - \boldsymbol{x}_1 \mid \boldsymbol{x}_0 \neq \boldsymbol{x}_1 \in \mathcal{M}\}.$$

Running Example 1 – DDH

Here $\mathcal{R} := \mathbf{Z}/q\mathbf{Z}$ and $a := 2$ which is consistent with $K_{\mathsf{hk}} = (\mathbf{Z}/q\mathbf{Z})^2$ and $K_{\mathsf{ehk}} = (\mathbf{Z}/q\mathbf{Z})^4$. Recall that $\mathsf{hash} : \mathbf{Z}/q\mathbf{Z}^2 \times G^2 \to G$, where G is a cyclic group of prime order q generated by g. Thus we set $f := g$, which generates $F := G$ and $n_f := q$. This implies that the algorithm $\log_f$ is the DL in G. Note that in a DDH group the DL problem is hard by assumption, so computing $\log_f$ can only be done efficiently for small values of the inner product. Thus $\mathcal{M}$ and $\mathcal{K}$ are subsets of $(\mathbf{Z}/q\mathbf{Z})^\ell$ s.t. $\forall \boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{k} \in \mathcal{K}$, $\log_g(g^{\langle \boldsymbol{m}, \boldsymbol{k} \rangle}) = \langle \boldsymbol{m}, \boldsymbol{k} \rangle \in \mathbf{Z}/q\mathbf{Z}$ is computable in time $\mathsf{poly}(\lambda)$. To summarise, for $\ell \in \mathbf{N}$,

$$\mathsf{H}_{\mathsf{ddh}} \text{ is a } (\mathbf{Z}/q\mathbf{Z}, 2, g, q, \ell, \mathcal{M}, \mathcal{K})\text{-}\mathsf{ipfe}\text{-compatible PHF.}$$

Running Example 2 – HSM-CL

Here $\mathcal{R} := \mathbf{Z}$ and $a := 1$ which is consistent with $K_{\mathsf{hk}} = \mathbf{Z}$ and $K_{\mathsf{ehk}} = \mathbf{Z}^2$. Recall that $\mathsf{hash} : \mathbf{Z} \times \widehat{G} \to \widehat{G}$, where $\widehat{G}$ which is a finite Abelian group, and F is a cyclic subgroup of $\widehat{G}$ of prime order q, generated by f. We set:

$$\mathcal{M} = \mathcal{K} = \{\boldsymbol{x} \in \mathbf{Z}^\ell \ : \ ||\boldsymbol{x}||_\infty < \sqrt{\frac{q}{2\ell}}\}.$$

Let us now describe our implementation for algorithm $\log_f$. For $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{k} \in \mathcal{K}$, let us denote $M := f^{\langle \boldsymbol{m}, \boldsymbol{k} \rangle}$; observe that since $||\boldsymbol{m}||_\infty$ and $||\boldsymbol{k}||_\infty < \sqrt{\frac{q}{2\ell}}$, it holds that $-q/2 < \langle \boldsymbol{m}, \boldsymbol{k} \rangle < q/2$. First one uses the Solve algorithm of Definition 3.1 to compute $\mathsf{sol} \leftarrow \mathsf{Solve}(M)$, then if $\mathsf{sol} \geqslant q/2$, one returns $(\mathsf{sol} - q)$, otherwise one returns sol. With this implementation it holds that $\log_f(f^{\langle \boldsymbol{m}, \boldsymbol{k} \rangle}) = \langle \boldsymbol{m}, \boldsymbol{k} \rangle$ in $\mathbf{Z}$. To summarise, for $\ell \in \mathbf{N}$,

$$\mathsf{H}_{\mathsf{hsm\text{-}cl}} \text{ is a } (\mathbf{Z}, 1, f, q, \ell, \mathcal{M}, \mathcal{K})\text{-}\mathsf{ipfe}\text{-compatible PHF.}$$

Running Example 3 – DDH-f

Here $\mathcal{R} := \mathbf{Z}$ and $a := 2$, which is consistent with $K_{\mathsf{hk}} = \mathbf{Z}^2$ and $K_{\mathsf{ehk}} = \mathbf{Z}^4$. Recall that $\mathsf{hash} : \mathbf{Z}^2 \times \widehat{G}^2 \to \widehat{G}$. For algorithm $\log_f$ we use the same as that described for running example 2. Consequently, for $\ell \in \mathbf{N}$, letting $\mathcal{M} = \mathcal{K} = \{\boldsymbol{x} \in \mathbf{Z}^\ell \ : \ ||\boldsymbol{x}||_\infty < \sqrt{\frac{q}{2\ell}}\}$, it holds that:

$$\mathsf{H}_{\mathsf{ddh\text{-}f}} \text{ is a } (\mathbf{Z}, 2, f, q, \ell, \mathcal{M}, \mathcal{K})\text{-}\mathsf{ipfe}\text{-compatible PHF.}$$

4.2.2 Associated Matrix

We here define the notion of a matrix $\mathbf{B}_{\boldsymbol{m}}$ *associated* to a vector $\boldsymbol{m}$. In our upcoming constructions, $\boldsymbol{m}$ will be the difference between the two challenge message vectors. As such, valid adversaries can request decryption keys associated to vectors $\boldsymbol{k} \in \mathcal{K}$ satisfying $\boldsymbol{k} \in \boldsymbol{m}^\perp$. The matrix $\mathbf{B}_{\boldsymbol{m}}$ is constructed in such a way that any such $\boldsymbol{k}$ can be written as a linear combination of the top $\ell - 1$ rows of $\mathbf{B}_{\boldsymbol{m}}$. Conversely, any $\boldsymbol{k} \notin \boldsymbol{m}^\perp$ – for which a decryption key trivially reveals which of the challenge messages was encrypted – has some contribution from the last row of $\mathbf{B}_{\boldsymbol{m}}$. The secret values of our protocols, when projected onto this last row, must conserve sufficient entropy for security to hold.

Definition 4.7. Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; ℓ and a be positive integers; consider an $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-$\mathsf{ipfe}$-compatible projective hash function H; and a non-zero vector $\boldsymbol{m} \in \mathcal{R}^\ell$. We say $\mathbf{B}_{\boldsymbol{m}} \in \mathcal{R}^{\ell \times \ell}$ is *a matrix associated to* $\boldsymbol{m}$ if, denoting

$(\boldsymbol{b}_1, \dots, \boldsymbol{b}_\ell)$ the rows of $\mathbf{B}_{\boldsymbol{m}}$ it holds that: (1) $\mathbf{B}_{\boldsymbol{m}}$ is invertible mod n_f; (2) $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{\ell-1})$ form a basis of $\boldsymbol{m}^\perp$; (3) $\boldsymbol{b}_\ell \notin \boldsymbol{m}^\perp$ and if $\mathcal{R} = \mathbf{Z}$ then $\boldsymbol{b}_\ell = \boldsymbol{m}$.

Lemma 4.8 states conditions to efficiently build a matrix associated to $\boldsymbol{m}$.

Lemma 4.8. Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; ℓ and a be positive integers; and consider an $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible projective hash function H, where n_f is either prime or hard to factor. From any $\boldsymbol{m} \in \Delta\mathcal{M}$ one can efficiently and deterministically construct a matrix $\mathbf{B}_{\boldsymbol{m}} \in \mathcal{R}^{\ell \times \ell}$ associated to $\boldsymbol{m}$.

Proof. If $\mathcal{R} = \mathbf{Z}/q\mathbf{Z}$ then by Definition 4.5, $q = n_f$ is prime. In this case we proceed as suggested in [ALS16, Appx. E]: given $\boldsymbol{m}$ one deterministically generates a $\mathbf{Z}/q\mathbf{Z}$-basis $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{\ell-1}) \in (\mathbf{Z}/q\mathbf{Z})^{(\ell-1)\times\ell}$ of $\boldsymbol{m}^\perp$. Let $\boldsymbol{b}_\ell \in (\mathbf{Z}/q\mathbf{Z})^\ell$ be a vector outside the subspace $\boldsymbol{m}^\perp$, also chosen in a deterministic manner. The resulting matrix $\mathbf{B} \in (\mathbf{Z}/q\mathbf{Z})^{\ell\times\ell}$ whose rows are the vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_\ell$ is invertible modulo q.

If $\mathcal{R} = \mathbf{Z}$ then first observe that by Definition 4.5, one has $\log_f(f^{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle$ for any $\boldsymbol{x} \in \mathcal{M}$, $\boldsymbol{y} \in \mathcal{K}$. Since f is of order n_f, this implies that vectors in $\mathcal{M}$ (resp $\mathcal{K}$) are of bounded norm, i.e. $\mathcal{M}$ and $\mathcal{K}$ are subsets of $\{\boldsymbol{x} \in \mathbf{Z}^\ell \ : \ ||\boldsymbol{x}||_\infty < \sqrt{\frac{q}{2\ell}}\}$.

Without loss of generality, assume the n_0 first coordinates of $\boldsymbol{m} \in \mathbf{Z}^\ell$ are zero (for some n_0), and all remaining entries are non-zero. The rows $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{\ell-1} \in \mathbf{Z}^\ell$ of the following matrix (due to [ALS16, Proof of Theorem 2]) form a basis of $\boldsymbol{m}^\perp$:

$$\left[\begin{array}{c|ccccc} \mathbf{I}_{n_0} & & & & & \\ \hline & -m_{n_0+2} & m_{n_0+1} & & & \\ & & -m_{n_0+3} & m_{n_0+2} & & \\ & & & \ddots & \ddots & \\ & & & & -m_\ell & m_{\ell-1} \end{array}\right] \in \mathcal{R}^{(\ell-1)\times\ell}.$$

Letting $\boldsymbol{b}_\ell := \boldsymbol{m}$, the matrix $\mathbf{B} \in \mathcal{R}^{\ell\times\ell}$ whose rows are the vectors $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_\ell)$ is invertible modulo n_f. If n_f is prime, from the norm bounds this is always true (this can be deduced from [ALS16, Proof of Theorem 2]). If n_f is composite, either $\mathbf{B}$ is invertible modulo n_f, otherwise its determinant reveals a non trivial factor of n_f [ALS16]. □

Notation 4.9. The fact Lemma 4.8 builds $\mathbf{B}_{\boldsymbol{m}}$ *deterministically* from $\boldsymbol{m}$ allows us to build – from a random variable M taking values in $\Delta\mathcal{M}$ – the matrix of random variables $\mathbf{B}_M$. We will use this notation in our definitions and proofs. We denote $\boldsymbol{b}_1^M, \dots, \boldsymbol{b}_\ell^M$ the rows of $\mathbf{B}_M$.

4.2.3 Confidentiality

In our generic construction for building inner product functional encryption from projective hash functions, the master secret key is a vector of ℓ hash keys of which adversaries can request linear combinations. These linear combinations correspond to the functional decryption keys granted to adversary. We here introduce a property called *vector smoothness*, ensuring confidentiality given this extra leakage of information. Vector smoothness is an extension of the definition of smoothness of [CS02], which ensures confidentiality given a PKE scheme's public parameters. In the context of IPFE, one also needs to deal with key derivation queries performed by the adversary. Precisely, vector smoothness ensures that given the projection of the master secret key on a hyperplane $\mathcal{H}$, its projection onto a line orthogonal to $\mathcal{H}$ remains uniformly distributed. This latter projection masks the challenge message in our constructions. This new property allows to capture the techniques used in [ALS16] (to build ind-fe-cpa-secure

IPFE schemes from DDH and DCR). Consequently, the proofs of Lemmas 4.12 to 4.14 share similarities with those of [ALS16, Thm. 1].

Definition 4.10 (δ_{vs}-vector-smooth over $\mathcal{X}$ on F)**.** Let ℓ and a be positive integers. Let $\mathcal{R}$ be a ring, either $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be an SMP, and H the associated PHF which we assume to be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible. For $i \in [\ell]$, let $\mathsf{hk}_i \leftarrow \mathsf{hashkg}(\mathcal{SM})$, and $\mathbf{hk} := (\mathsf{hk}_1, \ldots, \mathsf{hk}_\ell)^T$. Consider a random variable M taking values in $\Delta\mathcal{M}$ and the associated matrix $\mathbf{B}_M \in \mathcal{R}^{\ell\times\ell}$. Let $X \hookleftarrow \mathcal{X}\backslash\mathcal{L}$, and $Y \hookleftarrow \mathcal{U}(F)$. Then H is $\delta_{vs}(\ell)$-vector-smooth over $\mathcal{X}$ on F if the following tuples of random variables are $\delta_{vs}(\ell)$-close:

$$\mathcal{U} := \{M, X, \widehat{\mathsf{projkg}}(\mathbf{hk}), \{(\boldsymbol{b}_j^M)^T \cdot \mathbf{hk}\}_{j\in[\ell-1]}, \mathsf{hash}((\boldsymbol{b}_\ell^M)^T \cdot \mathbf{hk}, X) \cdot Y\} \text{ and}$$

$$\mathcal{V} := \left\{M, X, \widehat{\mathsf{projkg}}(\mathbf{hk}), \{(\boldsymbol{b}_j^M)^T \cdot \mathbf{hk}\}_{j\in[\ell-1]}, \mathsf{hash}((\boldsymbol{b}_\ell^M)^T \cdot \mathbf{hk}, X)\right\}.$$

We next give a convenient reformulation of vector smoothness for PHFs which possess homomorphic properties and are decomposable.

Lemma 4.11. Using the same notations as in Definition 4.10, assume H is further homomorphic, key homomorphic and $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable. Since $X \in \mathcal{X}\backslash\mathcal{L}$, there exist unique $(x, w) \in \mathsf{R}$ and $y \in \langle\Upsilon\rangle$ satisfying $X = x \cdot y$. Then H is δ_{vs}-vector-smooth over $\mathcal{X}$ on F if and only if the following distributions are δ_{vs}-close:

$$\mathcal{U}' := \{M, X, \widehat{\mathsf{projkg}}(\mathbf{hk}), \{(\boldsymbol{b}_j^M)^T \cdot \mathbf{hk}\}_{j\in[\ell-1]}, Y\} \text{ and}$$

$$\mathcal{V}' := \left\{M, X, \widehat{\mathsf{projkg}}(\mathbf{hk}), \{(\boldsymbol{b}_j^M)^T \cdot \mathbf{hk}\}_{j\in[\ell-1]}, \mathsf{hash}((\boldsymbol{b}_\ell^M)^T \cdot \mathbf{hk}, y)\right\}.$$

Proof. Consider distributions $\mathcal{U}$ and $\mathcal{V}$ of Definition 4.10. Consider $\boldsymbol{m} \in \Delta\mathcal{M}$, $(x_0, w_0) \in \mathsf{R}$, $y_0 \in \langle\Upsilon\rangle$, $\widehat{\mathsf{hp}}_i \in K_{\widehat{\mathsf{hp}}}$ for $i \in [\ell]$ and $v_j \in \mathcal{R}^a$ for $j \in [\ell - 1]$. The first three coordinates of $\mathcal{U}$ (or equivalently $\mathcal{V}$) fix $M = \boldsymbol{m}$; $x = x_0$; $w = w_0$; $y = y_0$; $\widehat{\mathsf{projkg}}(\mathsf{hk}_i) = \widehat{\mathsf{hp}}_i$ (which in turn fixes $\mathsf{hp}_i := \mathsf{projkg}(\mathsf{hk}_i)$ for some $\mathsf{hp}_i \in K_{\mathsf{hp}}$) for $i \in [\ell]$; and $\boldsymbol{b}_j^M)^T \cdot \mathbf{hk} = v_j$ for $j \in [\ell - 1]$. Consequently $\mathsf{hash}(\mathsf{hk}_i, x) = \mathsf{projhash}(\mathsf{hp}_i, x_0, w_0)$ for $i \in [\ell]$.

Let us denote $\mathbf{B}_{\boldsymbol{m}}$ the matrix associated to $\boldsymbol{m}$ built as per Lemma 4.8, and its rows $\boldsymbol{b}_1^{\boldsymbol{m}}, \ldots, \boldsymbol{b}_\ell^{\boldsymbol{m}}$. Since $M = \boldsymbol{m}$ it holds that $\mathbf{B}_M = \mathbf{B}_{\boldsymbol{m}}$. Using the key homomorphism of H we see that $\mathsf{hash}((\boldsymbol{b}_\ell^M)^T \cdot \mathbf{hk}, x) = \mathsf{hash}((\boldsymbol{b}_\ell^{\boldsymbol{m}})^T \cdot \mathbf{hk}, x_0)$ is fixed. And from the homomorphism of H it holds that $\mathsf{hash}((\boldsymbol{b}_\ell^M)^T \cdot \mathbf{hk}, X) = \mathsf{hash}((\boldsymbol{b}_\ell^{\boldsymbol{m}})^T \cdot \mathbf{hk}, x_0) \cdot \mathsf{hash}((\boldsymbol{b}_\ell^M)^T \cdot \mathbf{hk}, y)$. It is now clear that the statistical distance between $\mathcal{U}$ and $\mathcal{V}$ is equal to that between $\mathcal{U}'$ and $\mathcal{V}'$. □

Running Example 1 – $\mathsf{H}_{\mathsf{ddh}}$ is vector-smooth

Lemma 4.12, whose proof is inspired by [ALS16, Thm. 1], states $\mathsf{H}_{\mathsf{ddh}}$ is vector smooth.

Lemma 4.12. The projective hash function $\mathsf{H}_{\mathsf{ddh}}$ is 0-vector-smooth (over G on G).

Proof. For $i \in [\ell]$, let $\mathsf{hk}_i := (\kappa_{0,i}, \kappa_{1,i})$ denote independent random variables following the distribution $\mathcal{U}((\mathbf{Z}/q\mathbf{Z})^2)$; let $\boldsymbol{\kappa}_0 := (\kappa_{0,1}, \ldots, \kappa_{0,\ell})^T$, $\boldsymbol{\kappa}_1 := (\kappa_{1,1}, \ldots, \kappa_{1,\ell})^T$, and $\mathbf{hk} := (\mathsf{hk}_1, \ldots, \mathsf{hk}_\ell)^T$. Consider a random variable M taking values in $\Delta\mathcal{M}$ and the associated matrix $\mathbf{B}_M$. For $X \hookleftarrow \mathcal{X}\backslash\mathcal{L}$, there exist unique $\alpha \in \mathbf{Z}/q\mathbf{Z}$ and $\beta \in (\mathbf{Z}/q\mathbf{Z})^*$ s.t. $X = (g_0, g_1)^\alpha \odot (1, g_1)^\beta$. As noted in Lemma 4.11, for $\gamma \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, we need to evaluate the distance between:

$$\mathcal{U} = \left\{M, (g_0^\alpha, g_1^{\alpha+\beta}), \{g_0^{\kappa_{0,i}} g_1^{\kappa_{1,i}}\}_{i\in[\ell]}, \{\langle\boldsymbol{\kappa}_0, \boldsymbol{b}_i\rangle, \langle\boldsymbol{\kappa}_1, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]}, g_1^\gamma\right\}$$

$$\text{and } \mathcal{V} = \left\{M, (g_0^{\alpha}, g_1^{\alpha+\beta}), \{g_0^{\kappa_{0,i}} g_1^{\kappa_{1,i}}\}_{i\in[\ell]}, \{\langle \boldsymbol{\kappa}_0, \boldsymbol{b}_i\rangle, \langle \boldsymbol{\kappa}_1, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]}, g_1^{\beta\langle \boldsymbol{\kappa}_1, \boldsymbol{b}_\ell\rangle}\right\}.$$

Consider $\boldsymbol{m} \in \Delta\mathcal{M}$, $\alpha_0, \beta_0 \in \mathbf{Z}/q\mathbf{Z}$, $\boldsymbol{h} \in (\mathbf{Z}/q\mathbf{Z})^{\ell}$, $\boldsymbol{v}_0, \boldsymbol{v}_1 \in (\mathbf{Z}/q\mathbf{Z})^{\ell-1}$, and let us denote $a := \log_{g_0}(g_1)$. It suffices to study the distance between the random variables $Y := \beta\langle \boldsymbol{\kappa}_1, \boldsymbol{b}_\ell^M\rangle \bmod q$ and γ conditioned on the conjunction of the following events: $M = \boldsymbol{m}$; $(\alpha \bmod q, \beta \bmod q) = (\alpha_0 \bmod q, \beta_0 \bmod q)$; $\boldsymbol{\kappa}_0 + a\cdot \boldsymbol{\kappa}_1 \bmod q = \boldsymbol{h} \bmod q$; and for $j \in [\ell-1]$, $\langle \boldsymbol{\kappa}_0, \boldsymbol{b}_j^M\rangle = v_{0,j}$, and $\langle \boldsymbol{\kappa}_1, \boldsymbol{b}_j^M\rangle = v_{1,j}$. Let us denote $\mathbf{B}_{\boldsymbol{m}}$ the matrix associated to $\boldsymbol{m}$ built as per Lemma 4.8, and its rows $\boldsymbol{b}_1^{\boldsymbol{m}}, \dots, \boldsymbol{b}_\ell^{\boldsymbol{m}}$. Since $M = \boldsymbol{m}$ it holds that $\mathbf{B}_M = \mathbf{B}_{\boldsymbol{m}}$.

Let $(\boldsymbol{\kappa}_0^*, \boldsymbol{\kappa}_1^*)$ denote an arbitrary pair of vectors satisfying the same equations as $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1)$, i.e. those fixed by the aforementioned events. Then $\boldsymbol{\kappa}_0^* + a\boldsymbol{\kappa}_1^* = \boldsymbol{h} \bmod q$; $\langle \boldsymbol{\kappa}_0^*, \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = v_{0,j}$ and $\langle \boldsymbol{\kappa}_1^*; \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = v_{1,j}$ for $j \in [\ell-1]$. Since for $j \in [\ell-1]$, $\boldsymbol{b}_j^{\boldsymbol{m}} \in \boldsymbol{m}^{\perp}$, given the fixed information, the joint distribution of vectors $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1) \in (\mathbf{Z}/q\mathbf{Z})^2$ is:

$$\{(\boldsymbol{\kappa}_0^* - a\cdot\mu\cdot\boldsymbol{m} \bmod q, \boldsymbol{\kappa}_1^* + \mu\cdot\boldsymbol{m} \bmod q) \mid \mu \hookleftarrow \mathbf{Z}/q\mathbf{Z}\}$$

The conditional distribution of $\beta\langle \boldsymbol{\kappa}_1, \boldsymbol{b}_\ell^M\rangle$ is thus:

$$\{\beta(\langle \boldsymbol{\kappa}_1^*, \boldsymbol{b}_\ell^{\boldsymbol{m}}\rangle + \mu\langle \boldsymbol{m}, \boldsymbol{b}_\ell^{\boldsymbol{m}}\rangle) \bmod q \mid \mu \hookleftarrow \mathbf{Z}/q\mathbf{Z}\}$$

which is exactly $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$ since by construction, $\boldsymbol{b}_\ell^{\boldsymbol{m}} \notin \boldsymbol{m}^{\perp}$, so $\langle \boldsymbol{m}, \boldsymbol{b}_\ell^{\boldsymbol{m}}\rangle \neq 0 \bmod q$, and $\beta \in (\mathbf{Z}/q\mathbf{Z})^*$. Thus $\mathcal{U} = \mathcal{V}$ and $\mathsf{H}_{\mathsf{ddh}}$ is 0-vector-smooth. □

Running Example 2 – $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is vector-smooth

Lemma 4.13 states sufficient conditions for $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ to be vector smooth. We note that this lemma (and its proof) reflect the main ideas of the proof of ind-fe-cpa-security for our HSM-CL based IPFE, computing inner products in $\mathbf{Z}$, which we presented in [CLT18a, Thm. 7].

Lemma 4.13. If the hashkg algorithm of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ samples hashing keys from the Gaussian distribution $\mathcal{D}_{\mathbf{Z},\sigma}$ for $\sigma > \tilde{s}q^{3/2}\sqrt{|\log_2(\delta_{vs})|}$, then $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is δ_{vs}-vector-smooth over G on F.

Proof. Let $\mathbf{hk}$ denote a random variable following the distribution $\mathcal{D}_{\mathbf{Z}^{\ell},\sigma}$. Consider a random variable M taking values in $\Delta\mathcal{M}$ and the associated matrix $\mathbf{B}_M \in \mathbf{Z}^{\ell\times\ell}$. For $X \hookleftarrow G\backslash G^q$, there exist unique $\alpha \in \mathbf{Z}/s\mathbf{Z}$ and $\beta \in (\mathbf{Z}/q\mathbf{Z})^*$ s.t. $X = g_q^{\alpha} f^{\beta}$. As noted in Lemma 4.11, we need to evaluate the statistical distance between:

$$\mathcal{U} = \left\{M, g_q^{\alpha} f^{\beta}, \widehat{\mathsf{projkg}}(\mathbf{hk}), \{\langle \mathbf{hk}, \boldsymbol{b}_i^M\rangle\}_{i\in[\ell-1]}, f^{\gamma} \mid \gamma \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})\right\}$$

$$\text{and } \mathcal{V} = \left\{M, g_q^{\alpha} f^{\beta}, \widehat{\mathsf{projkg}}(\mathbf{hk}), \{\langle \mathbf{hk}, \boldsymbol{b}_i^M\rangle\}_{i\in[\ell-1]}, f^{\beta\langle \mathbf{hk}, \boldsymbol{b}_\ell^M\rangle}\right\}.$$

Consider $\boldsymbol{m} \in \Delta\mathcal{M}$, $\alpha_0 \in \mathbf{Z}/s\mathbf{Z}$, $\beta_0 \in \mathbf{Z}/q\mathbf{Z}$, $\widehat{\mathbf{hp}} \in (\mathbf{Z}/\varpi\mathbf{Z})^{\ell}$, and $\boldsymbol{v} \in \mathbf{Z}^{\ell-1}$. It suffices to study the distance between the random variables $Y := \beta\langle \mathbf{hk}, \boldsymbol{b}_\ell^M\rangle \bmod q$ and $\gamma \bmod q$ conditioned on the conjunction of the following events: $M = \boldsymbol{m}$; $\alpha \bmod s = \alpha_0 \bmod s$; $\beta \bmod q = \beta_0 \bmod q$; $\mathbf{hk} \bmod \varpi = \widehat{\mathbf{hp}} \bmod \varpi$ and for $j \in [\ell-1]$, $\langle \mathbf{hk}, \boldsymbol{b}_j^M\rangle = v_j$. Let us denote $\mathbf{B}_{\boldsymbol{m}}$ the matrix associated to $\boldsymbol{m}$ built as per Lemma 4.8, and its rows $\boldsymbol{b}_1^{\boldsymbol{m}}, \dots, \boldsymbol{b}_\ell^{\boldsymbol{m}}$. Since $M = \boldsymbol{m}$ it holds that $\mathbf{B}_M = \mathbf{B}_{\boldsymbol{m}}$. In the following, we evaluate the distribution followed by $\mathbf{hk}$ in $\mathbf{Z}$, conditioned on these events. To this end, let $\mathbf{hk}_0$ denote an arbitrary vector satisfying the same equations as $\mathbf{hk}$, *i.e.* for $j \in [\ell-1]$,

$$\langle \mathbf{hk}_0, \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = v_j \text{ in } \mathbf{Z} \quad \text{and} \quad \mathbf{hk}_0 \bmod \varpi = \widehat{\mathbf{hp}} \bmod \varpi.$$

We define $\Lambda := \{\boldsymbol{t} \in \mathbf{Z}^\ell \mid \langle \boldsymbol{t}, \boldsymbol{b}_i^{\boldsymbol{m}} \rangle = 0 \text{ for } i \in [\ell-1]; \boldsymbol{t} = \mathbf{0} \bmod \varpi\} \subset \mathbf{Z}^\ell$. Since hk is sampled from $\mathcal{D}_{\mathbf{Z}^\ell,\sigma}$, given the fixed information, hk is of the form $\mathsf{hk}_0 + T$ where T is a random variable with values in Λ. The variable T follows the same probability distribution as $\mathsf{hk} - \mathsf{hk}_0$ but taken over Λ, *i.e.* $\forall \boldsymbol{t} \in \Lambda$:

$$\Pr[T = \boldsymbol{t}] = \frac{\mathcal{D}_{Z^\ell,\sigma,-\mathsf{hk}_0}(\boldsymbol{t})}{\mathcal{D}_{Z^\ell,\sigma,-\mathsf{hk}_0}(\Lambda)} = \frac{\rho_{\sigma,-\mathsf{hk}_0}(\boldsymbol{t})}{\rho_{\sigma,-\mathsf{hk}_0}(\mathbf{Z}^\ell)} \cdot \frac{\rho_{\sigma,-\mathsf{hk}_0}(\mathbf{Z}^\ell)}{\rho_{\sigma,-\mathsf{hk}_0}(\Lambda)} = \mathcal{D}_{\Lambda,\sigma,-\mathsf{hk}_0}(\boldsymbol{t}).$$

So the conditional distribution followed by $\mathsf{hk} \in \mathbf{Z}^\ell$ is

$$\mathsf{hk}_0 + \mathcal{D}_{\Lambda,\sigma,-\mathsf{hk}_0}.$$

Now denoting $d \neq 0$ the gcd of the coefficients of $\boldsymbol{b}_\ell^{\boldsymbol{m}}$ and $\tilde{\boldsymbol{b}} = 1/d \cdot \boldsymbol{b}_\ell^{\boldsymbol{m}} \in \mathbf{Z}^\ell$, since $\boldsymbol{b}_\ell^{\boldsymbol{m}} = \boldsymbol{m}$ (*cf.* Definition 4.7), it holds that all vectors $\{\boldsymbol{b}_j^{\boldsymbol{m}}\}_{j \in [\ell-1]}$ belong to $\tilde{\boldsymbol{b}}^\perp$. Moreover, from the norm bonds on vectors in $\mathcal{M}$, it holds that $d \neq 0 \bmod q$. We consider the 1-dimensional lattice $\Lambda' := \{\boldsymbol{t} \in \mathbf{Z}^\ell | \langle \boldsymbol{t}, \boldsymbol{b}_i^{\boldsymbol{m}} \rangle = 0 \text{ for } i \in [\ell-1]\}$ which contains $\tilde{\boldsymbol{b}} \cdot \mathbf{Z}$. In fact as $\gcd(\tilde{b}_1, \ldots, \tilde{b}_\ell) = 1$, one has $\Lambda' = \tilde{\boldsymbol{b}} \cdot \mathbf{Z}$ ($\exists \boldsymbol{y} \in \mathbf{Z}^\ell$ s.t. $\Lambda' = \boldsymbol{y} \cdot \mathbf{Z}$, and $\tilde{\boldsymbol{b}} = \alpha \boldsymbol{y}$, so α must divide $\gcd(\tilde{b}_1, \ldots, \tilde{b}_\ell) = 1$). Moreover, for $\mu \in \mathbf{Z}$, in order for ϖ to divide each $\mu \tilde{b}_i$, ϖ must divide μ, so

$$\Lambda = \Lambda' \cap \varpi \cdot \mathbf{Z}^\ell = (\tilde{\boldsymbol{b}} \cdot \mathbf{Z} \cap \varpi \cdot \mathbf{Z}^\ell) = \varpi \cdot \tilde{\boldsymbol{b}} \cdot \mathbf{Z},$$

We now consider the distribution of $\langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle$, and then reduce it mod q, so as to prove that the random variable $\tilde{Y} := \langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle \bmod q$ follows a distribution close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Let us denote $\Lambda_0 := \varpi \cdot ||\tilde{\boldsymbol{b}}||_2^2 \cdot \mathbf{Z}$. It follows from Lemma 2.19 that the distribution of $\langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle$ is:

$$\langle \mathsf{hk}_0, \tilde{\boldsymbol{b}} \rangle + \mathcal{D}_{\Lambda_0, ||\tilde{\boldsymbol{b}}||_2 \cdot \sigma, -c} \quad \text{where } c := \langle \mathsf{hk}_0, \tilde{\boldsymbol{b}} \rangle \text{ in } \mathbf{Z}.$$

In order to prove that the above distribution, taken mod q, is statistically close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, we consider the distribution obtained by reducing the distribution $\mathcal{D}_{\Lambda_0, ||\tilde{\boldsymbol{b}}||_2 \cdot \sigma, -c}$ over Λ_0 modulo the sub-lattice $\Lambda_0' := q\Lambda_0$. Since ϖ and q are co-prime, it holds that $\Lambda_0/\Lambda_0' \simeq \mathbf{Z}/q\mathbf{Z}$, and so demonstrating that $\langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle \bmod q$ follows a distribution statistically close to $\mathcal{U}(\Lambda_0/\Lambda_0')$ will allow us to conclude. From Lemma 2.21 it follows that to achieve the required smoothing parameter $\eta_\epsilon(\Lambda_0')$ one must impose a lower bound on the standard deviation σ, i.e. $||\tilde{\boldsymbol{b}}||_2 \cdot \sigma > \eta_\epsilon(\Lambda_0')$. From [MR07] we know that, for $0 < \epsilon < 1/2$, and setting $\delta_{vs} := 2\epsilon$, it holds that:

$$\eta_\epsilon(\Lambda_0') \leqslant \sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}} \cdot \lambda_1(\Lambda_0') < \sqrt{\frac{|\log_2(\delta_{vs})|}{2}} \cdot \lambda_1(\Lambda_0')$$

Since $\lambda_1(\Lambda_0') = q \cdot \varpi \cdot ||\tilde{\boldsymbol{b}}||_2^2 < q \cdot \varpi \cdot ||\tilde{\boldsymbol{b}}||_2^2$, we require $\sigma > q \cdot \varpi \cdot ||\tilde{\boldsymbol{b}}||_2 \sqrt{2^{-1} |\log_2(\delta_{vs})|}$. Moreover, as $||\tilde{\boldsymbol{b}}||_2 < \sqrt{2q}$ (due to the norm bounds on vectors in $\mathcal{M}$, one has $||\tilde{\boldsymbol{b}}||_\infty < 2\sqrt{q/(2\ell)}$), choosing $\sigma > \tilde{s} \cdot q^{3/2} \sqrt{|\log_2(\delta_{vs})|}$ suffices to ensure that the distribution of $\langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle \bmod q$ is δ_{vs}-close to the uniform distribution over $\Lambda_0/\Lambda_0' \simeq \mathbf{Z}/q\mathbf{Z}$.

Finally $Y = \beta \cdot \langle \mathsf{hk}, \boldsymbol{b}_\ell^{\boldsymbol{m}} \rangle \bmod q = \beta \cdot d \cdot \langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle \bmod q$ where $\langle \mathsf{hk}, \tilde{\boldsymbol{b}} \rangle \bmod q$ is δ_{vs}-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, $\beta \neq 0 \bmod q$ and $d \neq 0 \bmod q$. This implies that Y also follows a distribution δ_{vs}-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Thus the statistical distance between the last coordinates of $\mathcal{U}$ and $\mathcal{V}$, given the first four, is at most δ_{vs}, which concludes the proof. □

Running Example 3 – $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is vector-smooth

Lemma 4.14 states sufficient conditions for $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ to be vector smooth. We note that this lemma (and its proof) reflect the main ideas of the proof of $\mathsf{ind\text{-}fe\text{-}cpa}$-security for our DDH-$f$ based IPFE, computing inner products in $\mathbf{Z}$, which we presented in [CLT18b, Thm. 5].

Lemma 4.14. If the hashkg algorithm of $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ samples hashing keys from the Gaussian distribution $\mathcal{D}_{\mathbf{Z},\sigma}$ for $\sigma > \tilde{s}q^{3/2}\sqrt{|\log_2(\delta_{vs})|}$, then $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is δ_{vs}-vector-smooth over G on F.

Proof. For $i \in [\ell]$, let $\mathsf{hk}_i := (\kappa_{0,i}, \kappa_{1,i})$ denote independent random variables following the distribution $\mathcal{D}_{\mathbf{Z}^2,\sigma}$; let $\boldsymbol{\kappa}_0 := (\kappa_{0,1}, \dots, \kappa_{0,\ell})^T$, $\kappa_1 := (\kappa_{1,1}, \dots, \kappa_{1,\ell})^T$, and $\textbf{hk} := (\mathsf{hk}_1, \dots, \mathsf{hk}_\ell)^T$. Consider a random variable M taking values in $\Delta\mathcal{M}$ and the associated matrix $\mathbf{B}_M \in \mathbf{Z}^{\ell\times\ell}$.

From the decomposability of $\mathsf{H}_{\mathsf{ddh\text{-}f}}$, for $X \hookleftarrow \mathcal{X}\backslash\mathcal{L}$ there exist unique $a \in \mathbf{Z}/s\mathbf{Z}$ and $\beta \in (\mathbf{Z}/q\mathbf{Z})^*$ satisfying $X = (g^a, h^a f^\beta)$. As noted in Lemma 4.11, we need to evaluate the statistical distance between:

$$\mathcal{U} = \Big\{M, X, ((\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1) \bmod \varpi, \boldsymbol{\kappa}_0 + \alpha\boldsymbol{\kappa}_1 \bmod q), \{\langle \boldsymbol{\kappa}_0, \boldsymbol{b}_j^M\rangle, \langle \boldsymbol{\kappa}_1, \boldsymbol{b}_j^M\rangle\}_{j\in[\ell-1]}, f^\gamma \mid \gamma \hookleftarrow \mathcal{U}(\mathbf{Z}/q\mathbf{Z})\Big\}$$

and $\mathcal{V} = \Big\{M, X, ((\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1) \bmod \varpi, \boldsymbol{\kappa}_0 + \alpha\boldsymbol{\kappa}_1 \bmod q), \{\langle \boldsymbol{\kappa}_0, \boldsymbol{b}_j^M\rangle, \langle \boldsymbol{\kappa}_1, \boldsymbol{b}_j^M\rangle\}_{j\in[\ell-1]}, f^{\beta\langle\boldsymbol{\kappa}_1, \boldsymbol{b}_\ell^M\rangle}\Big\}$.

Consider $\boldsymbol{m} \in \Delta\mathcal{M}$, $\bar{a} \in \mathbf{Z}/s\mathbf{Z}$, $\bar{\beta} \in \mathbf{Z}/q\mathbf{Z}$, $\bar{\boldsymbol{\kappa}_0}, \bar{\boldsymbol{\kappa}}_1 \in (\mathbf{Z}/\varpi\mathbf{Z})^\ell$, $\bar{\boldsymbol{\kappa}_2} \in (\mathbf{Z}/q\mathbf{Z})^\ell$, and $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathbf{Z}^\ell$. It suffices to study the distance between the random variables $Y := \beta\langle\boldsymbol{\kappa}_1, \boldsymbol{b}_\ell^M\rangle \bmod q$ and $\gamma \bmod q$ conditioned on the conjunction of the following events: $M = \boldsymbol{m}$; $a \bmod s = \bar{a} \bmod s$; $\beta \bmod q = \bar{\beta} \bmod q$; $(\bar{\boldsymbol{\kappa}_0}, \bar{\boldsymbol{\kappa}}_1) = (\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1) \bmod \varpi$; $\bar{\boldsymbol{\kappa}_2} = \boldsymbol{\kappa}_0 + \alpha\boldsymbol{\kappa}_1 \bmod q$ and for $j \in [\ell-1]$, $\langle\boldsymbol{\kappa}_0, \boldsymbol{b}_j^M\rangle = v_{0,j}$ and $\langle\boldsymbol{\kappa}_1, \boldsymbol{b}_j^M\rangle = v_{1,j}$. Let us denote $\mathbf{B}_{\boldsymbol{m}}$ the matrix associated to $\boldsymbol{m}$ built as per Lemma 4.8, and its rows $\boldsymbol{b}_1^{\boldsymbol{m}}, \dots, \boldsymbol{b}_\ell^{\boldsymbol{m}}$. Since $M = \boldsymbol{m}$ it holds that $\mathbf{B}_M = \mathbf{B}_{\boldsymbol{m}}$. In the following, we evaluate the distribution followed by $\textbf{hk}$ conditioned on these events. To this end, let $(\boldsymbol{\kappa}_0^*, \boldsymbol{\kappa}_1^*)$ denote an arbitrary pair of vectors satisfying the same equations as $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1)$, *i.e.* $\boldsymbol{\kappa}_0^* = \bar{\boldsymbol{\kappa}_0} \bmod \varpi$; $\boldsymbol{\kappa}_1^* = \bar{\boldsymbol{\kappa}_1} \bmod \varpi$; $\kappa_0^* + \alpha\kappa_1^* = \bar{\boldsymbol{\kappa}_2} \bmod q$; and for $j \in [\ell-1]$, $\langle\boldsymbol{\kappa}_0^*, \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = v_{0,j}, \langle\boldsymbol{\kappa}_1^*, \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = v_{1,j}$.

Consider the lattice $\Lambda := \{\boldsymbol{t} \in \mathbf{Z}^\ell | \langle\boldsymbol{t}, \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = 0 \text{ for } j \in [\ell-1]; \boldsymbol{t} = \mathbf{0} \bmod \varpi\} \subset \mathbf{Z}^\ell$. Since κ_0 and κ_1 are sampled from $\mathcal{D}_{\mathbf{Z}^\ell,\sigma}$, using similar arguments to those in proof of Lemma 4.13, one gets that the conditional joint distribution of $(\kappa_0, \kappa_1) \in \mathbf{Z}^\ell$ is

$$\{(\boldsymbol{\kappa}_0^* - \alpha\cdot\mu, \boldsymbol{\kappa}_1^* + \mu) \mid \mu \hookleftarrow \mathcal{D}_{\Lambda,\sigma,-\kappa_1^*}\}$$

Clearly the value of κ_1 fixes that of κ_0, so let us focus on the distribution of κ_1. Denoting $d \neq 0$ the gcd of the coefficients of $\boldsymbol{b}_\ell^{\boldsymbol{m}}$ and $\tilde{\boldsymbol{b}} = 1/d \cdot \boldsymbol{b}_\ell^{\boldsymbol{m}} \in \mathbf{Z}^\ell$, since $\boldsymbol{b}_\ell^{\boldsymbol{m}} = \boldsymbol{m}$ (*cf.* Definition 4.7), it holds that all vectors $\{\boldsymbol{b}_j^{\boldsymbol{m}}\}_{j\in[\ell-1]}$ belong to $\tilde{\boldsymbol{b}}^\perp$.

We consider the distribution of $\langle\kappa_1, \tilde{\boldsymbol{b}}\rangle$, and then reduce it mod q, so as to prove that the random variable $\langle\kappa_1, \tilde{\boldsymbol{b}}\rangle \bmod q$ follows a distribution close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Following the exact same reasoning as in proof of Lemma 4.13, one demonstrates that choosing $\sigma > \tilde{s}q^{3/2}\sqrt{|\log_2(\delta_{vs})|}$ ensures the distribution followed by the random variable $\langle\boldsymbol{\kappa}_1, \boldsymbol{b}_\ell^{\boldsymbol{m}}\rangle \bmod q$ is δ_{vs}-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Finally $Y = \beta\cdot d\cdot\langle\kappa_1, \tilde{\boldsymbol{b}}\rangle \bmod q$ where $\beta \neq 0 \bmod q$ and $d \neq 0 \bmod q$, consequently Y also follows a distribution δ_{vs}-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Thus $\mathcal{U}$ and $\mathcal{V}$ are δ_{vs}-close, which concludes the proof. □

4.2.4 Integrity

So as to guarantee security against active adversaries, who attempt to glean information by tampering with ciphertexts before requesting their decryption, we generalise the definition of a universal$_2$ PHF from [CS02]. To this end we define a new property on PHFs, called *vector universality*, which ensures ciphertext integrity in our upcoming constructions. The original definition of a universal$_2$ PHF of [CS02] allows to enforce ciphertext integrity for PKE schemes; in the context of IPFE, one also needs to deal with key derivation queries performed by the adversary. The definition of vector-universality, and proofs that our running examples possess it (Lemmas 4.16 to 4.18) are key to our achievements regarding IPFE schemes secure against active adversaries.

Intuition. As the definition is quite technical, we first give a little intuition. The high level idea is that we want the decryption algorithm to reject any ciphertext which, if decrypted, could leak harmful information. Just as in the context of PKE (*cf.* Section 3.4.2), these harmful ciphertexts, dubbed *invalid ciphertexts*, are exactly those whose first component lives in $\widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$. Hence, denoting this first ciphertext component x, one needs to ensure the decryption algorithm never decrypts a ciphertext with $x \notin \widehat{\mathcal{L}}$. To this end, upon encryption of a message vector of length ℓ, one computes ℓ evaluations of an extended projective hash function ehash over x, using independently sampled hashing keys $\mathsf{ehk}_1, \ldots, \mathsf{ehk}_\ell$. The resulting ciphertext contains x, the masked message components, and all the evaluations of ehash.

In our $\mathsf{ind\text{-}fe\text{-}cca}$-secure IPFE schemes a decryption key for $\boldsymbol{k} \in \mathcal{K}$ contains the linear combination $\overline{\mathsf{sk}_{\boldsymbol{k}}} := \sum_{i=1}^{\ell} k_i \mathsf{ehk}_i$. Using the key homomorphic property of the EPHF, a ciphertext will only be decrypted if $\mathsf{ehash}(\overline{\mathsf{sk}_{\boldsymbol{k}}}, x)$ yields the expected combination of the received ciphertext components.

Now if the ciphertext is invalid, *i.e.* if $x \notin \widehat{\mathcal{L}}$, it must be infeasible for an adversary to compute a ciphertext which will not be rejected, even given all the auxiliary information it gets from the scheme's public values and from its key derivation queries.

If the inequality of Definition 4.15 holds, one ensures that conditioned on the publicly available information (*i.e.* $\widehat{\mathsf{eprojkg}}(\mathbf{ehk}) = \widehat{\mathbf{ehp}}$); the adaptively chosen difference between challenge messages (*i.e.* $M = \boldsymbol{m}$); the evaluation of ehash given by the challenge ciphertext (*i.e.* $\mathsf{ehash}(\mathbf{ehk}, x^*, e^*) = \boldsymbol{\pi}^*$); and all the information available on $\mathbf{ehk}$ from key derivation queries (*i.e.* the evaluations of $\boldsymbol{b}^T \cdot \mathbf{ehk}$ for any $\boldsymbol{b}$ satisfying $\langle \boldsymbol{b}, \boldsymbol{m}_0 \rangle = \langle \boldsymbol{b}, \boldsymbol{m}_1 \rangle$), no adversary can predict an extended hash value π over an element $x \notin \widehat{\mathcal{L}}$ which would authorise decryption.

Definition 4.15 (δ_{vu}-vector-universal)**.** Let ℓ and a be positive integers. Let $\mathcal{R}$ be a ring, either $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be an SMP, and $\mathsf{eH} := (\mathsf{ehashkg}, \widehat{\mathsf{eprojkg}}, \mathsf{eprojkg}, \mathsf{ehash}, \widehat{\mathsf{eprojhash}}, \mathsf{eprojhash})$ be the associated EPHF, which we assume to be $(\mathcal{R}, 2a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-$\mathsf{ipfe}$-compatible. For $i \in [\ell]$, let $\mathsf{ehk}_i \leftarrow \mathsf{ehashkg}(\mathcal{SM})$, and denote $\mathbf{ehk} := (\mathsf{ehk}_1, \ldots, \mathsf{ehk}_\ell)^T$. Consider a random variable M taking values in $\Delta\mathcal{M}$ and the associated matrix $\mathbf{B}_M \in \mathcal{R}^{\ell \times \ell}$. We say eH is $\delta_{vu}(\ell)$-vector-universal if for any $\widehat{\mathbf{ehp}} \in (K_{\widehat{\mathsf{ehp}}})^\ell$; any $\boldsymbol{m} \in \Delta\mathcal{M}$; any $\boldsymbol{k} \in \mathcal{K}$ s.t. $\boldsymbol{k} \notin \boldsymbol{m}^\perp$; any $(x^*, e^*) \in \widehat{\mathcal{X}} \times E$, $(x, e) \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}} \times E$, s.t. $(x, e) \neq (x^*, e^*)$, and for any $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\ell-1}) \in (K_{\mathsf{ehk}})^{\ell-1}$; $\boldsymbol{\pi}^* \in \Pi^\ell$ and $\pi \in \Pi$ it holds that:

$$\Pr\big[\, \mathsf{ehash}(\boldsymbol{k}^T \cdot \mathbf{ehk}, x, e) = \pi \;\big|\; \mathsf{ehash}(\mathbf{ehk}, x^*, e^*) = \boldsymbol{\pi}^* \wedge \widehat{\mathsf{eprojkg}}(\mathbf{ehk}) = \widehat{\mathbf{ehp}} \\ \wedge ((\boldsymbol{b}_j^M)^T \cdot \mathbf{ehk} = \boldsymbol{v}_j \text{ for } j \in [\ell - 1]) \wedge M = \boldsymbol{m}\big] \leqslant \delta_{vu}(\ell).$$

Remark. Similarly to Lemma 4.11 let us assume H is further homomorphic, key homomorphic and $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable. Then using the same notations as in Definition 4.15, from the decomposability of eH we know there exist unique $z, z^* \in \widehat{\mathcal{L}}$ and $y, y^* \in \langle \Upsilon \rangle$, $y \neq 1$, such that $x = z \cdot y$ and $x^* = z^* \cdot y^*$. Since the output of $\widehat{\mathsf{eprojkg}}(\mathsf{ehk})$ fixes that of $\mathsf{ehash}(\mathsf{ehk}, z)$ and that of $\mathsf{ehash}(\mathsf{ehk}, z^*)$, and since eH is homomorphic, eH is $\delta_{vu}(\ell)$-vector-universal if for all $\widehat{\mathbf{ehp}} \in (K_{\widehat{\mathsf{ehp}}})^\ell$; any $\boldsymbol{k} \in \mathcal{K}$ s.t. $\boldsymbol{k} \notin \boldsymbol{m}^\perp$; and for any $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\ell-1}) \in (K_{\mathsf{ehk}})^{\ell-1}$; $\boldsymbol{\pi}_F^* \in F^\ell$ and $\pi_F \in F$ it holds that:

$$\Pr\big[\, \mathsf{ehash}(\boldsymbol{k}^T \cdot \mathbf{ehk}, z, e) = \pi \;\big|\; \mathsf{ehash}(\mathbf{ehk}, z^*, e^*) = \boldsymbol{\pi}_F^* \wedge \widehat{\mathsf{eprojkg}}(\mathbf{ehk}) = \widehat{\mathbf{ehp}} \\ \wedge ((\boldsymbol{b}_j^M)^T \cdot \mathbf{ehk} = \boldsymbol{v}_j \text{ for } j \in [\ell - 1]) \wedge M = \boldsymbol{m}\big] \leqslant \delta_{vu}(\ell).$$

We use this formulation to prove Lemmas 4.16 to 4.18.

Furthermore, for the proofs of these Lemmas, we denote:

- E_0 the event "$\mathsf{ehash}(\boldsymbol{k}^T \cdot \mathbf{ehk}, z, e) = \pi$";
- E_1 the event "$\mathsf{ehash}(\mathbf{ehk}, z^*, e^*) = \boldsymbol{\pi}^*_F$";
- E_2 the event "$\widehat{\mathsf{eprojkg}}(\mathbf{ehk}) = \widehat{\mathbf{ehp}}$";
- E_3 the event "$((\boldsymbol{b}^M_j)^T \cdot \mathbf{ehk} = \boldsymbol{v}_j$ for $j \in [\ell - 1]$";
- and E_4 the event "$M = \boldsymbol{m}$".

Running Example 1 – $\mathsf{eH}_{\mathsf{ddh}}$ is vector-universal

Lemma 4.16 states sufficient conditions for $\mathsf{H}_{\mathsf{ddh}}$ to be vector universal. Recall that for $\mathcal{SM}_{\mathsf{ddh}}$ it holds that $\mathcal{X} = \widehat{\mathcal{X}}$, $\widehat{\mathcal{L}} = \mathcal{L}$ and consequently $\widehat{\mathsf{eprojkg}} = \mathsf{eprojkg}$ and $\widehat{\mathsf{eprojhash}} = \mathsf{eprojhash}$.

Lemma 4.16. If $\Gamma : G^3 \mapsto \{0, \ldots, q-1\}$ is sampled from a δ_{cr}-hard CRHF generator, then $\mathsf{eH}_{\mathsf{ddh}}$ is δ_{vu}-vector-universal, where $\delta_{vu} = 1/q + \delta_{\mathsf{cr}}$.

Proof. For $i \in [\ell]$ let $\mathsf{ehk}_i := (\kappa_{0,i}, \kappa_{1,i}, \kappa_{2,i}, \kappa_{3,i})$ denote independent random variables following the distribution $\mathcal{U}((\mathbf{Z}/q\mathbf{Z})^4)$; let $\mathbf{ehk} := (\mathsf{ehk}_1, \ldots, \mathsf{ehk}_\ell)^T$ and $\boldsymbol{\kappa}_\mu := (\kappa_{\mu,1}, \ldots, \kappa_{\mu,\ell})^T \in (\mathbf{Z}/q\mathbf{Z})^\ell$ for $\mu \in \{0,1,2,3\}$. Consider a random variable M taking values in $\Delta\mathcal{M}$ and the associated matrix $\mathbf{B}_M \in \mathbf{Z}/q\mathbf{Z}^{\ell\times\ell}$. Consider a vector $\boldsymbol{m} \in (\mathbf{Z}/q\mathbf{Z})^\ell$ and matrix $\mathbf{B}_{\boldsymbol{m}} \in (\mathbf{Z}/q\mathbf{Z})^{\ell\times\ell}$ associated to $\boldsymbol{m}$ (Definition 4.7). Consider any $((x_0^*, x_1^*), e^*) \in G^2 \times G$, $((x_0, x_1), e) \in (G^2 \backslash \langle (g_0, g_1) \rangle)$ $\times G$, s.t. $((x_0, x_1), e) \neq ((x_0^*, x_1^*), e^*)$. By decomposability of $\mathsf{eH}_{\mathsf{ddh}}$ there exist unique (z_0, z_1), $(z_0^*,\ z_1^*) \in \widehat{\mathcal{L}}$ and $b, b^* \in \mathbf{Z}/q\mathbf{Z}$, $b \neq 0 \bmod q$, such that $(x_0, x_1) = (z_0, z_1) \odot (1, g_1^b)$ and $(x_0^*, x_1^*) = (z_0^*, z_1^*) \odot (1, g_1^{b^*})$. Let us denote $\gamma^* = \Gamma((x_0^*, x_1^*), e^*)$ and $\gamma = \Gamma((x_0, x_1), e)$. We must prove that for any $\mathbf{ehp} \in G^{2\ell}$; any $\boldsymbol{m} \in \Delta\mathcal{M}$; any $\boldsymbol{k} \in \mathcal{K}$ s.t. $\boldsymbol{k} \notin \boldsymbol{m}^\perp$; any $v_{\mu,j} \in \mathbf{Z}/q\mathbf{Z}$ for $\mu \in \{0,1,2,3\}$ and $j \in [\ell - 1]$; any $\boldsymbol{\pi}^* \in G^\ell$; and for any $\pi \in G$ it holds that:

$$\begin{aligned}\Pr[(g_1^b)^{\langle \boldsymbol{\kappa}_1 + \gamma\boldsymbol{\kappa}_3, \boldsymbol{k}\rangle} = \pi \mid (g_1^{b^*})^{\boldsymbol{\kappa}_1 + \gamma^*\boldsymbol{\kappa}_3} = \boldsymbol{\pi}^* \wedge (g_0^{\boldsymbol{\kappa}_0} g_1^{\boldsymbol{\kappa}_1}, g_0^{\boldsymbol{\kappa}_2} g_1^{\boldsymbol{\kappa}_3}) = \mathbf{ehp} \\ \wedge \langle \boldsymbol{\kappa}_\mu, \boldsymbol{b}_j^M \rangle = v_{\mu,j} \text{ for } j \in [\ell-1], \mu \in \{0,1,2,3\} \wedge M = \boldsymbol{m}],\end{aligned}$$

Observe that if $(x_0^*, x_1^*) \in \langle (g_0, g_1) \rangle$ then $b^* = 0 \bmod q$ and E_1 provides no information. We hereafter assume this is not the case. Thus $b \neq 0 \bmod q$ and $b^* \neq 0 \bmod q$ are both invertible mod q. We consider the information on $\mathbf{ehk}$ fixed by the conjunction of E_1, E_2, E_3 and E_4. As we shall see, given this information, the probability E_0 occurs is upper bounded by δ_{vu}.

1. For some $\boldsymbol{h} \in (\mathbf{Z}/q\mathbf{Z})^\ell$, event E_1 can equivalently be stated as: $\boldsymbol{h} = \boldsymbol{\kappa}_1 + \gamma^*\boldsymbol{\kappa}_3 \bmod q$.

2. For some $\boldsymbol{k}_0$ and $\boldsymbol{k}_1$ in $(\mathbf{Z}/q\mathbf{Z})^\ell$, E_2 can equivalently be stated as:

$$\begin{cases} \boldsymbol{k}_0 = \boldsymbol{\kappa}_0 + a\boldsymbol{\kappa}_1 \bmod q \\ \boldsymbol{k}_1 = \boldsymbol{\kappa}_2 + a\boldsymbol{\kappa}_3 \bmod q. \end{cases}$$

   Conditioned on E_1 and E_2, the joint distribution of $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3)$ is thus:

$$\{(\boldsymbol{k}_0 - a\boldsymbol{h} + a\gamma^*\boldsymbol{\mu}, \boldsymbol{h} - \gamma^*\boldsymbol{\mu}, \boldsymbol{k}_1 - a\boldsymbol{\mu}, \boldsymbol{\mu}) \mid \boldsymbol{\mu} \hookleftarrow (\mathbf{Z}/q\mathbf{Z})^\ell\}. \tag{4.1}$$

3. The conjunction of E_3 and E_4 sets: $\langle \boldsymbol{\kappa}_\mu, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = v_{\mu,j}$ for $j \in [\ell - 1], \mu \in \{0,1,2,3\}$.

We now evaluate the conditional distribution followed by $(\langle \boldsymbol{\kappa}_0, \boldsymbol{k}\rangle, \langle \boldsymbol{\kappa}_1, \boldsymbol{k}\rangle, \langle \boldsymbol{\kappa}_2, \boldsymbol{k}\rangle, \langle \boldsymbol{\kappa}_3, \boldsymbol{k}\rangle)$, conditioned on the conjunction of E_1, E_2, E_3 and E_4 Let $(\boldsymbol{\kappa}_0^*, \boldsymbol{\kappa}_1^*, \boldsymbol{\kappa}_2^*, \boldsymbol{\kappa}_3^*)$ denote an arbitrary quadruple of vectors satisfying the same equations as $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3)$, i.e. those fixed by E_1, E_2, E_3, E_4. Then

$$\begin{cases} \boldsymbol{\kappa}_0^* = \boldsymbol{k}_0 - a\boldsymbol{h} + a\gamma^*\boldsymbol{\kappa}_3^* \bmod q \\ \boldsymbol{\kappa}_1^* = \boldsymbol{h} - \gamma^*\boldsymbol{\kappa}_3^* \bmod q \\ \boldsymbol{\kappa}_2^* = \boldsymbol{k}_1 - a\boldsymbol{\kappa}_3^* \bmod q \\ \langle \boldsymbol{\kappa}_\mu^*, \boldsymbol{b}_j^{\boldsymbol{m}}\rangle = v_{\mu,j} \text{ for } j \in [\ell-1], \mu \in \{0,1,2,3\}. \end{cases}$$

Since for $j \in [\ell-1]$, $\boldsymbol{b}_j^{\boldsymbol{m}} \in \boldsymbol{m}^\perp$, the conditional joint distribution of $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3)$ is:

$$\{(\boldsymbol{\kappa}_0^* + \gamma^* \cdot a \cdot \mu \cdot \boldsymbol{m}, \boldsymbol{\kappa}_1^* - \gamma^* \cdot \mu \cdot \boldsymbol{m}, \boldsymbol{\kappa}_2^* - a \cdot \mu \cdot \boldsymbol{m}, \boldsymbol{\kappa}_3^* + \mu \cdot \boldsymbol{m}) \mid \mu \hookleftarrow \mathbf{Z}/q\mathbf{Z}\}.$$

The conditional distribution of $\langle \boldsymbol{\kappa}_3, \boldsymbol{k}\rangle$ is thus:

$$\{\langle \boldsymbol{\kappa}_3^*, \boldsymbol{k}\rangle + \mu\langle \boldsymbol{m}, \boldsymbol{k}\rangle \mid \mu \hookleftarrow \mathbf{Z}/q\mathbf{Z}\},$$

which is exactly $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$ since by definition $\boldsymbol{k} \notin \boldsymbol{m}^\perp$, so $\langle \boldsymbol{m}, \boldsymbol{k}\rangle \neq 0 \bmod q$.

Probability E_0 occurs. This is the probability $\mathfrak{p}$ that the random variable $X_1 := \langle \boldsymbol{\kappa}_1 + \gamma\boldsymbol{\kappa}_3, \boldsymbol{k}\rangle \bmod q$ takes a fixed value mod q. From Eq. (4.1) we can write:

$$X_1 = \langle \boldsymbol{h} + (\gamma - \gamma^*)\boldsymbol{\kappa}_3, \boldsymbol{k}\rangle.$$

Where $\langle \boldsymbol{h}, \boldsymbol{k}\rangle$ is fixed by E_1 and the value of $\boldsymbol{k}$. Thus if $\gamma \neq \gamma^* \bmod q$ then X_1 follows the uniform distribution modulo q and so the conditional probability E_0 occurs is $1/q$. Now since $\gamma^* = \Gamma((x_0^*, x_1^*), e^*)$ and $\gamma = \Gamma((x_0, x_1), e)$, the event $\gamma = \gamma^* \bmod q$ occurs with probability $\leqslant \delta_{\mathsf{cr}}$. We can conclude that $\mathfrak{p} \leqslant 1/q + \delta_{\mathsf{cr}}$, and denoting $\delta_{vu} := 1/q + \delta_{\mathsf{cr}}$, it holds that $\mathsf{H}_{\mathsf{ddh}}$ is δ_{vu}-vector-universal. □

Running Example 2 – $\mathsf{eH}_{\mathsf{hsm\text{-}cl}}$ is vector-universal

Recall that, for $\ell \in \mathbf{N}$, denoting $\mathcal{M} = \mathcal{K} = \{\boldsymbol{x} \in \mathbf{Z}^\ell \ : \ ||\boldsymbol{x}||_\infty < \sqrt{\frac{q}{2\ell}}\}$, $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is $(\mathbf{Z}, 1, f, q, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible. Lemma 4.17 states sufficient conditions for $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ to be vector universal.

Lemma 4.17. If the $\mathsf{ehashkg}$ algorithm of $\mathsf{eH}_{\mathsf{hsm\text{-}cl}}$ samples hashing keys from the Gaussian distribution $\widehat{\mathcal{D}} = \mathcal{D}_{\mathbf{Z},\sigma}$ for $\sigma > \sqrt{|\log_2(\delta)|}\tilde{s}q$, and $\Gamma : \widehat{G}^2 \mapsto \{0, \dots, q-1\}$ is sampled from a δ_{cr}-hard CRHF generator, then $\mathsf{eH}_{\mathsf{hsm\text{-}cl}}$ is δ_{vu}-vector-universal, where $\delta_{vu} := 1/q + \delta_{\mathsf{cr}} + \delta^{1/q}$.

Proof. For $i \in [\ell]$, $\beta \in \{0,1\}$, let $\mathsf{hk}_{\beta,i}$ denote independent random variables sampled from $\mathcal{D}_{\mathbf{Z},\sigma}$; let $\mathbf{hk}_\beta := (\mathsf{hk}_{\beta,1}, \dots, \mathsf{hk}_{\beta,\ell}) \in \mathbf{Z}^\ell$. Let M be a random variable taking values in $\Delta\mathcal{M}$ and $\mathbf{B}_M \in \mathbf{Z}^{\ell\times\ell}$ be the associated matrix.

Consider any $(x^*, e^*) \in \widehat{G}^2$, $(x, e) \in \widehat{G}\backslash\widehat{G}^q \times \widehat{G}$, s.t. $(x, e) \neq (x^*, e^*)$. By decomposability of $\mathsf{eH}_{\mathsf{hsm\text{-}cl}}$ there exist unique $z, z^* \in \widehat{G}^q$ and $b, b^* \in \mathbf{Z}/q\mathbf{Z}$, $b \neq 0 \bmod q$, such that $x = zf^b$ and $x^* = zf^{b^*}$. Let us denote $\gamma^* = \Gamma(x^*, e^*)$ and $\gamma = \Gamma(x, e)$. We must prove that for any $\widehat{\mathbf{ehp}} \in (\mathbf{Z}/\varpi\mathbf{Z})^{2\ell}$; any $\boldsymbol{m} \in \Delta\mathcal{M}$ (we denote $\boldsymbol{b}_1^{\boldsymbol{m}}, \dots, \boldsymbol{b}_\ell^{\boldsymbol{m}} \in \mathbf{Z}^\ell$ be the rows of the matrix $\mathbf{B}_{\boldsymbol{m}}$ associated to $\boldsymbol{m}$ built as per Lemma 4.8); any $\boldsymbol{k} \in \mathcal{K}$ s.t. $\boldsymbol{k} \notin \boldsymbol{m}^\perp$; any $v_{\beta,j} \in \mathbf{Z}$ for $\beta \in \{0,1\}$ and $j \in [\ell-1]$; any $\boldsymbol{\pi}^* \in F^\ell$; and any $\pi \in F$ it holds that:

$$\Pr[(f^b)^{\langle \mathbf{hk}_0 + \gamma\mathbf{hk}_1, \boldsymbol{k}\rangle} = \pi \mid (f^{b^*})^{\mathbf{hk}_0 + \gamma^*\mathbf{hk}_1} = \boldsymbol{\pi}^* \wedge \widehat{\mathsf{eprojkg}}(\mathbf{hk}_0, \mathbf{hk}_1) = \widehat{\mathbf{ehp}}$$

$$\wedge \, \langle \mathsf{hk}_\beta, \boldsymbol{b}_j^M \rangle = v_{\beta,j} \text{ for } j \in [\ell-1], \beta \in \{0,1\} \wedge M = \boldsymbol{m}] \leqslant \delta_{vu}.$$

Observe that if $x^* \in \widehat{G}^q$ then $b^* = 0 \bmod q$ and E_1 provides no information. We hereafter assume this is not the case. Thus $b \neq 0 \bmod q$ and $b^* \neq 0 \bmod q$ are both invertible mod q. We consider the information on ehk fixed by the conjunction of E_1, E_2, E_3 and E_4.

1. For some $\boldsymbol{h} \in (\mathbf{Z}/q\mathbf{Z})^\ell$, event E_1 can equivalently be stated as: $\boldsymbol{h} = \mathsf{hk}_0 + \gamma^* \mathsf{hk}_1 \bmod q$. So conditioned on E_1, the joint distribution of $(\mathsf{hk}_0 \bmod q, \mathsf{hk}_1 \bmod q)$ is:

$$\{(\boldsymbol{h} - \gamma^* \mathsf{hk}_1 \bmod q, \mathsf{hk}_1 \bmod q) \mid \mathsf{hk}_1 \hookleftarrow \mathcal{D}_{\mathbf{Z}^\ell,\sigma}\}.$$

2. For $\widehat{\mathsf{ehp}}_0$ and $\widehat{\mathsf{ehp}}_1 \in (\mathbf{Z}/\varpi\mathbf{Z})^\ell$ satisfying $\widehat{\mathsf{ehp}} = (\widehat{\mathsf{ehp}}_0, \widehat{\mathsf{ehp}}_1)$, event E_2 can equivalently be stated as: $\widehat{\mathsf{ehp}}_0 \bmod \varpi = \mathsf{hk}_0 \bmod \varpi$ and $\widehat{\mathsf{ehp}}_1 \bmod \varpi = \mathsf{hk}_1 \bmod \varpi$.
3. Event E_4 tells us that $\mathbf{B}_M = \mathbf{B}_{\boldsymbol{m}}$. Note that $\boldsymbol{b}_\ell^{\boldsymbol{m}} = \boldsymbol{m}$ since $\mathcal{R} = \mathbf{Z}$. The conjunction of events E_3 and E_4 can equivalently be stated as: $\langle \mathsf{hk}_\beta, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = v_{\beta,j}$ for $j \in [\ell-1], \beta \in \{0,1\}$.

We first evaluate the distribution followed by hk_1 conditioned on these events. To this end, let $\mathsf{hk}_1^* \in \mathbf{Z}^\ell$ denote an arbitrary vector satisfying the same equations as hk_1, i.e. for $j \in [\ell-1]$,

$$\langle \mathsf{hk}_1^*, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = \langle \mathsf{hk}_1, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = v_{\beta,j} \quad \text{and} \quad \mathsf{hk}_1^* \bmod \varpi = \widehat{\mathsf{ehp}}_1 \bmod \varpi.$$

We define $\Lambda := \{\boldsymbol{t} \in \mathbf{Z}^\ell \mid \langle \boldsymbol{t}, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = 0 \text{ for } j \in [\ell-1]; \boldsymbol{t} = \mathbf{0} \bmod \varpi\} \subset \mathbf{Z}^\ell$, such that, as in proof of Lemma 4.13, given the information fixed by E_2 and E_3, the distribution followed by $\mathsf{hk}_1 \in \mathbf{Z}^\ell$ is:

$$\mathsf{hk}_1^* + \mathcal{D}_{\Lambda,\sigma,-\mathsf{hk}_1^*}.$$

Denoting $d \neq 0$ the gcd of the coefficients of $\boldsymbol{b}_\ell^{\boldsymbol{m}}$ and $\tilde{\boldsymbol{b}} = 1/d \cdot \boldsymbol{b}_\ell^{\boldsymbol{m}} \in \mathbf{Z}^\ell$, since $\boldsymbol{b}_\ell^{\boldsymbol{m}} = \boldsymbol{m}$ (*cf.* Definition 4.7), it holds that all vectors $\{\boldsymbol{b}_i^{\boldsymbol{m}}\}_{i \in [\ell-1]}$ belong to $\tilde{\boldsymbol{b}}^\perp$. Now consider the 1-dimensional lattice $\Lambda' := \{\boldsymbol{t} \in \mathbf{Z}^\ell | \langle \boldsymbol{t}, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = 0 \text{ for } j \in [\ell-1]\}$ which contains $\tilde{\boldsymbol{b}}\mathbf{Z}$. In fact as $\gcd(\tilde{b}_1, \ldots, \tilde{b}_\ell) = 1$, one has $\Lambda' = \tilde{\boldsymbol{b}} \cdot \mathbf{Z}$, and also $\Lambda = \Lambda' \cap \varpi \cdot \mathbf{Z}^\ell = (\tilde{\boldsymbol{b}} \cdot \mathbf{Z} \cap \varpi \cdot \mathbf{Z}^\ell) = \varpi \cdot \tilde{\boldsymbol{b}} \cdot \mathbf{Z}$ (*cf.* detailed explanation in proof of Lemma 4.13). We now consider the distribution followed by $\langle \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle$, and then reduce it mod q, so as to prove that the random variable $\tilde{Y} := \langle \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle \bmod q$ follows a distribution close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Let us denote $\Lambda_0 := \varpi \cdot ||\tilde{\boldsymbol{b}}||_2^2 \cdot \mathbf{Z}$. It follows from Lemma 2.19 that the distribution followed by $\langle \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle$ is:

$$\langle \mathsf{hk}_1^*, \tilde{\boldsymbol{b}} \rangle + \mathcal{D}_{\Lambda_0, ||\tilde{\boldsymbol{b}}||_2 \cdot \sigma, -c} \ \text{ where } c = \langle \mathsf{hk}_1^*, \tilde{\boldsymbol{b}} \rangle \text{ in } \mathbf{Z}.$$

As in proof of Lemma 4.13, we reduce the distribution $\mathcal{D}_{\Lambda_0, ||\tilde{\boldsymbol{b}}||_2 \cdot \sigma, -c}$ over Λ_0 modulo the sublattice $\Lambda'_0 := q\Lambda_0$. Since $\sigma > \sqrt{|\log_2(\delta)|} \cdot \tilde{s} \cdot q = \sqrt{|\log_2(\delta^{\frac{1}{q}})|} \cdot \tilde{s} \cdot q^{3/2}$ it holds that $\langle \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle \bmod q$ is $\delta^{\frac{1}{q}}$-close to the uniform distribution over $\Lambda_0/\Lambda'_0 \simeq \mathbf{Z}/q\mathbf{Z}$. Using the information fixed by E_1, the distribution of $\langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle \bmod q$ is thus

$$\{\langle \boldsymbol{h}, \tilde{\boldsymbol{b}} \rangle + (\gamma - \gamma^*)(\langle \mathsf{hk}_1^*, \tilde{\boldsymbol{b}} \rangle + \nu) \bmod q | \ \nu \hookleftarrow (\mathcal{D}_{\Lambda_0, ||\tilde{\boldsymbol{b}}||_2 \sigma, -c} \bmod \Lambda'_0)\} \tag{4.2}$$

Since $(\mathcal{D}_{\Lambda_0, ||\tilde{\boldsymbol{b}}||_2 \sigma, -c} \bmod \Lambda'_0)$ is $\delta^{\frac{1}{q}}$-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, and $\boldsymbol{h}$ is fixed by E_1, if $\gamma \neq \gamma^* \bmod q$ then $\langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle \bmod q$ follows a distribution $\delta^{\frac{1}{q}}$-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$.

Probability E_0 occurs. This is the probability that the random variable $X_1 := \langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \boldsymbol{k} \rangle \bmod q$ takes a fixed value. Since the matrix $\mathbf{B}_{\boldsymbol{m}}$ is invertible mod q, $\boldsymbol{k} \bmod q$ can be uniquely expressed as a linear combination of the rows of $\mathbf{B}_{\boldsymbol{m}}$. We denote this decomposition

$$\boldsymbol{k} = \sum_{i \in [\ell]} \alpha_i \boldsymbol{b}_i^{\boldsymbol{m}} \bmod q \text{ with } \alpha_\ell \in (\mathbf{Z}/q\mathbf{Z})^* \text{ and } \alpha_i \in \mathbf{Z}/q\mathbf{Z} \text{ for } i \in [\ell - 1].$$

From the knowledge of $\boldsymbol{k}$ and E_3, for $i \in [\ell - 1]$ the values $\langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \alpha_i \boldsymbol{b}_i^{\boldsymbol{m}} \rangle$ are fixed. And so we need only consider the probability that $\alpha_\ell \langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \boldsymbol{b}_\ell^{\boldsymbol{m}} \rangle = d \cdot \alpha_\ell \langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle$ takes a fixed value mod q. But from Eq. (4.2) we know that, if $\gamma \neq \gamma^* \bmod q$ then $\langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle \bmod q$ follows a distribution $\delta^{\frac{1}{q}}$-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Note that, since $\gamma^* = \Gamma(x^*, e^*)$ and $\gamma = \Gamma(x, e)$, the event $\gamma = \gamma^*$ occurs with probability $\leqslant \delta_{\mathsf{cr}}$. It follows that the probability $\langle \mathsf{hk}_0 + \gamma \mathsf{hk}_1, \tilde{\boldsymbol{b}} \rangle$ takes a given value mod q is $\leqslant 1/q + \delta^{\frac{1}{q}} + \delta_{\mathsf{cr}}$, which concludes the proof. □

Running Example 3 – $\mathsf{eH}_{\mathsf{ddh\text{-}f}}$ is vector-universal

Recall that for $\ell \in \mathbf{N}$, denoting $\mathcal{M} = \mathcal{K} = \{\boldsymbol{x} \in \mathbf{Z}^\ell \ : \ ||\boldsymbol{x}||_\infty < \sqrt{\frac{q}{2\ell}}\}$, $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is $(\mathbf{Z}, 2, f, q, \ell, \mathcal{M}, \mathcal{K})$-$\mathsf{ipfe}$-compatible. Lemma 4.18 states sufficient conditions for $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ to be vector universal.

Lemma 4.18. If the $\mathsf{ehashkg}$ algorithm of $\mathsf{eH}_{\mathsf{ddh\text{-}f}}$ samples hashing keys from the Gaussian distribution $\widehat{\mathcal{D}} = \mathcal{D}_{\mathbf{Z},\sigma}$ for $\sigma > \sqrt{|\log_2(\delta)|}\tilde{s}q$, and $\Gamma : \widehat{G}^2 \mapsto \{0, \ldots, q-1\}$ is sampled from a δ_{cr}-hard CRHF generator, then $\mathsf{eH}_{\mathsf{ddh\text{-}f}}$ is δ_{vu}-vector-universal, where $\delta_{vu} := 1/q + \delta_{\mathsf{cr}} + \delta^{1/q}$.

Proof. For $i \in [\ell]$ let $\mathsf{ehk}_i := (\kappa_{0,i}, \kappa_{1,i}, \kappa_{2,i}, \kappa_{3,i})$ denote independent random variables following the distribution $\mathcal{D}_{\mathbf{Z}^4,\sigma}$; let $\mathbf{\mathsf{ehk}} := (\mathsf{ehk}_1, \ldots, \mathsf{ehk}_\ell)^T$ and $\boldsymbol{\kappa}_\mu := (\kappa_{\mu,1}, \ldots, \kappa_{\mu,\ell})^T \in \mathbf{Z}^\ell$ for $\mu \in \{0, 1, 2, 3\}$. Let M be a random variable taking values in $\Delta\mathcal{M}$ and $\mathbf{B}_M \in \mathbf{Z}^{\ell \times \ell}$ be the associated matrix.

Recall that we denote $\alpha := \log_g(h)$, and that $\widehat{\mathcal{L}} = \{(u_0 f^r, u_1 f^{\alpha r} | u_0, u_1 \in \widehat{G}^q; r \in \mathbf{Z}/q\mathbf{Z}\}$. Consider any $(x_0^*, x_1^*) \in \widehat{G}^2$, $(x_0, x_1) \in \widehat{G}^2 \backslash \widehat{\mathcal{L}}$ and $e, e^* \in \widehat{G}$, satisfying $((x_0, x_1), e) \neq ((x_0^*, x_1^*), e^*)$; from the decomposability of $\mathsf{eH}_{\mathsf{ddh\text{-}f}}$ we know there exist unique $(z_0, z_1), (z_0^*, z_1^*) \in \widehat{\mathcal{L}}$ and $b, b^* \in \mathbf{Z}/q\mathbf{Z}$, $b \neq 0 \bmod q$, such that $(x_0, x_1) = (z_0, z_1) \odot (1, f^b)$ and $(x_0^*, x_1^*) = (z_0^*, z_1^*) \odot (1, f^{b^*})$. Let us denote $\gamma^* = \Gamma((x_0^*, x_1^*), e^*)$ and $\gamma = \Gamma((x_0, x_1), e)$. We must prove that for any $((\bar{\boldsymbol{\kappa}}_0, \bar{\boldsymbol{\kappa}}_1, \bar{\boldsymbol{\kappa}}_2, \bar{\boldsymbol{\kappa}}_3), (\boldsymbol{k}_0, \boldsymbol{k}_1)) \in (\mathbf{Z}/\varpi\mathbf{Z})^4 \times (\mathbf{Z}/q\mathbf{Z})^2$; any $\boldsymbol{m} \in \Delta\mathcal{M}$ (we denote $\boldsymbol{b}_1^{\boldsymbol{m}}, \ldots, \boldsymbol{b}_\ell^{\boldsymbol{m}} \in \mathbf{Z}^\ell$ be the rows of the matrix $\mathbf{B}_{\boldsymbol{m}}$ associated to $\boldsymbol{m}$ built as per Lemma 4.8); any $\boldsymbol{k} \in \mathcal{K}$ s.t. $\boldsymbol{k} \notin \boldsymbol{m}^\perp$; any $v_{\mu,j} \in \mathbf{Z}/q\mathbf{Z}$ for $\mu \in \{0, 1, 2, 3\}$ and $j \in [\ell - 1]$; any $\boldsymbol{\pi}_F^* \in F^\ell$; and any $\pi_F \in F$ it holds that:

$$\begin{aligned} \Pr[f^{b\langle \boldsymbol{\kappa}_1 + \gamma \boldsymbol{\kappa}_3, \boldsymbol{k} \rangle} = \pi_F \mid f^{b^*(\boldsymbol{\kappa}_1 + \gamma^* \boldsymbol{\kappa}_3)} = \boldsymbol{\pi}_F^* & \\ \wedge ((\bar{\boldsymbol{\kappa}}_0, \bar{\boldsymbol{\kappa}}_1, \bar{\boldsymbol{\kappa}}_2, \bar{\boldsymbol{\kappa}}_3), (\boldsymbol{k}_0, \boldsymbol{k}_1)) = ((\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3) \bmod \varpi, (\boldsymbol{\kappa}_0 + \alpha \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2 + \alpha \boldsymbol{\kappa}_3) \bmod q) & \\ \wedge \langle \boldsymbol{\kappa}_\mu, \boldsymbol{b}_j \rangle = v_{\mu,j} \text{ for } j \in [\ell - 1], \mu \in \{0, 1, 2, 3\} \wedge M = \boldsymbol{m}] \leqslant \delta_{vu}. & \quad (4.3) \end{aligned}$$

Observe that if $(x_0^*, x_1^*) \in \widehat{\mathcal{L}}$, then $b^* = 0 \bmod q$, and E_1 provides no information. We assume this is not the case, and so b and b^* are invertible mod q. We now demonstrate that given the distribution followed by $\mathbf{\mathsf{ehk}}$ conditioned on the conjunction of E_1, E_2, E_3 and E_4, the probability E_0 occurs is bound by δ_{vu}. Let us first reformulate the information provided by these events for clarity.

1. For some $\boldsymbol{h} \in (\mathbf{Z}/q\mathbf{Z})^\ell$, event E_1 can equivalently be stated as: $\boldsymbol{h} = b^*(\boldsymbol{\kappa}_1 + \gamma^* \boldsymbol{\kappa}_3) \bmod q$.

2. The event E_2 fixes $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3) \bmod \varpi$, we hereafter focus on their distribution mod q. Conditioning on E_2 also tells us that: $\boldsymbol{k}_0 = \boldsymbol{\kappa}_0 + \alpha \boldsymbol{\kappa}_1 \bmod q$ and $\boldsymbol{k}_1 = \boldsymbol{\kappa}_2 + \alpha \boldsymbol{\kappa}_3 \bmod q$.

3. Event E_4 tells us that $\mathbf{B}_M = \mathbf{B}_{\boldsymbol{m}}$. Note that $\boldsymbol{b}_\ell^{\boldsymbol{m}} = \boldsymbol{m}$ since $\mathscr{R} = \mathbf{Z}$. The conjunction of events E_3 and E_4 can equivalently be stated as:

$$\langle \boldsymbol{\kappa}_\mu, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = v_{\mu,j} \text{ for } j \in [\ell - 1], \mu \in \{0, 1, 2, 3\}.$$

Let $(\boldsymbol{\kappa}_0^*, \boldsymbol{\kappa}_1^*, \boldsymbol{\kappa}_2^*, \boldsymbol{\kappa}_3^*) \in (\mathbf{Z}^\ell)^4$ denote an arbitrary tuple of vectors satisfying the same equations as $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3)$. Then, denoting $\Lambda := \{\boldsymbol{t} \in \mathbf{Z}^\ell | \langle \boldsymbol{t}, \boldsymbol{b}_j^{\boldsymbol{m}} \rangle = 0 \text{ for } j \in [\ell - 1]; \boldsymbol{t} = \boldsymbol{0} \bmod \varpi\} \subset \mathbf{Z}^\ell$, the conditional joint distribution of $(\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3) \bmod q$ is

$$\{\boldsymbol{\kappa}_0^* + \gamma^* \cdot \alpha \cdot \mu, \boldsymbol{\kappa}_1^* - \gamma^* \mu, \boldsymbol{\kappa}_2^* - \alpha\mu, \boldsymbol{\kappa}_3^* + \mu \mid \mu \hookleftarrow \mathcal{D}_{\Lambda, \sigma, -\boldsymbol{\kappa}_1^*}\}.$$

We now focus on the distribution of $\boldsymbol{\kappa}_3 \bmod q$, as it fixes $\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2 \bmod q$. Denoting $d \neq 0$ the gcd of the coefficients of $\boldsymbol{b}_\ell^{\boldsymbol{m}}$ and $\tilde{\boldsymbol{b}} = 1/d \cdot \boldsymbol{b}_\ell^{\boldsymbol{m}} \in \mathbf{Z}^\ell$, since $\boldsymbol{b}_\ell^{\boldsymbol{m}} = \boldsymbol{m}$ (*cf.* Definition 4.7), it holds that all vectors $\{\boldsymbol{b}_j^{\boldsymbol{m}}\}_{j \in [\ell - 1]}$ belong to $\tilde{\boldsymbol{b}}^\perp$. Using similar techniques to proofs of Lemmas 4.14 and 4.17, one gets that, since $\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3$ were sampled from $\mathcal{D}_{\mathbf{Z}^\ell, \sigma}$, for $\sigma > \sqrt{|\log_2(\delta)|}\tilde{s}q$, the conditional distribution followed by $\langle \boldsymbol{\kappa}_3, \tilde{\boldsymbol{b}} \rangle \bmod q$ is $\delta^{\frac{1}{q}}$-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$.

Probability that E_0 **occurs.** This is the probability $\mathfrak{p}$ that the random variable $X_1 := \langle \boldsymbol{\kappa}_1 + \gamma \boldsymbol{\kappa}_3, \boldsymbol{k} \rangle \bmod q$ takes a fixed value. As in proof of Lemma 4.17, $\boldsymbol{k} \bmod q$ can be uniquely expressed as

$$\boldsymbol{k} = \sum_{i \in [\ell]} \alpha_i \boldsymbol{b}_i^{\boldsymbol{m}} \bmod q \text{ with } \alpha_\ell \in (\mathbf{Z}/q\mathbf{Z})^* \text{ and } \alpha_i \in \mathbf{Z}/q\mathbf{Z} \text{ for } i \in [\ell - 1],$$

where the values $\langle \boldsymbol{\kappa}_1 + \gamma \boldsymbol{\kappa}_3, \alpha_i \boldsymbol{b}_i^{\boldsymbol{m}} \rangle$ are fixed, so we need only consider the conditional probability that $\alpha_\ell \langle \boldsymbol{\kappa}_1 + \gamma \boldsymbol{\kappa}_3, \boldsymbol{b}_\ell^{\boldsymbol{m}} \rangle = d \cdot \alpha_\ell \langle \boldsymbol{\kappa}_1 + \gamma \boldsymbol{\kappa}_3, \tilde{\boldsymbol{b}} \rangle$ takes a fixed value mod q. We denote $X_1' := d \cdot \alpha_\ell \langle \boldsymbol{\kappa}_1 + \gamma \boldsymbol{\kappa}_3, \tilde{\boldsymbol{b}} \rangle \bmod q$ this random variable. Since $b^* \neq 0 \bmod q$, given E_1 we can write:

$$X_1' = d \cdot \alpha_\ell \langle (b^*)^{-1} \boldsymbol{h} + \boldsymbol{\kappa}_3 (\gamma - \gamma^*), \tilde{\boldsymbol{b}} \rangle \bmod q$$

where $d \cdot \alpha_\ell \langle (b^*)^{-1} \boldsymbol{h}, \tilde{\boldsymbol{b}} \rangle \bmod q$ is fixed by E_1 and the value of $\boldsymbol{k}$. Moreover since $d \neq 0 \bmod q$ and $\alpha_\ell \neq 0 \bmod q$, if $\gamma \neq \gamma^* \bmod q$ then X_1' follows a distribution $\delta^{\frac{1}{q}}$-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$ and so the conditional probability E_0 occurs is $\leqslant \delta^{\frac{1}{q}} + 1/q$. Now since $\gamma^* = \Gamma((x_0^*, x_1^*), e^*)$ and $\gamma = \Gamma((x_0, x_1), e)$, the event $\gamma = \gamma^* \bmod q$ occurs with probability $\leqslant \delta_{\mathsf{cr}}$. We can conclude that $\mathfrak{p} \leqslant 1/q + \delta^{\frac{1}{q}} + \delta_{\mathsf{cr}}$, and denoting $\delta_{vu} := 1/q + \delta^{\frac{1}{q}} + \delta_{\mathsf{cr}}$, it holds that $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ is δ_{vu}-vector-universal. □

4.2.5 Inner Product Safe PHFs

We define the notions of active and passive inner product safe projective hash functions (aip-safe and pip-safe), which summarise the properties required to build ind-fe-cca and ind-fe-cpa secure IPFE schemes.

Definition 4.19 (pip-safe)**.** Let ℓ and a be positive integers. Let $\mathscr{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathsf{Gen}_{\mathcal{SM}}$ be an SMP generator outputting an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$; and let H be the associated PHF, which we assume to be $(\mathscr{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible. Then H is $(\mathscr{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs})$-passive inner product safe (pip-safe) if, denoting $F := \langle f \rangle$, it holds that:

- the order n_f of F is either prime or hard to factor;
- H is $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable, $\widehat{\Upsilon} \in \widehat{\mathcal{X}}$, $\Upsilon \in \mathcal{X}$;

- H is homomorphic;
- $\mathcal{SM}$ is $\delta_{\mathcal{L}}$-hard;
- and H is δ_{vs}-vector-smooth over $\mathcal{X}$ on F.

Definition 4.20 (aip-safe)**.** Let ℓ and a be positive integers. Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathsf{Gen}_{\mathcal{SM}}$ be a subgroup membership problem generator outputting an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$; and let H be the associated PHF, which we assume to be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible. Let eH be the EPHF obtained from H via the generic construction detailed in Section 3.4.3. The pair $(\mathsf{H}, \mathsf{eH})$ is said to be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs}, \delta_{vu})$-active inner product safe (aip-safe) if H is $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs})$-pip-safe and eH is δ_{vu}-vector-universal.

4.3 IPFE Secure against Passive Adversaries from PHFs

4.3.1 Generic Construction

We here provide an ind-fe-cpa-secure IPFE construction, and tight security reduction. Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathsf{Gen}_{\mathcal{SM}}$ be a subgroup membership problem generator outputting an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$; ℓ and a be positive integers; $\mathcal{M} \subseteq \mathcal{R}^\ell$ be the plaintext space; and $\mathcal{K} \subseteq \mathcal{R}^\ell$ be the space from which keys are derived. The scheme recovers $\langle \boldsymbol{m}, \boldsymbol{k} \rangle \in \mathcal{R}$ for $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{k} \in \mathcal{K}$. For security, the PHF associated to $\mathcal{SM}$, denoted H, must be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs})$-pip-safe. The resulting scheme is depicted in Fig. 4.1.

As we shall see in the upcomming running examples, when instantiated from DDH, the IPFE of Fig. 4.1 yields that of [ALS16] or equivalently, the ElGamal based scheme of [ABDP16]; when instantiated from HSM-CL or DDH-f it yields our FE schemes computing inner products in $\mathbf{Z}$ published in [CLT18a]. Moreover, we note that though the construction appears very similar to the ind-fe-cpa-secure construction of [BBL17], the requirements on the PHFs are different, and our security proof significantly lowers the bound on the adversary's advantage. A detailed comparison is given in Appendix A.

Correctness. As $K_{\mathsf{hk}} = \mathcal{R}^a$ one has $\mathbf{hk} \in (\mathcal{R}^\ell)^a$, so $\boldsymbol{k}^T \cdot \mathbf{hk} \in \mathcal{R}^a$. Next, by key homomorphism of H:

$$\prod_{i \in [\ell]} c_i^{k_i} = \prod_{i \in [\ell]} (\mathsf{hash}(\mathsf{hk}_i, c_0) f^{m_i})^{k_i} = f^{\langle \boldsymbol{k}, \boldsymbol{m} \rangle} \mathsf{hash}(\sum_{i \in [\ell]} k_i \mathsf{hk}_i, c_0) = f^{\langle \boldsymbol{k}, \boldsymbol{m} \rangle} \mathsf{hash}(\mathsf{sk}_{\boldsymbol{k}}, c_0),$$

thus $\prod_{i \in [\ell]} c_i^{k_i} \cdot \mathsf{hash}(\mathsf{sk}_{\boldsymbol{k}}, c_0)^{-1} = f^{\langle \boldsymbol{k}, \boldsymbol{m} \rangle} \in F$. Since H is $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible, for any $\boldsymbol{k} \in \mathcal{K}$, and $\boldsymbol{m} \in \mathcal{M}$, $\log_f(f^{\langle \boldsymbol{k}, \boldsymbol{m} \rangle}) = \langle \boldsymbol{k}, \boldsymbol{m} \rangle \in \mathcal{R}$. Consequently, for any $\mathsf{mpk}, \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\boldsymbol{k} \in \mathcal{K}$, and $\boldsymbol{m} \in \mathcal{M}$ it holds that $\mathsf{Dec}(\mathsf{mpk}, \mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k}), \mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m}))$ outputs $\langle \boldsymbol{k}, \boldsymbol{m} \rangle \in \mathcal{R}$.

Security. In Theorem 4.21 we demonstrate the ind-fe-cpa-security of the IPFE of Fig. 4.1. The structure of the proof resembles those of [ALS16] which are specific to precise assumptions. We do not include our proofs of [CLT18a] in this thesis since we are able to retrieve the same schemes and security bounds via the generic approach adopted here. Using the vector-smoothness property of the PHF, we are able to upper bound the amount of information adversaries can gain in an ind-fe-cpa experiment, just as, in proof of Theorem 3.20, smoothness allowed us to upper bound the information an adversary attacking a PKE scheme can gain in the ind-cpa experiment.

```
Setup$(1^\lambda, 1^\ell)$:
  $\mathcal{SM} \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$
  For $1 \leqslant i \leqslant \ell$ :
      Sample $\mathsf{hk}_i \leftarrow \mathsf{hashkg}(\mathcal{SM})$
      $\mathsf{hp}_i \leftarrow \mathsf{projkg}(\mathsf{hk}_i)$
  Return $\mathsf{mpk} := \mathbf{hp}$; $\mathsf{msk} := \mathbf{hk}$

Enc$(\mathsf{mpk}, \boldsymbol{m})$:
  If $\boldsymbol{m} \notin \mathcal{M}$ return $\perp$
  Sample $(c_0, w) \leftarrow \mathsf{R}$
  For $1 \leqslant i \leqslant \ell$ :
      $c_i \leftarrow \mathsf{projhash}(\mathsf{hp}_i, c_0, w) \cdot f^{m_i}$
  Return $\boldsymbol{ct} := (c_0, \boldsymbol{c})$

KeyDer$(\mathsf{msk}, \boldsymbol{k})$:
  If $\boldsymbol{k} \notin \mathcal{K}$ return $\perp$
  $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \boldsymbol{k}^T \cdot \mathbf{hk}$
  Return $(\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k})$

Dec$(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k}), \boldsymbol{ct})$:
  If $\boldsymbol{ct} \notin \widehat{\mathcal{X}} \times \Pi^\ell$ then return $\perp$
  $M \leftarrow (\prod_{i \in [\ell]} c_i^{k_i}) \cdot \mathsf{hash}(\mathsf{sk}_{\boldsymbol{k}}, c_0)^{-1}$
  If $M \notin F$ then return $\perp$
  Return $\mathsf{sol} = \log_f(M)$
```

Figure 4.1: IPFE that is ind-fe-cpa-secure from projective hash functions

Theorem 4.21. Let ℓ and a be positive integers. Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathsf{Gen}_{\mathcal{SM}}$ be a subgroup membership problem generator outputting an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$; and let H be the associated PHF, which we assume to be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-ipfe-compatible. If H is $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs})$-pip-safe then the IPFE scheme FE depicted in Fig. 4.1 is ind-fe-cpa-secure, and $\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}} \leqslant \delta_{\mathcal{L}} + \delta_{vs}$.

Proof. We proceed via a sequence of games, starting in the original ind-fe-cpa experiment, and ending in a game in which the adversary $\mathcal{A}$'s advantage is statistically close to $1/2$. The game steps are depicted in Fig. 4.2. Let S_i denote the event "The output of $\mathsf{Game_i}$ is 1".

Game_0. This is $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}$, so by definition:

$$\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}} = |\Pr[S_0\ ] - 1/2| .$$

Game_1. $\mathcal{C}$ computes $\boldsymbol{ct}$ using the hash keys instead of the projection keys and the witness. Though computed differently, the values of the ciphertext components remain unchanged, as is $\mathcal{A}$'s view:

$$\Pr[S_0] = \Pr[S_1]. \tag{4.4}$$

Game_2. Here $\mathcal{C}$ samples c_0 at random from $\mathcal{X} \backslash \mathcal{L}$ instead of from $\mathcal{L}$. Both games are indistinguishable under the $\delta_{\mathcal{L}}$-hardness of $\mathcal{SM}$, and it holds that:

$$|\Pr[S_1] - \Pr[S_2\ ]| \leqslant \delta_{\mathcal{L}}. \tag{4.5}$$

Let us now bound $\mathcal{A}$'s probability of guessing the bit β in Game_2. Observe that when $\mathcal{A}$ submits its' guess β' for β all the information $\mathcal{A}$ can use for its guess comes from: (1) the public key mpk; (2) the challenge ciphertext $\boldsymbol{ct}$; and (3) $\mathcal{A}$'s key derivation queries.

Intuition. Following [ALS16]'s proof methodology, we first delimit the information leaked in the challenge ciphertext by only considering the dimension in which both potential challenge ciphertexts differ. To this end we project this information onto the subspace generated by

1. Sample $(\mathsf{mpk}, \mathsf{msk}) := (\mathbf{hp}, \mathbf{hk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and $\beta \hookleftarrow \{0,1\}$
2. Send mpk to $\mathcal{A}$ and answer pre-challenge phase key derivation queries
3. Receive $\boldsymbol{m}_0$, $\boldsymbol{m}_1$ from $\mathcal{A}$
4. Sample $(x_0, w) \leftarrow \mathsf{R}$ and let $c_0 := x_0$
5. $\boxed{\boxed{\text{Sample } y_0 \hookleftarrow \langle\Upsilon\rangle,\ y_0 \neq 1 \text{ and overwrite } c_0 \leftarrow x_0 \cdot y_0 \in \mathcal{X}\backslash\mathcal{L}}}$
6. For $1 \leqslant i \leqslant \ell$:
7. $\quad\boxed{c_i := \mathsf{hash}(\mathsf{hk}_i, c_0) \cdot f^{m_{b,i}}}$
8. Let $\boldsymbol{ct} := (c_0, \boldsymbol{c})$
9. Send $\boldsymbol{ct}$ to $\mathcal{A}$ and answer post-challenge phase key derivation queries
10. Receive β' from $\mathcal{A}$
11. If $(\beta = \beta')$ return 1, else return 0.

$\boxed{\text{Framed}}$ text highlights the evolution from Game_0 to Game_1.
$\boxed{\boxed{\text{Double framed}}}$ text is only executed in Game_2.

Figure 4.2: Security games for proof of Theorem 4.21.

$\boldsymbol{m}_0 - \boldsymbol{m}_1$ (this encapsulates the information revealed on β). We then consider the distribution of the projection of $\mathbf{hk}$ on the subspace generated by $\boldsymbol{m}_0 - \boldsymbol{m}_1$, conditionally on $\mathcal{A}$'s view. Since $\mathcal{A}$ cannot query decryption keys for vectors $\boldsymbol{k}$ s.t. $\langle \boldsymbol{m}_0 - \boldsymbol{m}_1, \boldsymbol{k}\rangle \neq 0$, the δ_{vs}-vector-smoothness of H ensures that projecting $\mathbf{hk}$ onto the subspace generated by $\boldsymbol{m}_0 - \boldsymbol{m}_1$ induces a distribution $\{\mathsf{hash}(\langle \mathbf{hk}, \boldsymbol{m}_0 - \boldsymbol{m}_1\rangle, y) \mid y \in \langle\Upsilon\rangle\}$ which is δ_{vs}-close to $\mathcal{U}(F)$, and thus $\boldsymbol{m}_\beta$ is statistically hidden in $\boldsymbol{c}$.

Details. Consider the information leaked on β by the challenge ciphertext. The decomposability and homomorphic properties of H allow to write the coordinates of $\boldsymbol{c}$ as:

$$\begin{cases} c_0 = x_0 \cdot y_0 \in \mathcal{X}\backslash\mathcal{L},\ y_0 \neq 1 \\ c_i = f^{m_{b,i}} \cdot \mathsf{hash}(\mathsf{hk}_i, x_0) \cdot \mathsf{hash}(\mathsf{hk}_i, y_0) \in \Pi \ \text{ for } i \in [\ell]. \end{cases}$$

Since this decomposition of c_0 is unique (by Definition 3.18), and since $x_0 \in \mathcal{L}$, information theoretically, for $i \in [\ell]$ the value $\mathsf{hash}(\mathsf{hk}_i, x_0)$ is fixed by the values hp_i and c_0. We denote

$$z_i := f^{m_{b,i}} \cdot \mathsf{hash}(\mathsf{hk}_i, y_0) \in F \text{ for } i \in [\ell].$$

Any information fixed on the bit β by c_i is thus contained in z_i, and it suffices to consider the information which is leaked by $\boldsymbol{z} := (z_1, \ldots, z_\ell) \in F^\ell$.

Now using Lemma 4.8 one can deterministically build a matrix $\mathbf{B}_{\boldsymbol{m}} \in \mathcal{R}^{\ell\times\ell}$ associated to $\boldsymbol{m} := \boldsymbol{m}_1 - \boldsymbol{m}_0$, whose rows we denote $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell)$. By Definition 4.7 the matrix $\mathbf{B}_{\boldsymbol{m}}$ is invertible mod n_f; so since $\boldsymbol{z} \in F^\ell$, and $\mathsf{ord}(F) = n_f$, all the information fixed by $\boldsymbol{z}$ is contained in the vector $(\prod_{i\in[\ell]} z_i^{b_{1,i}}, \ldots, \prod_{i\in[\ell]} z_i^{b_{\ell,i}})$. It thus suffices to consider the information on β given by:

$$\prod_{i\in[\ell]} z_i^{b_{j,i}} = f^{\langle \boldsymbol{m}_\beta, \boldsymbol{b}_j\rangle} \cdot \mathsf{hash}(\boldsymbol{b}_j^T \cdot \mathbf{hk}, y_0) \qquad \text{for } j \in [\ell].$$

For $j \in [\ell-1]$ we have $\boldsymbol{b}_j \in \boldsymbol{m}^\perp$, so the value of $f^{\langle \boldsymbol{m}_\beta, \boldsymbol{b}_j\rangle} \cdot \mathsf{hash}(\boldsymbol{b}_j^T \cdot \mathbf{hk}, y_0)$ provides no information on β. Let us now consider that contained in:

$$f^{\langle \boldsymbol{m}_\beta, \boldsymbol{b}_\ell\rangle} \cdot \mathsf{hash}(\boldsymbol{b}_\ell^T \cdot \mathbf{hk}, y_0). \tag{4.6}$$

To this end we evaluate the distribution of $\mathsf{hash}(\boldsymbol{b}_\ell^T \cdot \mathbf{hk}, y_0)$, and so we need to consider the information leaked on $\mathbf{hk}$ via key derivation queries. First observe that any queried $\boldsymbol{k} \in \mathcal{K}$ must belong to $\boldsymbol{m}^\perp$, and so is a linear combination of vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell-1}$. We can thus apply the δ_{vs}-vector-smoothness over $\mathcal{X}$ on F of H, which ensures that given c_0, $\mathbf{hp}$ and $\boldsymbol{b}_j^T \cdot \mathbf{hk}$ for $j \in [\ell-1]$, the distribution of $\mathsf{hash}(\boldsymbol{b}_\ell^T \cdot \mathbf{hk}, y_0)$ is δ_{vs}-close to $\mathcal{U}(F)$, and statistically hides $f^{\langle \boldsymbol{m}_\beta, \boldsymbol{b}_\ell\rangle}$ in Eq. (4.6). Consequently:

$$|\Pr[S_2] - 1/2| \leqslant \delta_{vs} \tag{4.7}$$

Combining Eqs. (4.4), (4.5) and (4.7) concludes the proof since $\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}} \leqslant \delta_{\mathcal{L}} + \delta_{vs}$. □

Running Example 1 – Instantiation from DDH

Instantiating the IPFE of Fig. 4.1 with $\mathsf{H}_{\mathsf{ddh}}$ yields the DDH based IPFE of [ALS16], and the ElGamal based IPFE of [ABDP16]. Furthermore, we obtain the same upper $\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}$ as they do from their proofs.

Running Example 2 – Instantiation from HSM-CL

Instantiating the IPFE of Fig. 4.1 with $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ yields our HSM-CL based FE scheme computing inner products in $\mathbf{Z}$ published in [CLT18a]. We hereafter detail this instantiation.

Setting the parameters. We use the output $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$ of the Gen generator of Definition 3.1 and require that q is a μ bit prime, with $\mu \geqslant \lambda$. The message and key spaces are $\mathcal{M} = \mathcal{K} = \{\boldsymbol{x} \in \mathbf{Z}^\ell : ||\boldsymbol{x}||_\infty < \sqrt{\frac{q}{2\ell}}\}$. The decryption algorithm uses a centred modulus to recover $\langle \boldsymbol{k}, \boldsymbol{m}\rangle$ over $\mathbf{Z}$. To guarantee the scheme's security we sample the coordinates of the secret key from $\mathcal{D}_{\mathbf{Z},\sigma}$, i.e. discrete Gaussian entries of standard deviation $\sigma > \tilde{s}q^{3/2}\sqrt{\lambda}$, which yields $\delta_{vs} = 2^{-\lambda}$ (*cf.* Lemma 4.13). To sample encryption randomness (i.e. witnesses for $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$), it suffices to use $\mathcal{D}_{\mathbf{Z}^\ell,\sigma'}$ for $\sigma' > \tilde{s}\sqrt{\lambda}$, since $\{g_q^r, r \hookleftarrow \mathcal{D}_{\mathbf{Z}^\ell,\sigma'}\}$ is at distance less than $2^{-\lambda}$ from the uniform distribution in G^q (*cf.* Lemma 3.3).

Construction. Fig. 4.3 depicts the generic IPFE of Fig. 4.1 instantiated with $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$.

```
$\mathsf{Setup}(1^\lambda, 1^\mu, 1^\ell)$:
    Sample a $\mu$ bit prime $q$
    $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$
    For $1 \leqslant i \leqslant \ell$ :
        Sample $\mathsf{hk}_i \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma}$
        Let $\mathsf{hp}_i \leftarrow g_q^{\mathsf{hk}_i}$
    Return $\mathsf{msk} := \mathbf{hk}$
    and $\mathsf{mpk} := (\tilde{s}, g_q, f, q, \mathbf{hp})$

$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k})$
    If $\boldsymbol{k} \notin \mathcal{K}$, return $\perp$
    Let $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \langle \boldsymbol{k}, \mathbf{hk} \rangle \in \mathbf{Z}$
    Return $(\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k})$

$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m})$
    If $\boldsymbol{m} \notin \mathcal{M}$, return $\perp$
    Sample $r \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma'}$; set $c_0 \leftarrow g_q^r$
    For $1 \leqslant i \leqslant \ell$ :
        Let $c_i \leftarrow f^{m_i} \mathsf{hp}_i^r$
    Return $\boldsymbol{ct} := (c_0, \{c_i\}_{i \in [\ell]})$

$\mathsf{Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k}), \boldsymbol{ct})$
    If $\boldsymbol{ct} \notin \widehat{G}^{\ell+1}$, return $\perp$
    Let $M \leftarrow (\prod_{i \in [\ell]} c_i^{k_i}) \cdot (c_0^{-\mathsf{sk}_{\boldsymbol{k}}})$
    If $M \notin F$, return $\perp$
    $\mathsf{sol} \leftarrow \mathsf{Solve}(M)$
    If $\mathsf{sol} \geqslant q/2$, return $(\mathsf{sol} - q)$
    Else return $\mathsf{sol}$
```

Figure 4.3: FE scheme computing inner product in $\mathbf{Z}$ from the HSM-CL assumption.

Corollary 4.22 (of Theorem 4.21)**.** If the HSM-CL problem is hard, the IPFE scheme of Fig. 4.3 is ind-fe-cpa-secure.

Walking Example – Instantiation from DCR

Though we have not detailed the PHF which arises from the DCR assumption, one can also build such a PHF (and in fact in the original [CS02] article, one of their instantiations was based on DCR). Let us denote $\mathsf{H}_{\mathsf{dcr}}$ this projective hash function. Just as the (simplified ind-cpa-secure) encryption scheme of Camenisch Shoup [CS03], which relies on the DCR assumption, shares many similarities with $\Pi_{\mathsf{hsm\text{-}cl}}$ of Fig. 3.5, $\mathsf{H}_{\mathsf{dcr}}$ shares many similarities with $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$, and can in fact be made to satisfy all the required properties to instantiate the IPFE of Fig. 4.1. Furthermore, when instantiated from $\mathsf{H}_{\mathsf{dcr}}$, the IPFE of Fig. 4.1 yields the Paillier based construction of [ALS16], and we obtain the same upper bound for $\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\mathsf{ind\text{-}fe\text{-}cpa}}$ as they do from their (non generic) proof.

Running Example 3 – Instantiation from DDH-f

Instantiating the IPFE of Fig. 4.1 with $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ yields our DDH-f based FE scheme computing inner products in $\mathbf{Z}$ published in [CLT18a]. We hereafter detail this instantiation.

Setting the parameters. The parameters are the same as those of the previous instantiation from HSM-CL, only now encryption randomness is sampled from $\mathcal{D}_{\mathbf{Z},\sigma'}$ with $\sigma' > \tilde{s}q\sqrt{\lambda}$, since we need to induce a distribution $\{g^r, r \hookleftarrow \mathcal{D}_{\mathbf{Z}^\ell,\sigma'}\}$ at distance less than $2^{-\lambda}$ from the uniform distribution in G.

Construction. Fig. 4.4 depicts the generic IPFE of Fig. 4.1 instantiated with $\mathsf{H}_{\mathsf{ddh\text{-}f}}$.

Corollary 4.23 (of Theorem 4.21)**.** If the DDH-f problem is hard, the IPFE scheme of Fig. 4.4 is ind-fe-cpa-secure.

```
$\mathsf{Setup}(1^\lambda, 1^\mu, 1^\ell)$:
  1. Sample a $\mu$ bit prime $q$
  2. $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$
  3. Sample $\alpha \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma'}$
  4. Let $h \leftarrow g^\alpha$
  5. For $1 \leqslant i \leqslant \ell$ :
  6.     Sample $\kappa_{0,i}, \kappa_{1,i} \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma}$
  7.     Let $\mathsf{hp}_i \leftarrow g^{\kappa_{0,i}} h^{\kappa_{1,i}}$
  8. Return $\mathsf{msk} := (\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1)$
     and $\mathsf{mpk} := (\tilde{s}, g, h, f, q, \{\mathsf{hp}_i\}_{i\in[\ell]})$

$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k})$
  1. If $\boldsymbol{k} \notin \mathcal{K}$, return $\perp$
  2. Let $(\mathsf{sk}_0, \mathsf{sk}_1) \leftarrow (\langle \boldsymbol{k}, \boldsymbol{\kappa}_0\rangle, \langle \boldsymbol{k}, \boldsymbol{\kappa}_1\rangle)$
  3. Return $((\mathsf{sk}_0, \mathsf{sk}_1), \boldsymbol{k})$

$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m})$
  1. If $\boldsymbol{m} \notin \mathcal{M}$, return $\perp$
  2. Sample $r \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma'}$
  3. Let $z_0 \leftarrow g^r$ and $z_1 \leftarrow h^r$
  4. For $1 \leqslant i \leqslant \ell$ :
  5.     Let $c_i \leftarrow f^{m_i} \mathsf{hp}_i^r$
  6. Return $\boldsymbol{ct} := (z_0, z_1, \{c_i\}_{i\in[\ell]})$

$\mathsf{Dec}(\mathsf{mpk}, ((\mathsf{sk}_0, \mathsf{sk}_1), \boldsymbol{k}), \boldsymbol{ct})$
  1. If $\boldsymbol{ct} \notin \widehat{G}^{\ell+2}$, return $\perp$
  2. Let $M \leftarrow (\prod_{i\in[\ell]} c_i^{k_i}) \cdot (z_0^{\mathsf{sk}_0} z_1^{\mathsf{sk}_1})^{-1}$
  3. If $M \notin F$, return $\perp$
  4. $\mathsf{sol} \leftarrow \mathsf{Solve}(M)$
  5. If $\mathsf{sol} \geqslant q/2$, return $(\mathsf{sol} - q)$
  6. Else return $\mathsf{sol}$
```

Figure 4.4: FE scheme computing inner products in $\mathbf{Z}$ from the DDH-f assumption.

4.3.2 Computing Inner Products Modulo a Prime

In all three instantiations we have provided – from DDH, HSM-CL and DDH-f – inner products are in fact computed in $\mathbf{Z}$. Indeed, even though for DDH, one has $\mathcal{R} = \mathbf{Z}/q\mathbf{Z}$, since computing discrete logarithms is hard in a DDH group, the inner product must be small in order to efficiently decrypt, and hence one cannot actually use the fact the group G is of prime order to compute inner products *modulo a prime*. Such a functionality has interesting applications, namely Agrawal et al. [ALS16, Section 6] motivate functional encryption for the computation of linear functions modulo a prime p by demonstrating that such a scheme can be turned into a bounded collusion functional encryption scheme for all circuits[1], while Agrawal et (other) al. [ABP+17] provide a generic transformation from IPFE to trace-and-revoke systems. Naturally as they are performing linear algebra, their transformation requires the modulus to be prime and preferably quite large (of the order of 128 or 256 bits).

In this section we build efficient FE schemes computing inner products modulo a prime from HSM-CL and DDH-f, which we originally published in [CLT18a]. Here one computes inner products in $\mathbf{Z}/n_f\mathbf{Z}$ where $n_f = \mathsf{ord}(F)$, however hashing keys still live in $K_{\mathsf{hk}} := \mathbf{Z}^a$. This results in similar issues to those raised in the LWE and DCR-based schemes of [ALS16]. Namely, since decryption key queries are performed over the integers, an adversary may query keys for vectors that are linearly dependant over $(\mathbf{Z}/n_f\mathbf{Z})^\ell$ but independent over $\mathbf{Z}^\ell$. To solve this issue we require as in [ALS16] that the authority distributing decryption keys keeps track of all previously revealed decryption keys. This implies that the key generation algorithm must be *stateful*.

Our generic construction, as presented above, is not well suited for this setting, since now the ring in which inner products are computed, and that from which hashing keys are sampled are different. Essentially, problems occur when constructing the matrix associated to $\boldsymbol{m} \in (\mathbf{Z}/n_f\mathbf{Z})^\ell$. Using the constructive proof of Lemma 4.8, the matrix $\mathbf{B}_{\boldsymbol{m}}$ (built deterministically

[1] We note that this application typically calls for small values of p (e.g. $p = 2$).

from $\boldsymbol{m}$) would have components in $\mathbf{Z}/n_f\mathbf{Z}$; since inner products are now computed mod n_f, the components of $\mathbf{B}_{\boldsymbol{m}}$ are not of bounded norm, an so the determinant of $\mathbf{B}_{\boldsymbol{m}}$ could be a multiple of n_f, and hence $\mathbf{B}_{\boldsymbol{m}}$ may not be invertible modulo n_f. Furthermore when used to prove the security of our protocol, one needs to multiply $\mathbf{B}_{\boldsymbol{m}}$ by vectors of hashing keys, which live in $\mathbf{Z}^\ell$; therefore the coordinates of $\mathbf{B}_{\boldsymbol{m}}$ need to be lifted in $\mathbf{Z}$ before one can perform this operation. Thus, to capture the information an adversary can get from key derivation queries, we can no longer use the associated matrix built via Lemma 4.8. Consequently we adopt the solution used in [ALS16]; namely one recursively proves that each key derivation query reveals no information to the adversary on the challenge bit β. Then from the adversary's queries, one builds a matrix $\mathbf{X} \in \mathbf{Z}^{\ell\times\ell}$, which taken mod n_f is invertible. As with the associated matrix, the top $\ell - 1$ rows of $\mathbf{X}$, taken mod n_f, generate $\boldsymbol{m}^\perp \subset (\mathbf{Z}/n_f\mathbf{Z})^\ell$, whereas the last row of $\mathbf{X}$ does not belong to $\boldsymbol{m}^\perp$. Using this matrix in place of the associated matrix, one can then use similar techniques to those of proof of Theorem 4.21 to demonstrate that the scheme is $\mathsf{ind\text{-}fe\text{-}cpa}$.

We illustrate this with constructions from HSM-CL and DDH-f, note however that one cannot build IPFE modulo a prime from DDH in this way; indeed for large sizes of the inner product one cannot efficiently decrypt in a DDH group since the DL problem is hard by assumption.

Inner Products Modulo a Prime from HSM-CL

We here present an FE scheme computing inner products modulo the prime q, but from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ which samples hashing keys in $\mathbf{Z}$.

Setting the parameters. We use the output $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$ of the Gen generator of Definition 3.1; q is a μ-bit prime where $\mu \geqslant \lambda$. The message and vector spaces $\mathcal{M}$ and $\mathcal{K}$ are now $(\mathbf{Z}/q\mathbf{Z})^\ell$. Given an encryption of $\boldsymbol{m} \in (\mathbf{Z}/q\mathbf{Z})^\ell$ and a decryption key for $\boldsymbol{k} \in (\mathbf{Z}/q\mathbf{Z})^\ell$, the decryption algorithm recovers $\langle \boldsymbol{k}, \boldsymbol{m}\rangle \in \mathbf{Z}/q\mathbf{Z}$. To guarantee the scheme's security we sample the hashing key hk from $\mathcal{D}_{\mathbf{Z}^\ell,\sigma}$ with discrete Gaussian entries of standard deviation $\sigma > \sqrt{\lambda} \cdot q \cdot \tilde{s} \cdot (\sqrt{\ell} q)^{\ell-1}$. We require $\sigma' > \tilde{s}\sqrt{\lambda}$ to ensure that $\{g_q^r, r \hookleftarrow \mathcal{D}_{\mathbf{Z}^\ell,\sigma'}\}$ is at distance less than $2^{-\lambda}$ from the uniform distribution in G^q.

A note on efficiency. Observe that the standard deviation σ is chosen much larger than for the IPFE of Fig. 4.3 computing inner products in $\mathbf{Z}$. This is essentially due to the fact that vectors in $\mathcal{M}$ and $\mathcal{K}$ are no longer of bounded norm (for details see proof of Theorem 4.24). Though this does impact efficiency, we note that a similar blow up in key sizes occurs in the LWE and DCR based schemes of [ALS16]. In fact, the impact on their protocols is much worse since, where we can choose the bit size of q fairly small ($|q| \geqslant \lambda$), schemes resulting from LWE and DCR must have a much larger message space in order for security to hold. Consequently, our FE schemes computing inner products modulo a prime are still the most efficient such schemes to date.

Construction. The Setup and Enc algorithms proceed exactly as in Fig. 4.3, the only difference being that Enc operates on message vectors $\boldsymbol{m} \in (\mathbf{Z}/q\mathbf{Z})^\ell$ instead of $\boldsymbol{m} \in \mathbf{Z}^\ell$. In Fig. 4.5 we only define algorithms KeyDer and Dec, since they differ from those of Fig. 4.3.

Theorem 4.24. If the HSM-CL problem is hard, the FE scheme computing inner products in $\mathbf{Z}/q\mathbf{Z}$ of Fig. 4.5 is $\mathsf{ind\text{-}fe\text{-}cpa}$-secure.

Proof. The proof proceeds similarly to that in $\mathbf{Z}$ (*cf.* Corollary 4.22), starting with the real $\mathsf{ind\text{-}fe\text{-}cpa}$ experiment and ending in a game where the ciphertext statistically hides the random

$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k}, st)$

Answering the j^{th} key request for $\boldsymbol{k} \in (\mathbf{Z}/q\mathbf{Z})^\ell$. At any time the internal state st contains at most ℓ tuples $(\boldsymbol{k}_i, \overline{\boldsymbol{k}}_i, \mathsf{sk}_{\boldsymbol{k}_i})$ where the $\boldsymbol{k}_i$'s are previously queried vectors and $(\mathsf{sk}_{\boldsymbol{k}_i}, \overline{\boldsymbol{k}}_i)$ are corresponding secret keys.

```
If $\boldsymbol{k}$ is linearly independent of the $\boldsymbol{k}_i$'s mod $q$ :
    Set $\overline{\boldsymbol{k}} \in \{0, \dots, q-1\}^\ell$ with $\overline{\boldsymbol{k}} = \boldsymbol{k} \bmod q$; $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \langle \mathsf{hk}, \overline{\boldsymbol{k}} \rangle \in \mathbf{Z}$
    $st := (st, (\boldsymbol{k}, \overline{\boldsymbol{k}}, \mathsf{sk}_{\boldsymbol{k}}))$
If $\exists \{\gamma_i\}_{1 \leqslant i \leqslant j-1} \in \mathbf{Z}^{j-1}$ such that $\boldsymbol{k} = \sum_{i=1}^{j-1} \gamma_i \boldsymbol{k}_i \in (\mathbf{Z}/q\mathbf{Z})^\ell$ then:
    $\overline{\boldsymbol{k}} \leftarrow \sum_{i=1}^{j-1} \gamma_i \overline{\boldsymbol{k}}_i \in \mathbf{Z}^\ell$ ; $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \sum_{i=1}^{j-1} \gamma_i \mathsf{sk}_{\boldsymbol{k}_i} \in \mathbf{Z}$
Return $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\boldsymbol{k}})$
```

$\mathsf{Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \overline{\boldsymbol{k}}), \boldsymbol{ct})$

```
If $\boldsymbol{ct} \notin \widehat{G}^{\ell+1}$ return $\perp$
Let $M \leftarrow (\prod_{i \in [\ell]} c_i^{\overline{k}_i}) \cdot c_0^{-\mathsf{sk}_{\boldsymbol{k}}}$
If $M \notin F$ return $\perp$; else return $\mathsf{Solve}(M)$.
```

Figure 4.5: Stateful FE scheme computing inner products in $\mathbf{Z}/q\mathbf{Z}$ from HSM-CL.

bit β.Games 0 to 2 basically proceed identically to those of the proof of Corollary 4.22. The only difference is in the key derivation oracle that the adversary $\mathcal{A}$ has access to, which now executes the *stateful* key derivation algorithm. Thus we have a Game $2'$ for which:

$$|\Pr[S_2'] - \Pr[S_0]| \leqslant \delta_{\mathsf{hsm\text{-}cl}}.$$

Recall that $\mathcal{A}$ can issue queries (key, k) for any $\boldsymbol{k} \in (\mathbf{Z}/q\mathbf{Z})^\ell$ satisfying $\langle \boldsymbol{k}, \boldsymbol{m}_0 \rangle = \langle \boldsymbol{k}, \boldsymbol{m}_1 \rangle \in \mathbf{Z}/q\mathbf{Z}$. For each query, $\mathcal{A}$ is given a secret key $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\boldsymbol{k}})$ as in the real scheme. And in Game $2'$ we have:

$$\begin{cases} c_0 = f^v \cdot g_q^r \\ c_i = f^{m_{\beta,i} + v \cdot \mathsf{hk}_i} \cdot \mathsf{hp}_i^r, \quad \forall i \in [\ell] \end{cases}$$

where $v \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and $r \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma'}$. Therefore, the challenge ciphertext information-theoretically reveals:

$$\boldsymbol{z}_\beta := \boldsymbol{m}_\beta + v \cdot \mathbf{hk} \bmod q.$$

We define $\boldsymbol{m} := (m_1, \dots, m_\ell) = \boldsymbol{m}_1 - \boldsymbol{m}_0 \in (\mathbf{Z}/q\mathbf{Z})^\ell$, and, assuming $\mathcal{A}$ has performed j private key queries, for $1 \leqslant i \leqslant j$, we denote $\boldsymbol{k}_i \in (\mathbf{Z}/q\mathbf{Z})^\ell$ the vectors for which keys have been derived.

We want to demonstrate that from $\mathcal{A}$'s view, the bit β is statistically hidden in Game $2'$. However we cannot use the associated matrix and smoothness as in the proof of Theorem 4.21; indeed, if we build $\mathbf{B}_{\boldsymbol{m}}$ as in proof of Lemma 4.8 we cannot guarantee that $\mathbf{B}_{\boldsymbol{m}}$ is invertible modulo q, since $\det(\mathbf{B}_{\boldsymbol{m}} \mathbf{B}_{\boldsymbol{m}}^T)$ could be a multiple of q. Therefore, so as to ensure the queried vectors $\boldsymbol{k}_i$ do not in some way depend on β, we prove via induction that after the j first private key queries (where $j \in \{0, \dots, \ell - 1\}$), $\mathcal{A}$'s view remains statistically independent of β, thus proving that the challenge ciphertext in Game $2'$ statistically hides β such that $|Pr[S_2'] - 1/2| \leqslant 2^{-\lambda}$.

The induction proceeds on the value of j. For $j = 0$ the adversary can make no private key queries. With this restriction Games 2 and $2'$ are identical. It thus follows from proof of Theorem 4.21 that for $j = 0$ the induction hypothesis holds, i.e. $\mathcal{A}$'s view is indeed statistically independent of β.

Consider $j \in \{0, \dots, \ell - 1\}$. From the induction hypothesis one may assume that at this point the state $st = \{(\boldsymbol{k}_i, \overline{\boldsymbol{k}}_i, \mathsf{sk}_{\boldsymbol{k}_i}) \in (\mathbf{Z}/q\mathbf{Z})^\ell \times \mathbf{Z}^\ell \times \mathbf{Z}\}_{i \in [j]}$ is independent of β. Indeed if $\mathcal{A}$'s view after $j-1$ requests is independent of β then the j^{th} request performed by $\mathcal{A}$ must be so. W.l.o.g. one may assume that the key requests $\boldsymbol{k}_i$ performed by the adversary are linearly independent (otherwise $\mathcal{A}$ does not gain additional information from its request). This implies that the $\overline{\boldsymbol{k}}_i$'s are linearly independent mod q and generate a subspace of:

$$\boldsymbol{m}^\perp = \{\boldsymbol{x} \in (\mathbf{Z}/q\mathbf{Z})^\ell : \langle \boldsymbol{x}, \boldsymbol{m} \rangle = 0 \bmod q\}$$

Moreover the set $\{\overline{\boldsymbol{k}}_i\}_{i \in [j]}$ (generated during private key queries) can be extended to a basis $\{\overline{\boldsymbol{k}}_i\}_{i \in [\ell-1]}$ of $\boldsymbol{m}^\perp$. We define $\mathbf{X}_{\mathsf{top}} \in \mathbf{Z}^{(\ell-1)\times\ell}$ to be the matrix whose rows are the vectors $\overline{\boldsymbol{k}}_i$ for $i \in [\ell - 1]$.

$$\mathbf{X}_{\mathsf{top}} = \begin{bmatrix} \overline{\boldsymbol{k}}_1^T \\ \overline{\boldsymbol{k}}_2^T \\ \vdots \\ \overline{\boldsymbol{k}}_{\ell-1}^T \end{bmatrix}$$

Let $\boldsymbol{k}_{\mathsf{bot}} \in (\mathbf{Z}/q\mathbf{Z})^\ell$ be a vector such that $\boldsymbol{k}_{\mathsf{bot}} \notin \boldsymbol{m}^\perp$. This vector $\boldsymbol{k}_{\mathsf{bot}}$ is constructed deterministically from the set $\{\overline{\boldsymbol{k}}_i\}_{i \in [j]}$ and $\boldsymbol{m}$. We define $\overline{\boldsymbol{k}}_{\mathsf{bot}}$ to be the canonical lift of $\boldsymbol{k}_{\mathsf{bot}}$ over $\mathbf{Z}$, and $\mathbf{X}$ as:

$$\mathbf{X} = \left[\begin{array}{c} \mathbf{X}_{\mathsf{top}} \\ \hline \overline{\boldsymbol{k}}_{\mathsf{bot}}^T \end{array}\right]$$

The matrix $\mathbf{X}$ is built deterministically, invertible modulo q by construction, and independent of β by induction hypothesis and by construction. Since $\mathbf{X}$ is known to $\mathcal{A}$ (in the information theoretical sense), we need only prove that $\mathbf{X} \cdot \boldsymbol{z}_\beta$ is statistically independent of β. And since $\mathbf{X}_{\mathsf{top}} \cdot (\boldsymbol{m}_1 - \boldsymbol{m}_0) = 0 \bmod q$, we need only consider:

$$\langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{z}_\beta \rangle = \langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{m}_\beta \rangle + \upsilon \langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \mathbf{hk} \rangle \bmod q. \tag{4.8}$$

Now using $\mathbf{X}$ in place of the associated matrix, we can proceed as in proof of Lemma 4.13 to demonstrate that choosing $\sigma \geqslant \sqrt{\lambda} \cdot q \cdot \tilde{s} \cdot (\sqrt{\ell} q)^{\ell-1}$ ensures that $\langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \mathbf{hk} \rangle \bmod q$ follows a distribution $2^{-\lambda}$-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Note that we have a much larger standard deviation σ than that obtained in Lemma 4.13, this is due to the fact the entries of $\mathbf{X}$ are *not* of bounded norm. Indeed, denoting $\Lambda := \{\boldsymbol{t} \in \mathbf{Z}^\ell | \mathbf{X}_{\mathsf{top}} \cdot \boldsymbol{t} = \mathbf{0} \wedge \boldsymbol{t} = \mathbf{0} \bmod s\}$, the upper bound we get for $\lambda_1(q \cdot \Lambda)$ is $q \cdot \tilde{s} \cdot (\sqrt{\ell} q)^{\ell-1}$.

Now in Eq. (4.8), $\gcd(\upsilon, q) = 1$ with overwhelming probability, so $\upsilon \cdot \langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \mathbf{hk} \rangle$ statistically hides $\langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{m}_\beta \rangle$, and thereby β, thus concluding the proof. □

Inner Products Modulo a Prime from DDH-f

As in the HSM-CL based scheme of Fig. 4.5, the key generation algorithm is stateful to ensure adversaries cannot query keys for vectors that are linearly dependant over $(\mathbf{Z}/q\mathbf{Z})^\ell$ but independent over $\mathbf{Z}^\ell$.

Setting the parameters. These are the same as for the FE scheme computing inner products modulo q from HSM-CL, only now encryption randomness is sampled from $\mathcal{D}_{\mathbf{Z},\sigma'}$ with $\sigma' > \tilde{s} q \sqrt{\lambda}$ (as in Fig. 4.4).

Construction. Algorithms Setup and Enc proceed exactly as in Fig. 4.4. In Fig. 4.6 we only define algorithms KeyDer and Dec, which differ from those of Fig. 4.4.

```
$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k}, st)$
    Answering the $j^{th}$ key request for $\boldsymbol{k} \in (\mathbf{Z}/q\mathbf{Z})^\ell$. At any time the internal state $st$ contains
    at most $\ell$ tuples $(\boldsymbol{k}_i, \overline{\boldsymbol{k}}_i, \mathsf{sk}_{\boldsymbol{k}_i})$ where the $\boldsymbol{k}_i$'s are previously queried vectors and $(\mathsf{sk}_{\boldsymbol{k}_i}, \overline{\boldsymbol{k}}_i)$
    are corresponding secret keys.

If $\boldsymbol{k}$ is linearly independent of the $\boldsymbol{k}_i$'s modulo $q$ :
    Set $\overline{\boldsymbol{k}} \in \{0, \dots, q-1\}^\ell$ with $\overline{\boldsymbol{k}} = \boldsymbol{k} \mod q$
    $\mathsf{sk}_{\boldsymbol{k}} := (\mathsf{sk}_0, \mathsf{sk}_1) \leftarrow (\langle \overline{\boldsymbol{k}}, \boldsymbol{\kappa}_0 \rangle, \langle \overline{\boldsymbol{k}}, \boldsymbol{\kappa}_1 \rangle) \in \mathbf{Z} \times \mathbf{Z}$
    $st := (st, (\boldsymbol{k}, \overline{\boldsymbol{k}}, \mathsf{sk}_{\boldsymbol{k}}))$
If $\exists \{\gamma_i\}_{1 \leqslant i \leqslant j-1} \in \mathbf{Z}^{j-1}$ such that $\boldsymbol{k} = \sum_{i=1}^{j-1} \gamma_i \boldsymbol{k}_i \in (\mathbf{Z}/q\mathbf{Z})^\ell$ then:
    $\overline{\boldsymbol{k}} \leftarrow \sum_{i=1}^{j-1} \gamma_i \overline{\boldsymbol{k}}_i \in \mathbf{Z}^\ell$; $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \sum_{i=1}^{j-1} \gamma_i \mathsf{sk}_{\boldsymbol{k}_i} \in \mathbf{Z} \times \mathbf{Z}$
Return $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\boldsymbol{k}})$

$\mathsf{Dec}(\mathsf{mpk}, \boldsymbol{ct}, (\mathsf{sk}_{\boldsymbol{k}}, \overline{\boldsymbol{k}}))$

If $\boldsymbol{ct} \notin \widehat{G}^{\ell+2}$ return $\perp$; else parse $\overline{\boldsymbol{k}} = (\overline{k}_1, \dots, \overline{k}_\ell)$; $\mathsf{sk}_{\boldsymbol{k}} = (\mathsf{sk}_0, \mathsf{sk}_1)$
Let $M \leftarrow (\prod_{i \in [\ell]} c_i^{\overline{k}_i}) z_0^{-\mathsf{sk}_0} z_1^{-\mathsf{sk}_1}$. If $M \notin F$ return $\perp$; else return $\mathsf{Solve}(M)$.
```

Figure 4.6: Stateful FE scheme for inner products over $\mathbf{Z}/q\mathbf{Z}$ from DDH-f.

Theorem 4.25. If the DDH-f problem is hard, the FE scheme computing inner products in $\mathbf{Z}/q\mathbf{Z}$ of Fig. 4.6 is ind-fe-cpa-secure.

Proof. The proof proceeds similarly to that of Theorem 4.24. We follow the same steps as in proof of Theorem 4.21 up until the definition of Game 2, the only difference being that the adversary $\mathcal{A}$ queries the *stateful* key derivation algorithm. We denote Game i' the variant of Game i in which the key derivation algorithm is stateful. From the proof of Theorem Theorem 4.21, it holds that $|\Pr[S_2'] - \Pr[S_0']| = \delta_{\mathsf{ddh\text{-}f}}$.

As in the original Game 2, here in Game $2'$ the challenge ciphertext information theoretically reveals $\boldsymbol{z}_\beta := \boldsymbol{m}_\beta + \upsilon \cdot \boldsymbol{\kappa}_1 \mod q$.

We define $\boldsymbol{m} := (m_1, \dots, m_\ell) = \boldsymbol{m}_1 - \boldsymbol{m}_0 \in (\mathbf{Z}/q\mathbf{Z})^\ell$; and, assuming $\mathcal{A}$ has performed j private key queries, for $1 \leqslant i \leqslant j$, we denote $\boldsymbol{k}_i \in (\mathbf{Z}/q\mathbf{Z})^\ell$ the vectors for which keys have been derived.

From here on, to prove that in Game $2'$ the challenge ciphertext statistically hides the bit β, we prove via induction that after the j first private key queries, $\mathcal{A}$'s view remains statistically independent of β, thus proving that $|Pr[S_2'] - 1/2| \leqslant 2^{-\lambda}$. The induction proceeds on the value of j.

For $j = 0$ the adversary can make no private key queries, and so Games 2 and $2'$ are identical. It thus follows from proof of Theorem 4.21 that for $j = 0$ the induction hypothesis holds, i.e. $\mathcal{A}$'s view is indeed statistically independent of β. Now consider $j \in \{0, \dots, \ell - 1\}$. From the induction hypothesis one may assume that at this point the state $st = \{(\boldsymbol{k}_i, \overline{\boldsymbol{k}}_i, \mathsf{sk}_{\boldsymbol{k}_i})\}_{i \in [j]}$ is independent of β (if $\mathcal{A}$'s view after $j-1$ requests is independent of β then the j^{th} request must be so). W.l.o.g. one may assume that key requests $\boldsymbol{k}_i$ performed by $\mathcal{A}$ are linearly independent.

This implies that the $\overline{\boldsymbol{k}}_i$'s are linearly independent modulo q and generate a subspace of

$$\boldsymbol{m}^{\perp} = \{\boldsymbol{k} \in (\mathbf{Z}/q\mathbf{Z})^{\ell} : \langle \boldsymbol{k}, \boldsymbol{m} \rangle = 0 \bmod q\}.$$

The set $\{\overline{\boldsymbol{k}}_i\}_{i\in[j]}$ can be extended to a basis $\{\overline{\boldsymbol{k}}_i\}_{1\leqslant i\leqslant \ell-1}$ of $\boldsymbol{m}^{\perp q}$. We define $\mathbf{X}_{\mathsf{top}} \in \mathbf{Z}^{(\ell-1)\times\ell}$ to be the matrix whose rows are the vectors $\overline{\boldsymbol{k}}_i$ for $i \in [\ell-1]$. Let $\boldsymbol{k}_{\mathsf{bot}} \in (\mathbf{Z}/q\mathbf{Z})^{\ell}$ be a vector chosen deterministically, $\boldsymbol{k}_{\mathsf{bot}} \notin \boldsymbol{m}^{\perp}$, so that the adversary $\mathcal{A}$ can also easily compute $\boldsymbol{k}_{\mathsf{bot}}$. We define $\overline{\boldsymbol{k}}_{\mathsf{bot}}$ to be the canonical lift of $\boldsymbol{k}_{\mathsf{bot}}$ over $\mathbf{Z}$, and $\mathbf{X}$ as:

$$\mathbf{X} = \left[\begin{array}{c} \mathbf{X}_{\mathsf{top}} \\ \hline \overline{\boldsymbol{k}}_{\mathsf{bot}}^{T} \end{array}\right] \in \mathbf{Z}^{\ell\times\ell}.$$

The matrix $\mathbf{X}$ is invertible modulo q, statistically independent of β by induction and by construction, and computable by $\mathcal{A}$, thus we need only prove that $\mathbf{X} \cdot \boldsymbol{z}_{\beta}$ is statistically independent of β. And since $\mathbf{X}_{\mathsf{top}} \cdot (\boldsymbol{m}_1 - \boldsymbol{m}_0) = 0 \bmod q$, we need only consider

$$\langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{z}_{\beta} \rangle = \langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{m}_{\beta} \rangle + \upsilon \cdot \langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{\kappa}_1 \rangle \bmod q.$$

Now using $\mathbf{X}$ in place of the associated matrix, one can proceed as in proof of Lemma 4.14 to demonstrate that choosing $\sigma > \tilde{s} \cdot q^{\ell} \cdot \sqrt{\lambda} \cdot (\sqrt{\ell})^{\ell-1}$ suffices to ensure that from $\mathcal{A}$'s view, $\langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{\kappa}_1 \rangle$ follows a distribution statistically close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Finally, since υ is sampled uniformly at random from $\mathbf{Z}/q\mathbf{Z}$, $\upsilon \neq 0 \bmod q$ with all but negligible probability as q is a μ bit prime, with $\mu \geqslant \lambda$. Thus β is statistically hidden in the expression of $\langle \overline{\boldsymbol{k}}_{\mathsf{bot}}, \boldsymbol{z}_{\beta} \rangle$, which concludes the proof. □

4.4 IPFE Secure against Active Adversaries from PHFs

Securing cryptosystems against active adversaries is a major goal when designing cryptographic protocols, as it handles the much more realistic case where adversaries do not follow the prescribed protocol, and may attempt by any means to extract sensitive information. This also entails that, since the adversary is more powerful than the less dangerous passive adversary, devising protocols which are provably secure against active adversaries is considerably more complex. In this section, we extend the ind-fe-cpa-secure construction of Section 4.3 to handle active adversaries.

Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathsf{Gen}_{\mathcal{SM}}$ be a subgroup membership problem generator outputting an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$; ℓ and a be positive integers; $\mathcal{M} \subseteq \mathcal{R}^{\ell}$ be the plaintext space; and $\mathcal{K} \subseteq \mathcal{R}^{\ell}$ be the space from which keys are derived. For security, the pair of PHFs associated to $\mathcal{SM}$, denoted $(\mathsf{H}, \mathsf{eH})$ must be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs}, \delta_{vu})$-aip-safe, and we denote E the auxiliary input space of eH. The construction also requires a CRHF generator $\mathcal{H}$, such that $\hbar \hookleftarrow \mathcal{H}(1^{\lambda})$ maps $\{0,1\}^*$ to the efficiently recognisable set E; and a strongly unforgeable OTS scheme $\mathsf{OTS} := (\mathsf{Setup}_{\mathsf{OTS}}, \mathsf{KeyGen}_{\mathsf{OTS}}, \mathsf{Sign}, \mathsf{Verif})$. The resulting ind-fe-cca-secure IPFE scheme, depicted in Fig. 4.7, recovers $\langle \boldsymbol{m}, \boldsymbol{k} \rangle \in \mathcal{R}$ for $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{k} \in \mathcal{K}$, and resembles that of [BBL17].

Remark. The auxiliary input space E of eH is chosen to be the output of a CRHF; this ensures that adversaries – given a ciphertext computed from $(c_0, e) \in \mathcal{X} \times E$ – cannot compute a *different* ciphertext for the same values e and c_0. Indeed if it could do so, it needn't compute the hash values c_i for $i \in [\ell]$, since once could reuse those of the given ciphertext, and thereby break ciphertext integrity.

```
$\mathsf{Setup}(1^\lambda, 1^\ell)$:
 1. $\mathcal{SM} \leftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda)$
 2. $\hbar \leftarrow \mathcal{H}(1^\lambda)$; $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{OTS}}(1^\lambda)$
 3. For $1 \leqslant i \leqslant \ell$ :
 4.     Sample $\mathsf{hk}_i \leftarrow \mathsf{hashkg}(\mathcal{SM})$
 5.     $\mathsf{hp}_i \leftarrow \mathsf{projkg}(\mathsf{hk}_i)$
 6.     Sample $\mathsf{ehk}_i \leftarrow \mathsf{ehashkg}(\mathcal{SM})$
 7.     $\mathsf{ehp}_i \leftarrow \mathsf{eprojkg}(\mathsf{ehk}_i)$
 8. Return $\mathsf{mpk} := (\mathbf{hp}, \mathbf{ehp}, \hbar)$
    and $\mathsf{msk} := (\mathbf{hk}, \mathbf{ehk})$

$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m})$:
 1. If $\boldsymbol{m} \notin \mathcal{M}$ return $\perp$
 2. $(\mathsf{vk}, \mathsf{sk}_{\mathsf{OTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{OTS}}(\mathsf{pp})$
 3. $e \leftarrow \hbar(\mathsf{vk})$
 4. Sample $(c_0, w) \leftarrow \mathsf{R}$
 5. For $1 \leqslant i \leqslant \ell$ :
 6.     $c_i \leftarrow \mathsf{projhash}(\mathsf{hp}_i, c_0, w) \cdot f^{m_i}$
 7.     $\overline{c}_i \leftarrow \mathsf{eprojhash}(\mathsf{ehp}_i, c_0, w, e)$
 8. Set $\boldsymbol{ct} := (c_0, \boldsymbol{c}, \overline{\boldsymbol{c}})$
 9. $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{OTS}}, \boldsymbol{ct})$
10. Return $(\boldsymbol{ct}, \mathsf{vk}, \sigma)$

$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k})$:
 1. If $\boldsymbol{k} \notin \mathcal{K}$ return $\perp$
 2. $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \boldsymbol{k}^T \cdot \mathbf{hk} \in \mathcal{R}^a$
 3. $\overline{\mathsf{sk}}_{\boldsymbol{k}} \leftarrow \boldsymbol{k}^T \cdot \mathbf{ehk} \in \mathcal{R}^{2a}$
 4. Return $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}}, \boldsymbol{k})$

$\mathsf{Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}}, \boldsymbol{k}), (\boldsymbol{ct}, \mathsf{vk}, \sigma))$:
 1. If $\boldsymbol{ct} \notin \widehat{\mathcal{X}} \times \Pi^{2\ell}$ then return $\perp$
 2. If $\mathsf{Verif}(\mathsf{vk}, \boldsymbol{ct}, \sigma) = 0$
 3.     Then return $\perp$
 4. $e \leftarrow \hbar(\mathsf{vk})$
 5. If $\mathsf{ehash}(\overline{\mathsf{sk}}_{\boldsymbol{k}}, c_0, e) \neq \prod_{i \in [\ell]} \overline{c}_i^{k_i}$
 6.     Then return $\perp$
 7. $M \leftarrow (\prod_{i \in [\ell]} c_i^{k_i}) \cdot \mathsf{hash}(\mathsf{sk}_{\boldsymbol{k}}, c_0)^{-1}$
 8. If $M \notin F$ then return $\perp$
 9. Return $\mathsf{sol} \leftarrow \log_f(M)$
```

Figure 4.7: IPFE that is ind-fe-cca-secure from projective hash functions

Correctness. As $K_{\mathsf{hk}} = \mathcal{R}^a$ and $K_{\mathsf{ehk}} = \mathcal{R}^{2a}$, one has $\mathbf{hk} \in (\mathcal{R}^\ell)^a$, and $\mathbf{ehk} \in (\mathcal{R}^\ell)^{2a}$ so $\boldsymbol{k}^T \cdot \mathbf{hk} \in \mathcal{R}^a$ and $\boldsymbol{k}^T \cdot \mathbf{ehk} \in \mathcal{R}^{2a}$. Next, by correctness of OTS, ciphertexts output by Enc pass the check on line 2 of Dec; and by key homomorphism and correctness of eH:

$$\mathsf{ehash}(\overline{\mathsf{sk}}_{\boldsymbol{k}}, c_0, e) = \mathsf{ehash}(\boldsymbol{k}^T \cdot \mathbf{ehk}, c_0, e) = \prod_{i \in [\ell]} \mathsf{eprojhash}(\mathsf{ehp}_i, c_0, w, e)^{k_i} = \prod_{i \in [\ell]} \overline{c}_i^{k_i}.$$

Now correctness follows from that of the ind-fe-cpa-secure construction of Fig. 4.1.

Security. In Theorem 4.26 we demonstrate the ind-fe-cca-security of the IPFE of Fig. 4.7. The proof builds upon the ind-fe-cpa-security of the IPFE of Fig. 4.1, additionally using the vector-universality of eH to ensure ciphertext integrity. This property allows us to ensure any decryption queries performed by the adversary which could potentially leak information that is harmful to the security of the scheme are rejected with overwhelming probability. If this holds, we can demonstrate that an adversary cannot, via its decryption queries, learn more information than in an ind-cpa attack, and hence by the vector smoothness of H the scheme is secure. This extends the idea of the universal$_2$ property in the Cramer-Shoup generic construction for building ind-cca-secure PKE from PHFs. One notable difference is that here, we don't consider invalid *ciphertexts* (recall that these are the ciphertexts which, if decrypted, may leak sensitive information for a PKE scheme), but *invalid decryption queries*, where the adversary specifies both the ciphertext to be decrypted, and the vector $\boldsymbol{k} \in \mathcal{K}$ from which the decryption key should be derived.

Theorem 4.26. Let ℓ and a be positive integers. Let $\mathcal{R}$ be either the ring $\mathbf{Z}$ or $\mathbf{Z}/q\mathbf{Z}$ for some prime q; $\mathsf{Gen}_{\mathcal{SM}}$ be a subgroup membership problem generator outputting an instance $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$; let H be the associated PHF which we assume to be $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K})$-$\mathsf{ipfe}$-$\mathsf{compatible}$; and let eH be the resulting EPHF[2]. Assume $\mathcal{H}$ is a δ_{cr}-hard CRHF generator; and $\mathsf{OTS} := (\mathsf{Setup}_{\mathsf{OTS}}, \mathsf{KeyGen}_{\mathsf{OTS}}, \mathsf{Sign}, \mathsf{Verif})$ is a δ_{OTS}-secure suf-$\mathsf{1}$-cma signature scheme. If the pair $(\mathsf{H}, \mathsf{eH})$ is $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs}, \delta_{vu})$-$\mathsf{aip}$-$\mathsf{safe}$ then the IPFE scheme of Fig. 4.7 is ind-fe-cca-secure, and denoting q_{dec} an upper bound on the number of decryption queries made by the adversary for ind-fe-cca security, it holds that:

$$\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}} \leqslant \delta_{\mathcal{L}} + q_{\mathsf{dec}}\Big(\frac{n_f}{n_f - q_{\mathsf{dec}} + 1} \cdot \delta_{vu} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}\Big) + \delta_{vs}.$$

Remark. Before proving the theorem we give a brief comparison to the results obtained in [BBL17]. Details and comparisons of concrete instantiations can be found in Section 4.5. Though [BBL17] do not use the notions of vector smoothness and vector universality, we compare the properties they introduce for PHFs to ours (*cf.* Appendix A); this allows us to compare explicitly the security bounds they obtain to ours, as detailed hereafter.

The [BBL17] proof technique upper bounds the adversary's advantage[3] by $\delta_{\mathsf{bbl}} = \delta_{\mathcal{L}} + |\Delta\mathcal{M}|(\delta_{vs} + \delta_{vu}) + q_{\mathsf{dec}}|\Delta\mathcal{M}|(\delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$, where $|\Delta\mathcal{M}| \leqslant (4(\frac{n_f}{2\ell})^{1/2})^{\ell}$. From our security proof – since $q_{\mathsf{dec}} = \mathsf{poly}(\lambda)$, whereas n_f is exponential in λ – this advantage is upper bounded by $\delta_{\mathsf{us}} = \delta_{\mathcal{L}} + (\delta_{vs} + q_{\mathsf{dec}}\delta_{vu}) + q_{\mathsf{dec}}(\delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$. Since $|\Delta\mathcal{M}|$ grows exponentially with ℓ, and polynomially with n_f (which itself is exponential in λ), it is clear that their result requires much stronger properties from the underlying PHFs to compensate for the factor $|\Delta\mathcal{M}|$. To give a concrete example, for n_f of 128 bits, $\ell = 100$, and allowing the adversary to make $q_{\mathsf{dec}} = 2^{20}$ queries, one gets $|\Delta\mathcal{M}| = 2^{6218}$, and $\delta_{\mathsf{bbl}} = \delta_{\mathcal{L}} + 2^{6218}(\delta_{vs} + \delta_{vu}) + 2^{6238}(\delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$, whereas in this work $\delta_{\mathsf{us}} = \delta_{\mathcal{L}} + (\delta_{vs} + 2^{20}\delta_{vu}) + 2^{20}(\delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$. We note that even if hashing keys are sampled uniformly, which implies $\delta_{vs} = 0$, our security proof significantly reduces $\mathcal{A}$'s advantage, which allows us to use smaller keys, and significantly gain in efficiency.

To prove Theorem 4.26 we first introduce the notion of *valid* and *invalid* decryption queries: valid and rejected decryption queries reveal negligibly more information on the hash keys than what an adversary $\mathcal{A}$ could gain from projection keys and key derivation queries (*cf.* Lemmas 4.28 and 4.29). On the other hand the probability an invalid decryption query was not rejected, thereby potentially leaking relevant information, is negligible (*cf.* Lemma 4.30).

The analysis of this probability is conditioned on $\mathcal{A}$'s view, and performed *a posteriori*, i.e. when $\mathcal{A}$ guesses the challenge bit. Hence the notions of *valid* and *invalid* decryption queries can depend on the challenge messages (even though $\mathcal{A}$ may request decryption queries *before* choosing $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$).

Notation 4.27. Consider a decryption query $(\mathsf{decrypt}, (\boldsymbol{ct}, \mathsf{vk}, \sigma), \boldsymbol{k})$ performed by $\mathcal{A}$, where $\boldsymbol{ct} = (c_0, \boldsymbol{c}, \overline{\boldsymbol{c}})$, and $\boldsymbol{k} \in \mathcal{K}$. After the post-challenge phase of $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{IPFE},\mathcal{A}}$, one can categorise the query. It is said to be *valid* if either (1) $c_0 \in \widehat{\mathcal{L}}$, or (2) $\langle \boldsymbol{k}, \boldsymbol{m}_0 \rangle = \langle \boldsymbol{k}, \boldsymbol{m}_1 \rangle$ in $\mathcal{R}$, where $\boldsymbol{m}_0$, $\boldsymbol{m}_1$ are the challenge messages. Any decryption query which is not valid is said to be *invalid.*

[2] Obtained via the generic construction detailed in Section 3.4.3

[3] To simplify the comparison, we neglect a factor q_{dec} in their favour, their adversary in fact has advantage $\geqslant \delta_{\mathsf{bbl}}$

The above notion is related to that of invalid ciphertexts (*cf.* Definition 3.21). Indeed extending the notion of invalid ciphertexts from PKE to IPFE naturally yields that an IPFE ciphertext is invalid if $c_0 \notin \widehat{\mathscr{L}}$ (*cf.* Notation 4.35). Hence for a decryption query to be invalid, it is necessary (but not sufficient) that the queried ciphertext is invalid.

Proof of Theorem 4.26. We proceed via a sequence of games. Game_0 is the original $\mathsf{ind\text{-}fe\text{-}cca}$ experiment. In Game_2, with all but negligible probability, all the information revealed by decryption queries is contained in that revealed by public keys, the challenge ciphertext and key derivation queries (i.e. the information leaked in a chosen plaintext attack). The evolution of these games is highlighted in Fig. 4.8, the figure does not explicitly depict Game_0, as it follows immediately from the security definition and the scheme of Fig. 4.7. We then proceed as in proof of Theorem 4.21 to demonstrate that in Game_2, $\mathscr{A}$'s advantage is negligible. Let S_i denote the event "The output of *Game i* is 1".

1. Sample $\beta \hookleftarrow \{0,1\}$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
2. Send mpk to $\mathscr{A}$ and answer pre-challenge phase queries
3. Receive $\boldsymbol{m}_0$, $\boldsymbol{m}_1$ from $\mathscr{A}$
4. Let $(\mathsf{vk}, \mathsf{sk}_{\mathsf{OTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{OTS}}(\mathsf{pp})$ and $e \leftarrow \mathscr{h}(\mathsf{vk})$
5. Sample $(x_0, w) \leftarrow \mathsf{R}$ and let $c_0 := x_0$
6. $\boxed{\boxed{\text{Sample } y_0 \hookleftarrow \langle \Upsilon \rangle,\ y_0 \neq 1 \text{ and overwrite } c_0 \leftarrow x_0 \cdot y_0 \in \mathcal{X} \backslash \mathscr{L}}}$
7. For $1 \leqslant i \leqslant \ell$:
8. $\boxed{c_i := \mathsf{hash}(\mathsf{hk}_i, c_0) \cdot f^{m_{b,i}}}$ and $\boxed{\bar{c}_i := \mathsf{ehash}(\mathsf{ehk}_i, c_0, e)}$
9. Let $\boldsymbol{ct} := (c_0, \boldsymbol{c}, \bar{\boldsymbol{c}})$ and compute $\sigma \leftarrow \mathsf{OTS.Sign}(\mathsf{sk}_{\mathsf{OTS}}, \boldsymbol{ct})$
10. Send $(\boldsymbol{ct}, \mathsf{vk}, \sigma)$ to $\mathscr{A}$ and answer post-challenge phase queries
11. Receive β' from $\mathscr{A}$
12. If $(\beta = \beta')$ return 1, else return 0.

$\boxed{\text{Framed}}$ text highlights the evolution from Game_0 to Game_1.
$\boxed{\boxed{\text{Double framed}}}$ text is only executed in Game_2.

Figure 4.8: Evolution of security games for proof of Theorem 4.26.

Game_0. This is the original $\mathsf{ind\text{-}fe\text{-}cca}$ game, s.t.

$$\mathsf{Adv}_{\mathsf{FE},\mathscr{A}}^{\mathsf{ind\text{-}fe\text{-}cca}}(\lambda) = |\Pr[S_0\ ] - 1/2|\,.$$

Game_1. $\mathcal{C}$ computes $\boldsymbol{ct}$ using the hash keys instead of the projection keys and the witness. Though computed differently, the values of the ciphertext components remain unchanged, as is $\mathscr{A}$'s view:

$$\Pr[S_0] = \Pr[S_1]. \tag{4.9}$$

Game_2**.** Here $\mathcal{C}$ samples c_0 at random from $\mathcal{X}\backslash\mathcal{L}$ instead of $\mathcal{L}$. Both games are indistinguishable under the $\delta_{\mathcal{L}}$-hardness of $\mathcal{SM}$, and it holds that:

$$|\Pr[S_1] - \Pr[S_2\,]| \leqslant \delta_{\mathcal{L}}. \tag{4.10}$$

Let us now bound the probability S_2 occurs. When $\mathcal{A}$ submits its' guess β' for β, all $\mathcal{A}$'s queries are either valid or invalid, since the post-challenge phase is over. Furthermore, since $\mathcal{A}$ has finished collecting information, one can analyse $\mathcal{A}$'s conditional probability of guessing β given this information. The information $\mathcal{A}$ can use for its guess β' comes from: (1) the public key mpk; (2) the challenge ciphertext $\boldsymbol{ct}$; (3) key derivation queries; and (4) decryption queries.

Intuition. We first upper bound the probability any invalid decryption query was *not* rejected. Then, assuming all invalid decryption queries were rejected, we demonstrate that all decryption queries – either valid or rejected – do not provide further information to $\mathcal{A}$ than that revealed by projection keys and key derivation queries. This allows us to reduce the ind-fe-cca-security of the scheme of Fig. 4.7 to the ind-fe-cpa-security of the scheme of Fig. 4.1, only where the adversary is also given $\widehat{\mathbf{hp}} \leftarrow \widehat{\mathsf{projkg}}(\mathbf{hk})$.

Bounding the information revealed by decryption queries. Let $\textsc{Bad}$ denote the event an invalid decryption query was not rejected, and $\overline{\textsc{Bad}}$ the complement event. Denoting q_{dec} an upper bound on the number of decryption queries performed by $\mathcal{A}$, in Lemma 4.30, we demonstrate that:

$$\Pr[\textsc{Bad}] \leqslant q_{\mathsf{dec}}(\frac{n_f}{n_f - q_{\mathsf{dec}} + 1} \cdot \delta_{vu} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}). \tag{4.11}$$

It holds that $\Pr[S_2] = \Pr[S_2 \wedge \textsc{Bad}] + \Pr[S_2 \wedge \overline{\textsc{Bad}}] \leqslant \Pr[\textsc{Bad}] + \Pr[S_2 \wedge \overline{\textsc{Bad}}]$. We hereafter bound $\Pr[S_2 \wedge \overline{\textsc{Bad}}]$ (and thus assume $\textsc{Bad}$ does not occur). Precisely, in the following we demonstrate that

$$|\Pr[S_2 \wedge \overline{\textsc{Bad}}] - 1/2| \leqslant \delta_{vs}. \tag{4.12}$$

From Lemmas 4.28 and 4.29, decryption queries which do not cause $\textsc{Bad}$ to occur provide no further information to $\mathcal{A}$ on $\mathbf{hk}$ than what it can obtain from projection keys $\widehat{\mathbf{hp}} := \widehat{\mathsf{projkg}}(\mathbf{hk})$ and key derivation requests. Moreover, the secret key $\mathbf{hk}$ used to mask the bit β is sampled independently from $\mathbf{ehk}$ which has no influence on $\mathcal{A}$'s view of $\mathbf{hk}$. So to analyse $\mathcal{A}$'s view of β, it suffices to consider the distribution of $\mathbf{hk}$ from $\mathcal{A}$'s view, and given this distribution the information revealed by $(c_0, \boldsymbol{c})$ on β (we can ignore $\bar{\boldsymbol{c}}$ for this analysis). It thus suffices to prove the ind-fe-cpa security of a reduced version of the scheme, identical to that of Fig. 4.1, only where mpk also contains $\widehat{\mathbf{hp}} \leftarrow \widehat{\mathsf{projkg}}(\mathsf{hk})$ (at a high level, we have proven ciphertext integrity; we now prove the ciphertext ensures confidentiality).

Now the only difference between proving ind-fe-cpa-security of the reduced scheme and that of Fig. 4.1 is that $\mathcal{A}$ is now granted the projection keys in $\widehat{\mathbf{hp}}$ and not only the public projection keys in $\mathbf{hp}$. This information is only used when one calls upon the δ_{vs}-vector smoothness of the PHF, which by definition holds even given $\widehat{\mathbf{hp}}$. Thus $|\Pr[S_2 \wedge \overline{\textsc{Bad}}] - 1/2| \leqslant \delta_{vs}$. Consequently if $\textsc{Bad}$ does not occur, the bit β is statistically hidden from $\mathcal{A}$'s view. Putting together Eqs. (4.9) to (4.12) we have:

$$\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}} \leqslant \delta_{\mathcal{L}} + q_{\mathsf{dec}}(\frac{n_f}{n_f - q_{\mathsf{dec}} + 1} \cdot \delta_{vu} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}) + \delta_{vs}.$$

This concludes the proof that the IPFE scheme of Fig. 4.7 is ind-fe-cca-secure. □

Lemmas 4.28 to 4.30 hold for all security games of proof of Theorem 4.26.

Lemma 4.28. Consider the IPFE scheme of Fig. 4.7. Valid decryption queries fix no further information on hk or ehk than what can be deduced of projection keys $\widehat{\mathsf{hp}} := \widehat{\mathsf{projkg}}(\mathsf{hk})$, $\widehat{\mathsf{ehp}} := \widehat{\mathsf{eprojkg}}(\mathsf{ehk})$, and key derivation queries.

Proof. The lemma holds for all security games of proof of Theorem 4.26 since the projection keys $\widehat{\mathsf{hp}}$ and $\widehat{\mathsf{ehp}}$ (which fix the value of the scheme's public keys $\mathsf{hp} = \mathsf{projkg}(\mathsf{hk})$ and $\mathsf{ehp} = \mathsf{eprojkg}(\mathsf{ehk})$), and the information $\mathcal{A}$ can learn from key derivation requests do not change throughout the games. Consider a decryption query $(\mathsf{decrypt}, (\boldsymbol{ct}, \mathsf{vk}, \sigma), \boldsymbol{k})$ where $\boldsymbol{ct} = (c_0, \boldsymbol{c}, \overline{\boldsymbol{c}})$, and $\boldsymbol{k} \in \mathcal{K}$. Since the query is valid one of the following cases apply:
Case (1): $c_0 \in \widehat{\mathcal{L}}$. Algorithm Dec uses ehk to compute $\mathsf{ehash}(\boldsymbol{k}^T \cdot \mathsf{ehk}, c_0, e)$, where $e = \hbar(\mathsf{vk})$, and hk to compute $\mathsf{hash}(\boldsymbol{k}^T \cdot \mathsf{hk}, c_0)$. By correctness and by key homomorphism the PHFs it holds that:

$$\mathsf{hash}(\boldsymbol{k}^T \cdot \mathsf{hk}, c_0) = \widehat{\mathsf{projhash}}(\boldsymbol{k}^T \cdot \widehat{\mathsf{hp}}, c_0) \quad \text{and} \quad \mathsf{ehash}(\boldsymbol{k}^T \cdot \mathsf{ehk}, c_0, e) = \widehat{\mathsf{eprojhash}}(\boldsymbol{k}^T \cdot \widehat{\mathsf{ehp}}, c_0, e).$$

The above values are fixed by $\boldsymbol{k}$, c_0, $\widehat{\mathsf{hp}}$ and $\widehat{\mathsf{ehp}}$, so no more information is fixed on hk or ehk.
Case (2): $\langle \boldsymbol{k}, \boldsymbol{m}_0 \rangle = \langle \boldsymbol{k}, \boldsymbol{m}_1 \rangle$ in $\mathcal{R}$. Then $\mathcal{A}$ could query $(\mathsf{key}, \boldsymbol{k})$ and run the Dec algorithm itself. □

Lemma 4.29. Consider the IPFE scheme of Fig. 4.7. Decryption queries $(\mathsf{decrypt}, (\boldsymbol{ct}, \mathsf{vk}, \sigma), \boldsymbol{k})$ which are rejected provide no information on hk.

Proof. Let $\boldsymbol{ct} = (c_0, \boldsymbol{c}, \overline{\boldsymbol{c}})$ where $c_0 \in \widehat{\mathcal{X}}$, and $\boldsymbol{c}, \overline{\boldsymbol{c}} \in \Pi^\ell$. Since Lemma 4.28 ensures valid decryption queries do not leak information, it suffices to consider invalid decryption queries. If the rejection is due to the OTS verification algorithm, $\mathcal{A}$ learns nothing on ehk or hk. Now suppose the rejection is due to $\mathsf{ehash}(\overline{\mathsf{sk}}_{\boldsymbol{k}}, c_0, e) \neq \prod_{i \in [\ell]} \overline{c}_i^{k_i}$. Each such rejected decryption request – information theoretically – provides $\mathcal{A}$ with an inequality on ehk, however hk is sampled independently from ehk, consequently ehk has no influence on $\mathcal{A}$'s view of hk. Thus this rejection does not leak any information on hk. □

Lemma 4.30 bounds the probability invalid decryption queries are not rejected.

Lemma 4.30. Consider the IPFE scheme of Fig. 4.7. Recall that BAD denotes the event that, by the end of the post-challenge phase, $\mathcal{A}$ has performed an invalid decryption query to which $\mathcal{C}$ does not answer $\perp$. Assume $\mathcal{H}$ is a δ_{cr}-hard CRHF generator; and $\mathsf{OTS} := (\mathsf{Setup}_{\mathsf{OTS}}, \mathsf{KeyGen}_{\mathsf{OTS}}, \mathsf{Sign}, \mathsf{Verif})$ is a δ_{OTS}-secure $\mathsf{suf\text{-}1\text{-}cma}$ signature scheme. Then denoting q_{dec} an upper bound on the number of decryption queries performed by $\mathcal{A}$ it holds that:

$$\Pr[\textsc{Bad}] \leqslant q_{\mathsf{dec}}(\frac{n_f}{n_f - q_{\mathsf{dec}} + 1} \cdot \delta_{vu} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}).$$

Proof. First observe that only the challenge ciphertext $\boldsymbol{ct}$ differs between Game_0, Game_1 and Game_2. In Game_0 and Game_1 since $\boldsymbol{ct}$ is computed from $c_0 \in \mathcal{L}$, it leaks no further information on the hashing keys $\mathsf{hk}, \mathsf{ehk}$ than that revealed by the public projection keys. We thus bound the probability BAD occurs in Game_2, since this is the scenario where $\mathcal{A}$ has the most information on $\mathsf{hk}, \mathsf{ehk}$.

The proof proceeds in two steps: (1) use the vector universality of eH to upper bound $\Pr[\textsc{Bad}]$ for $q_{\mathsf{dec}} = 1$, i.e. the probability that $\mathcal{A}$'s first invalid decryption query is not rejected; (2) take into account the information leaked by the rejection of such decryption queries. This allows to bound the probability BAD occurs after q_{dec} decryption queries, for $q_{\mathsf{dec}} \geqslant 1$. Since hk is sampled independently from ehk, and has no influence on the latters' value, it suffices to consider the distribution of ehk from $\mathcal{A}$'s view (we ignore that of hk, which is not used for verifying ciphertext integrity). Moreover we consider $\mathcal{A}$'s success probability given $\widehat{\textsf{ehp}}$, which is at least that given $\textsf{ehp}$.

Let $M \in \Delta\mathcal{M}$ be a random variable ($\mathcal{A}$'s possible choices for $\boldsymbol{m}_0 - \boldsymbol{m}_1$); and $\mathbf{B}_M \in \mathcal{R}^{\ell\times\ell}$ be the associated matrix. The current analysis is performed *à posteriori*, which implies $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$ are fixed, so denoting $\boldsymbol{m} := (\boldsymbol{m}_1 - \boldsymbol{m}_0) \in \mathcal{R}^\ell$ we condition $\mathcal{A}$'s success probability on $M = \boldsymbol{m}$. Let $\boldsymbol{b}_1^{\boldsymbol{m}}, \dots, \boldsymbol{b}_\ell^{\boldsymbol{m}}$ denote the rows of $\mathbf{B}_{\boldsymbol{m}}$.

Step 1. Before its first invalid decryption query, $\mathcal{A}$ has access to:

1. the vector $\textsf{ehp}$, whose value is fixed by that of $\widehat{\textsf{ehp}}$;
2. the ciphertext $\boldsymbol{ct}$, which fixes $c_0 \in \mathcal{X}$, $e = \hbar(\textsf{vk}) \in E$; and $\overline{c}_i = \textsf{ehash}(\textsf{ehk}_i, c_0, e)$ for $i \in [\ell]$;
3. key derivation queries; since $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{\ell-1})$ is a basis of $\boldsymbol{m}^\perp$, and any $(\textsf{key}, \boldsymbol{k})$ queried by a valid $\mathcal{A}$ must satisfy $\boldsymbol{k} \in \boldsymbol{m}^\perp$, any such $\boldsymbol{k}$ is a linear combination of vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{\ell-1}$. Thus for some $\boldsymbol{v}_j \in \mathcal{R}^{2a}$, $j \in [\ell-1]$, the information leaked via key derivation queries on $\textsf{ehk}$ is upper bounded by the knowledge of events $\boldsymbol{v}_j = \boldsymbol{b}_j^T \cdot \textsf{ehk}$ for $j \in [\ell-1]$.
4. valid decryption queries, which reveal no more information than $\widehat{\textsf{ehp}}$, and key derivation queries (*cf.* Lemma 4.28).

Now consider the first *invalid* decryption query in performed by $\mathcal{A}$, denoted $(\textsf{decrypt}, (\boldsymbol{ct'}, \textsf{vk}', \sigma'), \boldsymbol{k'})$, to which the challenger answers $\textsf{res}$, where $\boldsymbol{ct'} = (c_0', \boldsymbol{c}', \overline{\boldsymbol{c}}')$ and $e' \leftarrow \hbar(\textsf{vk}')$. As such it satisfies $c_0' \in \widehat{\mathcal{X}} \backslash \widehat{\mathcal{L}}$ and $\boldsymbol{k'} \notin \boldsymbol{m}^\perp$. If we let $\pi := \prod_{i\in[\ell]} (\overline{c}_i')^{k_i'}$, applying the δ_{vu}-vector-universality of $\textsf{eH}$ ensures that, if $(c_0, e) \neq (c_0', e')$, the probability this first invalid decryption query satisfies $\textsf{res} \neq \perp$ is upper bounded by δ_{vu}.

Let us now upper bound the probability that $(c_0, e) = (c_0', e')$. Since $\mathcal{A}$ cannot query the challenge ciphertext to the decryption oracle, it holds that $(\boldsymbol{ct'}, \textsf{vk}', \sigma') \neq (\boldsymbol{ct}, \textsf{vk}, \sigma)$. We consider two cases:

- $\textsf{vk}' = \textsf{vk}$ and $e = e' \bmod n_f$ but $(\boldsymbol{ct}, \sigma) \neq (\boldsymbol{ct'}, \sigma')$. In this case, either $\textsf{OTS.Verif}(\textsf{vk}, \boldsymbol{ct'}, \sigma') = 0$ and the oracle rejects, or $\textsf{OTS.Verif}(\textsf{vk}, \boldsymbol{ct'}, \sigma') = 1$, *i.e.* $\mathcal{A}$ has forged a signature for $\boldsymbol{ct'}$ with verification key $\textsf{vk}$, thus breaking the unforgeability of $\textsf{OTS}$. This occurs with probability $\leqslant \delta_{\textsf{OTS}}$.
- $\textsf{vk}' \neq \textsf{vk}$ but $e = e' \bmod n_f$. In this case we have found a collision for $\hbar$. This occurs with probability $\leqslant \delta_{\textsf{cr}}$.

Thus the probability this first invalid decryption query is not rejected is upper bounded by $\delta_{vu} + \delta_{\textsf{cr}} + \delta_{\textsf{OTS}}$. It now remains to consider the information revealed by each rejected invalid decryption query.

Step 2. Consider an invalid query $(\textsf{decrypt}, (\boldsymbol{ct}, \textsf{vk}, \sigma), \boldsymbol{k})$ which is rejected. Let $(c_0, \boldsymbol{c}, \overline{\boldsymbol{c}}) = \boldsymbol{ct}$ and $\pi \leftarrow \prod_{i\in[\ell]} \overline{c}_i^{k_i}$. If the rejection is due to the $\textsf{OTS}$ verification algorithm, $\mathcal{A}$ learns nothing about $\textsf{ehk}$. Now suppose the rejection is due to $\textsf{ehash}(\boldsymbol{k}^T \cdot \textsf{ehk}, c_0, e) \neq \pi$. Since $c_0 \notin \widehat{\mathcal{L}}$, and $\textsf{eH}$ is $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable, there exist unique $x \in \widehat{\mathcal{L}}$ and $y \in \langle \Upsilon \rangle$ satisfying $c_0 = x \cdot y$. By the homomorphic properties of $\textsf{eH}$ we can write:

$$\textsf{ehash}(\boldsymbol{k}^T \cdot \textsf{ehk}, c_0, e) = \textsf{ehash}(\boldsymbol{k}^T \cdot \textsf{ehk}, x, e) \cdot \textsf{ehash}(\boldsymbol{k}^T \cdot \textsf{ehk}, y, e),$$

where information theoretically, the value of $\textsf{ehash}(\boldsymbol{k}^T \cdot \textsf{ehk}, x, e)$ is already fixed by the projection keys and c_0. Consequently, the rejection of this query rules out one possible value for $\textsf{ehash}(\boldsymbol{k}^T \cdot \textsf{ehk}, y, e) \in F$, and thereby a proportion of $1/n_f$ of the possible values for $\boldsymbol{k}^T \cdot \textsf{ehk}$.

Thus the probability that $\mathcal{A}$'s i^{th} invalid query is not rejected is upper bounded by $(\frac{n_f}{n_f-(i+1)} \cdot \delta_{vu} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$. This allows us to conclude that, after q_{dec} decryption queries:

$$\Pr[\textsc{Bad}] \leqslant q_{\mathsf{dec}}(\frac{n_f}{n_f - q_{\mathsf{dec}} + 1} \cdot \delta_{vu} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}).$$

□

Running Example 1 – Instantiation from DDH

Instantiating the IPFE of Fig. 4.7 with $\mathsf{H}_{\mathsf{ddh}}$ yields the IPFE scheme depicted in Fig. 4.9.

Setting the parameters. We use the output (G, g, q) of the $\mathsf{Gen}_{\mathsf{DL}}$ generator of Definition 2.2, and consider generators $g_0, g_1 \hookleftarrow G$. We also use two CRHF generators $\mathcal{H}_G$ and $\mathcal{H}$, such that $\hbar \hookleftarrow \mathcal{H}_G(1^\lambda)$ maps $\{0,1\}^*$ to G; and $\Gamma \hookleftarrow \mathcal{H}(1^\lambda)$ maps G^3 to $\{0, \dots, q-1\}$. Finally we use a strongly unforgeable OTS scheme $\mathsf{OTS} := (\mathsf{Setup}_{\mathsf{OTS}}, \mathsf{KeyGen}_{\mathsf{OTS}}, \mathsf{Sign}, \mathsf{Verif})$.

The message and key spaces are subsets of $(\mathbf{Z}/q\mathbf{Z})^\ell$. The decryption algorithm recovers $\langle \boldsymbol{k}, \boldsymbol{m} \rangle$ over $\mathbf{Z}/q\mathbf{Z}$ if it is sufficiently small for the discrete logarithm of $g^{\langle \boldsymbol{k}, \boldsymbol{m} \rangle}$ to be efficient. Hashing key coordinates are sampled from $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, as is the encryption randomness.

Corollary 4.31 (of Theorem 4.26)**.** If the DDH problem is hard, the IPFE scheme of Fig. 4.9 is $\mathsf{ind\text{-}fe\text{-}cca}$-secure.

Running Example 2 – Instantiation from HSM-CL

Instantiating the IPFE of Fig. 4.7 with $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ yields the IPFE scheme depicted in Fig. 4.10.

Setting the parameters. We use the same parameters as for the $\mathsf{ind\text{-}fe\text{-}cpa}$-secure scheme of Fig. 4.3. We also use two CRHF generators $\mathcal{H}_{\widehat{G}}$ and $\mathcal{H}$, such that $\hbar \hookleftarrow \mathcal{H}_{\widehat{G}}(1^\lambda)$ maps $\{0,1\}^*$ to $\widehat{G}$; and $\Gamma \hookleftarrow \mathcal{H}(1^\lambda)$ maps $\widehat{G}^2$ to $\{0, \dots, q-1\}$. Finally we use a strongly unforgeable OTS scheme $\mathsf{OTS} := (\mathsf{Setup}_{\mathsf{OTS}}, \mathsf{KeyGen}_{\mathsf{OTS}}, \mathsf{Sign}, \mathsf{Verif})$.

Corollary 4.32 (of Theorem 4.26)**.** If the HSM-CL problem is hard, the IPFE scheme of Fig. 4.10 is $\mathsf{ind\text{-}fe\text{-}cca}$-secure.

Walking Example – Instantiation from DCR

As in the $\mathsf{ind\text{-}fe\text{-}cpa}$ setting, one can build both a projective hash function $\mathsf{H}_{\mathsf{dcr}}$ and an extended projective hash function $\mathsf{eH}_{\mathsf{dcr}}$ from the decisional composite residuosity assumption which satisfy all the properties required in Theorem 4.26 to build $\mathsf{ind\text{-}fe\text{-}cca}$-secure IPFE. The resulting instantiation would very much resemble the HSM-CL based instantiation of running example 2. We do not detail this DCR based instantiation of our generic construction here. However in Section 4.5 we use this instantiation to demonstrate the concrete improvements our proof technique achieves compared to the work of [BBL17]. Indeed, as they also have an instantiation from DCR (which is their only scheme able to decrypt large inner products) the comparison is fair and meaningful.

Running Example 3 – Instantiation from DDH-f

Instantiating the IPFE of Fig. 4.7 with $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ yields the DDH-f based IPFE scheme depicted in Fig. 4.11.

```
$\mathsf{Setup}(1^\lambda, 1^\ell)$:
$(G, g, q) \leftarrow \mathsf{Gen}_{\mathsf{DL}}(1^\lambda)$
$g_0, g_1 \hookleftarrow G$
$\hbar \hookleftarrow \mathcal{H}_G(1^\lambda)$; $\Gamma \hookleftarrow \mathcal{H}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{OTS}}(1^\lambda)$
For $1 \leqslant i \leqslant \ell$ :
    $(\kappa_{0,i}, \kappa_{1,i}\kappa_{2,i}, \kappa_{3,i}, \kappa_{4,i}, \kappa_{5,i}) \hookleftarrow (\mathbf{Z}/q\mathbf{Z})^6$
    $\mathsf{hp}_i \leftarrow g_0^{\kappa_{0,i}} g_1^{\kappa_{1,i}}$
    $\mathsf{ehp}_{0,i} \leftarrow g_0^{\kappa_{2,i}} g_1^{\kappa_{3,i}}$, $\mathsf{ehp}_{1,i} \leftarrow g_0^{\kappa_{4,i}} g_1^{\kappa_{5,i}}$
Return $\mathsf{msk} := (\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3, \boldsymbol{\kappa}_4, \boldsymbol{\kappa}_5)$
    $\mathsf{mpk} := (G, q, g, g_0, g_1, \mathbf{hp}, \mathbf{ehp}_0, \mathbf{ehp}_1, \hbar, \Gamma, \mathsf{pp})$
```

```
$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m})$
If $\boldsymbol{m} \notin \mathcal{M}$, return $\perp$
$(\mathsf{vk}, \mathsf{sk}_{\mathsf{OTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{OTS}}(\mathsf{pp})$
$e \leftarrow \hbar(\mathsf{vk})$
$r \hookleftarrow \mathbf{Z}/q\mathbf{Z}$; let $(x_0, x_1) \leftarrow (g_0^r, g_1^r)$
$\gamma \leftarrow \Gamma(x_0, x_1, e)$
For $1 \leqslant i \leqslant \ell$ :
    $c_i \leftarrow g^{m_i} \mathsf{hp}_i^r$
    $\overline{c}_i \leftarrow (\mathsf{ehp}_{0,i} \mathsf{ehp}_{1,i}^\gamma)^r$
Let $\boldsymbol{ct} := (c_0, \boldsymbol{c}, \overline{\boldsymbol{c}})$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{OTS}}, \boldsymbol{ct})$
Return $(\boldsymbol{ct}, \mathsf{vk}, \sigma)$
```

```
$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k})$
If $\boldsymbol{k} \notin \mathcal{K}$, return $\perp$
$(\mathsf{sk}_0, \mathsf{sk}_1) \leftarrow (\langle \boldsymbol{k}, \boldsymbol{\kappa}_0 \rangle, \langle \boldsymbol{k}, \boldsymbol{\kappa}_1 \rangle) \in \mathbf{Z}$
For $2 \leqslant \mu \leqslant 5$, let $\overline{\mathsf{sk}}_\mu \leftarrow \langle \boldsymbol{k}, \boldsymbol{\kappa}_\mu \rangle \in \mathbf{Z}$
$\mathsf{sk}_{\boldsymbol{k}} := (\mathsf{sk}_0, \mathsf{sk}_1)$;
$\overline{\mathsf{sk}}_{\boldsymbol{k}} := (\overline{\mathsf{sk}}_2, \overline{\mathsf{sk}}_3, \overline{\mathsf{sk}}_4, \overline{\mathsf{sk}}_5)$
Return $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}}, \boldsymbol{k})$
```

```
$\mathsf{Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}}, \boldsymbol{k}), (\boldsymbol{ct}, \mathsf{vk}, \sigma))$
If $\boldsymbol{ct} \notin G^{2\ell+2}$, return $\perp$
If $\mathsf{Verif}(\mathsf{vk}, \boldsymbol{ct}, \sigma) = 0$, return $\perp$
$e \leftarrow \hbar(\mathsf{vk})$; $\gamma \leftarrow \Gamma(x_0, x_1, e)$
If $x_0^{\overline{\mathsf{sk}}_2 + \gamma \overline{\mathsf{sk}}_4} x_1^{\overline{\mathsf{sk}}_3 + \gamma \overline{\mathsf{sk}}_5} \neq \prod_{i \in [\ell]} \overline{c}_i^{k_i}$
    Then return $\perp$
$M \leftarrow (\prod_{i \in [\ell]} c_i^{k_i}) \cdot (x_0^{-\mathsf{sk}_0} x_1^{-\mathsf{sk}_1})$
Return $\log_g(M)$
```

Figure 4.9: Running Example 1 – ind-fe-cca-secure IPFE scheme from the DDH assumption.

Setting the parameters. We use the same parameters as for the ind-fe-cpa-secure scheme of Fig. 4.4. We also use two CRHF generators $\mathcal{H}_{\widehat{G}}$ and $\mathcal{H}$, such that $\hbar \hookleftarrow \mathcal{H}_{\widehat{G}}(1^\lambda)$ maps $\{0,1\}^*$ to $\widehat{G}$; and $\Gamma \hookleftarrow \mathcal{H}(1^\lambda)$ maps $\widehat{G}^3$ to $\{0, \ldots, q-1\}$. Finally we use a strongly unforgeable OTS scheme $\mathsf{OTS} := (\mathsf{Setup}_{\mathsf{OTS}}, \mathsf{KeyGen}_{\mathsf{OTS}}, \mathsf{Sign}, \mathsf{Verif})$.

Corollary 4.33 (of Theorem 4.26)**.** If the DDH-f problem is hard, the IPFE scheme of Fig. 4.11 is ind-fe-cca-secure.

4.5 Efficiency Comparisons

In this section, we first compare the efficiency of our functional encryption schemes computing inner products modulo a prime q (Figs. 4.5 and 4.6) to the schemes of [ALS16], which compute inner products modulo either some prime q (for their LWE based scheme), or modulo an RSA integer N (for their DCR based scheme). We then focus on our ind-fe-cca-secure generic construction, and illustrate the huge efficiency improvements we gain compared to the work of [BBL17]. We do not compare our ind-fe-cpa-secure FE schemes computing inner products in $\mathbf{Z}$, since compared to prior work, efficiency gains here are less significant. The main interest of our constructions of our ind-fe-cpa-secure constructions of Section 4.3 was to smoothly guide the reader to the construction and security proof for the ind-fe-cca-secure construction of Fig. 4.7.

```
$\mathsf{Setup}(1^\lambda, 1^\mu, 1^\ell)$:
    Sample a $\mu$ bit prime $q$
    $\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Gen}(1^\lambda, q)$
    $\hbar \hookleftarrow \mathcal{H}_G(1^\lambda)$; $\Gamma \hookleftarrow \mathcal{H}(1^\lambda)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{OTS}}(1^\lambda)$
    For $1 \leqslant i \leqslant \ell$ :
        Sample $\mathsf{hk}_i, \mathsf{ehk}_{0,i}, \mathsf{ehk}_{1,i} \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma}$
        Let $\mathsf{hp}_i \leftarrow g_q^{\mathsf{hk}_i}$
        Let $(\mathsf{ehp}_{0,i}, \mathsf{ehp}_{1,i}) \leftarrow (g_q^{\mathsf{ehk}_{0,i}}, g_q^{\mathsf{ehk}_{1,i}})$
    Return $\mathsf{msk} := (\mathbf{hk}, \mathbf{ehk}_0, \mathbf{ehk}_1)$ and
    $\mathsf{mpk} := (\mathsf{pp}_{\mathsf{CL}}, q, \mathbf{hp}, \mathbf{ehp}_0, \mathbf{ehp}_1, \hbar, \Gamma, \mathsf{pp})$

$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k})$
    If $\boldsymbol{k} \notin \mathcal{K}$, return $\perp$
    $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \langle \boldsymbol{k}, \mathbf{hk} \rangle$
    $\overline{\mathsf{sk}}_0 \leftarrow \langle \boldsymbol{k}, \mathbf{ehk}_0 \rangle$;
    $\overline{\mathsf{sk}}_1 \leftarrow \langle \boldsymbol{k}, \mathbf{ehk}_1 \rangle$
    Return $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_0, \overline{\mathsf{sk}}_1, \boldsymbol{k})$
```

```
$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m})$
    If $\boldsymbol{m} \notin \mathcal{M}$, return $\perp$
    $(\mathsf{vk}, \mathsf{sk}_{\mathsf{OTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{OTS}}(\mathsf{pp})$;
    $e \leftarrow \hbar(\mathsf{vk})$
    Sample $r \hookleftarrow \mathcal{D}_{\mathbf{Z},\sigma'}$; set $c_0 \leftarrow g_q^r$
    $\gamma \leftarrow \Gamma(c_0, e)$
    For $1 \leqslant i \leqslant \ell$ :
        $c_i \leftarrow f^{m_i} \mathsf{hp}_i^r$
        $\bar{c}_i \leftarrow (\mathsf{ehp}_{0,i} \mathsf{ehp}_{1,i}^{\gamma})^r$
    Let $\boldsymbol{ct} := (c_0, \boldsymbol{c}, \bar{\boldsymbol{c}})$;
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{OTS}}, \boldsymbol{ct})$
    Return $(\boldsymbol{ct}, \mathsf{vk}, \sigma)$

$\mathsf{Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_0, \overline{\mathsf{sk}}_1, \boldsymbol{k}), (\boldsymbol{ct}, \mathsf{vk}, \sigma))$
    If $\boldsymbol{ct} \notin \widehat{G}^{2\ell+1}$, return $\perp$
    If $\mathsf{Verif}(\mathsf{vk}, \boldsymbol{ct}, \sigma) = 0$, return $\perp$
    $e \leftarrow \hbar(\mathsf{vk})$; $\gamma \leftarrow \Gamma(c_0, e)$
    If $c_0^{\overline{\mathsf{sk}}_0 + \gamma \overline{\mathsf{sk}}_1} \neq \prod_{i \in [\ell]} \bar{c}_i^{k_i}$, return $\perp$
    $M \leftarrow (\prod_{i \in [\ell]} c_i^{k_i}) \cdot (c_0^{-\mathsf{sk}_{\boldsymbol{k}}})$
    If $M \notin F$, return $\perp$
    $\mathsf{sol} \leftarrow \mathsf{Solve}(M)$
    If $\mathsf{sol} \geqslant q/2$, return $(\mathsf{sol} - q)$
    Else return $\mathsf{sol}$
```

Figure 4.10: Running Example 2 – ind-fe-cca-secure IPFE from the HSM-CL assumption.

4.5.1 Modular IPFE Secure against Passive Adversaries

We put forth two generic constructions of FE for the evaluation of inner products modulo q. Both schemes are based on variants of Elgamal in the same group $\widehat{G}$ and both sample their master secret keys from Gaussian distributions with the same standard deviation. As a result their asymptotic complexities are the same. We compare an implementation of our HSM-CL based IPFE mod q of Fig. 4.5 within the class group of an imaginary quadratic field to the LWE based variant of [ALS16] (in the upcoming paragraph); and to their Paillier variant (in Table 4.1). These are the most relevant comparisons since DDH variants do not allow a full recovery of large inner products over $\mathbf{Z}/q\mathbf{Z}$.

Comparison with the LWE based scheme of [ALS16]. Parameter choices for lattice-based cryptography are complex, indeed [ALS16] do not provide a concrete set of parameters. This being said, using [ALS16, Theorem 3], and setting $\log q = \lambda$ as in Table 4.1, we give rough bit sizes for their LWE based FE scheme computing inner products over $\mathbf{Z}/q\mathbf{Z}$. Basic elements are integers modulo $\bar{q}$, where $\bar{q}$ is a parameter of the scheme which must be of size ℓ as $\bar{q} \approx 2^\ell$ is required for security to hold. The largest component in the master public key

```
$\mathsf{Setup}(1^\lambda, 1^\mu, 1^\ell)$:
Sample a $\mu$ bit prime $q$
$\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Gen}(1^\lambda, q)$
Sample $\alpha \hookleftarrow \mathscr{D}_{\mathbf{Z},\sigma'}$; let $h \leftarrow g^\alpha$
$\hslash \hookleftarrow \mathscr{H}_G(1^\lambda)$; $\Gamma \hookleftarrow \mathscr{H}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{OTS}}(1^\lambda)$
For $1 \leqslant i \leqslant \ell$ :
    $(\kappa_{0,i}, \kappa_{1,i}\kappa_{2,i}, \kappa_{3,i}, \kappa_{4,i}, \kappa_{5,i}) \hookleftarrow \mathscr{D}_{\mathbf{Z}^6,\sigma}$
    $\mathsf{hp}_i \leftarrow g^{\kappa_{0,i}} h^{\kappa_{1,i}}$
    $\mathsf{ehp}_{0,i} \leftarrow g^{\kappa_{2,i}} h^{\kappa_{3,i}}$
    $\mathsf{ehp}_{1,i} \leftarrow g^{\kappa_{4,i}} h^{\kappa_{5,i}}$
Return:
$\mathsf{msk} := (\boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3, \boldsymbol{\kappa}_4, \boldsymbol{\kappa}_5)$
$\mathsf{mpk} := (\mathsf{pp}_{\mathsf{CL}}, h, q, \mathbf{hp}, \mathbf{ehp}_0, \mathbf{ehp}_1, \hslash, \Gamma, \mathsf{pp})$
```

```
$\mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{k})$
If $\boldsymbol{k} \notin \mathscr{K}$, return $\perp$
Let $(\mathsf{sk}_0, \mathsf{sk}_1) \leftarrow (\langle \boldsymbol{k}, \boldsymbol{\kappa}_0 \rangle, \langle \boldsymbol{k}, \boldsymbol{\kappa}_1 \rangle)$
For $2 \leqslant \mu \leqslant 5$ :
    Let $\overline{\mathsf{sk}}_\mu \leftarrow \langle \boldsymbol{k}, \boldsymbol{\kappa}_\mu \rangle \in \mathbf{Z}$
$\mathsf{sk}_{\boldsymbol{k}} := (\mathsf{sk}_0, \mathsf{sk}_1)$;
$\overline{\mathsf{sk}}_{\boldsymbol{k}} := (\overline{\mathsf{sk}}_2, \overline{\mathsf{sk}}_3, \overline{\mathsf{sk}}_4, \overline{\mathsf{sk}}_5)$
Return $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}}, \boldsymbol{k})$
```

```
$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{m})$
If $\boldsymbol{m} \notin \mathcal{M}$, return $\perp$
$(\mathsf{vk}, \mathsf{sk}_{\mathsf{OTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{OTS}}(\mathsf{pp})$;
$e \leftarrow \hslash(\mathsf{vk})$
$r \hookleftarrow \mathscr{D}_{\mathbf{Z},\sigma'}$;
$(x_0, x_1) \leftarrow (g^r, h^r)$
$\gamma \leftarrow \Gamma(x_0, x_1, e)$
For $1 \leqslant i \leqslant \ell$ :
    $c_i \leftarrow f^{m_i} \mathsf{hp}_i^r$
    $\overline{c}_i \leftarrow (\mathsf{ehp}_{0,i} \mathsf{ehp}_{1,i}^\gamma)^r$
Let $\boldsymbol{ct} := (c_0, \boldsymbol{c}, \overline{\boldsymbol{c}})$;
Let $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{OTS}}, \boldsymbol{ct})$
Return $(\boldsymbol{ct}, \mathsf{vk}, \sigma)$
```

```
$\mathsf{Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}}, \boldsymbol{k}), (\boldsymbol{ct}, \mathsf{vk}, \sigma))$
If $\boldsymbol{ct} \notin \widehat{G}^{2\ell+2}$, return $\perp$
If $\mathsf{Verif}(\mathsf{vk}, \boldsymbol{ct}, \sigma) = 0$, return $\perp$
$e \leftarrow \hslash(\mathsf{vk})$; $\gamma \leftarrow \Gamma(x_0, x_1, e)$
If $x_0^{\overline{\mathsf{sk}}_2 + \gamma \overline{\mathsf{sk}}_4} x_1^{\overline{\mathsf{sk}}_3 + \gamma \overline{\mathsf{sk}}_5} \neq \prod_{i \in [\ell]} \overline{c}_i^{k_i}$
    Then return $\perp$
$M \leftarrow (\prod_{i \in [\ell]} c_i^{k_i}) \cdot (x_0^{-\mathsf{sk}_0} x_1^{-\mathsf{sk}_1})$
If $M \notin F$, return $\perp$
$\mathsf{sol} \leftarrow \mathsf{Solve}(M)$
If $\mathsf{sol} \geqslant q/2$, return $(\mathsf{sol} - q)$
Else return $\mathsf{sol}$
```

Figure 4.11: Running Example 3 – ind-fe-cca-secure IPFE from the DDH-f assumption.

mpk consists of $\lambda^2\ell^3$ elements, so mpk is of size greater than $\lambda^2\ell^4$. The component $z_{\boldsymbol{x}}$ in secret keys is the product of a vector from $(\mathbf{Z}/q\mathbf{Z})^\ell$ with a matrix, which yields a secret key vector made up of $\lambda\ell^2$ inner products, where each inner product is of size $\ell\lambda$. Thus these keys are of size $\lambda^2\ell^3$. Finally ciphertexts consist of $\lambda\ell^2$ elements, and are thus of size greater than $\lambda\ell^3$. As a result, although it may be hard to compare the complexities in λ, for a fixed security level, the complexity in ℓ for all the parameters of the LWE based scheme is in ℓ^3 or ℓ^4 whereas we are linear in ℓ as one can see in Table 4.1. For example, for $\lambda = 128, \ell = 100$, their decryption keys are of approximately 2^{34} bits vs. 13852 bits in our instantiation.

Comparison with the DCR based scheme of [ALS16]. To instantiate the HSM-CL based IPFE of Fig. 4.5, we use the instantiation for the CL framework which is detailed in Section 3.1.2; bit sizes of elements in Table 4.1 are chosen as detailed in Section 2.6 (in particular see Table 2.1). Ciphertexts in both protocols consist of $\ell + 1$ group elements. To simplify the comparison we consider linearly independent decryption key queries (ignoring the vectors in $\mathbf{Z}^\ell$). As a result, we have, for our scheme, the inner product of a vector from $(\mathbf{Z}/q\mathbf{Z})^\ell$ with a vector sampled from a discrete Gaussian with standard deviation greater than $\sqrt{\lambda}q\tilde{s}(\sqrt{\ell}q)^{\ell-1}$ over $\mathbf{Z}^\ell$ vs. the inner product of a vector of $(\mathbf{Z}/N\mathbf{Z})^\ell$ with a vector sampled from a discrete Gaussian with standard deviation greater than $\sqrt{\lambda}(\sqrt{\ell}N)^{\ell+1}$ over $\mathbf{Z}^\ell$. We note that using larger message spaces would be more favorable to their Paillier based scheme. Our protocols are the most suited for double or quadruple precision computations, where a 128 bits message space is large enough, since the DCR-based construction from [ALS16] would add a large overhead cost, while constructions from DDH could not decrypt the result.

	$\lambda = 112$		$\lambda = 128$	
bit size	HSM-CL	DCR	HSM-CL	DCR
msk	$\ell(112\ell + 687)$	$\ell(2048(\ell + 1) + 3)$	$\ell(128\ell + 928)$	$\ell(3072(\ell + 1) + 4)$
ciphertext	$1572(\ell + 1)$	$4096(\ell + 1)$	$2084(\ell + 1)$	$6144(\ell + 1)$
enc. expo.	687	2046	928	3070
dec. expo.	$\ell + 112(\ell + 1) + 684$	$\ell + 2048(\ell + 2)$	$\ell + 128(\ell + 1) + 928$	$\ell + 3072(\ell + 2)$

* ignoring an additive term $(\ell \pm 1)\log(\sqrt{\ell})$

Table 4.1: Comparing bit sizes of our modular HSM-CL-based IPFE of Fig. 4.5 to the DCR scheme of [ALS16]

In terms of timings, the exponents involved in our (multi-)exponentiations (for encryption and decryption) are significantly smaller than those in [ALS16], and the group size is also smaller. Indeed, the encryption of Paillier's variant involves $(\ell + 1)$ exponentiations to the power a $(|N| - 2)$-bit integer modulo N^2, whereas our protocol involves one exponentiation to the power a $|\sigma'|$-bit integer in $Cl(q^3\tilde{q})$, where $\sigma' > \tilde{s}\sqrt{\lambda}$ and ℓ (multi-)exponentiations whose maximum exponent size is also $|\sigma'|$. Decryptions involve respectively a multi-exponentiation whose maximum exponent size is lower than $\ell\sigma N = \ell\sqrt{\lambda}(\sqrt{\ell}N)^{\ell+1}N$ for [ALS16] and $\ell q\sigma = \ell q\sqrt{\lambda}q\tilde{s}(\sqrt{\ell}q)^{\ell-1}$ for our protocol.

Timings are computed theoretically as explained at the end of Section 2.6. Table 4.2 shows that our instantiation from HSM-CL fares better for decryption, while encryption is faster with an implementation from DCR. In the table we set $\ell = 100$, but as all parameters depend linearly in ℓ, one can extrapolate timings for other values of ℓ. We gain firstly from the fact that we can use smaller parameters for the same security level and secondly, because our security reductions allow to replace N^ℓ with q^ℓ in the derived keys. Thus the gain is not only in the constants and our scheme becomes more and more interesting as the security level and the dimension ℓ increase.

	$\lambda = 112, \ell = 100$		$\lambda = 128, \ell = 100$		$\lambda = 192, \ell = 100$	
	HSM-CL	DCR	HSM-CL	DCR	HSM-CL	DCR
secret key bitsize	12099	208999	13956	313448	21307	783464
encryption time	3.8s	**0.9s**	6.7s	**3s**	**26.7s**	32.2s
decryption time	**0.7s**	0.9s	**1s**	3.1s	**3.2s**	33.2s

Table 4.2: Timings: modular IPFE from HSM-CL [CLT18a] *vs.* IPFE from DCR [ALS16]

4.5.2 IPFE Secure against Active Adversaries

We here illustrate the huge gains in efficiency provided by our security proof for Theorem 4.26, compared to the work of [BBL17]. For all the upcoming comparisons we ommit the negligible probabilities of breaking the CRHFs and one time signatures (these are of little interest to our comparaison, their cost is marginally larger in [BBL17] due to a larger number of instances of the EPHFs).

Instantiations from DDH

We compare our DDH-based IPFE to that of [BBL17]. To instantiate the generic protocol of Section 4.4 from DDH we use $\mathsf{H}_{\mathsf{ddh}}$ of running example 1. This PHF (first introduced in [CS02]) is also that used by [BBL17]. Since hashing keys are sampled from $\mathcal{U}((\mathbf{Z}/q\mathbf{Z})^2)$, $\mathsf{H}_{\mathsf{ddh}}$ is 0-vector smooth, while $\mathsf{eH}_{\mathsf{ddh}}$ is $1/q$-vector universal. Consequently our security proof upper bounds the advantage of any $\mathsf{ind}\text{-}\mathsf{fe}\text{-}\mathsf{cca}$ adversary by $\delta_{\mathsf{us}} \leqslant \delta_{\mathsf{DDH}} + q_{\mathsf{dec}}(q - q_{\mathsf{dec}} + 1)^{-1}$, whereas, the [BBL17] proof technique bounds the adversary's advantage by: $\delta_{\mathsf{bbl}} \leqslant \delta_{\mathsf{DDH}} + q_{\mathsf{dec}} q^{-\nu} |\Delta\mathcal{M}|$, where $|\Delta\mathcal{M}| = (4(q(2\ell)^{-1})^{1/2})^{\ell}$ depends on the message space, and ν corresponds to the number of repeated instances of the hash function $\mathsf{eH}_{\mathsf{ddh}}$ required for $q^{-\nu}|\Delta\mathcal{M}|$ to be negligible, i.e. $\nu := \lfloor \log_2(|\Delta\mathcal{M}|) \cdot \lambda^{-1} + 1 \rfloor$. Thus our proof technique allows to reduce decryption keys by $(\nu - 1)$ ring elements, ciphertexts and public keys by $\ell(\mu - 1)$ group elements, while ensuring the same security guarantees.

In Table 4.3 we compare ciphertext and decryption key sizes needed to guarantee equivalent security levels for [BBL17] and for our work, for $\ell := 100$. We instantiate the DH group from elliptic curves to estimate group element sizes. Regarding computational complexity, encryption requires $\ell(\nu + 1)$ exponentiations in G while decryption requires $2\nu + 1$ multi-exponentiations in G. For the parameter sizes of Table 4.3, this yields a speed-up factor of 25.

Table 4.3: Comparing our IPFE scheme and that of [BBL17] from DDH

	$\lambda = 112$		$\lambda = 128$	
size	this work	[BBL17]*	this work	[BBL17]*
q (bits)	224	224	256	256
ν	1	50	1	50
decryption key $(\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}})$ (kB)	**0.17**	5.7	**0.2**	6.5
ciphertext (kB)	**5.6**	142.8	**6.4**	163.2

* For message space a subset of $(\mathbf{Z}/q\mathbf{Z})^{\ell}$, $|\Delta\mathcal{M}| = (4(\frac{q}{2\ell})^{1/2})^{\ell}$, $\nu = 1 + \frac{\log_2(|\Delta\mathcal{M}|)}{\lambda}$ and $\ell = 100$.

Instantiations from HSM-CL vs. DCR.

We here compare our HSM-CL-based IPFE (Fig. 4.10), which to lighten the upcomming discussion we call $\Sigma_{\mathsf{hsm-cl}}$ using class groups to both a DCR instantiation of our IPFE and the DCR variant of [BBL17], denoted respectively Σ_{dcr} and Σ_{bbl}.

We first observe that ciphertexts in $\Sigma_{\mathsf{hsm-cl}}$ and Σ_{dcr} consist of far less group elements, as explained hereafter. Let M_z denote an upper bound on the infinite norm of message vectors; $|\Delta\mathcal{M}| := (4M_z)^{\ell}$; as in the instantiation from DDH, ν corresponds to the number of repeated instances of the EPHF required to thwart $|\Delta\mathcal{M}|$ in the adversary's advantage. For Σ_{bbl} one must choose $\nu > \lambda + \log_2(2q_{\mathsf{dec}}|\Delta\mathcal{M}|)$ (this is the bound they give), whereas we can set $\nu = 1$

(for both $\Sigma_{\mathsf{hsm-cl}}$ and Σ_{dcr}). For all three schemes, ciphertexts consist of $((1+\nu)\ell+1)$ group elements. Regarding decryption keys, in $\Sigma_{\mathsf{hsm-cl}}$ they consist of three inner products between a vector from $(\mathbf{Z}/M_z\mathbf{Z})^\ell$ with a vector sampled from $\mathcal{D}_{\mathbf{Z}^\ell,\sigma}$, where $\sigma > \sqrt{2\lambda}q^{3/2}\tilde{s}$, whereas in Σ_{dcr} and Σ_{bbl}, they consist in one inner product of a vector of $(\mathbf{Z}/M_z\mathbf{Z})^\ell$ with a vector sampled in $\{0,\dots,\lfloor MN^2/4\rfloor\}^\ell$, and 2ν inner products of a vector of $(\mathbf{Z}/M_z\mathbf{Z})^\ell$ with a vector sampled in$\{0,\dots,\lfloor \nu MN^2/2\rfloor\}$, where M is chosen s.t. M_z/M is negligible.

We note that with HSM-CL, one has $M_z := (\frac{q}{2\ell})^{1/2}$, whereas with DCR, $M_z := (\frac{N}{2\ell})^{1/2}$. For a fair comparison however, to compute bit sizes in Table 4.4, we use the same bound $M_z = (\frac{2^\lambda}{2\ell})^{1/2}$ on the size of message and key components for both schemes. As a final note, comparing $\Sigma_{\mathsf{hsm-cl}}$ and Σ_{dcr} (both resulting from our construction), we see that an instantiation from DCR yields larger ciphertexts and decryption keys. For standard levels of security, Σ_{dcr} yields faster encryption and decryption than $\Sigma_{\mathsf{hsm-cl}}$, however for higher levels of security (192 bits and beyond), this trend is inverted. To compute bit sizes in Table 4.4 we let $\ell := 100$.

	$\lambda = 112$			$\lambda = 128$		
	HSM-CL	DCR		HSM-CL	DCR	
size	this work	(our proof)	[BBL17]	this work	(our proof)	[BBL17]
ν	1	1	5551	1	1	6367
group elt. (bits)	1572	4096	4096	2084	6144	6144
$(\mathsf{sk}_{\boldsymbol{k}}, \overline{\mathsf{sk}}_{\boldsymbol{k}})$ (kB)	**0.3**	0.9	3 168	**0.4**	1.3	5315
ciphertext (kB)	**39**	102.9	284 262	**52**	154.4	489 062
Enc. (sec.)	9.5	**3.4**	9962	16.13	**12.04**	38 342
Dec. (sec.)	0.16	**0.05**	203	0.27	**0.19**	804

Table 4.4: Our ind-fe-cca-secure IPFE from HSM-CL and DCR vs. the DCR scheme of [BBL17]

4.6 Applications and Perspectives for Future Work

4.6.1 Application to Non Zero Inner Product Encryption

We here briefly discuss a concrete application for ind-fe-cca-secure IPFE, namely the construction of non zero inner product encryption secure against active adversaries. This in turn allows to instantiate broadcast encryption schemes allowing for efficient revocation of users' ability to decrypt, and where the collusion and misbehaviour of users does not compromise the scheme's security.

Consider a positive integer λ. Non zero inner product encryption (NIPE), introduced in [KSW08], is a special from of functional encryption, which consists of the following algorithms:

- $\mathsf{N.Setup}(1^\lambda)$: a PPT algorithm which on input a security parameter λ and an integer ℓ outputs a master key pair $(\mathsf{msk}, \mathsf{mpk})$;
- $\mathsf{N.Enc}(\mathsf{mpk}, m, \boldsymbol{x})$: a PPT encryption algorithm which takes as input the public key mpk, a message m and a vector $\boldsymbol{x}$, computes a ciphertext $c_{\boldsymbol{x}}$ associated to $\boldsymbol{x}$, and outputs $ct := (c_{\boldsymbol{x}}, \boldsymbol{x})$;
- $\mathsf{N.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \boldsymbol{k})$: a PT key derivation algorithm which on input both master keys and a vector $\boldsymbol{k}$ computes a decryption key $\mathsf{sk}_{\boldsymbol{k}}$ associated to $\boldsymbol{k}$, and outputs $(\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k})$;
- $\mathsf{N.Dec}(\mathsf{mpk}, (\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k}), (c_{\boldsymbol{x}}, \boldsymbol{x}))$: a DPT decryption algorithm which on input the public key mpk, a decryption key $(\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k})$ associated to $\boldsymbol{k}$, and a ciphertext $(c_{\boldsymbol{x}}, \boldsymbol{x})$ encrypting m

associated to $\boldsymbol{x}$, outputs either the message m if $\langle \boldsymbol{x}, \boldsymbol{k} \rangle \neq 0$ (and the ciphertext is honestly generated), otherwise it outputs $\perp$.

We do not formally define adaptive security against chosen plaintext (ind-nipe-cpa) and chosen ciphertext (ind-nipe-cca) attacks for NIPE schemes, these notions are quite intuitive given the definitions provided in Section 4.1.2.

Recently Katsumata and Yamada [KY19] devised a generic yet simple construction allowing to build NIPE schemes which are ind-nipe-cpa-secure from IPFE schemes attaining the analogue ind-fe-cpa-security notion. In a nutshell, the construction works as follows: N.Setup is the IPFE Setup algorithm; N.Enc on input m and a vector $\boldsymbol{x}$ runs the IPFE algorithm $\mathsf{Enc}(\mathsf{mpk}, m\boldsymbol{x})$ to obtain $c_{\boldsymbol{x}}$, it then outputs $(c_{\boldsymbol{x}}, \boldsymbol{x})$; N.KeyDer is exactly KeyDer of the IPFE, outputting $(\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k})$. Finally N.Dec, on input mpk, $(\mathsf{sk}_{\boldsymbol{k}}, \boldsymbol{k})$, and $(c_{\boldsymbol{x}}, \boldsymbol{x})$ computes $z \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{k}}, c)$. Notice that if the protocol was executed correctly $z = m\langle \boldsymbol{x}, \boldsymbol{k} \rangle$; and since $\boldsymbol{k}$ and $\boldsymbol{x}$ are part of the decryption key and ciphertext respectively, one can check whether $\langle \boldsymbol{x}, \boldsymbol{k} \rangle = 0$; if so, or if $z = \perp$, N.Dec outputs $\perp$, else it outputs $z/\langle \boldsymbol{x}, \boldsymbol{k} \rangle$.

Katsumata et al. demonstrate that if there exists an adversary which breaks the ind-nipe-cpa-security of the NIPE, one can build another algorithm breaking the ind-fe-cpa-security of the underlying IPFE. The reduction is straightforward, and can easily be extended to the CCA setting. Indeed, using analogue notions of valid and invalid decryption queries to those used to prove security of Theorem 4.26, observe that any valid decryption query reveals no further information than that available in a chosen plaintext attack, whereas invalid decryption queries – which could leak sensitive information – would also be invalid for the underlying IPFE, and as such would be rejected. Thus our ind-fe-cca-secure IPFE yields for free a construction for ind-fe-cca-secure secure NIPE.

As demonstrated in [AL10], NIPE schemes can in turn be used to build identity-based revocation (IBR) schemes, a type of broadcast encryption scheme allowing for efficient revocation of users' ability to decrypt. Strengthening the security of NIPE schemes to deal with active adversaries allows to devise IBR schemes where the collusion and misbehaviour of users does not compromise the scheme's security.

4.6.2 Simulation Based Security

There exist two models in which one can prove security for functional encryption schemes. The first is the indistinguishability based model, which we have considered so far, where security holds if the adversary cannot distinguish the encryption of two messages of its choosing. This is the security model which was traditionally used to prove security of identity based encryption schemes [Sha84, Coc01, BF01], attribute based encryption schemes [SW05], or searchable encryption schemes [BDOP04], all of which were forerunners to the emergence of functional encryption [AL10, LOS^{+}10, OT10], formally defined in [BSW11, O'N10], which encompasses all of these primitives.

The second security model appeared comparatively late, with the work of [BSW11], in which they observed that for *general functionalities*, the indistinguishability based model is not sufficient for proving security of functional encryption, as it fails to rule out intuitively insecure systems. This lead to a definition of security in the stronger simulation based security model [BSW11, O'N10] (so strong in fact, that for some functionalities, it is impossible to achieve [BSW11, AGVW13]). This is the alternative model to game based definitions presented in Section 2.3.2. It was shown by Gorbunov et al. [GVW12] that adaptive simulation based security (sim-fe-cpa) implies adaptive indistinguishability based security, while O'Neill [O'N10] demonstrated that for a certain class of functions including the inner product function, $\mathsf{ind\text{-}fe\text{-}cpa}$-secure schemes are also $\mathsf{sim\text{-}fe\text{-}cpa}$-secure in the restricted model where the adversary makes all its' key requests *before* seeing the challenge ciphertext.

Though our $\mathsf{ind\text{-}fe\text{-}cpa}$-secure generic construction of Section 4.3, as is, does not attain simulation based security, in Section 4.6.2 we discuss sufficient additional requirements on the underlying PHFs to do so, at least against passive adversaries. For this discussion to make sense, we here provide the definition of simulation based security against passive adversaries for FE [BSW11]. As we have not studied the question of security against active adversaries in the simulation based model, defining the security model is unnecessary.

Intuitively, to attain simulation-based security, one must exhibit a PPT simulator $\mathcal{S}$, such that the view of any PPT adversary $\mathcal{A}$ can be simulated by $\mathcal{S}$ – which does not see the challenge message m^* – but is given access to pairs $\langle \boldsymbol{m^*}, \boldsymbol{k}_i \rangle$ where the $\boldsymbol{k}_i \in \mathcal{K}$ are the vectors for which $\mathcal{A}$ queries keys.

The real/ideal experiments. For $\lambda \in \mathbf{N}$ we denote by $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathcal{A}}(\lambda)$, $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathcal{A}}(\lambda)$ the random variables defined via the following experiments involving the scheme FE, the adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$, and a PPT simulator $\mathcal{S} = (\mathsf{Setup}^*, \mathsf{KeyDer}^*_0, \mathsf{Enc}^*, \mathsf{KeyDer}^*_1)$:

$\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathcal{A}}(\lambda)$

1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $\mathcal{A}_1(1^\lambda, \mathsf{mpk})$ adaptively issues key queries (key, k) where $k \in \mathcal{K}$ and receives $sk_k \leftarrow \mathsf{KeyDer}\ (\mathsf{msk}, k)$.
3. $\mathcal{A}_1$ outputs a message $m^* \in \mathcal{M}$ and a state st.
4. Let $c \leftarrow \mathsf{Enc}(\mathsf{mpk}, m^*)$, send $(\mathsf{mpk}, c, \mathsf{st})$ to $\mathcal{A}_2$.
5. $\mathcal{A}_2$ adaptively issues queries as did $\mathcal{A}_1$ in step 2.
6. Finally $\mathcal{A}_2$ outputs α.

$\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathcal{A}}(\lambda)$

1. $(\mathsf{mpk}^*, \mathsf{msk}^*) \leftarrow \mathsf{Setup}^*(1^\lambda)$.
2. $\mathcal{A}_1(1^\lambda, \mathsf{mpk}^*)$ adaptively issues queries (key, k) with $k \in \mathcal{K}$ and receives:
   $sk_k \leftarrow \mathsf{KeyDer}^*_0(\mathsf{msk}^*, k)$.
   Let $\{k_1, \ldots, k_{q_{\mathsf{key}}}\}$ denote the queried keys.
3. $\mathcal{A}_1$ outputs a message $m^* \in \mathcal{M}$ and a state st.
   Let $L_{\mathsf{ideal}} := \{(k_i, F(m^*, k_i), sk_{k_i})\}_{i=1}^{q_{\mathsf{key}}}$.
4. Let $(c^*, \mathsf{st}') \leftarrow \mathsf{Enc}^*(\mathsf{mpk}^*, \mathsf{msk}^*, L_{\mathsf{ideal}}, 1^{|m^*|})$.
   Send $(\mathsf{mpk}^*, c^*, \mathsf{st}')$ to $\mathcal{A}_2$.
5. $\mathcal{A}_2$ adaptively issues queries (key, k) where $k \in \mathcal{K}$ and receives $sk_k \leftarrow \mathsf{KeyDer}^*_1(\mathsf{msk}^*, \mathsf{st}', k)$.
6. Finally $\mathcal{A}_2$ outputs α.

Definition 4.34. An FE scheme FE for functionality F over a message space $\mathcal{M}$, and a key space $\mathcal{K}$ is simulation-secure against adaptive chosen plaintext attacks ($\mathsf{sim\text{-}fe\text{-}cpa}$) if there exists a PPT simulator $\mathcal{S}$, such that for any PPT adversary $\mathcal{A}$, it holds that:

$$\mathsf{Adv}^{\mathsf{sim\text{-}fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Big| \Pr\big[\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1\big] - \Pr\big[\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1\big] \Big| = \mathsf{negl}(\lambda).$$

Attaining Simulation Based Security

In the following we consider the IPFE construction of Fig. 4.1 which results from a $(\mathcal{R}, a, f, n_f, \ell, \mathcal{M}, \mathcal{K}, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_{vs})$-$\mathsf{pip\text{-}safe}$ projective hash function H. Generalising techniques due to [Wee17, ALMT20a], we here give intuition for additional properties required of the underlying PHF which are sufficient to render the IPFE construction of Fig. 4.1 $\mathsf{sim\text{-}fe\text{-}cpa}$-secure.

We first extend the notion of invalid ciphertexts for PKE (*cf.* Definition 3.21), to IPFE as follows.

Notation 4.35. For a vector of hashing keys $\mathsf{\mathbf{hk}}$, we call an *invalid ciphertext* encrypting $\boldsymbol{m} \in \mathcal{M}$ a ciphertext $\boldsymbol{ct} := (c_0, \boldsymbol{c})$ where $c_0 \in \mathcal{X} \backslash \mathcal{L}$, and $\boldsymbol{c} := \mathsf{hash}(\mathsf{\mathbf{hk}}, c_0) \cdot f^{\boldsymbol{m}}$. Note that for any $\boldsymbol{k} \in \mathcal{K}$, decrypting $\boldsymbol{ct}$ with $\mathsf{sk}_{\boldsymbol{k}} \leftarrow \boldsymbol{k}^T \cdot \mathsf{\mathbf{hk}}$ yields $\langle \boldsymbol{m}, \boldsymbol{k} \rangle \in \mathcal{R}$.

We follow the lines of [Wee17, ALMT20a] and assume $\mathcal{A}$ makes $\ell - 1$ key derivation queries for linearly independent vectors $\{\boldsymbol{k}_i\}_{i \in [\ell-1]}$. Recall that we must exhibit a PPT simulator $\mathcal{S}$, simulating the view of adversary $\mathcal{A}$ without seeing the challenge message m^*, but given access to pairs $\langle \boldsymbol{m}^*, \boldsymbol{k}_i \rangle$ where the $\boldsymbol{k}_i \in \mathcal{K}$ are the vectors queried by $\mathcal{A}$. Precisely $\mathcal{S}$ must implement the following algorithms:

1. Setup^*: Here $\mathcal{S}$ sets up master keys $(\mathsf{mpk}^*, \mathsf{msk}^*)$, where the distribution followed by mpk^* must be indistinguishable to that output by the real Setup algorithm. To this end $\mathcal{S}$ simply samples a vector of hashing keys $\mathsf{\mathbf{hk}} \in K_{hk}^{\ell}$, running the hashkg algorithm ℓ times, computes the associated vector of public projection keys $\mathsf{\mathbf{hp}} \leftarrow \mathsf{projkg}(\mathsf{\mathbf{hk}})$, and sets $\mathsf{mpk}^* := \mathsf{\mathbf{hp}}$ and $\mathsf{msk}^* := \mathsf{\mathbf{hk}}$.

2. KeyDer_0^*: Here $\mathcal{S}$ must answer 'pre-challenge' key derivation queries (i.e. queries that are made before $\mathcal{A}$ gets the (simulated) challenge ciphertext). For each key derivation query $\boldsymbol{k}$, $\mathcal{S}$ sends $\boldsymbol{k}^T \cdot \mathsf{\mathbf{hk}}$ to $\mathcal{A}$. Let K_{pre} denote the set of vectors $\boldsymbol{k} \in \mathcal{K}$ queried by $\mathcal{A}$ during this phase.

3. Enc^*: Here $\mathcal{S}$ must output a simulated challenge ciphertext. Since $\mathcal{S}$ does not know the challenge message $\boldsymbol{m}^* \in \mathcal{M}$, it encrypts a dummy message $\bar{\boldsymbol{m}}$ instead. This $\bar{\boldsymbol{m}}$ must be consistent with $\boldsymbol{m}^*$ from $\mathcal{A}$'s view. This means that $\forall \boldsymbol{k} \in K_{\mathsf{pre}}$, it must hold that $\langle \boldsymbol{m}^*, \boldsymbol{k} \rangle = \langle \bar{\boldsymbol{m}}, \boldsymbol{k} \rangle \in \mathcal{R}$. $\mathcal{S}$ computes an invalid ciphertext for $\bar{\boldsymbol{m}}$, i.e., for $(x, w) \in \mathsf{R}$ and $b \in (\mathbf{Z}/n_f\mathbf{Z})^*$, $y \leftarrow \Upsilon^b$, the simulated challenge ciphertext is $\bar{\boldsymbol{ct}} := (x \cdot y, \mathsf{hash}(\mathsf{\mathbf{hk}}, x \cdot y) \cdot f^{\bar{\boldsymbol{m}}})$. Note that it is important the ciphertext be invalid, since, as we shall see later, such a ciphertext can be seen as an encryption of any $\boldsymbol{m}' \in \mathcal{M}$, for another hashing key $\mathsf{\mathbf{hk}}'$ (fixed given $\bar{\boldsymbol{ct}}$ and $\boldsymbol{m}'$), and satisfying $\mathsf{projkg}(\mathsf{\mathbf{hk}}) = \mathsf{projkg}(\mathsf{\mathbf{hk}}')$.

4. KeyDer_1^*: Finally for any 'post-challenge' key derivation query $\boldsymbol{k}$, and given the corresponding value $z_{\boldsymbol{k}}^* := \langle \boldsymbol{m}^*, \boldsymbol{k} \rangle$, $\mathcal{S}$ must send a decryption key $\mathsf{sk}_{\boldsymbol{k}}'$ to $\mathcal{A}$ satisfying $\mathsf{Dec}(\mathsf{mpk}^*, \mathsf{sk}_{\boldsymbol{k}}', \bar{\boldsymbol{ct}}) = z_{\boldsymbol{k}}^*$. To this end, as in [ALMT20a], one embeds the value $z_{\boldsymbol{k}}^*$ into the functional decryption key $\mathsf{sk}_{\boldsymbol{k}}'$, so that, denoting $\bar{z}_{\boldsymbol{k}} := \langle \bar{\boldsymbol{m}}, \boldsymbol{k} \rangle$, the difference $z_{\boldsymbol{k}}^* - \bar{z}_{\boldsymbol{k}}$ serves as a shift on decryption. Precisely, recall that $\boldsymbol{k} \in \mathcal{R}^{\ell}$, $\mathsf{\mathbf{hk}} \in (\mathcal{R}^{\ell})^a$, and $z_{\boldsymbol{k}}^*, \bar{z}_{\boldsymbol{k}} \in \mathcal{R}$, then for some $\mathsf{hk}_{\mathcal{S}} \in K_{\mathsf{hk}} \subset \mathcal{R}^a$, $\mathcal{S}$ returns the following decryption key to $\mathcal{A}$:

   $$\mathsf{sk}_{\boldsymbol{k}}' := \boldsymbol{k}^T \cdot \mathsf{\mathbf{hk}} - (b^{-1} \bmod n_f)(z_{\boldsymbol{k}}^* - \bar{z}_{\boldsymbol{k}}) \cdot \mathsf{hk}_{\mathcal{S}} \text{ in } \mathcal{R}^a.$$

   We call $\mathsf{hk}_{\mathcal{S}} \in K_{\mathsf{hk}}$ the shift key, which must satisfy the following properties:

   (a) $\mathsf{hash}(\mathsf{hk}_{\mathcal{S}}, \Upsilon) = f$ and

   (b) $\forall u \in \mathcal{L}$, $\mathsf{hash}(\mathsf{hk}_{\mathcal{S}}, u) = \mathsf{hash}(0, u)$.

   Such a choice of $\mathsf{sk}_{\boldsymbol{k}}'$ ensures that $\mathsf{Dec}(\mathsf{mpk}^*, \mathsf{sk}_{\boldsymbol{k}}', \bar{\boldsymbol{ct}}) = z_{\boldsymbol{k}}^*$ as required. Notice that adding this shift is implicitly shifting the master secret key $\mathsf{\mathbf{hk}}$ by $(b^{-1} \bmod n_f)(\boldsymbol{m}^* - \bar{\boldsymbol{m}}) \cdot \mathsf{hk}_{\mathcal{S}}^T$; we denote $\mathsf{\mathbf{hk}}' := \mathsf{\mathbf{hk}} - (b^{-1} \bmod n_f)\big((\boldsymbol{m}^* - \bar{\boldsymbol{m}}) \cdot \mathsf{hk}_{\mathcal{S}}^T\big) \in (\mathcal{R}^{\ell})^a$.

We now explain intuitively why $\mathcal{A}$'s view in this simulation is indistinguishable from its view in a real execution. First observe that since the shift is a multiple of $\mathsf{hk}_{\mathcal{S}}$, from property (b) of $\mathsf{hk}_{\mathcal{S}}$ it holds that the public key $\mathsf{\mathbf{hp}}$ is consistent with $\mathsf{\mathbf{hk}}'$ from $\mathcal{A}$'s view, since for any $(u, w \in \mathsf{R})$, $\mathsf{hash}(\mathsf{\mathbf{hk}}, u) = \mathsf{hash}(\mathsf{\mathbf{hk}}', u) = \mathsf{hash}(\mathsf{hp}, u, w)$. Moreover, we have:

$$\bar{\boldsymbol{ct}} = (x \cdot y, \mathsf{projhash}(\mathsf{\mathbf{hp}}, x, w)\mathsf{hash}(\mathsf{\mathbf{hk}}, y) \cdot f^{\bar{\boldsymbol{m}}})$$

$$\begin{aligned}
&= (x \cdot y, \mathsf{hash}(\mathbf{hk}', x)\mathsf{hash}(\mathbf{hk}' + (b^{-1} \bmod n_f)(\boldsymbol{m}^* - \bar{\boldsymbol{m}}) \cdot \mathsf{hk}_{\mathscr{S}}^T, \Upsilon^b) \cdot f^{\bar{\boldsymbol{m}}}) \\
&= (x \cdot y, \mathsf{hash}(\mathbf{hk}', x)\mathsf{hash}(\mathbf{hk}', \Upsilon^b) \cdot \mathsf{hash}(\mathsf{hk}_{\mathscr{S}}, \Upsilon)^{(\boldsymbol{m}^* - \bar{\boldsymbol{m}})} \cdot f^{\bar{\boldsymbol{m}}}) \\
&= (x \cdot y, \mathsf{hash}(\mathbf{hk}', x)\mathsf{hash}(\mathbf{hk}', y) \cdot f^{\boldsymbol{m}^*}) = (x \cdot y, \mathsf{hash}(\mathbf{hk}', x \cdot y) \cdot f^{\boldsymbol{m}^*}),
\end{aligned}$$

Thus an invalid ciphertext encrypting $\boldsymbol{m}^*$ for the shifted key $\mathbf{hk}'$ is also an invalid ciphertext encrypting $\bar{\boldsymbol{m}}$ for $\mathbf{hk}$. Moreover $\mathscr{S}$'s answers to $\mathscr{A}$'s key derivation queries are exactly those one would obtain running the key derivation algorithm with $\mathbf{hk}'$. So $\mathscr{A}$'s view is exactly that of a real execution, where the sampled master secret key is $\mathbf{hk}'$ (modulo the fact we are using invalid ciphertexts in the simulated environment, but this change is unnoticeable under the $\delta_{\mathscr{L}}$-hardness of the subset membership problem). Finally using the smoothness of H allows to conclude that swapping $\mathbf{hk}$ for $\mathbf{hk}'$ is indistinguishable from $\mathscr{A}$'s view. Thus the real and simulated executions are indistinguishable to $\mathscr{A}$.

Running Example 1 – Shift key for $\mathsf{H}_{\mathsf{ddh}}$

For $\mathsf{H}_{\mathsf{ddh}}$, $\mathscr{L} = \langle(g_0, g_1)\rangle$, where g_0 and g_1 are randomly chosen elements of G. The generator of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ can choose g_0, g_1 such that it knows $w := \log_{g_0}(g_1)$. The shift key $\mathsf{hk}_{\mathscr{S}} := (-1 \bmod q, w^{-1} \bmod q)$ satisfies (a) and (b). Consequently with this $\mathsf{hk}_{\mathscr{S}}$ one can prove that the IPFE scheme resulting from $\mathsf{H}_{\mathsf{ddh}}$ (via the construction of Fig. 4.1) is $\mathsf{sim\text{-}fe\text{-}cpa}$-secure.

We note that the fact this construction is secure in the simulation based model is not new. Indeed, as noted previously, the IPFE scheme resulting from $\mathsf{H}_{\mathsf{ddh}}$ via the construction of Fig. 4.1 is exactly the DDH based scheme of Agrawal et al. [ALS16], they proved their scheme was $\mathsf{ind\text{-}fe\text{-}cpa}$-secure. Shortly after, Abdalla et al. [AGRW17] proved the same scheme was simulation secure against selective adversaries (which commit to the challenge messages prior to seeing the master public key). In [Wee17], Wee demonstrated that (still this same) scheme is secure in the simulation based model against semi-adaptive adversaries (the adversary is restricted to making all its key queries after it sees the challenge ciphertext), while the generic results of O'Neill [O'N10] tell us that since the scheme was proven $\mathsf{ind\text{-}fe\text{-}cpa}$-secure, it is also $\mathsf{sim\text{-}fe\text{-}cpa}$-secure in the restricted model where the adversary makes all its' key requests *before* seeing the challenge ciphertext. Finally, Agrawal et al. recently proved that the scheme is in fact $\mathsf{sim\text{-}fe\text{-}cpa}$-secure [ALMT20b]. Hence, though we do not claim to have proven anything new about the security of this well studied scheme, we generalise their approach.

Running Examples 2 and 3 – HSM-CL and DDH-f

The computation of $\mathsf{hk}_{\mathscr{S}}$ is not straight forward for any PHF, and in particular, for our running examples resulting from the CL framework, where the order of group $\widehat{G}$ is unknown, $\mathsf{hk}_{\mathscr{S}}$ cannot be computed efficiently. To see this consider our running example from HSM-CL, one would need: (a) $\mathsf{hk}_{\mathscr{S}} = 1 \bmod q$ and (b) $\mathsf{hk}_{\mathscr{S}} = 0 \bmod s$. However s is unknown, even to the group generator, and so finding a non zero multiple of s it not feasible in practice. We can thus not use the above proof technique to prove $\mathsf{sim\text{-}fe\text{-}cpa}$-security of an IPFE scheme resulting from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$.

Walking Example – Shift key for $\mathsf{H}_{\mathsf{dcr}}$

Despite the similarities between $\mathsf{H}_{\mathsf{dcr}}$ and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$, in $\mathsf{H}_{\mathsf{dcr}}$, which is the PHF arising from DCR, there is a trapdoor allowing to compute a shift key. Precisely, using the notations of Fig. 2.1, for $\mathsf{H}_{\mathsf{dcr}}$ we have $\mathscr{L} := \{r^N \bmod N^2 : r \hookleftarrow \mathbf{Z}/N^2\mathbf{Z}\}$, $\mathscr{X} := \mathbf{Z}/N^2\mathbf{Z}$, $\Upsilon := g = (N+1)$, and $f := (1+N)$; while, for a hashing key $\mathsf{hk} \in K_{\mathsf{hk}} = \mathbf{Z}$, and for $x \in \mathbf{Z}/N^2\mathbf{Z}$, the hashing algorithm computes $\mathsf{hash}(\mathsf{hk}, x) = x^{\mathsf{hk}} \in \mathbf{Z}/N^2\mathbf{Z}$. Observe that the factorisation of $N = PQ$, where P and Q are prime is used for the system's set up, and so must be known by the generator of $\mathsf{H}_{\mathsf{dcr}}$. Consequently this generator can compute $\Lambda \leftarrow (P-1)(Q-1)$ and compute the shift key satisfying $\mathsf{hk}_{\mathcal{S}} = 1 \bmod N$ and $\mathsf{hk}_{\mathcal{S}} = 0 \bmod \Lambda$. For any $x \in \mathcal{L}$, $x^{\Lambda} = x^0 = 1$, so (b) holds, since $\mathsf{hash}(\mathsf{hk}_{\mathcal{S}}, x) = \mathsf{hash}(0, x)$. Furthermore, since $(1+N)$ is of order N in $\mathbf{Z}/N^2\mathbf{Z}$, (a) also holds since $\mathsf{hash}(\mathsf{hk}_{\mathcal{S}}, \Upsilon) = \mathsf{hash}(\mathsf{hk}_{\mathcal{S}}, (1+N)) = (1+N)^{\mathsf{hk}_{\mathcal{S}}} = (1+N) = f$.

Chapter 5

Distributing EC-DSA

In Chapter 4, we saw how one can use projective hash functions to *generically* build inner product functional encryption schemes without sacrificing efficiency. We also saw that instantiating our constructions from assumptions in the CL framework and from the decisional composite residuosity assumption yields some of the most efficient such schemes to date. In particular, due in part to the fact that, in instantiations from the CL framework, one sets the message space to be of prime order q, where q can be chosen according to ones needs, we are able to substantially reduce the size of the keys and ciphertexts of our schemes.

In this chapter, we shall see another application of our projective hash functions: that of devising new threshold signature protocols for the standardised elliptic curve digital signature algorithm (EC-DSA). These protocols will also use the zero-knowledge proofs and arguments developed in Sections 3.6 and 3.7. We shall see that in this context, instantiations from the CL framework yield considerable efficiency gains compared to instantiations from DCR (or somewhat equivalently from the Paillier cryptosystem), thanks to the aforementioned for the choice of q.

State of the Art

Threshold signatures. A threshold signature scheme allows n mutually mistrusting users to share the capability of signing documents under a common public key. The threshold $t < n$ typically indicates that any subset of at least $t + 1$ users can collaborate in order to issue a valid signature. On the other hand, no coalition of t or less users can do so. In fact, even if an adversary corrupts up to t users, no information should leak on the underlying secret signing key. This latter property is very useful in practice as it significantly reduces the loss induced by a security break in. This explains why the study of threshold signatures (and more generally of threshold cryptography [Des88,DF90,GJKR96b,SG98,Sho00,Boy86,CH89,MR04a]) attracted significant interest from the early 1990s to the early 2000s.

Over the last few years, threshold signatures and, in particular, threshold EC-DSA signatures have received renewed attention. This mainly comes from the fact that EC-DSA is the signature scheme adopted in Bitcoin and other cryptocurrencies to validate transactions. Hence a secure, flexible and efficient protocol for threshold EC-DSA signatures would be very effective against the theft of Bitcoins: instead of storing a signing key in one single location one could share it among several servers so that none of them knows it in full and a quorum is needed to produce new signatures. This also means that an attacker must break into more than t servers to learn any meaningful information.

Of course, in order for a secure solution to be of any use in the cryptocurrency world, efficiency and flexibility are of fundamental importance. Here flexibility mainly refers to the possibility of arbitrarily setting the threshold. Efficiency, on the other hand, takes into account both the computational cost and the bandwidth consumption induced by the protocol.

Distributing EC-DSA. While for other signature schemes fast threshold variants are known (e.g. RSA signing [GJKR96a] and Schnorr signatures [Sch91, SS01]) constructing efficient threshold EC-DSA protocols has proved to be much harder. This is essentially due to a non-linear operation performed over the (secret) randomness used to produce EC-DSA signatures. Consequently, before the advent of cryptocurrencies, known solutions to the problem of devising threshold EC-DSA protocols fell short either in terms of flexibility or in terms of efficiency (or both). The state of the art was the work of Gennaro *et al.* [GJKR96a] where to implement a threshold of t servers one needed to share the key among a total of at least $n = 2t + 1$ servers, thus making sharings where all parties are required to participate to the signing process impossible.

The first protocol to overcome this limitation was that of Mackenzie and Reiter [MR01]. Their work focuses on the two-party case (i.e. where $t = 1$ and $n = 2$), which is non trivial since in such a setting there is no honest majority of parties. In their protocol, parties use Paillier's linearly homomorphic encryption scheme to combine their respective shares and complete the signature while keeping their inputs secret. It turns out that for their protocol to be secure against malicious adversaries, proving that each party followed the protocol correctly is not simple; they addressed this via expensive zero knowledge proofs. Consequenlty, if one whishes to garantee security against malicious adversaries, their protocol lacks the efficiency required to be used in practice. More recently Lindell [Lin17a] provided a much simpler and more efficient protocol. While his protocol also relies on Paillier's cryptosystem, he succeeds in removing almost all expensive zero knowledge proofs from the protocol, while proving that security holds in the simulation based model against malicious adversaries. Lindell's crucial observation is that, in a two party EC-DSA signing protocol, dishonest parties can create very little trouble since the verification algorithm of any signature scheme can be publicly evaluated. Hence a party can tell if the other cheated by running the verification algorithm on the jointly produced signature.

In a different style, Doerner et al. [DKLs18] provide a 2 out of n protocol requiring no additional assumptions than the security of centralised EC-DSA. They do not rely on linearly homomorphic encryption but rather on oblivious transfer for parties to combine their shares. Their protocol is fast, though its bandwidth consumption is higher than [Lin17a] due to this use of oblivious transfer.

Regarding the full threshold setting, i.e. protocols allowing to consider any threshold t such that $n \geqslant t + 1$, the work of Gennaro et al. [GGN16], whose efficiency was subsequently improved in [BGG17], shows how to generalise the Mackenzie-Reiter paradigm to any number of parties and with a full threshold. However implementing their protocols in practice requires some trusted set up, as their key generation requires multi-party Paillier key generation, and although two-party Paillier key generation is feasible [FLOP18], there is currently no practical multi-party variant.

More recently, building upon the ideas of [GGN16], Lindell et al. [LN18] propose a full threshold protocol which they prove secure under simulation-based definitions, for which they propose an implementation using Paillier's encryption scheme, and another relying on oblivious transfer. Concurrently, Gennaro et al. [GG18] put forth a protocol which does not require a trusted set up, and relies on Paillier's cryptosystem for parties to jointly compute signatures. Conversely to [LN18] they prove security in the game based model, reducing security to that of centralised EC-DSA.

Doerner et al., in [DKLs19], extend their protocol of [DKLs18] to the full threshold setting, their resulting protocol is proven secure in the universal composability paradigm (*cf.* Section 2.3.2). They obtain fast signature protocols, though again, the use of oblivious transfer incurs quite a high communication cost.

Our contributions

Generic solution for two-party EC-DSA. We start off considering the two party case, building upon the work of [Lin17a], and the observation that in Lindell's protocol, the use of Paillier's encryption scheme entails various complications.

First of all, a bandwidth overhead is incurred due to the difference between the order of the Paillier plaintext space, and that in which EC-DSA signatures are computed. Indeed since the Paillier plaintexts space is $\mathbf{Z}/N\mathbf{Z}$ whereas EC-DSA signatures live in $\mathbf{Z}/q\mathbf{Z}$ (q is prime), in order to avoid inconsistencies one needs to make sure N is taken large enough so that no wraparounds occur during the whole signature generation process. This also means that, when sending encrypted values, parties need to prove, via *range proofs*, that the encrypted plaintext is in the right range (*i.e.* sufficiently small).

A more subtle issue arises from the use of Paillier's encryption in the security proof. Indeed, to argue indistinguishability of an adversary's view in real and simulated executions, it seems necessary to set up a reduction to the indistinguishability of Paillier's cryptosystem. However if one knows the secret decryption key for the Paillier cryptosystem (i.e. the factorisation of N), then the underlying problem (DCR) is easy. This means one must design a proof technique that manages to successfully use Paillier's scheme *without* knowing the corresponding secret key sk. Problems occur if a corrupted player sends a *bad* ciphertext to the simulator (*i.e.* one that should not yield a valid signature), since the simulator can not use sk to decrypt and recognise that the ciphertext is bad.

Lindell [Lin17a] proposes two alternative proofs to overcome this. The first, in the game-based model, avoids the problem by allowing the simulator to abort with a probability that depends on the number of issued signatures q_s. This results in a proof of security that is not tight as the reduction looses a factor q_s. The second proof is simulation based, avoids the aborts, but requires the introduction of a new *interactive* non standard assumption regarding Paillier's encryption (detailed in Appendix C).

In order to overcome these issues, we observe that – as noted in Section 3.4.1 – for public key encryption schemes based on projective hash functions, the indistinguishability of ciphertexts is not compromised by the challenger knowing the decryption key hk. This key does *not* help break the subgroup membership problem underlying the scheme. Another interesting property (which we also use to prove security of our full threshold protocol) is that if one only knows the projection key hp, given an invalid ciphertext for a PHF based PKE (*cf.* Definition 3.21), this ciphertext could in fact decrypt to *any* message, and it is only given this message that one fixes the secret hashing key hk. This is particularly useful when simulating the view of an adversary, since one can replace a ciphertext decrypting to a given value with a ciphertext decrypting to any garbage value. We note that, for our two party protocol, subtle issues arise due to the fact the adversary has auxiliary information on the encrypted value. This leads us to introducing a new (non interactive) assumption, the *double encoding assumption*, whose hardness we advocate for PHFs arising from HSM-CL and DCR (*cf.* Section 5.2.1). Thus relying on linearly homomorphic schemes arising from projective hash functions, instead of Paillier, we provide a *generic* construction for two-party EC-DSA signatures from projective hash functions for which security does *not degrade* with the number of signature queries performed by the adversary; and whose simulation based security does not rely on *interactive* assumptions. Furthermore, observe that if we instantiate our protocol using either $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ of running example 2, or $\mathsf{H}_{\mathsf{ddh\text{-}f}}$ of running example 3 (*cf.* Chapter 3), we can *choose* the message space of the encryption scheme to be of the same prime order q as that used to compute

EC-DSA signatures. This avoids the need for range proofs and thereby significantly reduces bandwidth consumption compared to other two party EC-DSA protocols. We favour an instantiation from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ as resulting ciphertexts are smaller, and it allows to sample shorter keys (*cf.* Section 3.5.3), which improves both communication and computation complexity. We implement this instantiation in the Pari C Library [PAR20], and thereby obtain the first two party EC-DSA signing protocol which is practical (both in terms of computational efficiency and in terms of bandwidth consumption), does not require interactive assumptions and allows for a tight security reduction.

Bandwidth efficient full threshold EC-DSA. We next focus the full threshold dishonest majority setting. We revisit the [GG18] protocol which, as in Lindell's protocol for the two-party case, relies on Paillier's cryptosystem, thereby inducing the need for the order of the message space N to be much larger than q, and the need for range proofs so as to prevent malicious behaviour.

Building upon the ideas developed in our two party protocol we propose a new threshold EC-DSA variant of the [GG18] protocol. As our main goal is to improve efficiency, and in particular bandwidth consumption, we directly consider a protocol arising from the $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme of Fig. 3.5.

Our protocol eliminates all range proofs, while retaining comparable overall computational efficiency. Security does not degrade with the number of signatures queried by the adversary, and relies on the assumptions and tools introduced in Chapter 3. Precisely, as parties use the linearly homomorphic properties of $\Pi_{\mathsf{hsm\text{-}cl}}$ to jointly compute signatures, one must ensure no information leaks from this use of the encryption scheme. We thus require parties prove their ciphertexts are 'well formed' using the efficient zero-knowledge arguments of knowledge of Section 3.7. So as to reduce to the hardness of the strong root problem (*cf.* Definition 3.8) in these arguments of knowledge, we introduce a slight modification to the HSM-CL assumption. We then devise a setup protocol allowing parties to interactively set up the public parameters of the (accordingly modified) $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme.

We compare the efficiency of our protocol to those of Gennaro et al. [GG18] and of Lindell et al. [LN18], these are the best performing pre-existing protocols using similar construction techniques which achieve the same functionality. For all considered security levels our signing protocol reduces the bandwidth consumption by factors varying between 4.4 and 9, while key generation is consistently two times less expensive. In terms of timings, though for standard levels of security our signing protocol is up to four times slower than that of [GG18], for higher levels of security the trend is inverted.

Related publications and submissions. Most of the work in this chapter can be found in:

- [CCL+19] For the generic two party EC-DSA protocol of Section 5.2.
- [CCL+20] For the full threshold EC-DSA protocol of Section 5.3.

Road map

In Section 5.1 we formally define threshold signatures, the EC-DSA signature algorithm, and the security models adopted for our constructions. In Section 5.2.2 we define the notion of an EC-DSA-friendly projective hash function, from which, in Section 5.2.4, we provide a generic construction for two party EC-DSA, which we prove secure in the simulation-based model in Section 5.2.5. In Section 5.2.6 we provide an instantiation of this generic construction from the HSM-CL based projective hash function $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ (*cf.* running example 2 in Chapters 3 and 4).

Finally in Section 5.2.7 we compare implementations in the Pari C Library ([PAR20]) of our protocol to that of Lindell [Lin17a].

Next we consider the full threshold setting, in Section 5.3.1 we discuss the required public parameters, and in Section 5.3.2 we explain how parties can interactively set these up. In Section 5.3.3 we present our full-threshold EC-DSA signing protocol, whose security is proven in Section 5.3.4. Finally in Section 5.3.5 we compare the efficiency of this protocol to that of pre-existing protocols.

Notation 5.1. Throughout this chapter we consider a PPT algorithm group generator $\mathsf{Gen}_{\mathbf{G}}$ which on input 1^λ returns a description $(\mathbf{G}, P, q)$ of a group of EC points $(\mathbf{G}, +)$ of order q, generated by P. We further assume that the DL problem is hard for $\mathsf{Gen}_{\mathbf{G}}$.

5.1 Threshold Signature Algorithms

In this section, we present generalities required to understand this chapter. Precisely we define threshold signatures, the EC-DSA signing algorithm and the security models considered in our work.

5.1.1 Threshold Signature Scheme

Definition 5.2 ((t, n)-threshold signature scheme)**.** Let λ be a positive integer. For a threshold t and a number of parties $n > t$, a threshold signature protocol $\mathsf{T}\text{-}\Sigma$ for a signature scheme $\Sigma := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ consists of the following interactive protocols:

$\mathsf{ISetup}\langle 1^\lambda\rangle \to \langle \mathsf{pp}\rangle$ is a (possibly centralised) setup algorithm, which on input 1^λ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, and outputs public parameters pp for the signature scheme.

$\mathsf{IKeyGen}\langle \mathsf{pp}\rangle \to \langle (\mathsf{sk}_1, \mathsf{pk}); \ldots; (\mathsf{sk}_n, \mathsf{pk})\rangle$ s.t. $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$ where the values $\mathsf{sk}_1, \ldots,$ sk_n constitute a (t, n) threshold secret sharing of the signing key sk.

$\mathsf{ISign}\langle (\mathsf{sk}_1, m); \ldots; (\mathsf{sk}_n, m)\rangle \to \langle \sigma_1; \ldots; \sigma_n\rangle$ **or** $\langle \perp \rangle$ where $\perp$ is the error output, signifying the parties may abort the protocol; if all parties play symmetric roles in the protocol, for $i \in \{1, \ldots, n\}$, $\sigma_i = \sigma$, else $\sigma_i \in \{\mathsf{ok}, \sigma\}$, where ok signifies a party learns the protocol terminates successfully, and $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$.

The verification algorithm is non interactive and identical to that of Σ.

Correctness requires that for any $\lambda \in \mathbf{N}$, any $\mathsf{ISetup}\langle 1^\lambda; \ldots; 1^\lambda\rangle \to \langle \mathsf{pp}\rangle$, for all $\mathsf{IKeyGen}\langle \mathsf{pp}\rangle$ $\to \langle (\mathsf{sk}_1, \mathsf{pk}); \ldots; (\mathsf{sk}_n, \mathsf{pk})\rangle$, all messages m, and any subset $I = \{i_1, \ldots, i_k\}$, with $k > t$ and $i_1, \ldots, i_k \in \{1, \ldots, n\}$; if $\mathsf{ISign}\langle (\mathsf{sk}_{i_1}, m); \ldots; (\mathsf{sk}_{i_k}, m)\rangle \to \langle \sigma_1; \ldots; \sigma_k\rangle$ it holds that for $j \in \{1, \ldots, k\}$ if $\sigma_j \neq \mathsf{ok}$ then then $\mathsf{Verif}(\mathsf{pp}, \mathsf{vk}, m, \sigma_j) = 1$.

5.1.2 The Elliptic Curve Digital Signature Algorithm (EC-DSA)

We here present the specific signing protocol which is considered in this chapter, both in its centralised and distributed form, before positioning our work relatively to some of the many great achievements in the domain of threshold EC-DSA.

Centralised EC-DSA

EC-DSA is the elliptic curve analogue of the Digital Signature Algoritm (DSA). It was put forth by Vanstone [Van92] and accepted as ISO, ANSI, IEEE and FIPS standards. Let λ be a positive integer, and $(\mathbf{G}, P, q) \leftarrow \mathsf{Gen}_{\mathbf{G}}(1^\lambda)$. The EC-DSA scheme works on input $(\mathbf{G}, P, q)$;

uses a collision resistant hash function such as $\mathsf{SHA\text{-}2}$ (with the output converted to an integer); and consists of the following algorithms.

$\mathsf{KeyGen}(\mathbf{G}, q, P) \to (x, Q)$ where $x \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ is the secret signing key and $Q \leftarrow xP$ is the verification key.

$\mathsf{Sign}(x, m) \to (r, s)$ where r and s are computed as follows:

1. Compute m': the μ leftmost bits of $\mathsf{SHA} - 2(m)$.
2. Sample $k \hookleftarrow (\mathbf{Z}/q\mathbf{Z})^*$; compute $R \leftarrow kP$; denote $R = (r_x, r_y)$ and let $r \leftarrow r_x \bmod q$. If $r = 0$ choose another k.
3. Compute $s \leftarrow k^{-1}(m' + r \cdot x) \bmod q$.

$\mathsf{Verif}(Q, m, (r, s)) \to \{0, 1\}$ indicating whether or not the signature is accepted.

Difficulty of distributing EC-DSA. Compared to other digital signature schemes (e.g. RSA signing [GJKR96a] and Schnorr signatures [Sch91,SS01]), constructing efficient threshold protocols for the digital signature algorithm (DSA), and its elliptic curve variant EC-DSA, has proved to be quite challenging. We note that, though we focus on EC-DSA in this chapter, all of our results and those of prior work we cite applies to both the traditional DSA and EC-DSA. The main reason for this difficulty in sharing EC-DSA seems to result from the non-linear operation performed when inverting the (secret) randomness k to produce s.

Indeed, the natural approach to make the above algorithm distributed would be to share x additively among the participants and then start a multiparty computation protocol to produce the signature. In the two party case, this means that players start with shares x_1 and x_2 such that $Q = (x_1 + x_2)P$. The players can then proceed by generating random shares k_1, k_2 such that $R = (k_1 + k_2)P$. At this point, however, it is not clear how to compute, efficiently, shares k'_1, k'_2 such that $k'_1 + k'_2 = k'^{-1} \bmod q$.

On the widespread use of EC-DSA. One may wonder why EC-DSA has seen such widespread adoption, when other digital signature schemes lend themselves much better to threshold variants.

First observe that, compared to other signature protocols, for an equivalent level of security EC-DSA allows to use much smaller keys. As mentioned in Section 2.4.1, this is due to the fact that on elliptic curves, generic algorithms – which require $\Omega(\sqrt{p})$ group operations – are the best ones known for solving the DL problem, whereas in finite fields there exist algorithms with sub-exponential complexity $L_q[1/3]$. Similarly there exist algorithms in $L_N[1/3]$ for factoring integers. In this regard, a common RSA 3072-bit key provides a security level of 128 bits. However, EC-DSA requires only 256-bit sized keys to provide the same security level. These small keys and the speed of computing signatures explain why EC-DSA is particularly suitable for performing digital signatures in devices with constrained-resource.

Admittedly this efficiency of EC-DSA is not so much down to the algorithm itself, but rather to its use of elliptic curve cryptography. And indeed, the elliptic curve variant of the Schnorr signature algorithm [Sch91] comes with all the efficiency benefits of EC-DSA (e.g. short keys and fast signing time), is provably secure under standard assumptions [PS96], and does not involve any non-linear operations over secret data, therefore efficient threshold variants of the Schnorr signature algorithm are much easier to attain [SS01].

The reason Schnorr's signature algorithm has not seen the widespread adoption EC-DSA has is that, prior to publishing it, Schnorr filed multiple patents for his scheme which for years prevented its direct use. Interestingly DSA (and its EC variant EC-DSA) was designed as a variant of Schnorr specifically to circumvent these patents, and was subsequently widely deployed and standardised[1].

On the provable security of EC-DSA. Due to their efficiency and standardisation, EC-DSA signatures have seen widespread adoption over the internet (e.g. they are used for authentication in the Transport Layer Security protocol); hence organisations chose EC-DSA over other signatures as they considered sufficient its reputational security. However, though proofs of security, under certain assumptions, were found for digital signature schemes similar to DSA and EC-DSA, these proof techniques do not appear applicable to DSA and EC-DSA.

Currently proofs demonstrating that EC-DSA is existentially unforgeable under chosen message attacks rely either on strong assumptions on the group in which signatures are computed, or on the cryptographic hash function used to compute signatures. Indeed, Brown [Bro00] demonstrated EC-DSA is existentially unforgeable under chosen message attacks if one relies on the strong assumption that the underlying group is a generic group. Note that in this model adversaries to EC-DSA specific to elliptic curve groups are not ruled out. On the other hand Pointcheval and Vaudenay[2] [PV96, BPVY00], demonstrated that – if one hashes both the message m and the randomness r (*i.e.* $m' \leftarrow H(m|r)$), and if the hash function H behaves like a random oracle – then EC-DSA can be proven existentially unforgeable under chosen message attacks. However, the protocol they prove secure is slightly different to the real EC-DSA protocol, and such a reliance on the random oracle model is known to be controversial. Thus even though no attack has ever been found against EC-DSA, in the standard model the original EC-DSA scheme is not provably secure. For more details on the security of EC-DSA, we refer the reader to [FKP16, FKP17].

We also note that implementing EC-DSA securely is by no means a trivial task. Curves must be chosen appropriately, and implementation must include countermeasures against various attacks (e.g weak key generation, randomness reuse [BHH+14] and side channel attacks [DGH+13, ANT+20, JSSS20]).

5.1.3 Security Notions for Threshold Signatures

We here define the security models adopted to prove security of our constructions of Sections 5.2 and 5.3.

Game Based Security

Following [GJKR96b], we present a game-based definition of security analogous to existential unforgeability under chosen message attacks, adapted to the threshold setting: threshold unforgeability under chosen message attacks ($\mathsf{tu\text{-}cma}$).

Definition 5.3 (Threshold signature unforgeability [GJKR96b])**.** Consider a (t, n)-threshold signature scheme $\mathsf{IS} = (\mathsf{ISetup}, \mathsf{IKeyGen}, \mathsf{ISign}, \mathsf{Verif})$, and a PPT algorithm $\mathcal{A}$, having corrupted at most t players, and which is given the view of the protocols $\mathsf{IKeyGen}$ and ISign on input messages of its choice (chosen adaptively) as well as signatures on those messages. Let $\mathcal{M}$ be the set of queried messages. The scheme IS is threshold unforgeable under chosen message attacks ($\mathsf{tu\text{-}cma}$) if for any such $\mathcal{A}$, the probability $\mathsf{Adv}^{\mathsf{tu\text{-}cma}}_{\mathsf{IS},\mathcal{A}}$ that $\mathcal{A}$ can produce a signature on a message $m \notin \mathcal{M}$ is a negligible function of λ.

[1] The NIST proposed DSA for use in their Digital Signature Standard in 1991, and adopted it as FIPS 186 in 1994.

[2] Their work is on the DSA signature scheme, however results extend to elliptic curve variant.

Simulation Based Security and Ideal Functionalities

For our proof in the simulation based model (*cf.* Section 2.3.2), we consider static adversaries, that choose which parties are corrupted before the protocol begins.

We will use ideal functionalities for commitments, zero-knowledge proofs of knowledge (ZKPoK) and commitments to non interactive zero-knowledge (NIZK) proofs of knowledge between two parties P_1 and P_2. We give the intuition behind these ideal functionalities with the example of ZKPoK. We consider the case of a prover P_i with $i \in \{1,2\}$ who wants to prove the knowledge of a witness w for an element x which ensures that (x,w) satisfy the relation R, *i.e.* $(x,w) \in \mathsf{R}$. In an ideal world we can imagine an honest and trustful third party, which can communicate with both P_i and P_{3-i}. In this ideal scenario, P_i could give (x,w) to this trusted party, the latter would then check if $(x,w) \in \mathsf{R}$ and tell P_{3-i} if this is true or false. In the real world we do not have such trusted parties and must substitute them with a cryptographic protocol between P_1 and P_2. Roughly speaking, the Ideal/Real paradigm requires that whatever information an adversary $\mathcal{A}$ (corrupting either P_1 or P_2) could recover in the real world, it can also recover in the ideal world. The trusted third party can be viewed as the ideal functionality and we denote it by $\mathcal{F}$. If some protocol satisfies the above property regarding this functionality, we call it secure.

Formally, we denote $\mathcal{F}\langle x_1; x_2\rangle \to \langle y_1; y_2\rangle$ the joint execution of the parties via the functionality $\mathcal{F}$, with respective inputs x_i, and respective private outputs at the end of the execution y_i. Each transmitted message is labelled with a session identifier *sid*, which identifies an iteration of the functionality. The *ideal ZKPoK functionality* [HL10, Section 6.5.3], denoted $\mathcal{F}_{\mathsf{zk}}$, is defined for a relation R by $\mathcal{F}_{\mathsf{zk}}\langle (x,w); \emptyset\rangle \to \langle \emptyset; (x, \mathsf{R}(x,w))\rangle$, where $\emptyset$ is the empty output, signifying that the first party receives no output (*cf.* Fig. 5.1).

- Upon receiving $(\mathsf{prove}, sid, x, w)$ from a party P_i (for $i \in \{1,2\}$): if $(x,w) \notin \mathsf{R}$ or *sid* has been previously used then ignore the message. Otherwise, send (proof, sid, x) to party P_{3-i}

Figure 5.1: The $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R}}$ functionality

The *ideal commitment functionality*, denoted $\mathcal{F}_{\mathsf{com}}$, is depicted in Fig. 5.2. We also use an ideal functionality $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R}}$ for *commitments to NIZK proofs* for a relation R (*cf.* Fig. 5.3). Essentially, this is a commitment functionality, where the committed value is a NIZK proof.

- Upon receiving $(\mathsf{commit}, sid, x)$ from party P_i (for $i \in \{1,2\}$), record (sid, i, x) and send $(\mathsf{receipt}, sid)$ to party P_{3-i}. If some $(\mathsf{commit}, sid, *)$ is already stored, then ignore the message.
- Upon receiving $(\mathsf{decommit}, sid)$ from party P_i , if (sid, i, x) is recorded then send $(\mathsf{decommit},\ sid, x)$ to party P_{3-i}.

Figure 5.2: The $\mathcal{F}_{\mathsf{com}}$ functionality

The ideal functionality for two-party EC-DSA. The ideal functionality $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ (*cf.* Fig. 5.4) consists of two functions: a key generation function, called once, and a signing function, called an arbitrary number of times with the generated keys.

- Upon receiving $(\mathsf{com\text{-}prove}, sid, x, w)$ from a party P_i (for $i \in \{1,2\}$): if $(x,w) \notin \mathsf{R}$ or sid has been previously used then ignore the message. Otherwise, store (sid, i, x) and send $(\mathsf{proof\text{-}receipt}, sid)$ to P_{3-i}.
- Upon receiving $(\mathsf{decom\text{-}proof}, sid)$ from a party P_i (for $i \in \{1,2\}$): if (sid, i, x) has been stored then send $(\mathsf{decom\text{-}proof}, sid, x)$ to P_{3-i}

Figure 5.3: The $\mathcal{F}^{\mathsf{R}}_{\mathsf{com\text{-}zk}}$ functionality

Consider an Elliptic-curve group $\mathbf{G}$ of order q with generator a point P, then:

- Upon receiving $\mathsf{KeyGen}(\mathbf{G}, P, q)$ from both P_1 and P_2:
  1. Generate an EC-DSA key pair (Q, x), where $x \hookleftarrow (\mathbf{Z}/q\mathbf{Z})^*$ is chosen randomly and Q is computed as $Q \leftarrow x \cdot P$.
  2. Choose a hash function $H_q : \{0,1\}^* \to \{0,1\}^{\lfloor \log |q| \rfloor}$, and store $(\mathbf{G}, P, q, H_q, x)$.
  3. Send Q (and H_q) to both P_1 and P_2.
  4. Ignore future calls to KeyGen.
- Upon receiving $\mathsf{Sign}(sid, m)$ from both P_1 and P_2, where keys have already been generated from a call to Keygen and sid has not been previously used, compute an EC-DSA signature (r, s) on m, and send it to P_1 and P_2.

Figure 5.4: The $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ functionality

5.2 Two Party EC-DSA from PHFs

At Crypto 2017, Lindell [Lin17a] provided an elegant solution for two party EC-DSA. He adopts a similar strategy to Mackenzie et al. in [MR01], in that parties use the homomorphic properties of Paillier's cryptosystem to combine their secret shares. Using the simple observation that one can use the EC-DSA verification algorithm to check whether a party has cheated in the signature generation, he eliminates almost all the zero-knowledge proofs which were required in Mackenzie et al.'s protocol to handle malicious adversaries. In a little more detail, assume that in a preliminary phase party P_2 receives both P_1's public encryption key *and* an encryption $\mathsf{Enc}(x_1)$ of P_1's share of the secret signing key. Using the homomorphic properties of the encryption scheme, P_2 can now compute a ciphertext decrypting to $k_2^{-1}(H(m) + xr)$, which it sends to P_1; when P_1 decrypts, it multiplies the result by k_1^{-1} in order to obtain s.

The beauty here is that essentially, all a malicious P_1 can do is cheat in the generation of $R \leftarrow k_1 k_2 P$. However this operation is the well established Diffie-Hellman protocol for which very efficient and robust protocols exist. On the other hand, if P_2 is corrupted all it can do (except again cheat in the generation of R) is create a bad ciphertext as her final response for P_1. However, while P_2 can certainly try that, this would be easy to detect by simply checking the validity of the resulting signature.

Turning this nice intuition into a formal proof induces some caveats though, and, as noted in the chapter introduction, all the difficulties Lindell encounters in his proof are due to the use of Paillier's cryptosystem. In particular, regarding efficiency, this choice induces a number of range proofs in the overall protocol. In terms of security, recall that the reduction can not use the Paillier secret key while relying on the indistinguishability of the encryption scheme. This results in a security proof which, in the game based model degrades with the number of signatures the adversary is allowed to query, while in the simulation based model requires the reliance on a non standard interactive assumption.

In this section we present our generic two-party EC-DSA protocol from PHFs, which can be seen as a generalisation of Lindell's scheme, and allows to overcome the aforementioned issues due to the use of Paillier's encryption scheme. In its generic form, the protocol is not efficient enough for practical applications as it employs a general purpose zero-knowledge proof as underlying building block. Still, beyond providing a clean, general framework which is of interest in its own right, it allows us to abstract away the properties we want to realise. In particular, our protocol allows for a proof of security that is both tight and does not require interactive assumptions when proving simulation security. Indeed, in PKE schemes based on PHFs, indistinguishability of ciphertexts is not compromised by the challenger knowing the scheme's secret keys as it relies on a computational assumption and a statistical argument.

The correctness of our protocol follows from homomorphic properties that we require of the underlying PHF; furthermore, as we require the PHF be *homomorphically-extended* (Definition 3.12), the homomorphic properties of the PHF hold for any public key sampled from an efficiently recognisable set, thus no zero-knowledge proofs are required for the public key.

Towards efficient solutions, we show in Section 5.2.6 how to instantiate our generic construction using the HSM-CL based PHF of running example 2. Concretely, the main benefit of using the CL framework here, is that – as detailed in Section 3.5 – it allows to build linearly homomorphic PKE schemes where the plaintext space is $\mathbf{Z}/q\mathbf{Z}$ for arbitrarily large q. This also means that if one uses the very same q underlying the EC-DSA signature, one gets a concrete instantiation of our general protocol which naturally avoids all the inefficiencies resulting from N and q being different.

As final contribution, we propose a C implementation of our protocol[3]. Our results show that our improved security guarantees come almost at no additional cost. Indeed, while our scheme is slightly slower (by a factor 1.5 for key generation and 3.5 for signing) for 128-bit security level, we are actually better for larger parameters: for 256-bit security, we are more efficient both in terms of key generation and signing time (by respective factors of 4.2 and 1.3). In terms of bandwidth consumption, we fare better for all considered levels of security. We refer to Section 5.2.7 for precise implementation considerations and timings.

5.2.1 The Double Encoding Problem

For our security proof to go through we need a notion which deals with information leaked by the fact an interactive signing protocol concludes successfully or aborts, as this event may leak one bit of information to the adversary. We must ensure that a corrupted player can not devise ciphertexts allowing it to distinguish real and ideal executions, by causing an execution to conclude successfully in one case, while it would abort in the other. To this end, we require that – given a one way function (OWF) of $x \in \mathbf{Z}/q\mathbf{Z}$ (in our protocol this is the elliptic curve point $Q := xP$) – no PT adversary can produce two invalid encryptions of x.

Though the following assumption may seem quite ad-hoc, in the following paragraph we motivate that intuitively it seems to reduce to the hardness of inverting the one way function.

Definition 5.4 (Double encoding assumption)**.** Consider a positive integer $\lambda \in \mathbf{N}$, and a λ bit prime q. Further consider a collection of one way functions sampled via. an efficient algorithm Gen_{OW}, such that for $h \hookleftarrow \mathsf{Gen}_{OW}(1^\lambda, q)$, h has input space $\mathbf{Z}/q\mathbf{Z}$ (and arbitrary

[3] We also re-implemented Lindell's protocol to ensure a fair comparison

output space). Let $\mathsf{Gen}_{\mathcal{SM}}$ be a subgroup membership problem generator, which takes as input 1^λ, and a prime q, such that the resulting PHF H is $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable, for F of prime order q generated by f. Let $\mathcal{A}$ be an adversary for the double encoding (DE) problem, its advantage is defined as:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DE}}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Pr[&u_1, u_2, u_2u_1^{-1} \in \widehat{\mathcal{X}} \setminus \widehat{\mathcal{L}} \text{ and } \mathsf{hp} = \mathsf{projkg}(\mathsf{hk}) : \\ &\mathcal{SM} \hookleftarrow \mathsf{Gen}_{\mathcal{SM}}(1^\lambda, q), h \hookleftarrow \mathsf{Gen}_{OW}(1^\lambda, q), x \hookleftarrow \mathbf{Z}/q\mathbf{Z},\ y \leftarrow h(x), \\ &(\mathsf{hp}, (u_1, \mathsf{hash}(\mathsf{hk}, u_1)f^x), (u_2, \mathsf{hash}(\mathsf{hk}, u_2)f^x)) \hookleftarrow \mathcal{A}(h, \mathcal{SM}, y)].\end{aligned}$$

The DE problem is δ_{de}-hard for ($\mathsf{Gen}_{\mathcal{SM}}$, Gen_{OW}) if for all PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DE}}_{\mathcal{A}}(\lambda) \leqslant \delta_{\mathsf{de}}(\lambda)$. We say the DE assumption holds for ($\mathsf{Gen}_{\mathcal{SM}}$, Gen_{OW})(or the DE problem is hard for ($\mathsf{Gen}_{\mathcal{SM}}$, Gen_{OW})), if for any λ bit prime q the DE problem is δ_{de}-hard for ($\mathsf{Gen}_{\mathcal{SM}}$, Gen_{OW}) and $\delta_{\mathsf{de}}(\lambda) = \mathsf{negl}(\lambda)$.

On the hardness of the double encoding problem. If the PHF and the OWF arise from independent structures, it seems unlikely that one could solve the DE problem without breaking the one wayness of h, and subsequently computing two invalid encodings of x. Even if their structures are the same, it is unclear how one could do this. Of course if the OWF were the mapping of x to f^x, the DE problem would be easy. However in our applications we specifically require that computing x from f^x be easy, and consequently this mapping is *not* one way. We back the intuition that this problem is hard by considering two PHF instantiations which are relevant for our purposes. One from DCR (as detailed in [CS02]) and the other from HSM-CL (i.e. $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ of running example 2 in Chapter 3). Let us first recall the definition of a subgroup decomposition problem.

Definition 5.5. Consider a finite abelian group G, and subgroups G_1 and G_2 such that G is the direct product of G_1 and G_2. An algorithm $\mathcal{A}$ solves the subgroup decomposition (SD) problem in (G, G_1, G_2) if, given input $x \hookleftarrow G$, $\mathcal{A}$ outputs $y \in G_1, z \in G_2$ such that $x = yz$.

Recall that (*cf.* Chapter 3) for the projective hash functions $\mathsf{H}_{\mathsf{dcr}}$ and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ arising from DCR and HSM-CL, one has $K_{\mathsf{hk}} = \mathbf{Z}$, and for a hashing key $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM})$, and $x \in \widehat{\mathcal{X}}$, one has $\mathsf{hash}(\mathsf{hk}, x) = x^{\mathsf{hk}}$. This implies that the output space of the hashing algorithm is $\Pi = \widehat{\mathcal{X}} = \widehat{\mathcal{L}} \times \langle\widehat{\Upsilon}\rangle$, and in fact $\widehat{\Upsilon} = \Upsilon = f$ and $\langle\Upsilon\rangle = F$. Furthermore computing x from f^x can be done efficiently. Moreover $\mathsf{H}_{\mathsf{dcr}}$ and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ are homomorphic and key homomorphic.

In the following lemma we demonstrate that for both these PHFs, one can reduce the problem of inverting the OWF to the hardness of solving the SD problem in $(\widehat{\mathcal{X}}, \widehat{\mathcal{L}}, F)$, and the hardness of solving the DE problem.

Lemma 5.6. Consider PHFs arising from DCR and HSM-CL. Further consider a one way function h. Suppose there exists a PPT algorithm $\mathcal{B}_1$ solving the DE problem with non negligible probability; and a PPT algorithm $\mathcal{B}_2$ solving the SD problem with non negligible probability; then one can build a PPT algorithm breaking the one wayness of h with non negligible probability.

Proof. Consider $h \leftarrow \mathsf{Gen}_{OW}(1^\lambda, q)$, an adversary $\mathcal{A}$ attempting to invert h, and algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ as described in the lemma. $\mathcal{A}$ gets as input a value $y := h(x)$ for $x \hookleftarrow \mathbf{Z}/q\mathbf{Z}$. $\mathcal{A}$ runs $\mathcal{SM}\mathsf{Gen}_{\mathcal{SM}}(1^\lambda, q)$, and sends $(h, \mathcal{SM}, y)$ to $\mathcal{B}_1$. With significant probability $\mathcal{B}_1$ outputs $(\mathsf{hp}, (u_1, u_1^{\mathsf{hk}}f^x), (u_2, u_2^{\mathsf{hk}}f^x))$ where $u_1, u_2, u_2u_1^{-1} \in \widehat{\mathcal{X}} \setminus \widehat{\mathcal{L}}$ and $\mathsf{hp} = \mathsf{projkg}(\mathsf{hk})$. There exist unique values $z_1, z_2 \in \widehat{\mathcal{L}}$ and $b_1, b_2 \in \mathbf{Z}/q\mathbf{Z}$ such that $u_1 = z_1f^{b_1}$ and $u_2 = z_2f^{b_2}$. Let $e_1 := u_1^{\mathsf{hk}}f^x = z_1^{\mathsf{hk}}f^{b_1\mathsf{hk}+x}$ and $e_2 := z_2^{\mathsf{hk}}f^{b_2\mathsf{hk}+x}$. $\mathcal{A}$ calls upon $\mathcal{B}_2$ four times, with inputs u_1,

u_2, e_1 and e_2 respectively (these inputs can be re-randomised, but for simplicity we omit this level of detail), to obtain $z_1, z_2 \in \widehat{\mathcal{L}}$; $f^{b_1}, f^{b_2} \in F$; $z_1^{\mathsf{hk}}, z_2^{\mathsf{hk}} \in \widehat{\mathcal{L}}$; and $f^{b_1\mathsf{hk}+x}, f^{b_2\mathsf{hk}+x}$. Now $\mathcal{A}$ can efficiently compute $(b_1 \bmod q)$, $(b_2 \bmod q)$, $(b_1\mathsf{hk} + x \bmod q)$ and $(b_2\mathsf{hk} + x \bmod q)$. Since $u_2u_1^{-1} \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$, $b_1 \neq b_2 \bmod q$, and so there exists a unique solution for $x \bmod q$ which $\mathcal{A}$ can efficiently compute from the aforementioned equations, thus breaking the one wayness of h. □

Note that for the DCR based PHF there exists a trapdoor which renders the SD problem easy, which can be efficiently computed when generating the subset membership problem instance. Thus if the PHF arises from DCR, the DE problem is at least as hard as inverting the one way function.

For the HSM-CL based PHF (resulting from class group cryptography), i.e. $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ of running example 2, best known algorithms for solving the SD problem are sub-exponential, whereas for computing discrete logarithms in elliptic curves (which is the OWF we will consider in our construction) there currently exist only exponential algorithms. Consequently for this application the DE problem must have an exponential complexity.

5.2.2 EC-DSA-Friendly PHF

To build threshold EC-DSA from a PHF, we require a number of properties from the underlying PHF. In the following, we define the notion of an EC-DSA-friendly PHF, essentially it is a PHF which meets sufficient properties to ensure simulation based security in the protocol of Section 5.2.4.

Definition 5.7 (EC-DSA-friendly PHF)**.** Let λ be a positive integer, and let $(\mathbf{G}, P, q) \leftarrow \mathsf{Gen}_{\mathbf{G}}(1^\lambda)$. The output of generator $\mathsf{Gen}_{\mathbf{G}}$ defines the one way function $\mathsf{exp}_{\mathbf{G}}$ which to $x \in \mathbf{Z}/q\mathbf{Z}$ maps the EC point xP. Let $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be a subgroup membership problem, and consider the associated projective hash function $\mathsf{H} :=$ (hashkg, $\widehat{\mathsf{projkg}}$, projkg, hash, $\widehat{\mathsf{projhash}}$, $\mathsf{projhash}$). The projective hash function H is $(q, f, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_s, \delta_{\mathsf{de}})$-EC-DSA-friendly if:

- H is homomorphically extended (Definition 3.12) in $K'_{\mathsf{hp}} \subseteq K_{\mathsf{hp}}$; key homomorphic (Definition 3.13); and K_{hk} is a cyclic additive Abelian group;
- the co-domain Π of hash is a finite Abelian group which contains a cyclic subgroup F, generated by f, of order q;
- there exists an efficient isomorphism from $(\mathbf{Z}/q\mathbf{Z}, +)$ to $(F, \cdot)$, mapping $m \in \mathbf{Z}/q\mathbf{Z}$ to f^m, whose inverse $\log_f$ is also efficiently computable;
- H is $(\widehat{\Upsilon}, \Upsilon, F)$-decomposable, where $\widehat{\Upsilon} \in \widehat{\mathcal{X}}$, $\Upsilon \in \mathcal{X}$;
- $\mathcal{SM}$ is a $\delta_{\mathcal{L}}$-hard subgroup membership problem;
- H is δ_s-smooth over $\mathcal{X}$ on F;
- the DE problem is $\delta_{\mathsf{de}}(\lambda)$-hard for $(\mathsf{Gen}_{\mathcal{SM}}, \mathsf{Gen}_{\mathbf{G}})$ (*cf.* Definition 5.4).

Remark. Consider an EC-DSA-friendly PHF as defined above. For $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{SM})$, if $\mathsf{hash}(\mathsf{hk}, \Upsilon) = 1$ smoothness does not hold, hence we assume this is not the case. Throughout the rest of the paper, we denote Ψ the considered generator of K_{hk}, which satisfies $\mathsf{hash}(\Psi, \Upsilon) = f$. Consequently, for any $\mathsf{hk} \in K_{\mathsf{hk}}$, where $\mathsf{hk} = c \cdot \Psi$ (for some $c \in \mathbf{Z}$), and for any $y = \Upsilon^b$ (for some $b \in \mathbf{Z}$), one has $\mathsf{hash}(\mathsf{hk}, y) = f^{bc}$.

5.2.3 Zero-Knowledge Proofs

We use the $\mathcal{F}_{\mathsf{zk}}$, $\mathcal{F}_{\mathsf{com\text{-}zk}}$ hybrid model. Ideal ZK functionalities are used for the following relations, were the parameters of the elliptic curve $(\mathbf{G}, P, q) \leftarrow \mathsf{Gen}_{\mathbf{G}}(1^\lambda)$ are implicit public inputs:

1. $\mathsf{R}_{\mathsf{dl}} := \{(Q, w) | Q = wP\}$, proves the knowledge of the discrete logarithm of an elliptic curve point.

2. $\mathsf{R}_{\mathsf{phf\text{-}dl}} := \{(\mathsf{hp}, (c_1, c_2), Q_1); (x_1, w) | (c_1, c_2) = \mathsf{Enc}(\mathsf{hp}, x_1; (u, w)) \wedge (c_1, w) \in \mathsf{R} \wedge Q_1 = x_1 P\}$, proves the knowledge of the randomness used for encryption, and of the value x_1 which is both encrypted in the ciphertext (c_1, c_2) and the discrete logarithm of the elliptic curve point Q_1.

The functionalities $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R}_{\mathsf{dl}}}$, $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R}_{\mathsf{dl}}}$ can be instantiated using Schnorr proofs [Sch91]. For the $\mathsf{R}_{\mathsf{phf\text{-}dl}}$ proof, Lindell in [Lin17a] uses a proof of language membership as opposed to a proof of knowledge. Though his technique is quite generic, it cannot be used in our setting. Indeed, his approach requires that the ciphertext be *valid*, which means that the element c must be decryptable. As Lindell uses Paillier's encryption scheme, any element of $(\mathbf{Z}/N^2\mathbf{Z})^\times$ is a valid ciphertext. This is not the case for a PHF-based encryption scheme: as it incorporates redundancy not any pair in $\widehat{\mathcal{X}} \times \Pi$ is a valid ciphertext. For our instantiations, we use the protocols of Section 3.6. Note that in any case, we need not prove that x_1 is an integer in the range $\{0, \dots, q-1\}$ since both the message space of our PKE scheme and the EC group $\mathbf{G}$ are of order q.

5.2.4 Construction

Consider the description $(\mathbf{G}, P, q) \leftarrow \mathsf{Gen}_{\mathbf{G}}(1^\lambda)$; a subgroup membership problem $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ and the associated PHF H, which we assume to be $(q, f, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_s, \delta_{\mathsf{de}})$-EC-DSA-friendly. Further consider the resulting linearly homomorphic PKE scheme $\mathsf{PKE} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EvalSum}, \mathsf{EvalScal})$ (as described in Section 3.4). From these building blocks, in Fig. 5.5 we present a new two-party EC-DSA signing protocol.

5.2.5 Simulation Based Security

We here provide a proof that the protocol of Fig. 5.5 is secure in the Ideal/Real paradigm. To this end, we must argue the indistinguishability of an adversary's view – corrupting either party P_1 or P_2 – in real and simulated executions.

As noted in Section 3.4, in Cramer-Shoup like encryption schemes resulting from PHFs, thanks to the smoothness of the PHF the challenger in the $\mathsf{ind\text{-}cpa}$-security game knows hk, and this does not help solve the computational problem (i.e. the subset membership problem) underlying security. We insist on this point since in Lindell's protocol [Lin17a], many issues arise from the use of Paillier's cryptosystem, since if the challenger uses the secret key, one can no longer reduce the security of the protocol to the $\mathsf{ind\text{-}cpa}$-security of the Paillier encryption scheme. In particular this implies that in Lindell's game based proof, instead of letting the simulator use the Paillier secret key to decrypt the incoming ciphertext (and check the corrupted party P_2 did not send a different ciphertext c than that prescribed by the protocol), the simulator *guesses* when the adversary may have cheated by simulating an abort with a probability depending on the number of issued signatures. This results in a proof of security which is not tight.

P_1	$\mathsf{IKeyGen}(\mathbf{G}, P, q)$	P_2
$x_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$Q_1 \leftarrow x_1 P$	$\xrightarrow{(\mathsf{com\text{-}prove},1,Q_1,x_1)} \mathcal{F}^{\mathsf{R}_{\mathsf{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{proof\text{-}receipt},1)}$	
		$x_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$
P_1 aborts if $(\mathsf{proof}, 2, Q_2)$ not received.	$\xleftarrow{(\mathsf{proof},2,Q_2)} \mathcal{F}^{\mathsf{R}_{\mathsf{dl}}}_{\mathsf{zk}} \xleftarrow{(\mathsf{prove},2,Q_2,x_2)}$	$Q_2 \leftarrow x_2 P$
	$\xrightarrow{(\mathsf{decom\text{-}proof},1)} \mathcal{F}^{\mathsf{R}_{\mathsf{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{decom\text{-}proof},1,Q_1)}$	
$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$; $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$ Sample $(u, w) \in \mathsf{R}$		
$c_{\mathsf{key}} \leftarrow \mathsf{Enc}(\mathsf{hp}, x_1; (u, w))$	$\xrightarrow{(\mathsf{prove},3,(\mathsf{hp},c_{\mathsf{key}},Q_1),(x_1,w))} \mathcal{F}^{\mathsf{R}_{\mathsf{phf\text{-}dl}}}_{\mathsf{zk}} \xrightarrow{(\mathsf{proof},3,(\mathsf{hp},c_{\mathsf{key}},Q_1))}$	P_2 aborts if $(\mathsf{decom\text{-}proof}, 1, Q_1)$, $(\mathsf{proof}, 3, (\mathsf{hp}, c_{\mathsf{key}}, Q_1))$ not received or if $\mathsf{hp} \notin K_{\mathsf{hp}}$.
$Q \leftarrow x_1 Q_2$		$Q \leftarrow x_2 Q_1$

P_1	$\mathsf{ISign}(m, sid)$	P_2
$k_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$R_1 \leftarrow k_1 P$	$\xrightarrow{(\mathsf{com\text{-}prove},sid\Vert 1,R_1,k_1)} \mathcal{F}^{\mathsf{R}_{\mathsf{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{proof\text{-}receipt},sid\Vert 1)}$	$k_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $R_2 \leftarrow k_2 P$
P_1 aborts if $(\mathsf{proof}, sid\Vert 2, R_2)$ not received.	$\xleftarrow{(\mathsf{proof},sid\Vert 2,R_2)} \mathcal{F}^{\mathsf{R}_{\mathsf{dl}}}_{\mathsf{zk}} \xleftarrow{(\mathsf{prove},sid\Vert 2,R_2,k_2)}$	
	$\xrightarrow{(\mathsf{decom\text{-}proof},sid\Vert 1)} \mathcal{F}^{\mathsf{R}_{\mathsf{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{decom\text{-}proof},sid\Vert 1,R_1)}$	P_2 aborts if $(\mathsf{decom\text{-}proof}, sid\Vert 1, R_1)$ not received.
$R = (r_x, r_y) \leftarrow k_1 R_2$ $r \leftarrow r_x \bmod q$		$m' \leftarrow H(m)$ $R = (r_x, r_y) \leftarrow k_2 R_1$ $r \leftarrow r_x \bmod q$ $c_1 \leftarrow \mathsf{Enc}(\mathsf{hp}, k_2^{-1} \cdot m')$ $c_2 \leftarrow \mathsf{EvalScal}(\mathsf{hp}, c_{\mathsf{key}}, k_2^{-1} \cdot r \cdot x_2)$
$\alpha \leftarrow \mathsf{Dec}(\mathsf{hk}, c_3)$ $\hat{s} \leftarrow \alpha \cdot k_1^{-1}$, $s \leftarrow \min(\hat{s}, q - \hat{s})$ If $\mathsf{Verif}(Q, m, (r, s)) = 0$ then abort else return (r, s)	$\xleftarrow{\quad c_3 \quad}$	$c_3 \leftarrow \mathsf{EvalSum}(\mathsf{hp}, c_1, c_2)$

Figure 5.5: Two-Party EC-DSA Key Generation and Signing Protocols from PHFs

Moreover, though this technique suffices for a game-based definition, it does not for simulation based security definitions. Thus, in order to be able to prove their protocol using simulation, Lindell uses a non-standard interactive assumption (the $\mathsf{Paillier\text{-}EC}$ assumption, *cf.* Appendix C). Thanks to our use of PHFs we are able to avoid such an interactive assumption. Moreover, should one write a game based proof for our construction, the security loss present in [Lin17a] would not appear.

Theorem 5.8. Let λ be a positive integer and $(\mathbf{G}, P, q) \leftarrow \mathsf{Gen}_{\mathbf{G}}(1^\lambda)$, where the DL problem is δ_{dl}-hard for $\mathsf{Gen}_{\mathbf{G}}$. Let $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$ be a subgroup membership problem, and consider the associated projective hash function H, which we assume to be $(q, f, \widehat{\Upsilon}, \Upsilon, \delta_{\mathcal{L}}, \delta_s, \delta_{\mathsf{de}})$-EC-DSA-friendly. Then the protocol of Fig. 5.5 securely computes $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ in the $(\mathcal{F}_{\mathsf{zk}}, \mathcal{F}_{\mathsf{com\text{-}zk}})$-hybrid model in the presence of a malicious static adversary (under the ideal/real definition). Indeed there exists a simulator for the scheme such that no PT adversary – having corrupted either P_1 or P_2 – can distinguish a real execution of the protocol from a simulated one with probability greater than $2\delta_{\mathcal{L}} + \delta_{\mathsf{de}} + 2\delta_{\mathsf{dl}} + 1/q + \delta_s$.

Proof. In this proof, the simulator $\mathcal{S}$ only has access to an ideal functionality $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ for computing EC-DSA signatures, so all it learns in the ideal world is the public key Q which it gets as output of the KeyGen phase from $\mathcal{F}_{\text{ec-dsa}}$ and signatures (r, s) for messages m of its choice as output of the Sign phase. However in the real world, the adversary, having either corrupted P_1 or P_2 will also see all the interactions with the non corrupted party which lead to the computation of a signature. Thus $\mathcal{S}$ must be ale to simulate $\mathcal{A}$'s view of these interactions, while only knowing the expected output. To this end $\mathcal{S}$ must set up with $\mathcal{A}$ the same public key Q that it received from $\mathcal{F}_{\text{ec-dsa}}$, in order to be able to subsequently simulate interactively signing messages with $\mathcal{A}$, using the output of $\mathcal{F}_{\text{ec-dsa}}$ from the Sign phase.

$\mathcal{S}$ simulates P_2 – Corrupted P_1. We first show that if an adversary $\mathcal{A}_1$ corrupts P_1, one can construct a simulator $\mathcal{S}$ s.t. the output distribution of $\mathcal{S}$ is indistinguishable from $\mathcal{A}_1$'s view in an interaction with an honest party P_2. The main difference here with the proof of [Lin17a] arises from the fact we no longer use a ZKP from which $\mathcal{S}$ can extract the encryption scheme's secret key. Instead, $\mathcal{S}$ extracts the randomness used for encryption and the plaintext x_1 from the ZKPoK for $\mathsf{R}_{\text{phf-dl}}$, which allows it to recompute the ciphertext and verify it obtains the expected value c_{key}. Moreover since the message space of our encryption scheme is $\mathbf{Z}/q\mathbf{Z}$, if $\mathcal{A}_1$ does not cheat in the proofs (this is guaranteed by the $(\mathcal{F}_{\text{zk}}, \mathcal{F}_{\text{com-zk}})$-hybrid model), the obtained distributions are identical in the ideal and real executions (as opposed to statistically close as in [Lin17a]).

Key Generation Phase

1. Given input $\mathsf{KeyGen}(\mathbf{G}, P, q)$, the simulator $\mathcal{S}$ sends $\mathsf{KeyGen}(\mathbf{G}, P, q)$ to the ideal functionality $\mathcal{F}_{\text{ec-dsa}}$ and receives back a public key Q.

2. $\mathcal{S}$ invokes $\mathcal{A}_1$ on input $\mathsf{IKeyGen}(\mathbf{G}, P, q)$ and receives the commitment to a PoK of x_1 satisfying $Q_1 = x_1 P$ denoted $(\mathsf{com\text{-}prove}, 1, Q_1, x_1)$ as $\mathcal{A}_1$ intends to send to $\mathcal{F}_{\text{com-zk}}^{\mathsf{R}_{\text{dl}}}$. Thus $\mathcal{S}$ can extract x_1 and Q_1.

3. Using the extracted value x_1, $\mathcal{S}$ verifies that $Q_1 = x_1 P$. If so, it computes $Q_2 \leftarrow x_1^{-1} Q$ (using the value Q received from $\mathcal{F}_{\text{ec-dsa}}$); otherwise $\mathcal{S}$ samples a random Q_2 from $\mathbf{G}$.

4. $\mathcal{S}$ sends $(\mathsf{proof}, 2, Q_2)$ to $\mathcal{A}_1$ as if sent by $\mathcal{F}_{\text{zk}}^{\mathsf{R}_{\text{dl}}}$ thereby $\mathcal{S}$ simulates a ZKPoK of x_2 satisfying $Q_2 = x_2 P$.

5. $\mathcal{S}$ receives $(\mathsf{decom\text{-}proof}, 1)$ as $\mathcal{A}_1$ intends to send to $\mathcal{F}_{\text{com-zk}}^{\mathsf{R}_{\text{dl}}}$ and simulates P_2 aborting if $Q_1 \neq x_1 P$. $\mathcal{S}$ also receives $(\mathsf{prove}, 3, (\mathsf{hp}, c_{\text{key}}, Q_1), (x_1, w))$ as $\mathcal{A}_1$ intends to send to $\mathcal{F}_{\text{zk}}^{\mathsf{R}_{\text{phf-dl}}}$.

6. $\mathcal{S}$ computes u from w such that $(u, w) \in \mathsf{R}$, and using the extracted value x_1 verifies that $c_{\text{key}} = \mathsf{Enc}(\mathsf{hp}, x_1; (u, w))$; if not $\mathcal{S}$ simulates P_2 aborting.

7. $\mathcal{S}$ sends $\mathsf{continue}$ to $\mathcal{F}_{\text{ec-dsa}}$ for P_2 to receive output, and stores (x_1, Q, c_{key}).

When taking $\mathcal{F}_{\text{zk}}$ and $\mathcal{F}_{\text{com-zk}}$ as ideal functionalities, the only difference between the real execution as ran by an honest P_2, and the ideal execution simulated by $\mathcal{S}$ is that in the former $Q_2 \leftarrow x_2 P$ where $x_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, whereas in the latter $Q_2 \leftarrow x_1^{-1} Q$, where Q is the public key returned by the ideal functionality $\mathcal{F}_{\text{ec-dsa}}$. However since $\mathcal{F}_{\text{ec-dsa}}$ samples Q uniformly at random from $\mathbf{G}$, the distribution of Q_2 in both cases is identical.

Signing Phase

1. On input $\mathsf{Sign}(sid, m)$, $\mathcal{S}$ sends $\mathsf{Sign}(sid, m)$ to $\mathcal{F}_{\text{ec-dsa}}$ and receives back a signature (r, s).

2. $\mathcal{S}$ computes the point $R = (r, r_y)$ using the EC-DSA verification algorithm.

3. $\mathcal{S}$ invokes $\mathcal{A}_1$ with input $\mathsf{ISign}(sid, m)$ and simulates the first three interactions so that $\mathcal{A}_1$ computes R. The strategy is similar to that used to compute Q, in brief, it proceeds as follows:

   (a) $\mathcal{S}$ receives $(\mathsf{com\text{-}prove}, sid||1, R_1, k_1)$ from $\mathcal{A}_1$.

   (b) If $R_1 = k_1 P$ then $\mathcal{S}$ sets $R_2 \leftarrow k_1^{-1} R$; else $R_2 \hookleftarrow \mathbf{G}$. Then $\mathcal{S}$ sends $(\mathsf{proof}, sid||2, R_2)$ to $\mathcal{A}_1$.

   (c) $\mathcal{S}$ receives $(\mathsf{decom\text{-}proof}, sid||1)$ from $\mathcal{A}_1$. If $R_1 \neq k_1 P$ then $\mathcal{S}$ simulates P_2 aborting and instructs the trusted party computing $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ to abort.

4. $\mathcal{S}$ computes $c_3 \leftarrow \mathsf{Enc}(\mathsf{hp}, k_1 \cdot s \bmod q)$, where s was received from $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$, and sends c_3 to $\mathcal{A}_1$.

As with the computation of Q_2 in the key generation phase, R_2 is distributed identically in the real and ideal executions since R is randomly generated by $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$. The ZK proofs and verifications are also identically distributed in the $\mathcal{F}_{\mathsf{zk}}$, $\mathcal{F}_{\mathsf{com\text{-}zk}}$-hybrid model. Thus, the only difference between a real execution and the simulation is the way c_3 is computed. In the simulation it is an encryption of $k_1 \cdot s = k_1 \cdot k^{-1}(m' + r \cdot x) = k_2^{-1} \cdot (m' + r \cdot x) \bmod q$, whereas in a real execution c_3 is computed from c_{key}, using the homomorphic properties of the encryption scheme. However, notice that as long as there exist (u, w) such that $c_{\mathsf{key}} = \mathsf{Enc}(\mathsf{hp}, x_1; (u, w))$ where $Q = x_1 P$ – which is guaranteed by the ideal functionality $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R_{phf\text{-}dl}}}$ – and as long as the homomorphic operations hold – which is guaranteed for any hp in the efficiently verifiable ensemble K'_{hp} – the c_3 obtained in the real scenario is also an encryption of $s' = k_2^{-1} \cdot (m' + r \cdot x) \bmod q$. Thus c_3 is distributed identically in both cases.

This implies that the view of a corrupted P_1 is identical in the real and ideal executions of the protocol (in the $\mathcal{F}_{\mathsf{zk}}$, $\mathcal{F}_{\mathsf{com\text{-}zk}}$-hybrid model), *i.e.*, the simulator perfectly simulates the real environment, which completes the proof of this simulation case.

$\mathcal{S}$ simulates P_1 – Corrupted P_2. We now suppose an adversary $\mathcal{A}_2$ corrupts P_2 and describe the simulated execution of the protocol. We demonstrate via a sequence of games – where the first game is a real execution and the last game is a simulated execution – that both executions are indistinguishable. This proof methodology differs considerably to that of [Lin17a] since the main differences between a real and simulated execution are due to the ciphertext c_{key}, so the indistinguishability of both executions reduces to the hardness of the SMP, the smoothness of the underlying PHF, and the hardness of the DE problem. We first describe an ideal execution of the protocol:

Key Generation Phase

1. Given input $\mathsf{KeyGen}(\mathbf{G}, P, q)$, simulator $\mathcal{S}$ sends $\mathsf{KeyGen}(\mathbf{G}, P, q)$ to $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ and receives back Q.

2. $\mathcal{S}$ invokes $\mathcal{A}_2$ with input $\mathsf{IKeyGen}(\mathbf{G}, P, q)$ and sends $(\mathsf{proof\text{-}receipt}, 1)$ as $\mathcal{A}_2$ expects to receive from $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R_{dl}}}$.

3. $\mathcal{S}$ receives $(\mathsf{prove}, 2, Q_2, x_2)$ as $\mathcal{A}_2$ intends to send to $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R_{dl}}}$. $\mathcal{S}$ extracts x_2 from this interaction.

4. $\mathcal{S}$ verifies that Q_2 is a non zero point on the curve and that $Q_2 = x_2 P$. If so $\mathcal{S}$ computes $Q_1 \leftarrow (x_2)^{-1} Q$ and sends $(\mathsf{decom\text{-}proof}, 1, Q_1)$ to $\mathcal{A}_2$ as it expects to receive from $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R_{dl}}}$. If not $\mathcal{S}$ simulates P_1 aborting and halts.

5. $\mathcal{S}$ samples $\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$ and computes $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$. It also samples $\tilde{x}_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and $(u, w) \in \mathsf{R}$ and computes $c_{\mathsf{key}} \leftarrow \mathsf{Enc}(\mathsf{hp}, \tilde{x}_1; (u, w))$.

6. $\mathcal{S}$ sends $(\mathsf{proof}, 3, (\mathsf{hp}, c_{\mathsf{key}}, Q_1))$ to $\mathcal{A}_2$ as $\mathcal{A}_2$ expects to receive from $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R}_{\mathsf{phf\text{-}dl}}}$.

7. $\mathcal{S}$ sends $\mathsf{continue}$ to $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ for P_1 to receive output, and stores Q.

Signing Phase

1. Upon input $\mathsf{Sign}(sid, m)$, $\mathcal{S}$ sends $\mathsf{Sign}(sid, m)$ to $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ and receives back a signature (r, s).

2. $\mathcal{S}$ computes the point $R = (r, r_y)$ using the EC-DSA verification algorithm.

3. $\mathcal{S}$ invokes $\mathcal{A}_2$ with input $\mathsf{ISign}(sid, m)$ and sends $(\mathsf{proof\text{-}receipt}, sid||1)$ as $\mathcal{A}_2$ expects to receive from $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R}_{\mathsf{dl}}}$.

4. $\mathcal{S}$ receives $(\mathsf{prove}, sid||2, R_2, k_2)$ as $\mathcal{A}_2$ intends to send to $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R}_{\mathsf{dl}}}$. $\mathcal{S}$ extracts k_2 from this interaction.

5. $\mathcal{S}$ verifies that R_2 is a non zero point and that $R_2 = k_2 P$. If so it computes $R_1 \leftarrow k_2^{-1} R$ and sends $(\mathsf{decom\text{-}proof}, sid||1, R_1)$ to $\mathcal{A}_2$ as it expects to receive from $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R}_{\mathsf{dl}}}$. If not $\mathcal{S}$ simulates P_1 aborting and instructs the trusted party computing $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ to abort.

6. $\mathcal{S}$ receives $c_3 = (u_3, e_3)$ from $\mathcal{A}_2$, which it can decrypt using hk, i.e.
$$\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1} \right).$$
If $\alpha = k_2^{-1} \cdot (m' + r \cdot x_2 \cdot \tilde{x}_1) \bmod q$ then $\mathcal{S}$ sends $\mathsf{continue}$ to the trusted party $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$, so that the honest party P_1 receives output. Otherwise $\mathcal{S}$ instructs $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ to abort.

We now describe the sequence of games. Game_0 is the real execution of the protocol from P_1's view, and we finish in Game_6 which is the ideal simulation described above. In the following intermediary games, only the differences in the steps performed by $\mathcal{S}$ are depicted.

Game_0	Game_1
$Q \leftarrow x_1 x_2 P$	$Q \leftarrow x_1 x_2 P$
$\vdots$	$\vdots$
$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$	$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$
$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$	$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$
	Sample $(u, w) \in \mathsf{R}$
$c_{\mathsf{key}} \leftarrow \mathsf{Enc}(\mathsf{hp}, x_1)$	$c_{\mathsf{key}} \leftarrow (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{x_1})$
Send c_{key} to $\mathcal{A}_2$	Send c_{key} to $\mathcal{A}_2$
$\vdots$	$\vdots$
$R \leftarrow k_1 k_2 P$, $r \leftarrow r_x \bmod q$	$R \leftarrow k_1 k_2 P$, $r \leftarrow r_x \bmod q$
$\vdots$	$\vdots$
Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$	Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$
Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1} \right)$	Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1} \right)$
$\vdots$	$\vdots$
$s \leftarrow \alpha \cdot k_1^{-1}$	$s \leftarrow \alpha \cdot k_1^{-1}$
If not $\mathsf{Verif}(Q, m, (r, s))$ then abort	If not $\mathsf{Verif}(Q, m, (r, s))$ then abort
else return (r, s)	else return (r, s)

Game_2	Game_3
$Q \leftarrow x_1x_2P$	$Q \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$
	Extract x_2 from $(\mathsf{prove}, 2, Q_2, x_2)$
	$\tilde{x}_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$
⋮	⋮
$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$	$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$
$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$	$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$
$u \hookleftarrow \mathcal{X}\backslash\mathcal{L}$	$u \hookleftarrow \mathcal{X}\backslash\mathcal{L}$
$c_{\mathsf{key}} \leftarrow (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{x_1})$	$c_{\mathsf{key}} \leftarrow (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{\tilde{x}_1})$
Send c_{key} to $\mathcal{A}_2$	Send c_{key} to $\mathcal{A}_2$
⋮	⋮
$R \leftarrow k_1k_2P$, $r \leftarrow r_x \bmod q$	$(r, s) \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$, $r \leftarrow r_x \bmod q$
	Extract k_2 from $(\mathsf{prove}, sid\|\|2, R_2, k_2)$
⋮	⋮
Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$	Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$
⋮	⋮
Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1}\right)$	Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1}\right)$
$s \leftarrow \alpha \cdot k_1^{-1}$	(1) If $\alpha \neq k_2^{-1}(m' + r\tilde{x}_1x_2)$ then
If not $\mathsf{Verif}(Q, m, (r, s))$ then abort	(2) If $(\alpha k_2)P \neq mP + rQ$ abort
else return (r, s)	else return (r, s)

Game_4	Game_5	Game_6
$Q \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$	$Q \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$	$Q \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$
Extract x_2 from $(\mathsf{prove}, 2, Q_2, x_2)$	Extract x_2 from $(\mathsf{prove}, 2, Q_2, x_2)$	Extract x_2 from $(\mathsf{prove}, 2, Q_2, x_2)$
$\tilde{x}_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$	$\tilde{x}_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$	$\tilde{x}_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$
⋮	⋮	⋮
$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$	$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$	$\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$
$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$	$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$	$\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$
$u \hookleftarrow \mathcal{X}\backslash\mathcal{L}$	Sample $(u, w) \in \mathsf{R}$	
$c_{\mathsf{key}} \leftarrow (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{\tilde{x}_1})$	$c_{\mathsf{key}} \leftarrow (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{\tilde{x}_1})$	$c_{\mathsf{key}} \leftarrow \mathsf{Enc}(\mathsf{hp}, \tilde{x}_1)$
Send c_{key} to $\mathcal{A}_2$	Send c_{key} to $\mathcal{A}_2$	Send c_{key} to $\mathcal{A}_2$
⋮	⋮	⋮
$(r, s) \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$, $r \leftarrow r_x \bmod q$	$(r, s) \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$, $r \leftarrow r_x \bmod q$	$(r, s) \leftarrow \mathcal{F}_{\mathsf{ec\text{-}dsa}}$, $r \leftarrow r_x \bmod q$
Extract k_2 from $(\mathsf{prove}, sid\|\|2, R_2, k_2)$	Extract k_2 from $(\mathsf{prove}, sid\|\|2, R_2, k_2)$	Extract k_2 from $(\mathsf{prove}, sid\|\|2, R_2, k_2)$
⋮	⋮	⋮
Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$	Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$	Receive $c_3 := (u_3, e_3)$ from $\mathcal{A}_2$
⋮	⋮	⋮
Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1}\right)$	Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1}\right)$	Let $\alpha \leftarrow \log_f \left(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1}\right)$
If $\alpha \neq k_2^{-1}(m' + r\tilde{x}_1x_2)$	If $\alpha \neq k_2^{-1}(m' + r\tilde{x}_1x_2)$	If $\alpha \neq k_2^{-1}(m' + r\tilde{x}_1x_2)$
then abort	then abort	then abort
(check (2) removed)		

Let us now demonstrate that each game step is indistinguishable from the view of $\mathcal{A}_2$. Intuitively, in Game_1 the simulator uses the secret hashing key hk instead of the public projection key hp to compute c_{key}. Though the values are computed differently, they are distributed identically, and are perfectly indistinguishable. Next in Game_2 we replace the first element of the ciphertext (in Game_1 this is $u \in \mathcal{L}$) with an element $u \in \mathcal{X}\backslash\mathcal{L}$. By the hardness of the subset membership problem Game_1 and Game_2 are indistinguishable. Next in Game_3 we switch to the ideal world, so Q and R are received from $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$. The value x_1 such that $Q = x_1x_2P$ is unknown to $\mathcal{S}$ simulating P_1, and the value $\tilde{x}_1$ encrypted in c_{key} is sampled uniformly at random from $\mathbf{Z}/q\mathbf{Z}$, and is unrelated to Q. Proving indistinguishability between Game_2 and Game_3 is the most involved analysis of all our game steps. The smoothness of the PHF ensures that the ciphertext c_{key} follows identical distributions in both games from $\mathcal{A}_2$'s view; however difficulties arise due to the check performed by $\mathcal{S}$ on α after decrypting c_3. Indeed if $\mathcal{A}_2$ produces a ciphertext c_3 which passes the check in one game, but not in the other, clearly $\mathcal{A}_2$ can distinguish both games. To deal with this, in Game_3 we introduce an additional check (2). Check (2) is performed using the EC point Q, and compares α to $k_2^{-1}(m' + rx_1x_2)$. On the other hand check (1) is performed using the randomly sampled $\tilde{x}_1$, and compares α to $k_2^{-1}(m' + r\tilde{x}_1x_2)$. This extra check allows us to ensure that if $\mathcal{A}_2$ can cause one game to abort, while the other does not, it has either broken the double encoding problem, or fixes the value of $\tilde{x}_1$. Since from the smoothness of the PHF, $\tilde{x}_1$ follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$ from $\mathcal{A}_2$'s view, this cannot occur with probability greater than $1/q + \delta_s$. So Game_2 and Game_3 are indistinguishable. In Game_4 we remove check (2), and demonstrate that if $\mathcal{A}_2$ could distinguish both games, one could use $\mathcal{A}_2$ to break the DL problem in $\mathbf{G}$.

Next we use the hardness of the subset membership problem again to hop from Game_4 to Game_5, so that in the latter the first element of the ciphertext is once again in $\mathcal{L}$; and finally Game_5 to Game_6 are identical from an adversary's point of view since we simply use the public evaluation function of the hash function $\mathsf{projhash}$ instead of the private one.

We denote $\mathsf{E_i}$ the event algorithm $\mathcal{A}_2$ interacting with $\mathcal{S}$ in $\mathsf{Game_i}$ outputs 1. Thus by demonstrating that $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_6]|$ is negligible, we demonstrate that, from $\mathcal{A}_2$'s view, the real and ideal executions are indistinguishable.

Game_0 to Game_1. In Game_1 $\mathcal{S}$ uses the secret hashing key hk instead of the public projection key hp and the witness w to compute c_{key}. Both games are identical from $\mathcal{A}_2$'s view:

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_0]| = 0.$$

Game_1 to Game_2. In Game_2 the ciphertext component u of c_{key} is sampled from $\mathcal{X}\backslash\mathcal{L}$ instead of from $\mathcal{L}$. Both games are indistinguishable under the $\delta_{\mathcal{L}}$-hardness of $\mathcal{SM}$. Thus:

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| \leqslant \delta_{\mathcal{L}}.$$

Game_2 to Game_3. In Game_3 the points $Q = x_1x_2P$ and R come from the functionality $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$, while in Game_2 they are computed as in the real protocol. As a result in Game_3 the value $\tilde{x}_1$ encrypted in c_{key} is unrelated to x_1. Let us denote $c_{\mathsf{key}} := (u, e)$, where $e = \mathsf{hash}(\mathsf{hk}, u)f^{\tilde{x}_1}$, the invalid ciphertext which $\mathcal{S}$ sends to $\mathcal{A}_2$ in Game_3. Using the fact H is decomposable, and since $u \in \mathcal{X}\backslash\mathcal{L}$, we can write $u = zy$, for unique $z \in \mathcal{L}$ and $y \in \langle\Upsilon\rangle$. We denote $b \in \mathbf{Z}/q\mathbf{Z}$ the unique value such that $\mathsf{hash}(\Psi, y) = f^b$. Note that since $u \notin \widehat{\mathcal{L}}$, it holds that $b \neq 0 \bmod q$. Now to demonstrate that Game_2 and Game_3 are indistinguishable from $\mathcal{A}_2$'s view, we start by considering a fixed $\mathsf{hk}' \in K_{\mathsf{hk}}$ satisfying the following equations:

$$\begin{cases} \mathsf{projkg}(\mathsf{hk}') = \mathsf{hp} = \mathsf{projkg}(\mathsf{hk}), \\ \mathsf{hash}(\mathsf{hk}', y)f^{x_1} = \mathsf{hash}(\mathsf{hk}, y)f^{\tilde{x}_1}. \end{cases}$$

Note that the smoothness of H over $\mathcal{X}$ on F ensures that such a hk' exists (it is not necessarily unique). We can now see that in Game_3, c_{key} is an invalid encryption of both x_1 and of $\tilde{x}_1$, for respective hashing keys hk' and hk, but for the same public projection key hp, indeed:

$$\begin{aligned} c_{\mathsf{key}} &= (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{\tilde{x}_1}) = (u, \mathsf{projhash}(\mathsf{hp}, z, w) \cdot \mathsf{hash}(\mathsf{hk}, y) \cdot f^{\tilde{x}_1}) \\ &= (u, \mathsf{projhash}(\mathsf{hp}, z, w) \cdot \mathsf{hash}(\mathsf{hk}', y) \cdot f^{x_1}) = (u, \mathsf{hash}(\mathsf{hk}', u) \cdot f^{x_1}) \end{aligned}$$

Let us denote γ and $\gamma' \in \mathbf{Z}$ the values such that $\mathsf{hk} = \gamma \cdot \Psi$ and $\mathsf{hk}' = \gamma' \cdot \Psi$, so that $\mathsf{hash}(\mathsf{hk}, \Upsilon) = f^{\gamma}$ and $\mathsf{hash}(\mathsf{hk}', \Upsilon) = f^{\gamma'}$. Now since $\mathsf{hash}(\Psi, y) = f^b$, it holds that

$$b\gamma + \tilde{x}_1 = b\gamma' + x_1 \bmod q \quad \Leftrightarrow \quad \gamma' - \gamma = b^{-1}(\tilde{x}_1 - x_1) \bmod q. \tag{5.1}$$

The adversary $\mathcal{A}_2$ gets the EC-DSA public key Q, the public projection key $\mathsf{hp} = \mathsf{projkg}(\mathsf{hk})$, and c_{key} from $\mathcal{S}$ (at this point its view is identical to its' view in Game_2). Then $\mathcal{A}_2$ computes $c_3 = (u_3, e_3)$, which it sends to $\mathcal{S}$. The difference between Game_2 and Game_3 appears now in how $\mathcal{S}$ attempts to decrypt c_3. In Game_2 it would have used hk', whereas in Game_3 it uses hk.

Notation. We denote α the random variable obtained by decrypting c_3 (received in Game_3) with decryption key hk; we denote α' the random variable obtained by decrypting c_3 (received in Game_3) with decryption key hk'; we introduce a hypothetical Game_3', which is exactly as Game_3, only one decrypts c_3 with decryption key hk', thus obtaining α', and check (1) of Game_3 is replaced by 'If $\alpha \neq k_2^{-1}(m' + rx_1x_2)$'. Since both tests of Game_3' are redundant, we only keep check (2).

Observation. The view of $\mathcal{A}_2$ in Game_2 and in Game_3' is identical. By demonstrating that the probability $\mathcal{A}_2$'s view differs when $\mathcal{S}$ uses α in Game_3 from when it uses α' in Game_3' is negligible, we can conclude that $\mathcal{A}_2$ cannot distinguish Game_2 and Game_3 except with negligible probability.

Let us consider the ciphertext $c_3 = (u_3, e_3) \in \widehat{\mathcal{X}} \times \Pi$ sent by $\mathcal{A}_2$. By the decomposability of H we know there exist unique $z_3 \in \widehat{\mathcal{L}}$, $y_3 \in \langle \widehat{\Upsilon} \rangle$ satisfying $u_3 = z_3y_3$. Moreover there exists a unique $b_3 \in \mathbf{Z}/q\mathbf{Z}$ satisfying $\mathsf{hash}(\Psi, y_3) = f^{b_3}$. By the homomorphic properties of H the decryption algorithm applied to c_3 with decryption key hk (resp. hk') returns $\perp$ if $e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1} = e_3 \cdot \mathsf{hash}(\mathsf{hk}, z_3)^{-1} \cdot \mathsf{hash}(\mathsf{hk}, y_3)^{-1} \notin F$ (resp. $e_3 \cdot \mathsf{hash}(\mathsf{hk}', u_3)^{-1} = e_3 \cdot \mathsf{hash}(\mathsf{hk}', z_3)^{-1} \cdot \mathsf{hash}(\mathsf{hk}', y_3)^{-1} \notin F$). However since $z_3 \in \widehat{\mathcal{L}}$, and $\widehat{\mathsf{projkg}}(\mathsf{hk}') = \widehat{\mathsf{projkg}}(\mathsf{hk})$, by correctness of H it holds that $\mathsf{hash}(\mathsf{hk}', z_3) = \mathsf{hash}(\mathsf{hk}, z_3)$; while $\mathsf{hash}(\mathsf{hk}', y_3) = f^{\gamma' \cdot b_3}$ and $\mathsf{hash}(\mathsf{hk}, y_3) = f^{\gamma \cdot b_3}$ live in F. Consequently the decryption algorithm applied to c_3 with decryption key hk returns $\perp$ if and only if it does so with decryption key hk' (i.e. $\alpha =\perp$ if and only if $\alpha' =\perp$). In this case Game_3 is identical to Game_3' from $\mathcal{A}_2$'s view ($\mathcal{S}$ aborts in both cases). We hereafter assume decryption does not fail, which allows us to adopt the following notation: $e_3 = \mathsf{hash}(\mathsf{hk}, z_3)f^{h_3} = \mathsf{hash}(\mathsf{hk}', z_3)f^{h_3}$ with $h_3 \in \mathbf{Z}/q\mathbf{Z}$. We thus have:

$$\begin{aligned} \alpha &\leftarrow \log_f(e_3 \cdot \mathsf{hash}(\mathsf{hk}, u_3)^{-1}), \\ &= h_3 - b_3 \cdot \gamma \bmod q \end{aligned} \qquad \text{and} \qquad \begin{aligned} \alpha' &\leftarrow \log_f(e_3 \cdot \mathsf{hash}(\mathsf{hk}', u_3)^{-1}), \\ &= h_3 - b_3 \cdot \gamma' \bmod q \end{aligned}$$

such that, injecting Eq. (5.1), one gets:

$$\alpha - \alpha' = b_3(\gamma' - \gamma) = b_3b^{-1}(\tilde{x}_1 - x_1) \bmod q.$$

We now consider four cases:

1. $\alpha = \alpha' \bmod q$. This case occurs if $b_3 = 0$, i.e. $u_3 \in \widehat{\mathcal{L}}$ and so u_3 is a valid ciphertext; or if $\tilde{x}_1 = x_1 \bmod q$. If this occurs Game_2 and Game_3 are identical from $\mathcal{A}_2$'s view. Note that this is the only case where all checks pass for both α and α'.

2. $\alpha \neq \alpha' \bmod q$ but $\alpha - \alpha' = k_2^{-1}rx_2(\tilde{x}_1 - x_1) \bmod q$. This occurs if $b_3 = k_2^{-1}rx_2b \bmod q$, i.e. $\mathcal{A}_2$ homomorphic operations on c_{key}, and the difference between α and α' is that expected by the simulator. This results in identical views from $\mathcal{A}_2$'s perspective because α causes check (1) to pass if and only if α' causes check (2) to pass:

   $$\alpha = k_2^{-1}(m' + r\tilde{x}_1x_2) \Leftrightarrow \alpha' + k_2^{-1}rx_2(\tilde{x}_1 - x_1) = k_2^{-1}(m' + r\tilde{x}_1x_2) \Leftrightarrow \alpha' = k_2^{-1}(m' + rx_2x_1).$$

3. $\alpha \neq \alpha' \bmod q$ and $\alpha - \alpha' \neq k_2^{-1}rx_2(\tilde{x}_1 - x_1) \bmod q$. We here consider three sub-cases:

(a) Either both tests fail for α and test (2) fails for α'; i.e. $\alpha \neq k_2^{-1}(m' + r\tilde{x}_1 x_2) \bmod q$; and $\alpha, \alpha' \neq k_2^{-1}(m' + r x_1 x_2) \bmod q$. This results in identical views from $\mathcal{A}_2$'s perspective.

(b) Either the check on α' passes. This means that:

$$\alpha' = k_2^{-1}(m' + r x_1 x_2) \bmod q.$$

Since $\alpha - \alpha' \neq k_2^{-1} r x_2(\tilde{x}_1 - x_1) \bmod q$ necessarily check (1) on α fails; and since $\alpha \neq \alpha' \bmod q$ necessarily check (2) on α fails. Consequently if this sub-case occurs, $\mathcal{A}_2$'s view differs. We demonstrate that if the DE problem is hard, this case occurs with negligible probability.

Suppose that an algorithm $\mathcal{D}$ is able to cause this case to occur with non negligible probability $\mathfrak{p}$. Then we can devise an algorithm $\hat{\mathcal{S}}$ which uses $\mathcal{D}$ to break the DE assumption for $(\mathsf{Gen}_{\mathcal{SM}}, \mathsf{exp}_{\mathbf{G}})$. Algorithm $\hat{\mathcal{S}}$ gets as input a DE challenge point $Q = xP$ and the description $\mathcal{SM}$ of a subset membership problem, and must output hp, $(u_1, \mathsf{hash}(\mathsf{hk}', u_1) f^x)$ and $(u_2, \mathsf{hash}(\mathsf{hk}', u_2) f^x)$ where $\mathsf{hp} = \mathsf{projkg}(\mathsf{hk}')$; $u_1, u_2 \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$; $u_1 \neq u_2$; and $u_1/u_2 \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$. Precisely $\hat{\mathcal{S}}$ works as $\mathcal{S}$ would in Game_3, interacting with $\mathcal{D}$ instead of $\mathcal{A}_2$, the only difference being that instead of using the EC-DSA public key it receives from $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$, $\hat{\mathcal{S}}$ uses the DE challenge Q. Upon receiving c_3 from $\mathcal{D}$, $\hat{\mathcal{S}}$ computes $c_1 \leftarrow \mathsf{EvalScal}(\mathsf{hp}, \mathsf{EvalSum}(\mathsf{hp}, c_3, -k_2^{-1} m'), k_2 r^{-1})$. Finally $\hat{\mathcal{S}}$ computes $c_2 := (u_2, e_2) = c_{\mathsf{key}} \odot (1, f^{x_2})$ and outputs hp, c_1, c_2 to its' own double encoding challenger.

Analysis. Let us denote x_2, k_2 the values $\hat{\mathcal{S}}$ extracts from its interactions with $\mathcal{D}$. We further denote $x_1 := x \cdot x_2^{-1}$ (unknown to $\hat{\mathcal{S}}$). $\hat{\mathcal{S}}$ samples $\mathsf{hk} \hookleftarrow \mathsf{hashkg}(\mathcal{SM})$, and computes $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$. It then samples $\tilde{x}_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and computes $c_{\mathsf{key}} \leftarrow (u, \mathsf{hash}(\mathsf{hk}, u) \cdot f^{\tilde{x}_1})$ which can be interpreted as $(u, \mathsf{hash}(\mathsf{hk}', u) \cdot f^{x_1})$. Thus $c_2 = (u_2, e_2) = c_{\mathsf{key}} \odot (1, f^{x_2}) = (u, \mathsf{hash}(\mathsf{hk}', u) \cdot f^x)$, where $u_2 \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$ by construction. When $\hat{\mathcal{S}}$ receives c_3 from $\mathcal{D}$, with probability $\mathfrak{p}$, using decryption key hk', c_3 decrypts to $\alpha' = k_2^{-1}(m' + r x_1 x_2) \bmod q$. $\hat{\mathcal{S}}$ does not know hk', but using the homomorphic properties of the PKE, $\hat{\mathcal{S}}$ computes $c_1 := (u_1, e_1) \leftarrow (u_1, \mathsf{hash}(\mathsf{hk}', u_1) f^x)$. Since we ruled out the case 1. (where $\alpha = \alpha' \bmod q$), necessarily $u_1 \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$. And since we ruled out the case 2. (where $\alpha - \alpha' = k_2^{-1} r x_2 (\tilde{x}_1 - x_1) \bmod q$), necessarily $u_1/u_2 \in \widehat{\mathcal{X}}\backslash\widehat{\mathcal{L}}$. Thus with probability $\mathfrak{p}$, $\hat{\mathcal{S}}$ breaks the double encoding assumption, and consequently $\mathfrak{p} \leqslant \delta_{\mathsf{de}}$, which concludes that this case occurs with probability $\leqslant \delta_{\mathsf{de}}$.

(c) Else one of the checks on α passes.

   i. If $\alpha = k_2^{-1}(m' + r x_1 x_2) \bmod q$, then since $\alpha \neq \alpha' \bmod q$ necessarily check (2) on α' fails. However if this occurs, since $\mathcal{S}$ has extracted k_2, x_2 from the ZK proofs, it can compute x_1 from α, thereby breaking the DL problem in $\mathbf{G}$. This occurs with probability $\leqslant \delta_{\mathsf{dl}}$.

   ii. If $\alpha = k_2^{-1}(m' + r\tilde{x}_1 x_2) \bmod q$, then since $\alpha - \alpha' \neq k_2^{-1} r x_2 (\tilde{x}_1 - x_1) \bmod q$ necessarily check (2) on α' fails. Let us prove that information theoretically, this can not happen with probability greater than $1/q + \delta_s$. For clarity, we first recall the expression of c_{key} received by $\mathcal{A}_2$:

$$c_{\mathsf{key}} = (zy, \mathsf{projhash}(\mathsf{hp}, z)\mathsf{hash}(\mathsf{hk}, y) f^{\tilde{x}_1}) = (zy, \mathsf{projhash}(\mathsf{hp}, z) f^{(\tilde{x}_1 + b\gamma)})$$

   where $z \in \mathcal{L}$, $y \in \langle \Upsilon \rangle$, and $b \in (\mathbf{Z}/q\mathbf{Z})^*$ are unique, and $\mathsf{hash}(\Psi, y) = f^b$. We also recall the expression of c_3, sent by $\mathcal{A}_2$ to $\mathcal{S}$. Since c_3 decrypts to α with decryption key hk, we can write:

$$c_3 = (z_3 y_3, \mathsf{projhash}(\mathsf{hp}, z_3) f^{\alpha + b_3\gamma})$$

where $z_3 \in \widehat{\mathcal{L}}$, $y_3 \in \langle \widehat{\Upsilon} \rangle$, and $b_3 \in (\mathbf{Z}/q\mathbf{Z})^*$ are unique, and $\mathsf{hash}(\Psi, y_3) = f^{b_3}$. Let us denote $\pi_0 := \tilde{x}_1 + b\gamma \bmod q$, and $\pi_1 := \alpha + b_3\gamma \bmod q$. For this case to occur, it must hold that $\alpha = k_2^{-1}(m' + r\tilde{x}_1 x_2) \bmod q$, so

$$\begin{aligned} & \pi_1 = k_2^{-1}(m' + r\tilde{x}_1 x_2) + b_3\gamma \bmod q \\ \Leftrightarrow \quad & \tilde{x}_1 = (k_2\pi_1 - m' - k_2 b_3\gamma)(x_2 r)^{-1} \bmod q \end{aligned}$$

Substituting γ for $b^{-1}(\pi_0 - \tilde{x}_1)$ yields:

$$\begin{aligned} & \tilde{x}_1 = (k_2\pi_1 - m' - k_2 b_3 b^{-1}(\pi_0 - \tilde{x}_1))(x_2 r)^{-1} \bmod q \\ \Leftrightarrow \quad & \tilde{x}_1(1 - k_2 b_3 (b x_2 r)^{-1}) = (k_2\pi_1 - m' - k_2 b_3 b^{-1}\pi_0)(x_2 r)^{-1} \bmod q. \end{aligned}$$

As we dealt with $b_3 = k_2^{-1} r x_2 b \bmod q$ in case 2, here $b_3 \neq k_2^{-1} r x_2 b \bmod q$, and $1 - k_2 b_3 (b x_2 r)^{-1}$ is invertible mod q so we can write:

$$\tilde{x}_1 = (k_2\pi_1 - m' - k_2 b_3 b^{-1}\pi_0)(x_2 r)^{-1}(1 - k_2 b_3 (b x_2 r)^{-1})^{-1} \bmod q, \tag{5.2}$$

where π_0, b are fixed by c_{key}; π_1, b_3 are fixed by c_3; and m', r, k_2, x_2 are also fixed from $\mathcal{A}_2$'s view. So given $\mathcal{A}_2$'s view and $\mathcal{A}_2$'s output c_3, all the terms on the right hand side of Eq. (5.2) are fixed. However, given Q, hp and c_{key} (which is all the information $\mathcal{A}_2$ gets prior to outputting c_3), the δ_s-smoothness of H ensures that $\tilde{x}_1$ follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. For the current case to occur, Eq. (5.2) must hold, thus from being given a view where $\tilde{x}_1$ follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, $\mathcal{A}_2$ has succeeded in fixing this random variable to be the exact value sampled by $\mathcal{S}$. This occurs with probability $\leqslant 1/q + \delta_s$.

Combining the above, we get that Game_2 and Game_3 differ from $\mathcal{A}_2$'s view if and only if we are in case 3. (b) or 3. (c), which occur with probability $\leqslant 1/q + \delta_s + \delta_{\mathsf{de}} + \delta_{\mathsf{dl}}$. Thus:

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| \leqslant 1/q + \delta_s + \delta_{\mathsf{de}} + \delta_{\mathsf{dl}}.$$

Game_3 to Game_4. In Game_4 check (2) is removed. Both games differ if and only if check (1) fails in both of them, while check (2) passes. If this happens $\mathcal{S}$ has decrypted c_3 to the value $\alpha = k_2^{-1}(m' + r x_1 x_2) \bmod q$. Since $\mathcal{S}$ has extracted k_2, x_2 from the simulated proofs of knowledge, r from the EC-DSA signature it received and knows m', it can compute x_1 from α, thereby computing the DL of point Q. Thus distinguishing these games reduces to the hardness of breaking the DL problem in $\mathbf{G}$. We conclude that:

$$|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| \leqslant \delta_{\mathsf{dl}}.$$

Game_4 to Game_5. The change here is exactly that between Game_1 and Game_2, thus both games are indistinguishable under the hardness of the subset membership problem and:

$$|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_4]| \leqslant \delta_{\mathcal{L}}.$$

Game_5 to Game_6. The change here is exactly that between Game_0 and Game_1, thus both games are perfectly indistinguishable, even for an unbounded adversary, thus:

$$|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_5]| = 0.$$

Real/Ideal executions. Putting together the above probabilities, we get that:

$$|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_0]| \leqslant 2\delta_{\mathscr{L}} + \delta_{\mathsf{de}} + 2\delta_{\mathsf{dl}} + 1/q + \delta_s,$$

and so, assuming the hardness of the subset membership problem, the smoothness of H, and the hardness of the double encoding problem, it holds that the real and ideal executions are computationally indistinguishable from $\mathcal{A}_2$'s view, which concludes the proof of the theorem. □

5.2.6 Instantiation from the HSM-CL Based PHF

The protocol results from a direct application of the generic construction of Fig. 5.5 using the $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ projective hash function (*cf.* running example 2 of Chapter 3). The ideal functionality $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R}_{\mathsf{cl\text{-}dl}}}$ is instantiated using the zero-knowledge proof of knowledge $\Sigma_{\mathsf{cl\text{-}dl}}$ for relation $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ of Section 3.6.1. We first explicit the DE problem for $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$.

The double encoding problem. The DE problem (*cf.* Definition 5.4) is δ_{de}-hard for $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ and the function $\mathsf{exp}_{\mathbf{G}} : x \mapsto xP$ if for any PPT $\mathcal{A}$, it holds that:

$$\delta_{\mathsf{de}} \geqslant \Pr\left[\begin{array}{c|l} u_1, u_2 \in \widehat{G} \setminus \widehat{G}^q, & \mathsf{pp}_{\mathbf{G}} := (\mathbf{G}, P, q) \\ u_2 \cdot u_1^{-1} \in \widehat{G} \setminus \widehat{G}^q & \mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q) \\ \text{and } \mathsf{hp} = g_q^{\mathsf{hk}} & x \hookleftarrow \mathbf{Z}/q\mathbf{Z},\ Q \leftarrow x \cdot P \\ & (\mathsf{hp}, (u_1, u_1^{\mathsf{hk}} \cdot f^x), (u_2, u_2^{\mathsf{hk}} \cdot f^x)) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathbf{G}}, \mathsf{pp}_{\mathsf{CL}}, Q) \end{array}\right].$$

In Lemma 5.9 we demonstrate that $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is EC-DSA-friendly.

Lemma 5.9. Let λ be a positive integer, and let $(\mathbf{G}, P, q) \leftarrow \mathsf{Gen}_{\mathbf{G}}(1^\lambda)$. Let $\mathsf{exp}_{\mathbf{G}}$ be the function which to $x \in \mathbf{Z}/q\mathbf{Z}$ maps the EC point xP. Consider a generator $\mathsf{Gen}_{\mathsf{CL}}$ of a HSM-CL group with an easy DL subgroup, which, on input 1^λ and the prime q, outputs $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q)$. Let $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ be the resulting SMP and PHF (*cf.* running example 2 of Chapter 3). If $\widehat{\mathcal{D}}$ (from which the hashkg algorithm of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ samples hashing keys) is δ-close to $\mathcal{U}(\mathbf{Z}/\widehat{n}\mathbf{Z})$, and the DE problem is δ_{de}-hard for $(\mathcal{SM}_{\mathsf{hsm\text{-}cl}}, \mathsf{exp}_{\mathbf{G}})$ then $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is $(q, f, f, \delta_{\mathsf{hsm\text{-}cl}}, \delta_s, \delta_{\mathsf{de}})$-EC-DSA-friendly where $\delta_s \leqslant 2\delta$.

Proof. As demonstrated in Section 3.3.3, $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is homomorphically extended for public projection keys in $\widehat{G}$; key homomorphic; and $\mathbf{Z}$ is an abelian cyclic group. In Section 3.3.5 we saw that $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is (f, F)-decomposable, where f is of order q; and using the Solve algorithm of Definition 3.1 the mapping $m \in \mathbf{Z}/q\mathbf{Z}$ to f^m, and its inverse $\log_f$ are efficiently computable. By definition of the HSM-CL assumption $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is a $\delta_{\mathsf{hsm\text{-}cl}}$-hard SMP; and as demonstrated in Lemma 3.16, if $\widehat{\mathcal{D}}$ is δ-close to $\mathcal{U}(\mathbf{Z}/\widehat{n}\mathbf{Z})$, H is δ_s-smooth over G on F with $\delta_s \leqslant 2\delta$; finally the DE problem for $(\mathcal{SM}_{\mathsf{hsm\text{-}cl}}, \mathsf{exp}_{\mathbf{G}})$ is assumed to be $\delta_{\mathsf{de}}(\lambda)$-hard. □

The two party EC-DSA protocol from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is detailed in Fig. 5.6, and the following corollary states its security.

Corollary 5.10 (of Theorem 5.8)**.** If Gen is the generator of a HSM-CL group with easy DL subgroup F, and $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is $(q, f, f, \delta_{\mathsf{hsm\text{-}cl}}, \delta_s, \delta_{\mathsf{de}})$-EC-DSA-friendly, the protocol of Fig. 5.6 securely computes $\mathcal{F}_{\mathsf{ec\text{-}dsa}}$ in the $(\mathcal{F}_{\mathsf{zk}}, \mathcal{F}_{\mathsf{com\text{-}zk}})$-hybrid model in the presence of a malicious static adversary (under the ideal/real definition).

P_1	$\mathsf{IKeyGen}(\mathbf{G}, P, q)$	P_2
$(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$ $x_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $Q_1 \leftarrow x_1 P$		
	$\xrightarrow{(\mathsf{com\text{-}prove},1,Q_1,x_1)} \mathcal{F}^{\mathsf{R_{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{proof\text{-}receipt},1)}$	
		$x_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $Q_2 \leftarrow x_1 P$
P_1 aborts if $(\mathsf{proof}, 2, Q_2)$ not received.		
	$\xleftarrow{(\mathsf{proof},2,Q_2)} \mathcal{F}^{\mathsf{R_{dl}}}_{\mathsf{zk}} \xleftarrow{(\mathsf{prove},2,Q_2,x_2)}$	
	$\xrightarrow{(\mathsf{decom\text{-}proof},1)} \mathcal{F}^{\mathsf{R_{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{decom\text{-}proof},1,Q_1)}$	
$\mathsf{hk}, \rho \hookleftarrow \widehat{\mathcal{D}}_q$ $\mathsf{hp} \leftarrow g_q^{\mathsf{hk}}$		
$c_{\mathsf{key}} = (u, e) \leftarrow (g_q^\rho, \mathsf{hp}^\rho f^{x_1})$	$\xrightarrow{(\mathsf{prove},3,(\mathsf{hp},c_{\mathsf{key}},Q_1),(x_1,\rho))} \mathcal{F}^{\mathsf{R_{cl\text{-}dl}}}_{\mathsf{zk}} \xrightarrow{(\mathsf{proof},3,(\mathsf{hp},c_{\mathsf{key}},Q_1))}$	P_2 aborts if $(\mathsf{decom\text{-}proof}, 1, Q_1)$, $(\mathsf{proof}, 3,\ (\mathsf{hp}, c_{\mathsf{key}}, Q_1))$ not received or $\mathsf{hp} \notin \widehat{G}$.
$Q \leftarrow x_1 Q_2$		$Q \leftarrow x_2 Q_1$

P_1	$Sign(m, \mathsf{sid})$	P_2
$k_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $R_1 \leftarrow k_1 P$		
	$\xrightarrow{(\mathsf{com\text{-}prove},\mathsf{sid}\|\|1,R_1,k_1)} \mathcal{F}^{\mathsf{R_{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{proof\text{-}receipt},\mathsf{sid}\|\|1)}$	
		$k_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $R_2 \leftarrow k_2 P$
P_1 aborts if $(\mathsf{proof}, sid\|\|2, R_2)$ not received.	$\xleftarrow{(\mathsf{proof},\mathsf{sid}\|\|2,R_2)} \mathcal{F}^{\mathsf{R_{dl}}}_{\mathsf{zk}} \xleftarrow{(\mathsf{prove},\mathsf{sid}\|\|2,R_2,k_2)}$	
	$\xrightarrow{(\mathsf{decom\text{-}proof},\mathsf{sid}\|\|1)} \mathcal{F}^{\mathsf{R_{dl}}}_{\mathsf{com\text{-}zk}} \xrightarrow{(\mathsf{decom\text{-}proof},\mathsf{sid}\|\|1,R_1)}$	P_2 aborts if $(\mathsf{decom\text{-}proof}, sid\|\|1, R_1)$ not received. $m' \leftarrow H(m)$
$R = (r_x, r_y) \leftarrow k_1 R_2$ $r \leftarrow r_x \bmod q$		$R = (r_x, r_y) \leftarrow k_2 R_1$ $r \leftarrow r_x \bmod q$ $\tau \hookleftarrow \widehat{\mathcal{D}}_q$ $c_1 = (c_{1,1}, c_{1,2}) \leftarrow (g_q^\tau, \mathsf{hp}^\tau f^{k_2^{-1} m'})$ $c_2 = (c_{2,1}, c_{2,2}) \leftarrow (u^{k_2^{-1} r x_2}, e^{k_2^{-1} r x_2})$ $c_3 = (c_{3,1}, c_{3,2}) \leftarrow (c_{1,1} c_{2,1}, c_{1,2} c_{2,2})$
	$\xleftarrow{\quad c_3 \quad}$	
$\alpha \leftarrow \mathsf{Solve}(c_{3,2}/c_{3,1}^{\mathsf{hk}})$ $\hat{s} \leftarrow \alpha \cdot k_1^{-1}$ $s \leftarrow \min(\hat{s}, q - \hat{s})$ If not $\mathsf{Verif}(Q, m, (r, s))$ P_1 aborts Else Return (r, s)		

Figure 5.6: Simulation secure two-party EC-DSA from the HSM-CL assumption.

Efficiency Improvements

Consider a positive integer d. Using the ZKPoK for $\mathsf{R}_{\mathsf{lcm\text{-}dl}}$ of Section 3.6.2, with an lcm $y \leftarrow \mathsf{lcm}(1, 2, 3, \ldots, 2^d - 1)$, instead of $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ to prove $c_{\mathsf{key}} = (u, e)$ is well formed, one can divide by d the number of rounds for this ZKPoK. This entails a few modification for the overall protocol. Indeed with a proof based on the lcm trick, P_2 is only convinced that c_{key}^y is well formed. Consequently P_2 must raise each component of c_{key} to the power y before performing homomorphic operations, and for correctness P_2 must also multiply the message m' by $y \bmod q$. When P_1 decrypts it multiplies the decrypted value by $y^{-1} \bmod q$.

On the choice of d**.** At the end of this ZKPoK for $\mathsf{R}_{\mathsf{lcm\text{-}dl}}$, during which P_1 proves to P_2 that it knows x_1 and ρ satisfying $u = g_q^\rho$ and $e = \mathsf{hp}^\rho f^{x_1}$, P_2 is only convinced that $u^y = g_q^{y\rho}$ and $e^y = \mathsf{hp}^{y\rho} f^{yx_1}$, where $y = \mathsf{lcm}(1, \ldots, 2^d - 1)$. So for P_2 to perform operations on c_{key} which are returned to P_1, without risking leaking information to P_1, P_2 must first raise u and e to the power y at the end of the key generation phase.

Thus, as explained at the end of Section 3.6.2, d cannot be chosen arbitrarily large, as the exponentiation of ciphertext components to the power y must take reasonable time. Hence we set the challenge set to be $\mathcal{C} := \{0, 1\}^{10}$, and $y = \mathsf{lcm}(1, \ldots, 2^{10} - 1)$, which is a 1479 bits integer, so exponentiating to the power y remains efficient. To achieve a soundness error of $2^{-\lambda}$ the protocol must be repeated $\lambda/10$ times.

Trading statistical soundness for computational soundness. If one assumes the LO and SR problems are hard for $\mathsf{Gen}_{\mathsf{CL}}$ (*cf.* Section 3.2.4), and if the (deterministic) generator g_q used in the protocol of Fig. 5.6 is replaced by a random power $\hat{g}_q$ of g_q, then one can use the ZKAoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ of Fig. 3.10. For this generator to appear random to all parties, on can use a public coin setup process (e.g. that of [LM19, Section 8.1]), which provides a description of G, F and G^q and of a random generator $\hat{g}_q$ of G^q. Alternatively, one can have multiple parties jointly run an interactive setup protocol, so that the resulting generator $\hat{g}_q$ is random to all parties (see Section 5.3.2 for a detailed description of this interactive protocol).

Note that as presented, the ZKAoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ of Fig. 3.10 is only secure against honest verifiers. In order to plug it into the protocol of Fig. 5.6, the ZKAoK must be made secure against malicious adversaries. To this end we can use the techniques mentioned in Section 3.7 of [Gro04, GMY06].

5.2.7 Implementation and Efficiency Comparisons

In this section we compare an implementation of our protocol of Fig. 5.6 (using the original ZKPoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$) with Lindell's protocol of [Lin17a]. For a fair comparison, we implement both protocols with the Pari C Library ([PAR20]), as this library handles arithmetic in class groups, $\mathbf{Z}/n\mathbf{Z}$ and elliptic curves. In particular, in this library, exponentiations in $\mathbf{Z}/n\mathbf{Z}$ and in class groups both use the same sliding window method. The running times are measured on a single core of an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz (even though key generation can easily be parallelised). We do not implement commitments (this does not bias the comparison as they appear with equal weight in both schemes), and only measure computation time and do not include communication (again this is fair as communication is similar).

We ran our implementation on the standard NIST curves P-256, P-384 and P-521, corresponding to levels of security 128, 192 and 256. For the encryption scheme, we start with a 112 bit security, as in [Lin17a], but also study the case where its level of security matches the security of the EC.

Again as in [Lin17a], we fixed the number of rounds in ZKPs to reach a statistical soundness error of 2^{-40}. For the distributions we also set the parameters to get statistical error of 2^{-40}. The ZKP for R_{dl} are implemented with Schnorr's protocol.

In the following, we review the theoretical complexity and experimental results of both schemes, before comparing them. In terms of theoretical complexity, exponentiations in the encryption schemes dominate the computation as EC operations are much cheaper. Thus, we only count these exponentiations; we will see this results in an accurate prediction of experimental timings.

Lindell's Protocol with Paillier's Encryption Scheme

The key generation uses on average 360 Paillier exponentiations (of the form $r^N \bmod N^2$) but not all of them are full exponentiations. The signing phase uses only 2 Paillier exponentiations.

The timings corresponds to the mean of several experiments (30 to 1000 depending on the security level). The timings are quite stable other than the generation of the RSA modulus in the key generation. We use standard RSA integers (*i.e.*, not strong prime factors) as this would be too slow for high security levels. For example, for 256 bits security (15360 bits modulus), the generation of the modulus takes 95 seconds (mean of 30 experiments) with a standard deviation of 56s. For the rest of the protocol the experimental timings are roughly equal to the number of exponentiations multiplied by the cost of one exponentiation.

The results are summarised in Fig. 5.7a. Timings are given in milliseconds and sizes in bits. The columns corresponds to the elliptic curve used for EC-DSA, the security parameter in bits for the encryption scheme, the timings of the key generation and of the signing phase and the total communication in bits for each phase. Modulus sizes are set according to the NIST recommendations. Note that for the first line, we use a 2048 bits modulus[4] as in [Lin17a] and obtain similar experimental results.

Curve	Sec. Param.	Keygen (ms)	Signing (ms)	Keygen (b)	Signing (b)
P-256	112	**2 133**	**20**	881 901	5 636
P-256	128	**6 340**	**49**	1 317 101	7 684
P-384	192	65 986	**437**	3 280 429	17 668
P-521	256	429 965	2 415	6 549 402	33 832

(a) Lindell's Protocol with Paillier

Curve	Sec. Param.	Keygen (ms)	Signing (ms)	Keygen (b)	Signing (b)
P-256	112	5 521	101	**178 668**	**4 748**
P-256	128	9 350	170	**227 526**	**5 706**
P-384	192	**35 491**	649	**427 112**	**10 272**
P-521	256	**103 095**	**1 888**	**688 498**	**16 078**

(b) Our Protocol with HSM-CL

Figure 5.7: Experimental results (timings in ms, sizes in bits)

[4] As opposed to 4096 as preconised by the NIST.

Our Protocol from $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$

The key generation uses a total of 160 class group exponentiations (of the form g_q^r in the class group of discriminant $\Delta_q = -q^3 \cdot \tilde{q}$). This corresponds to the 40 rounds of the $\mathsf{R}_{\mathsf{cl\text{-}dl}}$ ZKPoK of Fig. 3.7. Note that exponentiations in the group $F = \langle f \rangle$ are almost free as seen in Section 3.1.2.

We point out that using the lcm trick of Section 3.6.2 with an lcm of $y := (1, \dots, 2^{10} - 1)$, one would reduce to 4 the number of executions of the ZKPoK (to obtain the same soundness error of 2^{-40}), hence P_1 would only be performing 16 class group exponentiations, while P_2 would have to perform two class group exponentiations at the end of the key generation protocol (these can be performed offline).

Signing uses 3 class group exponentiations (one encryption and one decryption).

We use the same number of experiments as for Lindell's protocol. Here timings are very stable. Indeed during key generation, we only compute the public key $\mathsf{hp} \leftarrow g_q^{\mathsf{hk}}$ with one exponentiation, as the output of Gen (mainly the discriminant Δ_q of the class group and the generator g_q) is a common public parameter that only depends on the cardinality q of the elliptic curve. As a result this can be considered as an input of the protocol, as the same group can be used by all users. Moreover, doing this does not change the global result of the comparison with Lindell's protocol: the running time of Gen is dominated by the generation of $\tilde{q}$, a prime of size much smaller than the factors of the RSA modulus. So even if we add this running time in the Keygen column, this does not affect the results of our comparisons for any of the considered security levels.

The results are summarised in Fig. 5.7b. Timings are given in milliseconds and sizes in bits. The columns correspond to the elliptic curve used for EC-DSA, the security parameter in bits for the encryption scheme, the timings of the key generation and of the signing phase and the total communication in bits for each phase.

Comparison

Fig. 5.7 shows that Lindell's protocol is faster for both key generation and signing for standard security levels for the encryption scheme (112 and 128 bits of security) while our solution remains of the same order of magnitude. However for high security levels our solution becomes faster, in terms of key generation from a 192-bits security level, and for both key generation and signing from a 256-bits security level.

In terms of communication, our solution outperforms the scheme of Lindell for all levels of security by a factor 5 to 10 for Keygen. For Signing, we gain 15% for basic security to a factor 2 at 256-bits security level. In terms of rounds, our protocol uses 126 rounds for Keygen and Lindell's protocol uses 175 rounds, so we get a 28% gain. Both protocols use 7 rounds for Signing. This situation can be explained by the fact that, even though we use less than half the number of exponentiations in the key generation as we do not need a range proof, and though we can use smaller parameters (*cf.* Section 2.6), the group law is more complex in class groups (one multiplication in the group is slower). Consequently the key generation for our solution only takes less time from the 192 bits level (while being of the same order of magnitude below this level). For signing, we increase the cost by one exponentiation due to the Elgamal structure of the HSM-CL-based PKE. However, we note that one can pre-process this encryption by computing $(g_q^\tau, \mathsf{hp}^\tau)$ in an offline phase and computing $c_1 \leftarrow (g_q^\tau, \mathsf{hp}^\tau f^{k_2^{-1} m'})$ which results in only one multiplication in the online phase. As a result we would have only one exponentiation in the online signing for the decryption operation. The same holds for Lindell's protocol with Paillier. With this improvement both protocols take the same time for signing at the 192 bits level.

Increasing the number of rounds to obtain a 2^{-60} soundness error. This impacts only KeyGen, where both protocols use 40 iterations of ZKPs to achieve a 2^{-40} soundness error. Lindell's protocol performs 9 exponentiations per iteration while ours performs 4. Multiplying all timings by 3/2 should reflect achieving a 2^{-60} soundness error, and indeed this is what we observe in practice. Complexity is linear in the number of iterations and the ratio between our timings and those of [Lin17a] remains constant.

5.3 Full Threshold EC-DSA

In this section we present new techniques to realise efficient threshold variants of the EC-DSA signature scheme. Our resulting protocols are particularly efficient in terms of bandwidth consumption and, as several recent works (e.g. [GG18,LN18,DKLs19,CMP20,GKSS20,GG20]) allow to consider any threshold t such that $n \geqslant t+1$.

As mentioned previously, the difficulty when trying to devise a threshold variant of this scheme comes from the fact that one has to compute both $R = k^{-1}P$ and a multiplication of the two secret values k, x. The Gennaro and Goldfeder protocol [GG18] protocol addresses this as follows. Starting from two secrets $a = a_1 + \cdots + a_n$, $b = b_1 + \cdots + b_n$ additively shared among the parties (i.e. P_i holds a_i and b_i), players compute $ab = \sum_{i,j} a_i b_j$ by computing additive shares of each $a_i b_j$. This can be achieved via a simple two party protocol, originally proposed by Gilboa [Gil99] in the setting of two party RSA key generation, which parties execute in a pairwise way. Slightly more in detail, this latter protocol relies on linearly homomorphic encryption and Gennaro and Goldfeder implement it using Paillier's cryptosystem as underlying building block. This choice, however, becomes problematic when dealing with malicious adversaries, as Paillier plaintexts live in $(\mathbf{Z}/N\mathbf{Z})$ whereas EC-DSA signatures live in $\mathbf{Z}/q\mathbf{Z}$. To avoid inconsistencies, one needs to choose N significantly larger than q, so that no wrap arounds occur during the execution of the whole protocol. To prevent malicious behaviour, this also induces the need of expensive range proofs.

We revisit the [GG18] protocol and propose a new threshold EC-DSA variant based on the HSM-CL based encryption scheme $\Pi_{\mathsf{hsm\text{-}cl}}$ (*cf.* Fig. 3.5). In our protocol the aforementioned multiplication step can be done efficiently and without resorting to range proofs. We thus avoid all the required range proofs, while retaining comparable overall (computational) efficiency. Security – which does not degrade with the number of signatures queried by the adversary in the tu-cma game (*cf.* Definition 5.3 – relies on the assumptions and tools introduced in Chapter 3. As in many previous works on multiparty EC-DSA (*e.g.* [MR01, Lin17a, GG18]), the linearly homomorphic properties of $\Pi_{\mathsf{hsm\text{-}cl}}$ enables parties to perform operations collaboratively while keeping their inputs secret. To ensure no information leaks from this use of the PKE, parties must prove their ciphertexts are 'well formed'. To this end, we will use the efficient ZKAoK of Fig. 3.9, which proves knowledge of the plaintext and of the randomness used to compute a ciphertext for $\Pi_{\mathsf{hsm\text{-}cl}}$. In order to use this ZKAoK, we first introduce a slight modification to the HSM-CL assumption in Section 5.3.1. Then in Section 5.3.2 we explain how parties interactively set up the public parameters of $\Pi_{\mathsf{hsm\text{-}cl}}$. In Section 5.3.3 we present our (t, n)-threshold EC-DSA signing protocol, whose security is demonstrated in Section 5.3.4.

We compare the speed and communication costs of our protocol to [GG18] and to that of Lindell and Nof [LN18], these are the best performing pre-existing protocols using similar construction techniques which achieve the same functionality. Our comparisons show that for all considered security levels our signing protocol reduces the bandwidth consumption by factors varying between 4.4 and 9, while key generation is consistently two times less expensive. Moreover, we even outperform (for all security levels) the stripped down implementation of the [GG18] protocol where a number of range proofs are omitted[5]. We believe this to be an important aspect of our schemes. In terms of timings, though for standard levels of security our signing protocol is up to four times slower than that of [GG18], for higher levels of security the trend is inverted: for 256-bit security we are twice as fast as all other secure schemes considered[6].

Remark. In this section, we start from a slightly different formulation of centralised EC-DSA to that presented in Section 5.1.2. Namely in the Sign algorithm, one samples $k \hookleftarrow (\mathbf{Z}/q\mathbf{Z})^*$ and computes $R \leftarrow k^{-1}P$, while s is computed as $k(m' + rx)$. Since q is prime, both formulations of the signing protocol are equivalent, however this slight syntactic modification is more convenient for our method of distributing k in the full threshold protocol.

5.3.1 A Note on the Underlying Assumptions

In our full threshold protocol of Section 5.3.3, we use the zero-knowledge arguments of knowledge of Section 3.7. These guarantee that parties honestly compute their ciphertexts, and secures the protocol against malicious adversaries. As seen in Section 3.7, the soundness of these ZKAoK relies on the SR and LO assumptions (*cf.* Section 3.2.4). Precisely, they require that it be hard to compute roots of $\hat{g}_q$, where $\hat{g}_q$ is a random power of the generator g_q of G^q output by Gen. This further requires a small modification to the $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme (of Fig. 3.5), in that one substitutes g_q for $\hat{g}_q$ throughout the scheme. Accordingly, for our full threshold protocol we modify slightly the HSM-CL assumption by considering a random element $\hat{g}_q$ instead of using the somewhat deterministic generator g_q of G^q.

Definition 5.11 (HSM-CL assumption)**.** Let λ be a positive integer and let $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator for a group with an easy DL subgroup. Let $\mathcal{D}$ (resp. $\mathcal{D}_q$) be a distribution over the integers such that the distribution $\{g^x, x \hookleftarrow \mathcal{D}\}$ (resp. $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$) is at distance less than $\delta(\lambda)$ from the uniform distribution in G (resp. in G^q), for some $\delta(\lambda) = \mathsf{negl}(\lambda)$. Let $\mathcal{A}$ be an adversary for the HSM-CL problem, its advantage is defined as:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{hsm\text{-}cl}}(\lambda) \stackrel{\text{def}}{=} \Big| 2 \cdot \Pr\big[b = b^\star : (\tilde{s}, f, g_q, \widehat{G}, F) \leftarrow \mathsf{Gen}(1^\lambda, q), t \hookleftarrow \mathcal{D}_q, \hat{g}_q = g_q^t,\\ x \hookleftarrow \mathcal{D}, x' \hookleftarrow \mathcal{D}_q, b \hookleftarrow \{0,1\}, Z_0 \leftarrow g^x, Z_1 \leftarrow \hat{g}_q^{x'},\\ b^\star \leftarrow \mathcal{A}(q, \tilde{s}, f, g_q, \hat{g}_q, \widehat{G}, F, Z_b, \mathsf{Solve}(.))\big] - 1 \Big|\end{aligned}$$

The HSM-CL problem is $\delta_{\mathsf{hsm\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{hsm\text{-}cl}}(\lambda) \leqslant \delta_{\mathsf{hsm\text{-}cl}}(\lambda)$. We say the HSM-CL assumption holds for $\mathsf{Gen}_{\mathsf{CL}}$ (or the HSM-CL problem is hard for $\mathsf{Gen}_{\mathsf{CL}}$), if the HSM-CL problem is $\delta_{\mathsf{hsm\text{-}cl}}$-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\delta_{\mathsf{hsm\text{-}cl}}(\lambda) = \mathsf{negl}(\lambda)$.

Clearly instantiating the projective hash function $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ (*cf.* running example 2) using $\hat{g}_q$ instead of g_q does not change the properties proven for $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$. Throughout this section, when referring to $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$, we implicitly mean that we use $\hat{g}_q$ instead of g_q. In particular, this variant of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ satisfies the same smoothness and homomorphic properties as the original. Consequently instantiating the linearly homomorphic PKE $\Pi_{\mathsf{hsm\text{-}cl}}$ (*cf.* Fig. 3.5) using $\hat{g}_q$ instead of g_q also yields an $\mathsf{ind\text{-}cpa}$-secure PKE.

[5] As Gennaro and Goldfeder themselves point out in [GG18], omitting these proofs leaks information on the shared signing key. While they conjecture that this information is limited enough for the protocol to remain secure, no formal analysis is provided.

[6] But still twice as slow as the stripped down [GG18] protocol.

In addition to the HSM-CL assumption, we use the LO assumption for Gen which states that it is hard to find low order elements in the group $\widehat{G}$ (*cf.* Definition 3.7); and the SR assumption for class groups, which states that it is hard to find roots in $\widehat{G}$ of random elements of the subgroup $\langle g_q \rangle$ (*cf.* Definition 3.8). These assumptions allow us to use the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$ of Fig. 3.9 which significantly increase the challenge space of our proofs (compared to ZKPoKs), and thereby reduce the number of rounds in the protocol to achieve a satisfying soundness. This yields improvements both in terms of bandwidth and of computational complexity.

5.3.2 Interactive Setup for the HSM-CL Based Encryption Scheme

We first explain how parties interactively set up the public parameters of the HSM-CL based encryption scheme $\Pi_{\mathsf{hsm\text{-}cl}}$. This ensures all parties' view of $\hat{g}_q$ is that of a random element in G^q, so that the assumptions underlying the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$ of Fig. 3.9 hold. Though – for clarity – we describe this interactive setup as a separate protocol, it can be done in parallel to the $\mathsf{IKeyGen}$ protocol of threshold EC-DSA, thereby only increasing by one the number of rounds of the threshold signing protocol.

Generating a random generator $\hat{g}_q$. As mentioned in the previous section, in order to use the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$ it must hold that $\hat{g}_q$ is a random element of the subgroup $\langle g_q \rangle$ where $\mathsf{pp}_{\mathsf{CL}} := (\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(1^\lambda, q)$. Precisely if a malicious prover P^* could break the soundness of the ZKAoK, an adversary trying to break the SR problem, given input a random $\hat{g}_q$, should be able to feed this input to P^*, and use P^* to solve it's own challenge. Consequently, as the ZKAoK will be used peer-to-peer by all parties in the threshold EC-DSA protocol, they will collaboratively generate – in the interactive $\mathsf{IKeyGen}$ – the public parameters $(\tilde{s}, f, g_q, \widehat{G}, F)$, and a common $\hat{g}_q$ which is random to each party. We call this interactive sub-protocol ISetup, since it allows parties to collaboratively set up the public parameters for $\Pi_{\mathsf{hsm\text{-}cl}}$. All parties then use this $\hat{g}_q$ to compute their public keys and as a basis for $\Pi_{\mathsf{hsm\text{-}cl}}$.

In our instantiation from class groups described in Section 3.1.2, and in particular in our description of the group generator Gen of Fig. 3.1, given the pair of primes $\tilde{q}$ and q, the generation of the public parameters $\mathsf{pp}_{\mathsf{CL}}$ is somewhat deterministic. The only step which is not obviously deterministic is step 5 of the algorithm of Fig. 3.1, where it is not specified how one chooses the small prime r. This step can be made deterministic by initially setting r to be a fixed small value, and then incrementing r until it satisfies the required conditions (this takes polynomial time under the generalised Riemann hypothesis). Let us overload the notation $\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Gen}(\tilde{q}, q)$ to refer to this deterministic setup.

We note that in the generation of $\mathsf{pp}_{\mathsf{CL}}$ described in Fig. 3.1, the prime $\tilde{q}$ is produced using an algorithm $\mathsf{next\text{-}prime}$, which on input a random seed $r \in \mathbf{N}$ and a prime q, returns the next prime greater than r satisfying $q\tilde{q} \equiv -1 \pmod 4$ and $(q/\tilde{q}) = -1$.

We first define the functionality computed by ISetup, running in two steps.

Definition 5.12. For a number of parties n, and an integer A, ISetup consists of the following interactive protocols:

Step 1 $\langle (1^k, q) \rangle \rightarrow \langle \tilde{q} \rangle$ **or** $\langle \perp \rangle$ where $\perp$ is the error output, signifying the parties may abort the protocol, and $\tilde{q}$ is a k bit prime which is produced as prescribed in algorithm $\mathsf{Gen}(1^k, q)$ of Fig. 3.1.

Step 2 $\langle (\tilde{q}, q) \rangle \rightarrow \langle (\tilde{s}, f, g_q, \widehat{G}, F, \hat{g}_q) \rangle$ **or** $\langle \perp \rangle$ where $(\tilde{s}, g, f, g_q, \widehat{G}, G, F, G^q) \leftarrow \mathsf{Gen}(\tilde{q}, q)$; and denoting $\mathcal{D}_q$ a distribution as per Definition 5.11, $\hat{g}_q$ is computed as $t \hookleftarrow \mathcal{D}_q$, $\hat{g}_q \leftarrow g_q^t$.

Let us now describe our protocol instantiating ISetup of Definition 5.12. Let λ be a positive integer. Let $\mathcal{D}_q$ be a distribution over the integers such that the distribution $\{g_q^x, x \hookleftarrow \mathcal{D}_q\}$ is

P_i	$\mathsf{ISetup}(1^k, q)$	All players $\{P_j\}_{j\neq i}$
$r_i \hookleftarrow \{0,1\}^k$		
$[\mathsf{c}_i, \mathsf{d}_i] \leftarrow \mathsf{Com}(r_i)$	$\xRightarrow{\mathsf{c}_i}$	
	$\xRightarrow{\mathsf{d}_i}$	$r_i \leftarrow \mathsf{Open}(\mathsf{c}_i, \mathsf{d}_i)$
$\tilde{q} \leftarrow \mathsf{next\text{-}prime}(\bigoplus_{j=1}^{n} r_j, q)$		
Compute g_q from $q, \tilde{q}$		
$t_i \hookleftarrow \mathcal{D}_q$ and $g_i \leftarrow g_q^{t_i}$		
$(\tilde{\mathsf{c}}_i, \tilde{\mathsf{d}}_i) \leftarrow \mathsf{Com}(g_i)$	$\xRightarrow{\tilde{\mathsf{c}}_i}$	
	$\xRightarrow{\tilde{\mathsf{d}}_i}$	$g_i \leftarrow \mathsf{Open}(\tilde{\mathsf{c}}_i, \tilde{\mathsf{d}}_i)$
$\pi_i := \mathsf{ZKPoK}_{g_i}\{(t_i) : g_i = g_q^{t_i}\}$	$\xleftrightarrow{\pi_i}$	if a proof fails abort
$\hat{g}_q \leftarrow \prod_{j=1}^{n} g_q^{t_j} = \prod_{j=1}^{n} g_j$		

Figure 5.8: Threshold CL setup used in $\mathsf{IKeyGen}$

at distance less than $\delta(\lambda)$ from the uniform distribution in G^q for some $\delta(\lambda) = \mathsf{negl}(\lambda)$. For n parties to jointly run ISetup, they proceed as depicted in Fig. 5.8, i.e. performing the following steps:

Step 1 — Generation of public prime $\tilde{q}$ of bit-size k.

1. Each P_i samples a random $r_i \hookleftarrow \{0,1\}^k$, computes $(\mathsf{c}_i, \mathsf{d}_i) \leftarrow \mathsf{Com}(r_i)$ and broadcasts c_i.
2. After receiving $\{\mathsf{c}_j\}_{j\neq i}$, each P_i broadcasts d_i thus revealing r_i.
3. All players compute the common output $\tilde{q} \leftarrow \mathsf{next\text{-}prime}(\bigoplus_{j=1}^{n} r_j, q)$.

Step 2 — Generation of $\hat{g}_q$.

1. From $\tilde{q}$, and the order q of the EC, all parties can use the deterministic algorithm $\mathsf{Gen}(\tilde{q}, q)$ to compute the deterministic generator g_q.
2. Next each player P_i performs the following steps:
   (a) Sample a random $t_i \hookleftarrow \mathcal{D}_q$; compute $g_i \leftarrow g_q^{t_i}$; $(\tilde{\mathsf{c}}_i, \tilde{\mathsf{d}}_i) \leftarrow \mathsf{Com}(g_i)$, and broadcast $\tilde{\mathsf{c}}_i$.
   (b) Receive $\{\tilde{\mathsf{c}}_j\}_{j\neq i}$. Broadcast $\tilde{\mathsf{d}}_i$ thus revealing g_i.
   (c) Perform a ZKPoK of t_i such that $g_i = g_q^{t_i}$. If a proof fails, abort.
3. Each party P_i computes $\hat{g}_q := \prod_{j=1}^{n} g_j = g_q^{\sum t_j}$, and has output $(\tilde{s}, f, g_q, \widehat{G}, F, \hat{g}_q)$.

Theorem 5.13 states the security of the interactive protocol ISetup of Fig. 5.8.

Theorem 5.13. If the commitment scheme is non-malleable and equivocal; and the proofs π_i are zero knowledge proofs of knowledge of discrete logarithm in $\langle g_q \rangle$, then the protocol of Fig. 5.8 securely computes ISetup with abort, in the presence of a malicious adversary corrupting any $t < n$ parties, with point-to-point channels.

Proof. We describe a simulator $\mathcal{S}$ simulating P_1 against all the other (potentially corrupted) parties, since all parties play symmetric roles, this is without loss of generality. $\mathcal{S}$ gets as input

$\tilde{q}$ from the first step of the ideal functionality, and upon calling step 2 of ISetup with primes $(\tilde{q}, q)$, it receives $\tilde{s}, f, g_q, \widehat{G}, F, \hat{g}_q$ and some $t \in \mathbf{Z}$.

$\mathcal{S}$ must simulate Step 1 so that all players output this same $\tilde{q}$. Next $\mathcal{S}$ must simulate Step 2 so that each player P_i outputs $\tilde{s}, f, g_q, \widehat{G}, F, \hat{g}_q$.

Simulating step 1 — Generation of $\tilde{q}$.

1. $\mathcal{S}$ samples $r_1 \hookleftarrow \{0,1\}^k$, computes $(\mathsf{c}_1, \mathsf{d}_1) \leftarrow \mathsf{Com}(r_1)$ and broadcasts c_1.
2. $\mathcal{S}$ broadcasts d_1, revealing r_1, and receives $\{r_j\}_{j>1}$.
3. $\mathcal{S}$ samples r_1' uniformly at random in $\{0,1\}^k$, subject to the condition $\tilde{q} = \mathsf{next\text{-}prime}(r_1' \oplus \bigoplus_{j=2}^n r_j, q)$. Then $\mathcal{S}$ computes an equivocated decommitment d_1' which opens to r_1', rewinds the adversary to 2 and broadcasts d_1' instead of d_1.
4. All players compute the common output $\tilde{q} \leftarrow \mathsf{next\text{-}prime}(r_1' \oplus \bigoplus_{j=2}^n r_j, q)$.

Simulating step 2 — Generation of $\hat{g}_q$.

1. From $\tilde{q}$ and q all parties use $\mathsf{Gen}(\tilde{q}, q)$ to compute the deterministic generator g_q.
2. $\mathcal{S}$ (simulating P_1) does the following:
   (a) Sample $t_1 \hookleftarrow \mathcal{D}_q$; compute $g_1 \leftarrow g_q^{t_1}$; $(\tilde{\mathsf{c}}_1, \tilde{\mathsf{d}}_1) = \mathsf{Com}(g_1)$, and broadcast $\tilde{\mathsf{c}}_1$.
   (b) Receive $\{\tilde{\mathsf{c}}_j\}_{j \neq 1}$. Broadcast $\tilde{\mathsf{d}}_1$ thus revealing g_1.
   (c) Perform a ZKPoK for $\pi_1 := \mathsf{ZKPoK}_{g_1}\{t_1 : g_1 = g_q^{t_1}\}$.
   (d) Receive $\{\tilde{\mathsf{c}}_j\}_{j \neq 1}$, recover $g_j \leftarrow \mathsf{Open}(\tilde{\mathsf{c}}_j, \tilde{\mathsf{d}}_j)$ for each j.
   (e) Compute $h \leftarrow \prod_{j=2}^n g_j$; $g_1' \leftarrow \hat{g}_q \cdot h^{-1}$; and an equivocated decommitment d' opening to g_1', rewind the adversary to 2. (b) and broadcast d' instead of $\tilde{\mathsf{d}}_1$. In 2. (c) simulate the ZKPoK.
3. If all proofs are correct, the protocol goes along with $\leftarrow g_1' h = \hat{g}_q$.

Let us now demonstrate that if the commitment scheme is non-malleable and equivocal; and the proofs π_i are zero knowledge proofs of knowledge then a simulated execution of steps 1 and 2 above is – from the view of (potentially corrupted) parties $P_2, \ldots, P_n$ – indistinguishable from a real execution. Moreover when the simulation – on input $(\mathsf{pp}_{\mathsf{CL}}, \hat{g}_q)$, where $\mathsf{pp}_{\mathsf{CL}}$ is computed deterministically from primes $\tilde{q}$ and q – does not abort, all parties output $\tilde{q}$ in step 1, and $\hat{g}_q$ in step 2.

Step 1: The only difference between real and simulated protocols is the way r_1 is computed. In the simulation $\mathcal{S}$ does not know r_1, but it chooses a r_1' such that $\tilde{q} = \mathsf{next\text{-}prime}(r_1' \oplus \bigoplus_{j=2}^n r_j, q)$. Let $R = \bigoplus_{j=2}^n r_j$ and $H_q = \{x \in \{0,1\}^k : \mathsf{prev\text{-}prime}(\tilde{q}) \leqslant x \oplus R \leqslant \tilde{q} - 1\}$ be the set of all the elements x such that $\tilde{q} = \mathsf{next\text{-}prime}(x \oplus R, q)$. Since r_1 belongs to the set H_q, and it has been chosen uniformly at random, as long as r_1' is chosen uniformly at random in the same set, the real and simulated executions are indistinguishable.

Step 2: The only difference is in point 2. (e), where $\mathcal{S}$ computes g_1' instead of using g_1. Since g_1 and $\hat{g}_q \cdot h^{-1}$ follow the same distribution, real and simulated executions are indistinguishable.

Moreover, we observe that the simulation can fail in three points: in step 1 if someone refuses to decommit after rewinding and in step 2, if some π_i fails or if someone refuses to decommit after rewinding. Since the commitment scheme is non-malleable and equivocal, in Step 1 $\mathcal{S}$ can rewind and equivocate the commitment to r_1, and if there are no aborts, all parties decommit to their correct values. As a consequence, all parties output $\tilde{q}$ at the end of Step 1. In step 2, all parties compute the correct g_q using $\tilde{q}$ from the deterministic $\mathsf{Gen}(\tilde{q}, q)$, if not there is an abort caused by the soundness of the proof π_i corresponding to the corrupted P_i. Finally, if no abort has occurred, in 2. (e), then $\mathcal{S}$ can equivocate the decommitment to g_1 and all parties decommit to the correct values thanks to the non-malleability of the scheme. If no party refuses to decommit after rewinding, the protocol ends with $\hat{g}_q$ (and $\tilde{q}$ from step 1).

Thus the simulation is indistinguishable from a real execution of the protocol from the adversary's view, which concludes proof of Theorem 5.13. □

Remark. Given the simulator described in proof of Theorem 5.13, it is clear that for each execution of ISetup (*cf.* Fig. 5.8), which interactively sets the public parameters of the CL framework, our reduction for the SR problem can program the outputs $\tilde{q}$ and $\hat{g}_q$ if the reduction controls at least one uncorrupted player.

Indeed consider an adversary $\mathcal{A}$ for the SR problem for generator Gen. $\mathcal{A}$ gets as input a description of $\mathsf{pp}_{\mathsf{CL}}$ output by $\mathsf{Gen}(1^\lambda, q)$, which includes $\tilde{q}$ and the order q of the EC, and a random element $Y \in G^q$. Now all $\mathcal{A}$ needs to do is perform the same steps as $\mathcal{S}$ in proof of Theorem 5.13, substituting $\hat{g}_q$ for Y.

Remark. The randomness of $\tilde{q}$ is not crucial to the security of the EC-DSA protocol: conversely to RSA prime factors, here $\tilde{q}$ is public. Indeed in our ISetup algorithm, the output of $\mathsf{next\text{-}prime}$ is biased, but it allows for an efficient solution.

Instantiating the proofs π_i. In Step 2. 2.(c) of the ISetup protocol, each P_i computes $\pi_i \leftarrow \mathsf{ZKPoK}_{g_i}\,\{(t_i) : g_i = g_q^{t_i}\}$. In fact it suffices for them to compute $\mathsf{ZKPoK}_{g_i}\{(z_i) : g_i^y = g_q^{z_i}\}$, where $y \leftarrow \mathsf{lcm}(1, 2, 3, \ldots, 2^{10})$ using the lcm trick of Section 3.6.2. Then in Step 2.3. all players compute $\hat{g}_q \leftarrow (\prod_{j=1}^n g_j)^y$. The resulting $\hat{g}_q$ has the required properties to be plugged into the $\mathsf{IKeyGen}$ protocol. We use this modified interactive setup for our efficiency comparisons of Section 5.3.5.

5.3.3 Full Threshold EC-DSA Protocol

We now describe the overall (t, n)-threshold EC-DSA protocol. Participants run on input $(\mathbf{G}, q, P)$ used by the EC-DSA signature scheme. In Fig. 5.9, and in phases 1, 3, 4, 5 of Fig. 5.10, all players perform the same operations (on their respective inputs) w.r.t. all other parties, so we only describe the actions of some party P_i. In particular if P_i broadcasts some value v_i, implicitly P_i receives v_j broadcast by P_j for all $j \in [n]$, $j \neq i$. Broadcasts from P_i to all other players are denoted by double arrows, whereas peer-to-peer communications are denoted by single arrows. Conversely Phase 2 of Fig. 5.10 is performed by all pairs of players $\{(P_i, P_j)\}_{i \neq j}$. Each player will thus perform $(n-1)$ times the set of instructions on the left (performed by P_i on the figure) and $(n-1)$ times those on the right (performed by P_j).

Key generation.

We assume that prior to the interactive key generation protocol $\mathsf{IKeyGen}$, all parties run the ISetup protocol of Fig. 5.8 at the outcome of which they all output a common random generator $\hat{g}_q$. We denote by $\Pi_{\mathsf{hsm\text{-}cl}} = (\mathsf{KeyGen}_{\mathsf{hsm\text{-}cl}}, \mathsf{Enc}_{\mathsf{hsm\text{-}cl}}, \mathsf{Dec}_{\mathsf{hsm\text{-}cl}})$ the HSM-CL-based PKE of Fig. 3.5. Each party uses $\hat{g}_q$ to generate its' encryption key pair, and to verify the ZKAoK in the ISign protocol. In practice $\mathsf{IKeyGen}$ and ISetup can be ran in parallel, this increases the number of rounds in $\mathsf{IKeyGen}$ by 1 broadcast per party if the ZKPs are made non interactive, and by 2 broadcasts if it is performed interactively between players. The $\mathsf{IKeyGen}$ protocol (also depicted in Fig. 5.9) proceeds as follows:

P_i	$\mathsf{IKeyGen}(\mathbf{G}, P, q)$	All players $\{P_j\}_{j \neq i}$
$u_i \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$[\mathsf{kgc}_i, \mathsf{kgd}_i] \leftarrow \mathsf{Com}(u_i P)$		
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}_{\mathsf{hsm\text{-}cl}}(1^\lambda)$	$\xRightarrow{\mathsf{pk}_i \text{ and } \mathsf{kgc}_i}$	
	$\xRightarrow{\mathsf{kgd}_i}$	
Perform (t, n)-VSS share of u_i:		$Q_i \leftarrow \mathsf{Open}(\mathsf{kgc}_i, \mathsf{kgd}_i)$
$p_i(X) = u_i + \sum_{k=1}^{t} a_{i,k} X^k \bmod q$		
Denote $\{\sigma_{i,j} := p_i(j)\}_{j \in [n]}$		$Q = \sum_{i=1}^{n} Q_i$
and $\{V_{i,k} := a_{i,k} P\}_{k \in [t]}$	$\xrightarrow{\text{Send } \sigma_{i,j} \textit{ to } P_j}$	
	$\xRightarrow{\{V_{i,k}\}_{k \in [t]}}$	
$\{\sigma_{k,i}\}_k$ are additive shares of $x_i := \sum_{k \in [n]} p_k(i)$		
where $\{x_i\}_{i \in [n]}$ are (t, n) Shamir shares of x.		
$\pi_{\mathsf{kg},i} := \mathsf{ZKPoK}_{X_i}\{(x_i) : X_i = x_i P\}$	$\xleftrightarrow{\pi_{\mathsf{kg},i}}$	

Figure 5.9: Threshold Key Generation

1. Each P_i samples a random $u_i \hookleftarrow \mathbf{Z}/q\mathbf{Z}$; computes $[\mathsf{kgc}_i, \mathsf{kgd}_i] \leftarrow \mathsf{Com}(u_i P)$ and generates a pair of keys $(\mathsf{sk}_i, \mathsf{pk}_i)$ for $\Pi_{\mathsf{hsm\text{-}cl}}$. Each P_i broadcasts $(\mathsf{pk}_i, \mathsf{kgc}_i)$.
2. Each P_i broadcasts kgd_i. Let $Q_i \leftarrow \mathsf{Open}(\mathsf{kgc}_i, \mathsf{kgd}_i)$. The EC-DSA public key is set to $Q \leftarrow \sum_{i=1}^{n} Q_i$. Party P_i performs a (t, n) Feldman-VSS of u_i, with Q_i as the free term in the exponent. Each player adds the private shares received during the n Feldman VSS protocols. The resulting values x_i are a (t, n) Shamir secret sharing of the signing key x. Observe that all parties know $\{X_i := x_i P\}_{i \in [n]}$.
3. Each P_i proves in ZK that it knows x_i using Schnorr's protocol [Sch91].

Signing.

The signature generation protocol runs on input m and the output of the $\mathsf{IKeyGen}$ protocol of Fig. 5.9. We denote $S \subseteq [n]$ the subset of players which collaborate to sign m. Assuming $|S| = t$ one can convert the (t, n) shares $\{x_i\}_{i \in [n]}$ of x into (t, t) shares $\{w_i\}_{i \in S}$ of x using the appropriate Lagrangian coefficients. Since the $X_i = x_i P$ and Lagrangian coefficients are public values, all parties can compute $\{W_i := w_i P\}_{i \in S}$. After a brief discussion on the parameters required to securely instantiate the protocol, we describe the steps of the algorithm. A global view of the interactions is also provided in Fig. 5.10.

Setting the parameters. Note that in Phase 1 of the signature protocol, we use the ZKAoK for relation $\mathsf{R}_{\mathsf{Enc}}$ of Fig. 3.9. As explained in Section 3.7, if secret values are sampled from $\mathcal{D}_q = \mathcal{D}_{\mathbf{Z},\sigma'}$ for $\sigma' = \tilde{s}\sqrt{\lambda}$ (as prescribed for the $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme, *cf.* Section 3.5.3), then with overwhelming probability, these exponents live in the set $\{-10\tilde{s}\sqrt{\lambda}, \dots, 10\tilde{s}\sqrt{\lambda}\}$. We assume this is the case. Let us denote C the size of the challenge set used for the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$ of Fig. 3.9. According to Theorem 3.31, we thus set the bound $A \in \mathbf{N}$ of Fig. 5.10 to be an integer such that $10\tilde{s}\sqrt{\lambda}C/A$ is a negligible function of the security parameter λ. This ensures that the ZKAoK is indeed zero-knowledge.

Steps of the signing protocol.

P_i	**Phase 1**	All players $\{P_j\}_{j\neq i}$
$k_i, \gamma_i \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$r_i \hookleftarrow \{1, \dots, A-1\}$		
$c_{k_i} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, k_i; r_i)$		
$[\mathsf{c}_i, \mathsf{d}_i] \leftarrow \mathsf{Com}(\gamma_i P)$	$\xRightarrow{\mathsf{c}_i, c_{k_i}}$	if a proof fails, abort
$\pi_i := \mathsf{ZKAoK}_{\mathsf{pk}_i, c_{k_i}}\{(k_i, r_i) :$ $((\mathsf{pk}_i, c_{k_i}); (k_i, r_i)) \in \mathsf{R}_{\mathsf{Enc}}\}$	$\xleftrightarrow{\pi_i}$	
P_i	**Phase 2**	P_j
$\beta_{j,i}, \nu_{j,i} \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$B_{j,i} \leftarrow \nu_{j,i} \cdot P$		
$c_{\beta_{j,i}} \leftarrow \mathsf{Enc}(\mathsf{pk}_j, -\beta_{j,i})$		
$c_{\nu_{j,i}} \leftarrow \mathsf{Enc}(\mathsf{pk}_j, -\nu_{j,i})$		
$c_{k_j\gamma_i} \leftarrow \mathsf{EvalAdd}(\mathsf{EvalScal}(c_{k_j}, \gamma_i), c_{\beta_{j,i}})$		
$c_{k_j w_i} \leftarrow \mathsf{EvalAdd}(\mathsf{EvalScal}(c_{k_j}, w_i), c_{\nu_{j,i}})$	$\xrightarrow{c_{k_j\gamma_i}, c_{k_j w_i}, B_{j,i}}$	
		$\alpha_{j,i} \leftarrow \mathsf{Dec}(\mathsf{sk}_j, c_{k_j\gamma_i})$
		$\mu_{j,i} \leftarrow \mathsf{Dec}(\mathsf{sk}_j, c_{k_j w_i})$
		If $\mu_{j,i}P + B_{j,i} \neq k_j W_i$ then abort
$\delta_i \leftarrow k_i\gamma_i + \sum_{j\neq i}(\alpha_{i,j} + \beta_{j,i})$		
$\sigma_i \leftarrow k_i w_i + \sum_{j\neq i}(\mu_{i,j} + \nu_{j,i})$		
P_i	**Phase 3**	All players $\{P_j\}_{j\neq i}$
	$\xRightarrow{\delta_i}$	$\delta = \sum_{i\in S} \delta_i = k\gamma$
P_i	**Phase 4**	All players $\{P_j\}_{j\neq i}$
	$\xRightarrow{\mathsf{d}_i}$	$\Gamma_i \leftarrow \mathsf{Open}(\mathsf{c}_i, \mathsf{d}_i)$
$\pi_{\gamma_i} := \mathsf{ZKPoK}_{\Gamma_i}\{(\gamma_i) : \Gamma_i = \gamma_i P\}$	$\xleftrightarrow{\pi_{\gamma_i}}$	if a proof fails, abort
		$R \leftarrow \delta^{-1}(\sum_{i\in S} \Gamma_i)$
		let $R = (r_x, r_y)$ and $r \leftarrow r_x \bmod q$.
P_i	**Phase 5**	All players $\{P_j\}_{j\neq i}$
$s_i \leftarrow mk_i + r\sigma_i$		
$\ell_i, \rho_i \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$V_i \leftarrow s_i R + \ell_i P$ and $A_i \leftarrow \rho_i P$		
$[\widehat{\mathsf{c}}_i, \widehat{\mathsf{d}}_i] \leftarrow \mathsf{Com}(V_i, A_i)$	$\xRightarrow{\widehat{\mathsf{c}}_i}$	
$\widehat{\pi}_i := \mathsf{ZKPoK}_{(V_i, A_i)}\{(s_i, \ell_i, \rho_i) :$ $V_i = s_i R + \ell_i P \wedge A_i = \rho_i P\}$	$\xRightarrow{\widehat{\mathsf{d}}_i}$	
	$\xleftrightarrow{\widehat{\pi}_i}$	if a proof fails, abort
		$V \leftarrow -mP - rQ + \sum_{i\in S} V_i$
$U_i \leftarrow \rho_i V$ and $T_i \leftarrow \ell_i A$		and $A \leftarrow \sum_{i\in S} A_i$
$[\tilde{\mathsf{c}}_i, \tilde{\mathsf{d}}_i] \leftarrow \mathsf{Com}(U_i, T_i)$	$\xRightarrow{\tilde{\mathsf{c}}_i}$	
	$\xRightarrow{\tilde{\mathsf{d}}_i}$	if $\sum_{i\in S} T_i \neq \sum_{i\in S} U_i$ then abort.
	$\xRightarrow{s_i}$	$s \leftarrow \sum_{i\in S} s_i$,
		if (r, s) is not a valid signature, abort,
		else return (r, s).

Figure 5.10: Threshold signature protocol

Phase 1: Each party P_i samples $k_i, \gamma_i \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and $r_i \hookleftarrow \{1, \dots, A-1\}$ uniformly at random. It computes $c_{k_i} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, k_i; r_i)$, a ZKAoK π_i that the ciphertext is well formed, and $[\mathsf{c}_i, \mathsf{d}_i] \leftarrow \mathsf{Com}(\gamma_i P)$. Each P_i broadcasts $(\mathsf{c}_i, c_{k_i}, \pi_i)$.

Phase 2: *Intuition: denoting* $k := \sum_{i\in S} k_i$ *and* $\gamma := \sum_{i \in S} \gamma_i$ *it holds that* $k\gamma = \sum_{i,j\in S} k_j \gamma_i$ *and* $kx = \sum_{i,j \in S} k_j w_i$. *The aim of Phase 2 is to convert the multiplicative shares* k_j *and* γ_i *of* $k_j\gamma_i$ *(resp.* k_j *and* w_i *of* $k_j w_i$*) into additive shares* $\alpha_{j,i} + \beta_{j,i} = k_j \gamma_i$ *(resp.* $\mu_{j,i} + \nu_{j,i} = k_j w_i$*). Phase 2 is performed peer-to-peer between each pair* $\{(P_i, P_j)\}_{i \neq j}$*, so that at the end of the phase 2* P_i *knows* $\{\alpha_{i,j}, \beta_{j,i}, \mu_{i,j}, \nu_{j,i}\}_{j\in S, j\neq i}$.
Each peer-to-peer interaction proceeds as follows:

(a) P_i samples $\beta_{j,i}, \nu_{j,i} \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, and computes $B_{j,i} \leftarrow \nu_{j,i} \cdot P$. It uses the homomorphic properties of $\Pi_{\mathsf{hsm\text{-}cl}}$ and the ciphertext c_{k_j} broadcast by P_j in Phase 1 to compute $c_{k_j\gamma_i}$ and $c_{k_j w_i}$: encryptions under pk_j of $k_j\gamma_i - \beta_{j,i}$ and $k_j w_i - \nu_{j,i}$ respectively.

(b) P_i sends $(c_{k_j\gamma_i}, c_{k_j w_i}, B_{j,i})$ to P_j, who decrypts both ciphertexts to recover respectively $\alpha_{j,i}$ and $\mu_{j,i}$. If a decryption fails, P_j aborts.

(c) Since W_i is public, P_j verifies that P_i used the same share w_i as that used to compute the public key Q by checking $\mu_{j,i} \cdot P + B_{j,i} = k_j W_i$. If the check fails, P_j aborts.

P_i computes $\delta_i \leftarrow k_i \gamma_i + \sum_{j\neq i}(\alpha_{i,j} + \beta_{j,i})$ and $\sigma_i \leftarrow k_i w_i + \sum_{j \neq i}(\mu_{i,j} + \nu_{j,i})$.

Phase 3: Each P_i broadcasts δ_i. All players compute $\delta \leftarrow \sum_{i\in S} \delta_i$.

Phase 4: (a) Each P_i broadcasts d_i which decommits to Γ_i.

(b) Each P_i proves knowledge of γ_i s.t. $\Gamma_i = \gamma_i P$. All players compute $R \leftarrow \delta^{-1}(\sum_{i\in S}\Gamma_i)$ $= k^{-1} \cdot P$. Let $R = (r_x, r_y)$ and $r \leftarrow r_x \bmod q$.

Phase 5: (a) Each P_i computes $s_i = k_i m + \sigma_i r$, samples $\ell_i, \rho_i \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ uniformly at random, computes $V_i \leftarrow s_i R + \ell_i P$; $A_i \leftarrow \rho_i P$; and $[\widehat{\mathsf{c}}_i, \widehat{\mathsf{d}}_i] \leftarrow \mathsf{Com}(V_i, A_i)$. Each P_i broadcasts $\widehat{\mathsf{c}}_i$.

(b) Each party P_i decommits by broadcasting $\widehat{\mathsf{d}}_i$ along with a NIZKPoK of (s_i, ℓ_i, ρ_i) s.t. $(V_i = s_i R + \ell_i P) \wedge (A_i = \rho_i P)$. It checks all the proofs it gets from other parties. If a proof fails P_i aborts.

(c) All parties compute $V \leftarrow -mP - rQ + \sum_{i\in S} V_i$, $A \leftarrow \sum_{i\in S} A_i$. Each party P_i computes $U_i \leftarrow \rho_i V$, $T_i \leftarrow \ell_i A$ and the commitment $[\tilde{\mathsf{c}}_i, \tilde{\mathsf{d}}_i] \leftarrow \mathsf{Com}(U_i, T_i)$. It then broadcasts $\tilde{\mathsf{c}}_i$.

(d) Each P_i decommits to (U_i, T_i) by broadcasting $\tilde{\mathsf{d}}_i$.

(e) All players check $\sum_{i\in S} T_i = \sum_{i \in S} A_i$. If the check fails they abort.

(f) Each P_i broadcasts s_i s.t. all players can compute $s \leftarrow \sum_{i\in S} s_i$. They check that (r, s) is a valid EC-DSA signature, if so, they output (r, s), otherwise they abort the protocol.

5.3.4 Security

The security proof is a reduction to the unforgeability of standard EC-DSA. We demonstrate that if there exists a PPT algorithm $\mathcal{A}$ which breaks the threshold EC-DSA protocol of Figs. 5.9 and 5.10, then we can construct a forger $\mathcal{F}$ which uses $\mathcal{A}$ to break the unforgeability of standard EC-DSA. To this end $\mathcal{F}$ must simulate the environment of $\mathcal{A}$, so that $\mathcal{A}$'s view of its interactions with $\mathcal{F}$ are indistinguishable from $\mathcal{A}$'s view in a real execution of the protocol. Precisely, we show that if an adversary $\mathcal{A}$ corrupts $\{P_j\}_{j>1}$, one can construct a forger $\mathcal{F}$ simulating P_1 s.t. the output distribution of $\mathcal{F}$ is indistinguishable from $\mathcal{A}$'s view in an interaction with an honest party P_1 (all players play symmetric roles in the protocol so it is sufficient to provide a simulation for P_1). $\mathcal{F}$ gets as input an EC-DSA public key Q, and has access to a signing oracle for messages of its choice. After this query phase, $\mathcal{F}$ must output a forgery, i.e. a signature σ for a message m of its choice, which it did not receive from the oracle.

Simulating the key generation protocol

On input a public key $Q := xP$, the forger $\mathcal{F}$ must set up in its simulation with $\mathcal{A}$ this same public key Q (without knowing x). This will allow $\mathcal{F}$ to subsequently simulate interactively signing messages with $\mathcal{A}$, using the output of its' (standard) EC-DSA signing oracle. The main differences with the proof of [GG18] arise from the fact $\mathcal{F}$ knows it's own decryption key sk_1, but does not extract that of other players. As in our two party protocol, we use the PKE $\Pi_{\mathsf{hsm\text{-}cl}}$ which results from the $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ projective hash function, whose security is statistical, thus the fact $\mathcal{F}$ uses its' secret key does not compromise security, and we can still reduce the security of the protocol to the smoothness of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$. However as we do not prove knowledge of secret keys associated to public keys in the key generation protocol, $\mathcal{F}$ can not extract the decryption keys of corrupted players. The simulation is described in Fig. 5.11.

Simulating the signature generation

On input m, $\mathcal{F}$ must simulate the ISign protocol from $\mathcal{A}$'s view.

First observe that since, in its simulation, $\mathcal{F}$ never simulates the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$, it is sufficient that the argument of knowledge be honest verifier zero knowledge for our purposes. Indeed, the argument is only used in Phase 1 of the protocol, and $\mathcal{F}$ knows the value k_1 for which it is proving knowledge.

We let

$$\tilde{k_i} := \mathsf{Dec}(\mathsf{sk}_i, c_{k_i}),$$

which $\mathcal{F}$ can extract from the proof π_i performed in Phase 1, and

$$\tilde{k} := \sum_{i \in S} \tilde{k_i}.$$

We further denote:

$$k \in \mathbf{Z}/q\mathbf{Z} \text{ the value satisfying } R = k^{-1}P \text{ in Phase 4 of the signing protocol.}$$

Notice that if any of the players mess up the computation of R by revealing wrong shares δ_i, we may have $k \neq \tilde{k} \bmod q$. As in [GG18], we distinguish two types of executions of the protocol:

$$\text{an execution where } \tilde{k} = k \bmod q \text{ is said to be } \textit{semi-correct}, \\ \text{whereas} \\ \text{an execution where } \tilde{k} \neq k \bmod q \text{ is } \textit{non semi-correct}.$$

Both executions will be simulated differently. At the end of Phase 4, when both simulations diverge, $\mathcal{F}$ knows k and $\tilde{k}$, so it can detect if it is in a semi-correct execution or not and chose how to simulate P_1.

We point out that $\mathcal{F}$ does not know the secret share w_1 of x associated with P_1, but it knows the shares $\{w_j\}_{j\in S, j\neq 1}$ of all the other players. Indeed $\mathcal{F}$ can compute these from the

1. $\mathcal{F}$ receives a public key Q from it's EC-DSA challenger.
2. Repeat the following steps (by rewinding $\mathcal{A}$) until $\mathcal{A}$ sends correct decommitments for $P_2, \ldots, P_n$ on both iterations.
3. $\mathcal{F}$ selects a random value $u_1 \in \mathbf{Z}/q\mathbf{Z}$, computes $[\mathsf{kgc}_1, \mathsf{kgd}_1] \leftarrow \mathsf{Com}(u_1 P)$ and broadcasts kgc_1. $\mathcal{F}$ receives $\{\mathsf{kgc}_j\}_{j \in [n], j \neq 1}$.
4. $\mathcal{F}$ broadcasts kgd_1 and receives $\{\mathsf{kgd}_j\}_{j \in [n], j \neq 1}$. For $i \in [n]$, let $Q_i \leftarrow \mathsf{Open}(\mathsf{kgc}_i, \mathsf{kgd}_i)$ be the revealed commitment value of each party. Each player performs a (t, n) Feldman-VSS of the value Q_i, with Q_i as the free term in the exponent.
5. $\mathcal{F}$ samples a $\Pi_{\mathsf{hsm\text{-}cl}}$ key pair $(\mathsf{pk}_1, \mathsf{sk}_1) \hookleftarrow \mathsf{KeyGen}(1^\lambda)$.
6. $\mathcal{F}$ broadcasts pk_1 and receives the public keys $\{\mathsf{pk}_j\}_{j \in [n], j \neq 1}$.
7. $\mathcal{F}$ rewinds $\mathcal{A}$ to the decommitment step and
   - equivocates P_1's commitment to $\widehat{\mathsf{kgd}}$ which opens to $\widehat{Q}_1 \leftarrow Q - \sum_{j=2}^{n} Q_j$.
   - simulates the Feldman-VSS with free term $\widehat{Q}_1$.
8. $\mathcal{A}$ broadcasts the decommitments $\{\widehat{\mathsf{kgd}}_j\}_{j \in [n], j \neq 1}$. Let $\{\widehat{Q}_j\}_{j=2\ldots n}$ be the committed values revealed by $\mathcal{A}$ at this point (which could be $\perp$ if $\mathcal{A}$ refuses to decommit).
9. All players compute the verification key $\widehat{Q} \leftarrow \sum_{i=1}^{n} \widehat{Q}_i$. If any $\widehat{Q}_i = \perp$ then $\widehat{Q} \leftarrow \perp$.
10. Each player P_i adds the private shares it received during the n Feldman VSS protocols to obtain x_i (the $\{x_i\}_{i \in [n]}$ are a (t, n) Shamir secret sharing of the secret key $x = \sum_i u_i$). Note that due to the free term in the exponent, the values $X_i = x_i P$ are public.
11. $\mathcal{F}$ simulates the ZKPoK that it knows x_1 corresponding to X_1. For $j \in [n]$, $j \neq 1$, $\mathcal{F}$ receives from $\mathcal{A}$ a ZKPoK of x_j such that $X_j = x_j \cdot P$, from which $\mathcal{F}$ can extract the values $\{x_j\}_{j \in [n], j \neq 1}$.

Figure 5.11: Simulating P_1 in $\mathsf{IKeyGen}$

values $\{x_j\}_{j \in [n], j \neq 1}$ extracted during key generation. It also knows $W_1 = w_1 P$ from the key generation protocol. Moreover $\mathcal{F}$ knows the encryption keys $\{\mathsf{pk}_j\}_{j \in S}$ of all players, and it's own decryption key sk_1.

In the simulation, depicted in Fig. 5.11, $\mathcal{F}$ aborts whenever $\mathcal{A}$ refuses to decommit any of the committed values, fails a ZK proof, if a decryption fails, or if the signature (r, s) does not verify.

Simulating a semi-correct execution

Lemma 5.14. Let C denote the size of the challenge space used in the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$. Assuming the SR assumption and the LO_C assumption hold for $\mathsf{Gen}_{\mathsf{CL}}$; $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is $\delta_{\mathsf{hsm\text{-}cl}}$-hard; $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is smooth; and the commitment scheme is non-malleable and equivocal; then on input m the simulation either outputs a valid signature (r, s) or aborts, and is computationally indistinguishable from a semi-correct real execution.

Phase 1:

As in a real execution, $\mathcal{F}$ samples $k_1, \gamma_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and $r_1 \hookleftarrow \{1, \dots, A-1\}$ uniformly at random. It computes $c_{k_1} \leftarrow \mathsf{Enc}(\mathsf{pk}_1, k_1; r_1)$, the associated ZKAoK π_1, and $[\mathsf{c}_1, \mathsf{d}_1] \leftarrow \mathsf{Com}(\gamma_1 P)$. It broadcasts $(\mathsf{c}_1, c_{k_1}, \pi_1)$ before receiving $\{\mathsf{c}_j, c_{k_j}, \pi_j\}_{j \in S, j \neq 1}$. $\mathcal{F}$ checks the proofs are valid and extracts the encrypted values $\{k_j\}_{j \in S, j \neq 1}$ from which it computes $\tilde{k} \leftarrow \sum_{i \in S} k_i$.

Phase 2:

(a) For $j \in S, j \neq 1$, $\mathcal{F}$ computes $\beta_{j,1}$, $c_{k_j \gamma_1}$ as in a real execution. However since it only knows $W_1 = w_1 P$ (but not w_1), it samples a random $\mu_{j,1} \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ and sets $c_{k_j w_1} \leftarrow \mathsf{Enc}(\mathsf{pk}_j, \mu_{j,1})$, and $B_{j,1} \leftarrow k_j \cdot W_1 - \mu_{j,1} \cdot P$. Then $\mathcal{F}$ sends $(c_{k_j \gamma_1}, c_{k_j w_1}, B_{j,1})$ to P_j.

(b) When $\mathcal{F}$ receives $(c_{k_1 \gamma_i}, c_{k_1 w_j}, B_{1,j})$ from P_j it decrypts as in a real execution of the protocol to obtain $\alpha_{1,j}$ and $\mu_{1,j}$

(c) $\mathcal{F}$ verifies that $\mu_{1,j} P + B_{1,j} = k_1 W_j$. If so, since $\mathcal{F}$ also knows k_1 and w_j, it computes $\nu_{1,j} \leftarrow k_1 w_j - \mu_{1,j} \bmod q$

$\mathcal{F}$ computes $\delta_1 \leftarrow k_1 \gamma_1 + \sum_{k \neq 1} \alpha_{1,k} + \sum_{k \neq 1} \beta_{k,1}$. Even though $\mathcal{F}$ cannot compute σ_1 (since it does not know w_1), it can compute :

$$\begin{aligned} \sum_{i>1} \sigma_i = \sum_{i>1} (k_i w_i + \sum_{j \neq i} \mu_{i,j} + \nu_{j,i}) \\ = \sum_{i>1} \sum_{j \neq i} (\mu_{i,j} + \nu_{j,i}) + \sum_{i>1} k_i w_i = \sum_{i>1} (\mu_{i,1} + \nu_{1,i}) + \sum_{i>1; j>1} k_i w_j \end{aligned}$$

since it knows all the values $\{k_j\}_{j \in S}$, $\{w_j\}_{j \in S, j \neq 1}$, it chooses the random values $\mu_{i,1}$ and it can compute all of the shares $\nu_{1,j} = k_1 w_j - \mu_{1,j} \bmod q$.

Phase 3:

cF broadcasts δ_1 and receives all the $\{\delta_j\}_{j \in S, j \neq 1}$ from $\mathcal{A}$. Let $\delta \leftarrow \sum_{i \in S} \delta_i$.

Phase 4:

(a) $\mathcal{F}$ broadcasts d_1 which opens to Γ_1, and $\mathcal{A}$ reveals $\{\mathsf{d}_j\}_{j \in S, j \neq 1}$ which open to $\{\Gamma_j\}_{j \in S, j > 1}$.

(b) $\mathcal{F}$ proves knowledge of γ_1 s.t. $\Gamma_1 = \gamma_1 P$. For $j \in S, j \neq 1$, $\mathcal{F}$ receives the ZKPoK π_{γ_j} from which it extracts γ_j. Then $\mathcal{F}$ computes $\gamma \leftarrow \sum_{i \in S} \gamma_i \bmod q$ and $k \leftarrow \delta \cdot \gamma^{-1} \bmod q$.

(c) If $k = \tilde{k} \bmod q$ (semi-correct execution), $\mathscr{F}$ proceeds as follows:

- $\mathscr{F}$ requests a signature (r, s) for m from its EC-DSA signing oracle.
- $\mathscr{F}$ computes $R \leftarrow s^{-1}(mP{+}rQ) \in \mathbf{G}$ (where $r = r_x \bmod q$ for $R = (r_x, r_y)$).
- $\mathscr{F}$ rewinds $\mathscr{A}$ to the decommitment step at Phase 4. (a) and equivocates P_1's commitment to open to $\widehat{\Gamma}_1 \leftarrow \delta R - \sum_{i>1} \Gamma_i$. It also simulates the ZKPoK of $\widehat{\gamma}_1$ satisfying $\widehat{\Gamma}_1 = \widehat{\gamma}_1 P$. Note that $\delta^{-1}(\widehat{\Gamma}_1 + \sum_{i>1} \Gamma_i) = R$.

  Phase 5: Now $\mathscr{F}$ knows $\sum_{j \in S, j \neq 1} s_j$ held by $\mathscr{A}$ since $s_j = k_j m + \sigma_j r$.

  – $\mathscr{F}$ computes s_1 held by P_1 as $s_1 \leftarrow s - \sum_{j \in S, j \neq 1} s_j$.

  – $\mathscr{F}$ continues the steps of Phase 5 as in a real execution.

(d) Else $k \neq \tilde{k} \bmod q$ (non-semi-correct execution), and $\mathscr{F}$ proceeds as follows:

- $\mathscr{F}$ computes $R \leftarrow \delta^{-1}(\sum_{i \in S} \Gamma_i) = k \cdot P$. Let $R = (r_x, r_y)$ and $r \leftarrow r_x \bmod q$.
- in Phase 5, $\mathscr{F}$ does the following:

  – sample $s_1, \ell_1, \rho_1 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, let $V_1 \leftarrow s_1 R + \ell_1 P$; $A_1 \leftarrow \rho_1 P$; $[\widehat{\mathsf{c}}_1, \widehat{\mathsf{d}}_1] \leftarrow \mathsf{Com}(V_1, A_1)$ and broadcast $\widehat{\mathsf{c}}_1$;

  – receive $\{\widehat{\mathsf{c}}_j\}_{j \neq 1}$ and broadcast $\widehat{\mathsf{d}}_1$. Prove knowledge of (s_1, ℓ_1, ρ_1) satisfying $V_1 = s_1 R + \ell_1 P$ and $A_1 = \rho_1 P$;

  – for $j \in S$, $j \neq 1$, receive $\widehat{\mathsf{d}}_j$ and the ZKPoK $\widehat{\pi}_i$ of (s_j, ℓ_j, ρ_j) relative to V_j and A_j;

  – let $V \leftarrow -mP - rQ + \sum_{i \in S} V_i$; $A \leftarrow \sum_{i \in S} A_1$; $T_1 \leftarrow \ell_1 A$ and sample $U_1 \hookleftarrow \mathbf{G}$;

  – let $[\tilde{\mathsf{c}}_1, \tilde{\mathsf{d}}_1] \leftarrow \mathsf{Com}(U_1, T_1)$ and broadcast $\tilde{\mathsf{c}}_1$. Upon receiving $\{\tilde{\mathsf{c}}_j\}_{j \neq 1}$ from $\mathscr{A}$, broadcast $\tilde{\mathsf{d}}_1$ and receive the $\{\tilde{\mathsf{d}}_j\}_{j \neq 1}$;

  – now since $\sum_{i \in S} T_1 \neq \sum_{i \in S} U_1$ both $\mathscr{A}$ and $\mathscr{F}$ abort.

Figure 5.11: Simulating P_1 in ISign

Proof. The differences between the real and simulated views are the following:

1. $\mathscr{F}$ does not know w_1. So for $j > 1$ it cannot compute $c_{k_j w_1}$ as in a real execution of the protocol. However under the SR and the LO$_C$ assumptions, $\mathscr{F}$ can extract k_j from proof π_j in Phase 1. It then samples $\mu_{j,1} \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, computes $B_{j,1} \leftarrow k_j W_1 - \mu_{j,1} P$, and $c_{k_j w_1} \leftarrow \mathsf{Enc}(\mathsf{pk}_j, \mu_{j,1})$. The resulting view of $\mathscr{A}$ is identical to an honestly generated one since both in real and simulated executions $\mu_{j,1}$ follows the distribution $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, while $B_{j,1}$ follows the uniform distribution in $\mathbf{G}$ and passes the check $B_{j,1} + \mu_{j,1} P = k_j W_1$ performed by $\mathscr{A}$. Moreover c_{k_j} was proven to be a valid ciphertext, so ciphertexts computed using homomorphic operations over c_{k_j} and fresh ciphertexts computed with pk_j follow identical distributions from $\mathscr{A}$'s view.

2. $\mathscr{F}$ computes $\widehat{\Gamma}_1 \leftarrow \delta \cdot R - \sum_{i>1} \Gamma_i$, and equivocates its commitment c_1 so that d_1 opens to $\widehat{\Gamma}_1$. Let us denote $\widehat{\gamma}_1 \in \mathbf{Z}/q\mathbf{Z}$ the value satisfying $\widehat{\Gamma}_1 = \widehat{\gamma}_1 P$. Here $\widehat{\gamma}_1$ is unknown to $\mathscr{F}$, but $\mathscr{F}$ can simulate a ZKPoK of $\widehat{\gamma}_1$. Let us further denote $\widehat{k} \in \mathbf{Z}/q\mathbf{Z}$ the randomness (unknown to $\mathscr{F}$) used by its' signing oracle to produce (r, s). It holds that $\delta = \widehat{k}(\widehat{\gamma}_1 + \sum_{j \in S, j>1} \gamma_j)$. Finally, let us denote $\widehat{k}_1 := \widehat{k} - \sum_{j \in S, j>1} k_j$. Since δ was made public in Phase 3, by

opening to $\widehat{\Gamma}_1 = \widehat{\gamma}_1 P$ instead of $\Gamma_1 = \gamma_1 P$, $\mathcal{F}$ is implicitly using $\widehat{k}_1 \neq k_1$, even though $\mathcal{A}$ received an encryption of k_1 in Phase 1. As we shall see the smoothness of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ and the hardness of the $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ subset membership problem ensure this change is unnoticeable to $\mathcal{A}$.

Claim. If $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is δ_s-smooth, and $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is $\delta_{\mathsf{hsm\text{-}cl}}$-hard, then no PPT adversary $\mathcal{A}$ – interacting with $\mathcal{F}$ – can notice the value of k_1 in the computation of R being replaced by the (implicit) value $\widehat{k}$ with probability greater than $2\delta_{\mathsf{hsm\text{-}cl}} + 3/q + 4\delta_s$.

Proof. To see this consider the following sequence of games. We denote E_i the probability $\mathcal{A}$ outputs 1 in $\mathsf{Game_i}$. The technique here is similar to that of proof of Theorem 5.8, when considering a corrupted P_2 and $\mathcal{S}$ simulated P_1.

Game_0 *to* Game_1. $\mathcal{F}$ uses the secret key sk_1 instead of the public key pk_1 and r_1 to compute $c_{k_1} \leftarrow (u_1, u_1^{\mathsf{sk}_1} f^{k_1})$ where $u_1 = g_q^{r_1}$. Both games are perfectly indistinguishable from $\mathcal{A}$'s view:

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_0]| = 0.$$

Game_1 *to* Game_2. In Game_2 one replaces the first element of c_{k_1} (in Game_1 this is $u_1 \in G^q$) with $\tilde{u}_1 \in G \backslash G^q$. There exist unique $r_1 \in \mathbf{Z}/s\mathbf{Z}$ and $b_1 \in \mathbf{Z}/q\mathbf{Z}$ such that, denoting $c_{k_1} = (\tilde{u}_1, \tilde{u}_1^{\mathsf{sk}_1} f^{k_1})$, one has $\tilde{u}_1 = g_q^{r_1} f^{b_1}$. Under the $\delta_{\mathsf{hsm\text{-}cl}}$-hardness of $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ it holds that:

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| \leqslant \delta_{\mathsf{hsm\text{-}cl}}.$$

Game_2 *to* Game_3. In Game_3 the points $Q = xP$ and $R = \widehat{k}^{-1}P$ come from the EC-DSA oracle, while in Game_2 they are computed as in the real protocol. As a result, the value k_1 encrypted in c_{k_1} is unrelated to $\widehat{k}$. Let us denote $\widehat{k}_1 := \widehat{k} - \sum_{j>1} k_j$, this is the value which – if used by $\mathcal{F}$ instead of k_1 – would lead to the joint computation of $R = \widehat{k}^{-1}P$.

To demonstrate that Game_2 and Game_3 are indistinguishable from $\mathcal{A}$'s view, we proceed similarly to as in proof of Theorem 5.8 (where we were also arguing the indistinguishability of Game_2 and Game_3). Indeed, we again observe that for a given public key, there are in fact q possible different secret keys. Given an invalid ciphertext, and the value $z \in \mathbf{Z}/q\mathbf{Z}$ to which it should decrypt, one fixes the value of the secret key (since it must be *the* secret key which decrypts to z). However if one only knows the public key, this invalid ciphertext could potentially encrypt *any value in* $\mathbf{Z}/q\mathbf{Z}$. The main difference here compared to proof of Theorem 5.8 is that the adversary does not – prior to seeing the invalid ciphertext – have any auxiliary information on the encrypted value (recall that in proof of Theorem 5.8, the adversary also knew the elliptic curve point Q satisfying $Q = zP$). Hence in this proof we need not rely on the DE assumption.

Hence, as in proof of Theorem 5.8, we start by considering a fixed $\widehat{\mathsf{sk}}_1 \in \mathbf{Z}$ satisfying the following equations:

$$\begin{cases} \widehat{\mathsf{sk}}_1 = \mathsf{sk}_1 \bmod \varpi, \\ \widehat{\mathsf{sk}}_1 = \mathsf{sk}_1 + {b_1}^{-1}(k_1 - \widehat{k}_1) \bmod q. \end{cases}$$

Note that the smoothness of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ over G on F ensures that such a $\widehat{\mathsf{sk}}_1$ exists (it is not necessarily unique). We can now see that in Game_3, c_{k_1} is an invalid encryption of both $\widehat{k}_1$ and k_1, for respective secret keys $\widehat{\mathsf{sk}}_1$ and sk_1, but for the same public key pk_1, indeed:

$$\begin{aligned} c_{k_1} &= (\tilde{u}_1, \tilde{u}_1^{\mathsf{sk}_1} f^{k_1}) = (g_q^{r_1} f^{b_1}, (g_q^{r_1} f^{b_1})^{\mathsf{sk}_1} \cdot f^{k_1}) = (g_q^{r_1} f^{b_1}, (g_q^{\mathsf{sk}_1})^{r_1} \cdot f^{k_1 + b_1 \mathsf{sk}_1}) \\ &= (g_q^{r_1} f^{b_1}, (g_q^{\widehat{\mathsf{sk}}_1})^{r_1} f^{\widehat{k}_1 + \widehat{\mathsf{sk}}_1 \cdot b_1}) = (\tilde{u}_1, \tilde{u}_1^{\widehat{\mathsf{sk}}_1} f^{\widehat{k}_1}) \end{aligned}$$

$\mathcal{A}$ receives the point Q, the public key $\mathsf{pk}_1 = g_q^{\mathsf{sk}_1}$, and c_{k_1} from $\mathcal{S}$ (at this point $\mathcal{A}$'s view is identical to its view in Game_2). The adversary $\mathcal{A}$ corrupting P_j then computes $c_{k_1\gamma_j}$ and $c_{k_1 w_j}$. To simplify notations we denote $c_\alpha := (u_\alpha, e_\alpha) = c_{k_1\gamma_j}$ and $c_\mu := (u_\mu, e_\mu) = c_{k_1 w_j}$. $\mathcal{A}$ then sends c_α and c_μ to $\mathcal{S}$. The difference between Game_2 and Game_3 appears now in how $\mathcal{S}$ attempts to decrypt c_α and c_μ. In Game_2 it would have used $\widehat{\mathsf{sk}}_1$, whereas in Game_3 it uses sk_1.

Notation. We denote α (resp. μ) the random variable obtained by decrypting c_α (resp. c_μ) (received in Game_3) with decryption key sk_1; we denote α' (resp. μ') the random variable obtained by decrypting c_α (resp. c_μ) with $\widehat{\mathsf{sk}}_1$; we introduce a hypothetical Game_3', which is exactly as Game_3, only one decrypts c_α (resp. c_μ) with decryption key $\widehat{\mathsf{sk}}_1$, thus obtaining α' (resp. μ'). Moreover in Game_3' the check performed in Phase 2 is 'If $\mu' P + B_{1,j} \neq \widehat{k}_1 W_j$ then abort'.

Observation. $\mathcal{A}$'s view in Game_2 and in Game_3' is identical. By demonstrating that the probability $\mathcal{A}$'s view differs when $\mathcal{S}$ uses α, μ in Game_3 from when it uses α', μ' in Game_3' is negligible, we can conclude that $\mathcal{A}$ cannot distinguish Game_2 and Game_3 with significant probability.

The smoothness of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ ensures that given pk_1, which fixes $\mathsf{sk}_1 \bmod s$, the distribution followed by $\mathsf{sk}_1 \bmod q$ remains δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. In particular this ensures that $\mathcal{A}$'s view of α and α' are δ_s-close. Indeed, $\mathcal{A}$ receives an invalid encryption of k_1, which information theoretically masks k_1. At this point $\mathcal{A}$'s view of k_1 is that of a random variable following a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. $\mathcal{A}$ then computes c_α which is sends to $\mathcal{S}$. Then $\mathcal{A}$ receives either (a one way function of) k_1, or (a one way function of) some random value unrelated to k_1, which follows the uniform distribution modulo q. Finally $\mathcal{A}$ must decide which it received.

For μ and μ', the indistinguishability of $\mathcal{A}$'s view of both random variables is a little more delicate, since $\mathcal{A}$ gets additional information from the check on the curve performed by $\mathcal{S}$, namely in Game_3 if $\mu P + B_{1,j} \neq k_1 W_j$ then $\mathcal{S}$ aborts. We call the output of this check test. And in Game_3', if $\mu' P + B_{1,j} \neq \widehat{k}_1 W_j$ then $\mathcal{S}$ aborts. We call the output of this check test'. Notice that if $\mathsf{test} = \mathsf{test}'$, both games are δ_s-close from $\mathcal{A}$'s view (since the only change is in the ciphertext c_{k_1}). Let us upper bound the probability $\mathfrak{p}$ that $\mathsf{test} \neq \mathsf{test}'$. This will allow us to conclude that

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| \leqslant \mathfrak{p} + \delta_s.$$

Consider the ciphertext $c_\mu = (u_\mu, e_\mu) \in \widehat{G} \times \widehat{G}$ sent by $\mathcal{A}$. There exist unique $z_\mu \in \widehat{G}^q$, $b_\mu \in \mathbf{Z}/q\mathbf{Z}$ such that $u_\mu = z_\mu f^{b_\mu}$. Since $\mathsf{sk}_1 = \widehat{\mathsf{sk}}_1 \bmod \varpi$, $\mu = \perp$ if and only if $\mu' = \perp$. This occurs when $e_\mu z_\mu^{-\mathsf{sk}_1} = e_\mu z_\mu^{-\widehat{\mathsf{sk}}_1} \notin F$. In this case Game_3 is δ_s-close to Game_3' from $\mathcal{A}$'s view ($\mathcal{S}$ aborts in both cases). We hereafter assume decryption does not fail, which allows us to adopt the following notation: $e_\mu = z_\mu^{\mathsf{sk}_1} f^{h_\mu} = z_\mu^{\widehat{\mathsf{sk}}_1} f^{h_\mu}$ with $h_\mu \in \mathbf{Z}/q\mathbf{Z}$. We thus have:

$$\mu = \log_f(e_\mu u_\mu^{-\mathsf{sk}_1}) = h_\mu - b_\mu \mathsf{sk}_1 \bmod q \quad \text{and} \quad \mu' = \log_f(e_\mu u_\mu^{-\widehat{\mathsf{sk}}_1}) = h_\mu - b_\mu \widehat{\mathsf{sk}}_1 \bmod q.$$

And consequently:

$$\mu - \mu' = b_\mu(\widehat{\mathsf{sk}}_1 - \mathsf{sk}_1) = b_\mu {b_1}^{-1}(k_1 - \widehat{k}_1) \bmod q.$$

We consider three cases:

(a) $\mu = \mu'$ mod q. This may happen for two reasons:

i. If $k_1 = \widehat{k}_1$ mod q, then Game_3 and $\mathsf{Game}_3{}'$ are identical.

ii. Else $b_\mu = 0$ mod q, i.e. $u_\mu \in \widehat{G}^q$ and c_μ is a valid ciphertext. Since we ruled out $k_1 = \widehat{k}_1$ mod q in the previous case, if $\mathsf{test} = \mathsf{true}$, necessarily $\mathsf{test}' = \mathsf{false}$, and vice versa. Both cases being symmetric, we consider the case $\mathsf{test} = \mathsf{true}$. From $\mathcal{A}$'s view, prior to outputting c_μ the only fixed information relative to k_1 is that contained $c_{k_1} = (g_q^{r_1} f^{b_1}, (g_q^{r_1} f^{b_1})^{\mathsf{sk}_1} f^{k_1})$. This fixes $\pi_0 := b_1 \cdot \mathsf{sk}_1 + k_1$ mod q. However from $\mathcal{A}$'s view, given pk_1, the random variable sk_1 follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$; thus k_1 also follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Now $\mathcal{A}$ returns $B_{1,j}$ and $c_\mu = (z_\mu, z_\mu^{\mathsf{sk}_1} f^\mu)$, where $z_\mu \in \widehat{G}^q$, which satisfy $\mu P + B_{1,j} = k_1 W_j$ since $\mathsf{test} = \mathsf{true}$. Thus $\mathcal{A}$ has information theoretically fixed the value of k_1 mod q, given a view of k_1 which is δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. Hence this occurs with probability $\leqslant 1/q + \delta_s$.

(b) $\mu \neq \mu'$ mod q but $\mu - \mu' = w_j(k_1 - \widehat{k}_1)$ mod q, i.e. $b_\mu = w_j b_1$ mod q. This results in $\mathcal{S}$ aborting on μ' in Game_2 if and only if $\mathcal{S}$ aborts on μ in Game_3. This occurs if the adversary performs homomorphic operations on c_{k_1}, and the difference between the random variables is that expected by $\mathcal{S}$. Indeed:

$$\mu = k_1 w_j \Leftrightarrow \mu' + w_j(k_1 - \widehat{k}_1) = k_1 w_j \Leftrightarrow \mu' = \widehat{k}_1 w_j.$$

(c) $\mu \neq \mu'$ mod q and $\mu - \mu' \neq w_j(k_1 - \widehat{k}_1)$ mod q. We here consider three sub-cases:

i. Either $\mathsf{test}' = \mathsf{test} = \mathsf{false}$; in this case Game_3 is δ_s-close to $\mathsf{Game}_3{}'$ from $\mathcal{A}$'s view.

ii. Either $\mathsf{test}' = \mathsf{true}$; this means that:

$$\mu' = \widehat{k}_1 w_j - \nu_{1,j} \bmod q.$$

Now since $\mu - \mu' \neq w_j(k_1 - \widehat{k}_1)$ mod q necessarily $\mathsf{test} = \mathsf{false}$. Consequently if this event occurs, $\mathcal{A}$'s view differs. Let us prove that information theoretically, this can not happen with probability greater than $1/q + \delta_s$. For clarity, we first recall the expression of c_{k_1} received by $\mathcal{A}$:

$$c_{k_1} = (g_q^{r_1} f^{b_1}, \mathsf{pk}_1^{r_1} f^{\widehat{\mathsf{sk}}_1 b_1 + \widehat{k}_1})$$

where $b_1 \neq 0$ mod q. We also recall the expression of c_μ, sent by $\mathcal{A}$ to $\mathcal{S}$. Since c_μ decrypts to μ' with decryption key $\widehat{\mathsf{sk}}_1$, we can write:

$$c_\mu = (z_\mu f^{b_\mu}, z_\mu^{\widehat{\mathsf{sk}}_1} f^{\mu' + b_\mu \widehat{\mathsf{sk}}_1}).$$

Let us denote $\pi_0 := \widehat{\mathsf{sk}}_1 b_1 + \widehat{k}_1$ mod q and $\pi_1 := \mu' + b_\mu \widehat{\mathsf{sk}}_1$. For this case to occur, it must hold that $\mu' = \widehat{k}_1 w_j - \nu_{1,j}$ mod q, so

$$\pi_1 = \widehat{k}_1 w_j - \nu_{1,j} + b_\mu \widehat{\mathsf{sk}}_1 \bmod q.$$

Substituting $\widehat{\mathsf{sk}}_1$ for $(\pi_0 - \widehat{k}_1) b_1^{-1}$ yields:

$$\begin{aligned} \pi_1 &= \widehat{k}_1 w_j - \nu_{1,j} + b_\mu b_1^{-1}(\pi_0 - \widehat{k}_1) \bmod q \\ \Leftrightarrow \pi_1 + \nu_{1,j} - b_\mu b_1^{-1} \pi_0 &= \widehat{k}_1 (w_j - b_\mu b_1^{-1}) \bmod q \end{aligned}$$

As we dealt with $b_\mu = w_j b_1 \bmod q$ in case (b), here $w_j - b_\mu b_1^{-1}$ is invertible mod q so we can write:

$$\widehat{k}_1 = (\pi_1 + \nu_{1,j} - b_\mu b_1^{-1} \pi_0)(w_j - b_\mu b_1^{-1})^{-1} \bmod q \tag{5.3}$$

where π_0, b_1 are fixed by c_{k_1}; π_1, b_μ are fixed by c_μ; w_j is fixed by W_j; and $\nu_{1,j}$ is fixed by $B_{1,j}$. So given $\mathscr{A}$'s view and $\mathscr{A}$'s output ($B_{1,j}$ and c_μ), all the terms on the right hand side of Eq. (5.3) are fixed. However, given pk_1, c_{k_1} and W_j (which is all the relevant information $\mathscr{A}$ gets prior to outputting c_μ), the δ_s-smoothness of $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ ensures that $\widehat{k}_1$ follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$. If the current case occurs, Eq. (5.3) must hold, thus from being given a view where $\widehat{k}_1$ follows a distribution δ_s-close to $\mathcal{U}(\mathbf{Z}/q\mathbf{Z})$, $\mathscr{A}$ succeeds in fixing this random variable to be the exact value used by $\mathscr{S}$. This occurs with probability $\leqslant 1/q + \delta_s$.

iii. Else $\mathsf{test} = \mathsf{true}$; this means that $\mu = k_1 w_j - \nu_{1,j} \bmod q$. Since $(\mu - \mu' \neq w_j(k_1 - \widehat{k}_1) \bmod q)$ necessarily test' fails, and $\mathscr{A}$'s view differs. Reasoning as in the previous case, but setting $\pi_0 := \mathsf{sk}_1 b_1 + k_1 \bmod q$ and $\pi_1 := \mu + b_\mu \mathsf{sk}_1$, one demonstrates that this case occurs with probability $\leqslant 1/q + \delta_s$.

Combining the above, we get that $\mathsf{test}' \neq \mathsf{test}$ if and only if we are in case (a) ii. (c) ii. or (c) iii., which occurs with probability $\leqslant 3(1/q + \delta_s)$. Thus:

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| \leqslant 3/q + 4\delta_s.$$

Game_3 *to* Game_4. In Game_4, the first element u_1 of c_{k_1} is once again sampled in G^q. Both games are indistinguishable under the hardness of $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ and:

$$|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| \leqslant \delta_{\mathsf{hsm\text{-}cl}}.$$

Game_4 *to* Game_5. In Game_5 $\mathscr{S}$ uses the public key pk_1 to encrypt k_1. The change here is exactly that between Game_0 and Game_1, both games are perfectly indistinguishable, and:

$$|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_4]| = 0.$$

Real/Ideal executions. Putting together the above probabilities, we get that:

$$|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_0]| \leqslant 2\delta_{\mathsf{hsm\text{-}cl}} + 3/q + 4\delta,$$

which concludes the proof of the claim. □

3. We now tackle the third and last difference between the real and simulated executions of the signature protocol. Justifying that this difference is unnoticeable to the adversary will allow us to conclude the proof of Lemma 5.14. Notice that $\mathscr{S}$ does not know σ_1, and thus cannot compute s_1 as in a real execution. Instead it computes $s_1 = s - \sum_{j \in S, j \neq 1} s_j = s - \sum_{j \in S, j \neq 1}(k_j m + \sigma_j r)$ where implicitly $s = \widehat{k}(m + rx)$. So $s_1 = \widehat{k}_1 m + r(\widehat{k}x - \sum_{j \in S, j \neq 1} \sigma_j)$, and $\mathscr{S}$ is implicitly setting $\widehat{\sigma}_1 := \widehat{k}x - \sum_{j \in S, j \neq 1} \sigma_j$ so $\widehat{k}x = \widehat{\sigma}_1 + \sum_{j \in S, j \neq 1} \sigma_j$. We note that, since the real execution is semi correct, the correct shares of k for the adversary are the k_i that $\mathscr{S}$ knows and $R = \widehat{k}P = (\widehat{k}_1 + \sum_{j \in S, j \neq 1} k_j)$. Therefore the value s_1 computed by $\mathscr{S}$ is consistent with a correct share for P_1 for a valid signature (r, s), which makes Phase 5 indistinguishable from the real execution to the adversary. In particular, observe that if none of the parties aborted during Phase 2, the output shares are correct. So if

$\mathcal{A}$ here uses the values $\{\sigma_j\}_{j\in S, j>1}$ as computed in a real execution of the protocol, it expects the signature generation protocol to output a valid signature. And indeed with $\mathcal{S}$'s choice of $\widehat{\sigma}_1$ and $\widehat{k}_1$, the protocol will terminate, outputting the valid signature (r,s) it received from its signing oracle. Conversely, if $\mathcal{A}$ attempts to cheat in Phase 5 by using a different set of σ_j's than those prescribed by the protocol, the check $\sum_{i\in S} T_i = \sum_{i\in S} U_i$ will fail, and all parties abort as in a real execution of the protocol.

□

Non semi-correct executions

Lemma 5.15. Let $\mathsf{Gen}_{\mathbf{G}}$ be the algorithm which on input a security parameter λ outputs the EC description $(\mathbf{G}, P, q)$, and let C denote the size of the challenge space used in the ZKAoK for $\mathsf{R}_{\mathsf{Enc}}$. Assuming the SR assumption and the LO$_C$ assumption hold for $\mathsf{Gen}_{\mathsf{CL}}$; that the DDH assumption holds for $\mathsf{Gen}_{\mathbf{G}}$; and the commitment scheme is non-malleable and equivocal; then the simulation is computationally indistinguishable from a non-semi-correct real execution.

Proof. We construct three games between the simulator $\mathcal{S}$ (running P_1) and the adversary $\mathcal{A}$ (running all other players). In Game_0, $\mathcal{S}$ runs the real protocol. The only change between Game_0 and Game_1 is that in Game_1, $\mathcal{S}$ chooses U_1 as a random group element. In Game_2 the simulator $\mathcal{S}$ runs the simulation described in Fig. 5.11.

Game_0 *to* Game_1. We prove that if there exists an adversary $\mathcal{A}$ distinguishing Game_0 and Game_1, $\mathcal{A}$ can be used to break the DDH assumption in $\mathbf{G}$. Let $\tilde{A} = a\cdot P$, $\tilde{B} = b \cdot P$, $\tilde{C} = c\cdot P$ be the DDH challenge where $c = ab$ or $c \hookleftarrow \mathbf{Z}/q\mathbf{Z}$. The DDH adversary $\mathcal{S}_0$ runs $\mathcal{A}$, simulating the key generation phase and setting the EC-DSA verification key to be $Q = \tilde{B}$. $\mathcal{S}_0$ also extracts the values $\{x_j\}_{j\in[n], j\neq 1}$ chosen by $\mathcal{A}$ from the ZKPoK of step 11 of the $\mathsf{IKeyGen}$ simulation of Fig. 5.11. At this point $Q = \tilde{B}$ and $\mathcal{S}_0$ knows x_i and the decryption key sk_1 matching pk_1, but not b and therefore not x_1.

Next $\mathcal{S}_0$ runs the simulated signing protocol for a non-semi-correct execution. Recall that $S \subseteq [n]$ denotes the subset of players collaborating in ISign. Denoting $t := |S|$, the (t,n) shares $\{x_i\}_{i\in[n]}$ are converted into (t,t) shares $\{w_i\}_{i\in S}$ as per the protocol. Thus $b = \sum_{i\in S} w_i$ where $\mathcal{S}_0$ knows $\{w_j\}_{j\in S, j\neq 1}$ but not w_1. We denote $w_A := \sum_{j\in S, j\neq 1} w_j$ which satisfies $w_1 = b - w_A$. $\mathcal{S}_0$ runs the signing protocol normally for Phases $1,2,3,4$. It extracts the values $\{\gamma_j\}_{j\in S, j\neq 1}$ from the PoK in Phase 4, and knows γ_1 since it ran P_1 normally. Therefore $\mathcal{S}_0$ knows k such that $R = k^{-1}\cdot P$ since $k = (\sum_i \gamma_i)^{-1}\delta \bmod q$. It also knows k_1 (chosen normally according to the protocol) and $\{k_j\}_{j\in S, j\neq 1}$ which – assuming the SR and LO assumptions hold – it can extract from the proofs in Phase 1.

Before moving to the simulation of Phase 5, let's look at Phase 2 of the protocol for the computation of the shares σ_i. We note that since $\mathcal{S}_0$ knows sk_1 it also knows all the shares $\mu_{1,j}$ since it can decrypt the ciphertext $c_{k_1 w_j}$ received from P_j. However $\mathcal{S}_0$ does not know w_1 therefore it sends the encryption of a random $\mu_{j,1}$ to P_j and sets (implicitly) $\nu_{j,1} = k_j w_1 - \mu_{j,1}$. At the end the share σ_1 held by P_1 is

$$\sigma_1 = k_1 w_1 + \sum_{j\in S, j\neq 1} (\mu_{1,j} + \nu_{j,1}) = \tilde{k} w_1 + \sum_{j\in S, j\neq 1} (\mu_{1,j} - \mu_{j,1}) \text{ where } \tilde{k} = \sum_{i\in S} k_i.$$

Recall that since this is a non-semi-correct execution $\tilde{k} \neq k$ where $R = k^{-1}\cdot P$. Since $w_1 = b - w_A$ we have $\sigma_1 = \tilde{k}b + \mu_1$ where $\mu_1 = \sum_{j\in S, j\neq 1}(\mu_{1,j} - \mu_{j,1}) - \tilde{k}w_A$ with $\mu_1, \tilde{k}$ known to $\mathcal{S}_0$. This allows $\mathcal{S}_0$ to compute the correct value $\sigma_1 \cdot P = \tilde{k}\tilde{B} + \mu_1 \cdot P$ and therefore the correct value of $s_1 \cdot R$ as:

$$s_1 \cdot R = (k_1 m + r\sigma_1)\cdot R = k^{-1}(k_1 m + r\sigma_1)\cdot P$$

$$= k^{-1}(k_1 m + r\mu_1) \cdot P + k^{-1}(\tilde{k}r) \cdot \tilde{B} = \hat{\mu}_1 \cdot P + \hat{\beta}_1 \cdot \tilde{B}$$

where $\hat{\mu}_1 = k^{-1}(k_1 m + r\mu_1)$ and $\hat{\beta}_1 = k^{-1}\tilde{k}r$ are known to $\mathcal{F}_0$.

In the simulation of Phase 5, $\mathcal{F}_0$ selects a random ℓ_1 and sets $V_1 := s_1 \cdot R + \ell_1 \cdot P$, $A_1 = \rho_1 \cdot P = \tilde{A} = a \cdot P$. It simulates the ZKP (since it does not know ρ_1 or s_1) and extracts s_i, ℓ_i, ρ_i from $\mathcal{A}$'s proofs s.t. $V_i = s_i \cdot R + \ell_i \cdot P = k^{-1}s_i \cdot P + \ell_i \cdot P$ and $A_i = \rho_i \cdot P$. Let $s_A = \sum_{j\in S, j\neq 1} k^{-1}s_j$. Note that, substituting the above relations (and setting $\ell = \sum_{i\in S}\ell_i$), we have: $V = -m \cdot P - r \cdot Q + \sum_{i\in S} V_i = \ell \cdot P + s_1 \cdot R + (s_A - m) \cdot P - r \cdot Q$. Moreover $Q = \tilde{B}$ so $-r \cdot Q = -r \cdot \tilde{B}$, and:

$$V = \ell \cdot P + \hat{\mu}_1 \cdot P + \hat{\beta}_1 \cdot \tilde{B} + (s_A - m) \cdot P - r \cdot \tilde{B} = (\ell + \theta) \cdot P + \kappa \cdot \tilde{B}$$

where $\mathcal{F}_0$ knows $\theta = \hat{\mu}_1 + s_A - m$ and $\kappa = \hat{\beta}_1 - r$. Note that for executions that are not semi-correct $\kappa \neq 0$.

Next $\mathcal{F}_0$ computes $T_1 \leftarrow \ell_1 \cdot A$ (correctly), but computes U_1 as $U_1 \leftarrow (\ell + \theta) \cdot \tilde{A} + \kappa \cdot \tilde{C}$, using this U_1 it continues as per the real protocol and aborts on the check $\sum_{i\in S} T_i = \sum_{i\in S} U_i$.

Observe that when $\tilde{C} = ab \cdot P$, by our choice of $a = \rho_1$ and $b = x$, we have that $U_1 = (\ell + \theta)\rho_1 \cdot P + \kappa \cdot \rho_1 \tilde{B} = \rho_1 \cdot V$ as in Game_0. However when $\tilde{C}$ is a random group element, U_1 is uniformly distributed as in Game_1. Therefore under the LO, SR, and DDH assumptions Game_0 and Game_1 are indistinguishable.

Game_1 *to* Game_2. In Game_2, $\mathcal{F}$ broadcasts a random $\tilde{V}_1 = \tilde{s}_1 \cdot R + \ell_1 \cdot P$. This is indistinguishable from the correct $V_1 = s_1 \cdot R + \ell_1 \cdot P$ thanks to the mask $\ell_1 \cdot P$ which (under the DDH assumption) is computationally indistinguishable from a random value, since the adversary only knows A_1. To be precise, let $\tilde{A} = (a - \delta) \cdot P, \tilde{B} = b \cdot P$ and $\tilde{C} = ab \cdot P$ be the DDH challenge where δ is either 0 or random in $\mathbf{Z}/q\mathbf{Z}$. The simulator proceeds as in Game_0 (i.e. the regular protocol) until Phase 5. In Phase 5, $\mathcal{F}_0$ broadcasts $V_1 = \tilde{s}_1 \cdot R + \tilde{A}$ and $A_1 = \tilde{B}$. It simulates the ZKPoK (it does not know ℓ_1 or ρ_1), and extracts s_i, ℓ_i, ρ_i from the adversary s.t. $V_i = s_i \cdot R + \ell_i \cdot P = k^{-1}s_i \cdot P + \ell_i \cdot P$ and $A_i = \rho_i \cdot P$.

Next $\mathcal{F}_0$ samples a random U_1 and sets $T_1 \leftarrow \tilde{C} + \sum_{j\in S, j\neq 1} \rho_j \cdot \tilde{A}$ before aborting. Note that when $\tilde{A} = aP$, we implicitly set $a = \ell_1$ and $b = \rho_1$ and have $V_1 = s_1 \cdot R + \ell_1 \cdot P$ and $T_1 = \ell_1 \cdot A$ as in Game_1. However when $\tilde{A} = aP - \delta P$ with a random δ, then this is equivalent to having $V_1 = \tilde{s}_1 R + \ell_1 P$ and $T_1 = \ell_1 A$ with a randomly distributed $\tilde{s}_1$ as in Game_2. Therefore under the DDH assumption Game_1 and Game_2 are indistinguishable. □

Concluding the proof

As mentioned when introducing the simulation of ISign (Fig. 5.11), the forger $\mathcal{F}$ simulating $\mathcal{A}$'s environment can detect if it is in a semi-correct-execution or not, i.e. whether $\mathcal{A}$ decides to be malicious and terminate the protocol with an invalid signature. Consequently $\mathcal{F}$ always knows how to simulate $\mathcal{A}$'s view and all simulated executions of the protocol are indistinguishable of real ones. Moreover if $\mathcal{A}$, having corrupted up to t parties in the threshold EC-DSA protocol, outputs a forgery, since $\mathcal{F}$ set up with $\mathcal{A}$ the same public key Q as it received from its' EC-DSA challenger, $\mathcal{F}$ can use this signature as its own forgery, thus breaking the existential unforgeability of standard EC-DSA.

Denoting $\mathsf{Adv}^{\mathsf{tu\text{-}cma}}_{\mathsf{t-ecdsa},\mathcal{A}}$, $\mathcal{A}$'s advantage in breaking the existential unforgeability of our threshold protocol, and $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{ecdsa},\mathcal{F}}$ the forger $\mathcal{F}$'s advantage in breaking the existential unforgeability of centralised EC-DSA, from Lemmas 5.14 and 5.15 it holds that if the DDH assumption holds for generator $\mathsf{Gen}_{\mathbf{G}}$; the SR assumption and the LO_C assumption hold for $\mathsf{Gen}_{\mathsf{CL}}$; $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is a hard SMP; $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is smooth; and the commitment scheme is non-malleable and equivocal then: $|\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{ecdsa},\mathcal{F}} - \mathsf{Adv}^{\mathsf{tu\text{-}cma}}_{\mathsf{t-ecdsa},\mathcal{A}}| = \mathsf{negl}$. Unforgeability of centralised EC-DSA states that $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{ecdsa},\mathcal{F}}$ is negligible, which implies that $\mathsf{Adv}^{\mathsf{tu\text{-}cma}}_{\Pi,\mathcal{A}}$ is too.We can thus state the following theorem, which captures the security of the protocol.

Theorem 5.16. Assuming centralised EC-DSA is an euf-cma signature scheme; the DDH assumption holds for generator $\mathsf{Gen}_{\mathbf{G}}$; the SR and the LO_C assumptions hold for $\mathsf{Gen}_{\mathsf{CL}}$; $\mathcal{SM}_{\mathsf{hsm\text{-}cl}}$ is a hard SMP; $\mathsf{H}_{\mathsf{hsm\text{-}cl}}$ is smooth; and the commitment scheme is non-malleable and equivocal, then the (t, n)-threshold EC-DSA protocol of Figs. 5.9 and 5.10 is a tu-cma threshold signature scheme.

5.3.5 Efficiency Comparisons

We analyse the theoretical complexity of our protocol, using the timings computed for a multiplication of Section 2.6.3, and estimating the resulting time taken for exponentiations. Regarding bandwidth consumption we count the communication of group elements. We compare the communication cost of our protocol to those of [GG18, LN18] for the standard NIST curves P-256, P-384 and P-521, corresponding to levels of security 128, 192 and 256. For the encryption scheme, we start with a 112 bit security, as in the [GG18, LN18] implementations, but also study the case where its level of security matches that of the EC.

We compare our work to best performing pre-existing protocols using similar construction techniques (from homomorphic encryption) which achieve the same functionality, i.e. (t, n)-threshold EC-DSA for any t s.t. $n \geqslant t+1$. We do not provide a comparison to [DKLs18,DKLs19] as they use oblivious transfer which leads to protocols with a much higher communication cost. Similarly, and as noted in [DKO$^+$19] a direct comparison to [DKO$^+$19, SA19] is difficult as they rely on preprocessing to achieve efficient signing, which is a level of optimisation we have not considered. We don't compare to [GGN16,BGG17] as [GG18] is already faster and cheaper in terms of communication complexity.

The computed communication cost is for our protocol as described in Section 5.3.3, and as such is provably secure. Conversely the implementation which [GG18] provided omits a number of range proofs present in their described protocol. Though this substantially improves the efficiency of their scheme, they themselves note that removing these proofs creates an attack which leaks information on the secret signing key shared among the servers. They conjecture this information is limited enough for the protocol to remain secure, however since no formal analysis is performed, the resulting scheme is not proven secure. For a fair comparison we estimate the communication cost and timings of both their secure protocol and the stripped down version. In terms of bandwidth we outperform even their stripped down protocol.

In both protocols, when possible ZKPs are performed non interactively, replacing the challenge by a hash value, whose size depends on the security parameter λ. We note that our interactive setup for the $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme uses a ZKPoK where challenges are of size 10 bits (using the lcm trick), it must thus be repeated $\lambda/10$ times. We note however that the PoK of integer factorisation used in the key generation of [GG18] has similar issues.

For non-malleable equivocal commitments, we use a cryptographic hash function H and define the commitment to x as $h = H(x, r)$ for a uniformly chosen r of length λ. We assume H behaves as a random oracle.

The communication cost comparison is done by counting the number of bits that are both sent and received by a given party throughout the protocol[7]. In terms of timings, we do not count exponentiations and multiplications over the EC group as these are very cheap compared

[7] Broadcasting one element is counted as sending one element.

to computations in the class group of in $\mathbf{Z}/N^2\mathbf{Z}$, furthermore both protocols essentially perform identical operations on the curve.

The [LN18] protocol with Paillier encryption. We use the figures Lindell et al. provide in [LN18, Tab. 1] to compare our protocol to theirs. We note that – to their advantage – their key generation should include additional costs which are not counted in our figures (e.g. local Paillier key generation, verification of the ZKP of correctness of the Paillier key); hence we do not take it into account in our color-coded comparison. The resulting costs are given in Fig. 5.12a.

The [GG18] protocol with Paillier encryption. The main cost in their $\mathsf{IKeyGen}$ protocol is the ZKPoK of integer factorisation, which is instantiated using [PS00, Thm. 8]. Precisely each prover commits to K values mod N, the challenge lives mod B, the final opening is an element of size A, where, as prescribed in [GPS06], we take $\log(A) = \log(N)$, $\log(B) = \lambda$ and $K = \frac{\lambda+\log(|N|)}{log(C)}$ where $C = 2^{60}$ is chosen s.t. Floyd's cycle-finding algorithm is efficient in a space of size smaller than C. For their signature protocol, the cost of the ZKPs used in the MtA protocol are counted using [GG18, Appendix A].

The results are summarised in Fig. 5.12b. Since the range proofs (omitted in the stripped down version) only occur in the signing protocol, the timings and communication cost of their interactive key generation is identical in both settings, we thus only provide these figures once. Furthermore, as it is not provably secure, we do not take their stripped down protocol into account in our color-coded comparison.

The communication cost of each protocol is given in Bytes. The columns correspond to the EC used for EC-DSA, the security parameter λ in bits for the encryption scheme, the timings of the key generation and of the signing phase and the total communication in bytes for each interactive protocol.

Our protocol with the $\Pi_{\mathsf{hsm\text{-}cl}}$ encryption scheme. For key generation we take into account the interactive key generation for $\Pi_{\mathsf{hsm\text{-}cl}}$, which is done in parallel with $\mathsf{IKeyGen}$, consequently the number of rounds of $\mathsf{IKeyGen}$ increases by only one broadcast per player. In $\mathsf{IKeyGen}$, each party performs 2 class group exponentiations of $\log(\tilde{s}) + 40$ bits (where $\tilde{s} \approx \sqrt{q \cdot \tilde{q}}$), to compute generators g_i and public keys pk_i, and $n\lambda/10$ exponentiations of $\log(\tilde{s}) + 90$ bits for the proofs and checks in the ISetup sub-protocol.

Signing uses $2 + 10t$ exponentiations of $\log(\tilde{s}) + 40$ bits (for computing ciphertexts and homomorphic operations), $2(t + 1)$ exponentiations of $\log(\tilde{s}) + 80 + \lambda$ bits (for the ZKAoK) and $2t$ exponentiations of size q (for homomorphic scalar multiplication of ciphertexts).

The results are summarised in Fig. 5.12c. The columns correspond to the EC used for EC-DSA, the security parameter λ in bits for the encryption scheme, the timings of the key generation and of the signing phase and the total communication in bytes for $\mathsf{IKeyGen}$ and ISign.

Rounds. Our protocols require the same number of rounds as those of [LN18]. Our $\mathsf{IKeyGen}$ requires 5 rounds (only 4 assuming a standardised setup), compared to 4 in [GG18]. Our signing protocol requires 8 rounds as opposed to 9 in [GG18].

Comparison. Fig. 5.12 shows that the protocols of [LN18, GG18] are faster for both key generation and signing for standard security levels for the encryption scheme (112 and 128 bits of security) while our solution remains of the same order of magnitude. However for high security levels, our signing protocol is fastest from a 192-bits security level.

In terms of communication, our solution outperforms the other two protocols for all levels of security, factors vary according to the number of users n and the desired threshold t.

Curve	λ (bits)	IKeyGen (ms)	ISign (ms)	IKeyGen (Bytes)	ISign (Bytes)
P-256	112	$> 95n + 95$	$181t$	$> 6\,336(n-1)$	$16\,064t$
P-256	128	$> 331n + 331$	$632t$	$> 9\,152(n-1)$	$22\,208t$
P-384	192	$> 3\,548n + 3\,548$	$6\,773t$	$> 22\,176(n-1)$	$51\,744t$
P-521	256	$> 20\,579n + 20\,579$	$39\,288t$	$> 43\,672(n-1)$	$99\,845t$

(a) [LN18]'s secure t out of n protocol.

		Provably secure (with range proofs)				Stripped down	
Curve	λ bits	IKeyGen (ms)	ISign (ms)	IKeyGen (Bytes)	ISign (Bytes)	ISign (ms)	ISign (Bytes)
P-256	112	$77n + 9$	$170t$	$32(n+t) + 9\,990n - 64$	$23\,308t + 588$	$34t$	$4\,932t + 588$
P-256	128	$399n + 30$	$582t$	$32(n+t) + 21\,392n - 64$	$33\,568t + 608$	$120t$	$7\,008t + 608$
P-384	192	$10, 564n + 323$	$6\,129t$	$48(n+t) + 128088n - 96$	$81\,072t + 912$	$1290t$	$16\,656t + 912$
P-521	256	$121\,605n + 1871$	$35\,246t$	$65(n+t) + 503\,591n - 130$	$159\,391t + 1\,232$	$7483t$	$32\,470t + 1\,231$

(b) [GG18]'s t out of n protocol.

We note that using a public coin setup for CL, which provides a description of $\widehat{G}$, of the subgroups F and G_q and of a *random* generator g_q of G_q, one could completely omit the interactive setup phase for the CL encryption scheme and have all parties use the output of this public coin process. Such a setup has been described in [LM19, Section 8.1]. In our protocol this would reduce the bandwidth consumption of IKeyGen by a factors varying from 6 to 16 (for increasing levels of security). Moreover in terms of timings, the only exponentiation in the class group would be each party computing its own ciphertext, and so the only operations linear in the number of users n would be on the curve (or integers modulo q), which are extremely efficient.

Curve	λ (bits)	$\mathsf{IKeyGen}$ (ms)	ISign (ms)	$\mathsf{IKeyGen}$ (Bytes)	ISign (Bytes)
P-256	112	$466n + 79$	$459t + 175$	$\mathbf{32(n+t) + 2951n - 64}$	$\mathbf{3\,670t + 1\,747}$
P-256	128	$939n + 137$	$922t + 299$	$\mathbf{32(n+t) + 4\,297n - 64}$	$\mathbf{4\,455t + 2\,052}$
P-384	192	$5\,276n + 541$	$\mathbf{3\,538t + 1\,149}$	$\mathbf{48(n+t) + 10\,851n - 96}$	$\mathbf{8\,022t + 3\,560}$
P-521	256	$21\,110n + 1\,597$	$\mathbf{10\,291t + 3\,351}$	$\mathbf{65(n+t) + 22\,942n - 130}$	$\mathbf{12\,576t + 5\,433}$

(c) Our secure t out of n protocol – With an interactive setup for $\Pi_{\mathsf{hsm\text{-}cl}}$.

Figure 5.12: Comparative sizes (in bits), timings (in ms) and communication cost (in Bytes)

Chapter 6

Conclusion and Open Problems

In this thesis we first focused on building new tools from the framework of a group G with an easy discrete logarithm subgroup F introduced by Laguillaumie and Castagnos in [CL15]. Their work grounded itself on the hardness of the DDH problem in G. We started off formalising new hardness assumptions from which we devised families of projective hash functions with homomorphic properties. From each of these assumptions we also built public key encryption schemes, which are all linearly homomorphic and benefit of having a prime order message space, where this prime can (with some restrictions) be chosen independently of the security parameter. These properties make the encryption schemes (or somewhat equivalently, the corresponding PHFs) particularly interesting building blocks for devising advanced cryptographic primitives.

In view of using the aforementioned encryption schemes to build threshold signatures, where one must ensure parties send honestly computed ciphertexts, we devise zero knowledge proofs and arguments for the CL framework. We provide proofs for various statements, including a proof of knowledge of the plaintext and randomness used to compute a ciphertext. This latter proof is sufficient to convince a verifier that a ciphertext was honestly generated. Our proofs deal with difficulties arising from the fact the group G we work in is of unknown order, and that elements of this group are not efficiently recognisable (one can only recognise elements of some larger group $\widehat{G}$ containing G). We provide various trade-offs between security and efficiency (proofs which are statistically convincing are costly in terms of communication, whereas computationally convincing proofs require minimal computation and interactions).

Having designed these tools, we then set about employing them to fashion advanced cryptographic primitives, starting with inner product functional encryption. By defining two new properties for projective hash functions: vector smoothness and vector universality, we are able to devise both ind-fe-cpa and ind-fe-cca-secure inner product functional encryption schemes where, conversely to previous such constructions, our security reduction is tight. In particular, when instantiating our ind-fe-cpa construction from DDH or DCR, we retrieve the constructions of [ALS16, ABDP16], with the same security bound; and when instantiated from our assumptions in the CL framework, we obtain new constructions. Instantiations from DCR and from the CL framework yield the most efficient such schemes to date.

Regarding our ind-fe-cca-secure construction, when instantiated from concrete assumptions such as HSM-CL or DCR, this results in schemes which are the first ind-fe-cca-secure inner product functional encryption schemes capable of decrypting the inner product *whatever its size*, and where running times of our algorithms and ciphertext sizes are reasonable for implementation in large scale information systems.

In a different style, but still building upon projective hash functions in the CL framework, we then set about improving the state of the art for threshold EC-DSA. We devised the first two party EC-DSA protocol which is both efficient in terms of speed and computational bandwidth, has a tight security proof and does not rely on any interactive assumptions. We also built a competitive full threshold protocol which outperforms comparable protocols (attaining equivalent functionality) in terms of bandwidth consumption.

These constructions demonstrate that, if one adopts the approach of building threshold EC-DSA from linearly homomorphic encryption, the Paillier cryptosystem is by no means adapted to the task, as it introduces the need for range proofs, and in some instances of artificial and interactive assumptions. Conversely, with the support of our zero knowledge proofs an arguments for the CL framework, the linearly homomorphic encryption schemes arising from the CL framework are well suited. Indeed, they allow to choose the order of the encryption scheme's message space to be the (standardised) prime order of the elliptic curve group used to compute EC-DSA signatures, thereby resulting in seamless interactions between the encryption scheme and the signature computation.

All our generic constructions from projective hash functions can be instantiated from class group cryptography; this was in fact the instantiation provided by Castagnos and Laguillaumie for the CL framework. Since best known algorithms for computing the class number – which is the problem underlying the security of the CL framework – have higher asymptotic complexity than those for factoring integers, or computing the discrete logarithm in finite fields, group elements in our protocols can be chosen smaller than for those relying on e.g. the DCR assumption, while maintaining an equivalent level of security. This has significant impact on the communication complexity of interactive protocols.

Open Problems

We have initiated the task of devising tools from cryptography based on ideal class groups of imaginary quadratic orders, via the interface provided by the CL framework. As illustrated in this thesis, these tools allow for the modular design of advanced cryptographic primitives. The presented tool set is by no means complete, and those tools provided can certainly be further honed. What is more our constructions can be further refined, be it either by enhancing functionality, reinforcing security or improving efficiency; and there are undoubtedly innumerable other advanced cryptosystems which can arise from the framework. We here highlight a few open problems, though there are of course many other technical and interesting questions left unanswered.

More flexibility for the CL framework. It would be interesting to study to what extend one can choose the message space in the CL framework, e.g. could we have a message space of order 2^k for a given integer k? This could offer an alternative to the Joye-Libert cryptosystem [JL13], and would have interesting applications in devising general multi-party computation protocols supporting operations modulo 2^k (as suggested in the SPD$\mathbf{Z}_{2^k}$ protocol of Cramer et al. [CDE+18]), it could in particular be used to instantiate the Mon$\mathbf{Z}_{2^k}$a protocol. Working modulo 2^k matches modern CPU computations and allows protocol designers to directly apply optimisations and tricks that are possible there.

Another possible extension for the CL would be devising threshold encryption schemes. In particular can we have a distributed (and decentralised) key generation? Since our encryption schemes in the CL framework have an Elgamal like structure, and that for Elgamal encryption, efficient threshold variants are known, one can hope the answer is affirmative. The main challenge here is dealing with the unknown order of the CL group.

Improving our ZK proofs. In Chapter 3 we present zero-knowledge proofs and arguments of knowledge for groups of unknown order. The zero-knowledge proofs – whose soundness holds statistically – currently require that exponents be sampled from folded uniform distributions. What would the impact be (on the security of these proofs, mainly in terms of zero-knowledge) of sampling exponents from folded Gaussians? Sampling exponents from folded Gaussians instead of folded uniforms allows for shorter keys, and hence better overall computational complexity of the resulting protocols.

Stronger security and enriched functionalities for IPFE. In Section 4.6.2, building upon the techniques of Agrawal et al. [ALMT20a] we explained how – if the group order is known – one can attain simulation based security against *passive* adversaries from our generic construction.

However we cannot apply these techniques to the CL framework, as our group order is unknown. The natural questions which spring to mind are hence: could one develop an alternative approach to attaining simulation security which would work in groups of unknown order, and could thus be used in the CL framework? And what properties are required of projective hash functions to build inner product functional encryption schemes secure against *active* adversaries in the simulation based model?

Independently to this strengthening of security, one could study the extension of our constructions to deal with the multi-input or multi-client setting, or yet again to allow for a decentralised key distribution.

Stronger security and enriched functionalities for threshold EC-DSA. Regarding threshold EC-DSA, in the past few months a profusion of articles have appeared, bringing great improvements to the state of the art [CMP20, GKSS20, DJN$^+$20, GG20]. These all use preprocessing techniques to obtain a non interactive signing protocol, so that, for parties to issue a signature, they need only perform local operations to get an additive share of the overall signature. In terms of security, they also provide new features. We hereafter give a brief overview of the added value each of these articles brings to threshold EC-DSA. The work of Canetti et al. [CMP20], which also builds upon the [GG18] protocol, significantly reduces the number of rounds required for parties to jointly compute a signature; and proves security within the universal composability security framework. In concurrent work [GG20] also extend the [GG18] protocol to handle identifiable aborts (allowing to detect which player is responsible if the protocol fails to conclude successfully). Conversely Gągol et al. [GKSS20] build upon the full threshold protocol of [LN18] to further handle identifiable aborts and provide robustness for the signing protocol (if a minimum number of parties honestly participate in the signing phase, they are guaranteed to output a valid signature). Finally Damgård et al. – using very different techniques (they do not rely on homomorphic encryption) – provide a protocol whose universally composable security, albeit in the honest majority setting, relies solely on the hardness of EC-DSA. They also explain how, via an extra layer added to their basic protocol, one can enforce fairness for the signing protocol (all parties get output or nobody does).

We believe one should (quite easily) be able to adapt some of the ideas in the aforementioned works, in particular, attaining a non interactive, robust and fair signing protocol with identifiable aborts should be fairly straightforward. Regarding the security model, using the techniques of [CMP20] (i.e. considering an ideal threshold signature functionality instead of an ideal EC-DSA functionality), we believe our full threshold protocol may be proven secure in the UC security framework, however this analysis will certainly be more involved.

Other advanced cryptosystems from the CL framework. A variety of advanced cryptographic primitives can be built using the malleability of linearly homomorphic encryption, one could look into the design of lossy trapdoor functions [PW08] from the CL framework, or use it for general purpose multi party computation, building upon the ideas underlying the

BeDOZa [BDOZ11, Orl11] or SPDZ [DPSZ12] protocols.

Instantiating our constructions from more varied assumptions. Finally, it would be interesting to consider instantiating our projective hash functions from more varied cryptographic assumptions, e.g. in the lattice setting. It is known that lattice based projective hash functions have been particularly difficult to realise; this somewhat hints towards the fact that such an instantiation would not be straightforward. However it would be worth investigating how well suited the lattice based projective hash functions of Benhamouda et al. [BBDQ18] are to an instantiation of our framework. One could then compare resulting protocols to e.g. the LWE based inner product functional encryption schemes of [ALS16].

Bibliography

[ABDP15] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, *Lecture Notes in Computer Science* 9020, pages 733–751. Springer, Heidelberg, March / April 2015. 15, 24, 105, 106, 107

[ABDP16] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011, 2016. `http://eprint.iacr.org/2016/011`. 20, 29, 106, 125, 128, 207

[ABG19] M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *Advances in Cryptology – ASIACRYPT 2019, Part III*, *Lecture Notes in Computer Science* 11923, pages 552–582. Springer, Heidelberg, December 2019. 107

[ABKW19] M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 11443, pages 128–157. Springer, Heidelberg, April 2019. 107

[ABP$^+$17] S. Agrawal, S. Bhattacherjee, D. H. Phan, D. Stehlé, and S. Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 2277–2293. ACM Press, October / November 2017. 15, 24, 106, 130

[ACF$^+$18] M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Advances in Cryptology – CRYPTO 2018, Part I*, *Lecture Notes in Computer Science* 10991, pages 597–627. Springer, Heidelberg, August 2018. 106

[ACGU20] M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. Cryptology ePrint Archive, Report 2020/577, 2020. `https://eprint.iacr.org/2020/577`. 107

[AGRW17] M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *Advances in Cryptology – EUROCRYPT 2017, Part I*, *Lecture Notes in Computer Science* 10210, pages 601–626. Springer, Heidelberg, April / May 2017. 106, 152

[AGVW13] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *Advances in Cryptology – CRYPTO 2013,* *Part II*, *Lecture Notes in Computer Science* 8043, pages 500–518. Springer, Heidelberg, August 2013. 149

[AL10] N. Attrapadung and B. Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, *Lecture Notes in Computer Science* 6056, pages 384–402. Springer, Heidelberg, May 2010. 149

[ALMT20a] S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *Public-Key Cryptography – PKC 2020*, pages 34–64, Cham, 2020. Springer International Publishing. 106, 109, 150, 151, 209

[ALMT20b] S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, *Lecture Notes in Computer Science* 12110, pages 34–64. Springer, Heidelberg, May 2020. 107, 152

[ALS16] S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Advances in Cryptology – CRYPTO 2016, Part III*, *Lecture Notes in Computer Science* 9816, pages 333–362. Springer, Heidelberg, August 2016. 15, 20, 21, 24, 29, 51, 63, 64, 106, 107, 108, 109, 114, 115, 125, 126, 128, 129, 130, 131, 143, 144, 145, 146, 152, 207, 210, 231

[ANT$^+$20] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. Ladderleak: Breaking ecdsa with less than one bit of nonce leakage. Cryptology ePrint Archive, Report 2020/615, 2020. `https://eprint.iacr.org/2020/615`. 161

[BBBF18] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO 2018, Part I*, *Lecture Notes in Computer Science* 10991, pages 757–788. Springer, Heidelberg, August 2018. 18, 27

[BBDQ18] F. Benhamouda, O. Blazy, L. Ducas, and W. Quach. Hash proof systems over lattices revisited. In *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 10770, pages 644–674. Springer, Heidelberg, March 2018. 210

[BBF18] D. Boneh, B. Bünz, and B. Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. `https://eprint.iacr.org/2018/712`. 71, 72

[BBF19] D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *Advances in Cryptology – CRYPTO 2019, Part I*, *Lecture Notes in Computer Science* 11692, pages 561–586. Springer, Heidelberg, August 2019. 71

[BBHM02] I. Biehl, J. Buchmann, S. Hamdy, and A. Meyer. A signature scheme based on the intractability of computing roots. *Designs, Codes and Cryptography*, 25(3):223–236, Mar 2002. 71

[BBL17] F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 10175, pages 36–66. Springer, Heidelberg, March 2017. 11, 21, 29, 30, 77, 106, 107, 108, 110, 112, 125, 135, 137, 142, 143, 147, 148, 231, 234, 235, 236, 237

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156 – 189, 1988. 57

[BCP03] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *Advances in Cryptology – ASIACRYPT 2003*, *Lecture Notes in Computer Science* 2894, pages 37–54. Springer, Heidelberg, November / December 2003. 89, 91

[BCPV13] O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, *Lecture Notes in Computer Science* 7954, pages 534–551. Springer, Heidelberg, June 2013. 46

[BDGM20] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Candidate iO from homomorphic encryption schemes. In *Advances in Cryptology – EUROCRYPT 2020, Part I*, *Lecture Notes in Computer Science* 12105, pages 79–109. Springer, Heidelberg, May 2020. 42

[BDOP04] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT 2004*, *Lecture Notes in Computer Science* 3027, pages 506–522. Springer, Heidelberg, May 2004. 149

[BDOZ11] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, *Lecture Notes in Computer Science* 6632, pages 169–188. Springer, Heidelberg, May 2011. 42, 210

[Bea96] D. Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 629–638. ACM Press, May 1996. 45

[Bel04] K. Belabas. On quadratic fields with large 3-rank. *Mathematics of Computation*, 73(248):2061–2074, 2004. 72

[Ben87] J. D. C. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, USA, 1987. AAI8809191. 42

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, *Lecture Notes in Computer Science* 2139, pages 213–229. Springer, Heidelberg, August 2001. 149

[BFS20] B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In *Advances in Cryptology – EUROCRYPT 2020, Part I*, *Lecture Notes in Computer Science* 12105, pages 677–706. Springer, Heidelberg, May 2020. 18, 27, 71

[BG93] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *Advances in Cryptology – CRYPTO'92*, *Lecture Notes in Computer Science* 740, pages 390–420. Springer, Heidelberg, August 1993. 58

[BGG17] D. Boneh, R. Gennaro, and S. Goldfeder. Using level-1 homomorphic encryption to improve threshold dsa signatures for bitcoin wallet security. In *LATINCRYPT*, 2017. 156, 201

[BH01] J. Buchmann and S. Hamdy. A survey on IQ cryptography. In *Public Key Cryptography and Computational Number Theory*, pages 1–15. De Gruyter Proceedings in Mathematics, 2001. 71

[BHH$^+$14] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic curve cryptography in practice. In *FC 2014: 18th International Conference on Financial Cryptography and Data Security*, *Lecture Notes in Computer Science* 8437, pages 157–175. Springer, Heidelberg, March 2014. 161

[BHJ$^+$13] F. Böhl, D. Hofheinz, T. Jager, J. Koch, J. H. Seo, and C. Striecks. Practical signatures from standard assumptions. In *Advances in Cryptology – EUROCRYPT 2013*, *Lecture Notes in Computer Science* 7881, pages 461–485. Springer, Heidelberg, May 2013. 45

[BJK15] A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *Advances in Cryptology – ASIACRYPT 2015, Part I*, *Lecture Notes in Computer Science* 9452, pages 470–491. Springer, Heidelberg, November / December 2015. 106

[BJS10] J.-F. Biasse, M. J. Jacobson, and A. K. Silvester. Security estimates for quadratic field based cryptosystems. In *ACISP 10: 15th Australasian Conference on Information Security and Privacy*, *Lecture Notes in Computer Science* 6168, pages 233–247. Springer, Heidelberg, July 2010. 18, 28, 49, 50

[BKS18] Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. *Journal of Cryptology*, 31(2):434–520, April 2018. 105

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979. 15, 25

[Boy86] C. Boyd. Digital multisignature. *Cryptography and Coding*, pages 241–246, 1986. 16, 25, 155

[BP97] N. Bari and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology – EUROCRYPT'97*, *Lecture Notes in Computer Science* 1233, pages 480–494. Springer, Heidelberg, May 1997. 70

[BPVY00] E. F. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In *PKC 2000: 3rd International Workshop on Theory and Practice in Public Key Cryptography*, *Lecture Notes in Computer Science* 1751, pages 276–292. Springer, Heidelberg, January 2000. 161

[Bro00] D. Brown. The exact security of ecdsa. 12 2000. 161

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC 2011: 8th Theory of Cryptography Conference*, *Lecture Notes in Computer Science* 6597, pages 253–273. Springer, Heidelberg, March 2011. 14, 24, 105, 109, 110, 149, 150

[Bue76] D. A. Buell. Class groups of quadratic fields. *Mathematics of Computation*, 30(135):610–623, 1976. 72

[BW88] J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, June 1988. 18, 27

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001. 38

[Cas19] G. Castagnos. *Cryptography based on quadratic fields: cryptanalyses, primitives and protocols*. Habilitation à diriger des recherches, Université de Bordeaux, November 2019. 49

[CC07] G. Castagnos and B. Chevallier-Mames. Towards a DL-based additively homomorphic encryption scheme. In *ISC 2007: 10th International Conference on Information Security*, *Lecture Notes in Computer Science* 4779, pages 362–375. Springer, Heidelberg, October 2007. 89

[CCL$^+$19] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *Advances in Cryptology – CRYPTO 2019, Part III*, *Lecture Notes in Computer Science* 11694, pages 191–221. Springer, Heidelberg, August 2019. 19, 28, 64, 158

[CCL$^+$20] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 12111, pages 266–296. Springer, Heidelberg, May 2020. 19, 28, 64, 158

[CD98] R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *Advances in Cryptology – CRYPTO'98*, *Lecture Notes in Computer Science* 1462, pages 424–441. Springer, Heidelberg, August 1998. 57

[CDE$^+$18] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *Advances in Cryptology – CRYPTO 2018, Part II*, *Lecture Notes in Computer Science* 10992, pages 769–798. Springer, Heidelberg, August 2018. 208

[CDG$^+$18a] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *Advances in Cryptology – ASIACRYPT 2018, Part II*, *Lecture Notes in Computer Science* 11273, pages 703–732. Springer, Heidelberg, December 2018. 107

[CDG$^+$18b] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. `https://eprint.iacr.org/2018/1021`. 107

[CH89] R. A. Croft and S. P. Harris. Public-key cryptography and reusable shared secret. *Cryptography and Coding*, pages 189–201, 1989. 155

[CIL17] G. Castagnos, L. Imbert, and F. Laguillaumie. Encryption switching protocols revisited: Switching modulo p. In *Advances in Cryptology – CRYPTO 2017, Part I*, *Lecture Notes in Computer Science* 10401, pages 255–287. Springer, Heidelberg, August 2017. 18, 27, 51, 52

[CJLN09] G. Castagnos, A. Joux, F. Laguillaumie, and P. Q. Nguyen. Factoring pq^2 with quadratic forms: Nice cryptanalyses. In *Advances in Cryptology – ASIACRYPT 2009*, *Lecture Notes in Computer Science* 5912, pages 469–486. Springer, Heidelberg, December 2009. 18, 27

[CKY09] J. Camenisch, A. Kiayias, and M. Yung. On the portability of generalized Schnorr proofs. In *Advances in Cryptology – EUROCRYPT 2009*, *Lecture Notes in Computer Science* 5479, pages 425–442. Springer, Heidelberg, April 2009. 60, 98

[CL84] H. Cohen and H. W. Lenstra Jr. Heuristics on class groups. In *Number Theory*, pages 26–36, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg. 72

[CL09] G. Castagnos and F. Laguillaumie. On the security of cryptosystems with quadratic decryption: The nicest cryptanalysis. In *Advances in Cryptology – EUROCRYPT 2009*, *Lecture Notes in Computer Science* 5479, pages 260–277. Springer, Heidelberg, April 2009. 18, 27

[CL15] G. Castagnos and F. Laguillaumie. Linearly homomorphic encryption from DDH. In *Topics in Cryptology – CT-RSA 2015*, *Lecture Notes in Computer Science* 9048, pages 487–505. Springer, Heidelberg, April 2015. 17, 27, 42, 46, 49, 52, 63, 64, 65, 66, 69, 89, 90, 93, 207, 229

[CLT18a] G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In *Advances in Cryptology – ASIACRYPT 2018, Part II*, *Lecture Notes in Computer Science* 11273, pages 733–764. Springer, Heidelberg, December 2018. 19, 28, 64, 107, 108, 109, 116, 125, 128, 129, 130, 146, 231

[CLT18b] G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. Cryptology ePrint Archive, Report 2018/791, 2018. `https://eprint.iacr.org/2018/791`. 117

[CLT20] G. Castagnos, F. Laguillaumie, and I. Tucker. A tighter proof for cca secure inner product functional encryption: Genericity meets efficiency. 2020. 64, 108

[CMP20] R. Canetti, N. Makriyannis, and U. Peled. Uc non-interactive, proactive, threshold ecdsa. Cryptology ePrint Archive, Report 2020/492, 2020. `https://eprint.iacr.org/2020/492`. 182, 209

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, *Lecture Notes in Computer Science* 2260, pages 360–363. Springer, Heidelberg, December 2001. 149

[Coh00] H. Cohen. *A course in computational algebraic number theory.* Springer-Verlag, 2000. 48, 49, 50

[Cox89] D. Cox. *Primes of the Form X2 + Ny2: Fermat, Class Field Theory, and Complex Multiplication*. Monographs and textbooks in pure and applied mathematics. Wiley, 1989. 46, 47, 48, 49

[CP93] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO'92*, *Lecture Notes in Computer Science* 740, pages 89–105. Springer, Heidelberg, August 1993. 20, 29, 93

[CPP06] B. Chevallier-Mames, P. Paillier, and D. Pointcheval. Encoding-free ElGamal encryption without random oracles. In *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, *Lecture Notes in Computer Science* 3958, pages 91–104. Springer, Heidelberg, April 2006. 89

[CPP16] G. Couteau, T. Peters, and D. Pointcheval. Encryption switching protocols. In *Advances in Cryptology – CRYPTO 2016, Part I*, *Lecture Notes in Computer Science* 9814, pages 308–338. Springer, Heidelberg, August 2016. 42

[Cra97] R. Cramer. Modular design of secure yet practical cryptographic protocols. 1997. 57

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *Advances in Cryptology – CRYPTO'97*, *Lecture Notes in Computer Science* 1294, pages 410–424. Springer, Heidelberg, August 1997. 57, 58

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'98*, *Lecture Notes in Computer Science* 1462, pages 13–25. Springer, Heidelberg, August 1998. 91

[CS02] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology – EUROCRYPT 2002*, *Lecture Notes in Computer Science* 2332, pages 45–64. Springer, Heidelberg, April / May 2002. 16, 17, 19, 20, 26, 28, 29, 63, 64, 73, 75, 76, 79, 82, 83, 85, 86, 87, 108, 114, 118, 129, 147, 165, 236

[CS03] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology – CRYPTO 2003*, *Lecture Notes in Computer Science* 2729, pages 126–144. Springer, Heidelberg, August 2003. 64, 91, 129

[DDM16] P. Datta, R. Dutta, and S. Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, *Lecture Notes in Computer Science* 9614, pages 164–195. Springer, Heidelberg, March 2016. 106

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. 46

[Des88] Y. Desmedt. Society and group oriented cryptography: A new concept. In *Advances in Cryptology – CRYPTO'87*, *Lecture Notes in Computer Science* 293, pages 120–127. Springer, Heidelberg, August 1988. 16, 25, 155

[DF90] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology – CRYPTO'89*, *Lecture Notes in Computer Science* 435, pages 307–315. Springer, Heidelberg, August 1990. 16, 25, 155

[DF02] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – ASIACRYPT 2002*, *Lecture Notes in Computer Science* 2501, pages 125–142. Springer, Heidelberg, December 2002. 20, 29, 58, 59, 60, 71, 98, 100

[DG03] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th Annual ACM Symposium on Theory of Computing*, pages 426–437. ACM Press, June 2003. 46

[DGH$^+$13] J.-L. Danger, S. Guilley, P. Hoogvorst, C. Murdica, and D. Naccache. A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards. *Journal of Cryptographic Engineering*, 3, 11 2013. 161

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, *Lecture Notes in Computer Science* 1992, pages 119–136. Springer, Heidelberg, February 2001. 42, 89

[DJN$^+$20] I. Damgård, T. P. Jakobsen, J. B. Nielsen, J. I. Pagter, and M. B. Østergård. Fast threshold ecdsa with honest majority. Cryptology ePrint Archive, Report 2020/501, 2020. `https://eprint.iacr.org/2020/501`. 209

[DKLs18] J. Doerner, Y. Kondi, E. Lee, and a. shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pages 980–997. IEEE Computer Society Press, May 2018. 156, 201

[DKLs19] J. Doerner, Y. Kondi, E. Lee, and a. shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pages 1051–1066. IEEE Computer Society Press, May 2019. 156, 182, 201

[DKO$^+$19] A. P. K. Dalskov, M. Keller, C. Orlandi, K. Shrishak, and H. Shulman. Securing dnssec keys via threshold ecdsa from generic mpc. *IACR Cryptology ePrint Archive*, 2019:889, 2019. 201

[DOT18] P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 10770, pages 245–277. Springer, Heidelberg, March 2018. 106

[DPP20] X. T. Do, D. H. Phan, and D. Pointcheval. Traceable inner product functional encryption. In *Topics in Cryptology – CT-RSA 2020*, *Lecture Notes in Computer Science* 12006, pages 564–585. Springer, Heidelberg, February 2020. 107

[DPSZ12] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, *Lecture Notes in Computer Science* 7417, pages 643–662. Springer, Heidelberg, August 2012. 210

[EHK$^+$13] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In *Advances in Cryptology – CRYPTO 2013, Part II*, *Lecture Notes in Computer Science* 8043, pages 129–147. Springer, Heidelberg, August 2013. 106

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. of FOCS 87*, pages 427–437. IEEE Computer Society, 1987. 35

[FKP16] M. Fersch, E. Kiltz, and B. Poettering. On the provable security of (ec)dsa signatures. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1651–1662, New York, NY, USA, 2016. Association for Computing Machinery. 161

[FKP17] M. Fersch, E. Kiltz, and B. Poettering. On the one-per-message unforgeability of (EC)DSA and its variants. In *TCC 2017: 15th Theory of Cryptography Conference, Part II*, *Lecture Notes in Computer Science* 10678, pages 519–534. Springer, Heidelberg, November 2017. 161

[FLOP18] T. K. Frederiksen, Y. Lindell, V. Osheter, and B. Pinkas. Fast distributed RSA key generation for semi-honest and malicious adversaries. In *Advances in Cryptology – CRYPTO 2018, Part II*, *Lecture Notes in Computer Science* 10992, pages 331–361. Springer, Heidelberg, August 2018. 156

[FO97] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO'97*, *Lecture Notes in Computer Science* 1294, pages 16–30. Springer, Heidelberg, August 1997. 60

[Fuj16] E. Fujisaki. Improving practical UC-secure commitments based on the DDH assumption. In *SCN 16: 10th International Conference on Security in Communication Networks*, *Lecture Notes in Computer Science* 9841, pages 257–272. Springer, Heidelberg, August / September 2016. 46

[Gal02] S. D. Galbraith. Elliptic curve Paillier schemes. *Journal of Cryptology*, 15(2):129–138, March 2002. 89

[Gen04] R. Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In *Advances in Cryptology – CRYPTO 2004*, *Lecture Notes in Computer Science* 3152, pages 220–236. Springer, Heidelberg, August 2004. 46

[GG18] R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1179–1194. ACM Press, October 2018. 21, 30, 156, 158, 182, 183, 191, 201, 202, 203, 209

[GG20] R. Gennaro and S. Goldfeder. One round threshold ecdsa with identifiable abort. Cryptology ePrint Archive, Report 2020/540, 2020. `https://eprint.iacr.org/2020/540`. 182, 209

[GGG$^+$14] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *Advances in Cryptology – EUROCRYPT 2014*, *Lecture Notes in Computer Science* 8441, pages 578–602. Springer, Heidelberg, May 2014. 106

[GGH13a] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology – EUROCRYPT 2013*, *Lecture Notes in Computer Science* 7881, pages 1–17. Springer, Heidelberg, May 2013. 105

[GGH$^+$13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, October 2013. 105

[GGHZ16] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Functional encryption without obfuscation. In *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, *Lecture Notes in Computer Science* 9563, pages 480–511. Springer, Heidelberg, January 2016. 15, 24, 105

[GGN16] R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS 16: 14th International Conference on Applied Cryptography and Network Security*, *Lecture Notes in Computer Science* 9696, pages 156–174. Springer, Heidelberg, June 2016. 21, 30, 156, 201

[Gil99] N. Gilboa. Two party RSA key generation. In *Advances in Cryptology – CRYPTO'99*, *Lecture Notes in Computer Science* 1666, pages 116–129. Springer, Heidelberg, August 1999. 182

[GJKR96a] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *Advances in Cryptology – CRYPTO'96*, *Lecture Notes in Computer Science* 1109, pages 157–172. Springer, Heidelberg, August 1996. 16, 25, 156, 160

[GJKR96b] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Advances in Cryptology – EUROCRYPT'96*, *Lecture Notes in Computer Science* 1070, pages 354–371. Springer, Heidelberg, May 1996. 155, 161

[GKP$^+$13] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology – CRYPTO 2013, Part II*, *Lecture Notes in Computer Science* 8043, pages 536–553. Springer, Heidelberg, August 2013. 105

[GKSS20] A. Gagol, J. Kula, D. Straszak, and M. Swietek. Threshold ecdsa for decentralized asset custody. Cryptology ePrint Archive, Report 2020/498, 2020. `https://eprint.iacr.org/2020/498`. 182, 209

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 13, 17, 23, 26, 37, 39, 42

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304. ACM Press, May 1985. 56

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. 45

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987. 16, 25, 35, 37

[GMY06] J. A. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, April 2006. 98, 179

[GO94] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. 37

[Gol01] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, Cambridge, UK, 2001. 36, 57

[GPS06] M. Girault, G. Poupard, and J. Stern. On the fly authentication and signature schemes based on groups of unknown order. *Journal of Cryptology*, 19(4):463–487, October 2006. 20, 29, 94, 96, 202, 237, 240

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008. 54, 55

[Gro04] J. Groth. *Honest Verifier Zero-knowledge Arguments Applied.* PhD thesis, Denmark, 2004. 98, 179

[GVW12] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology – CRYPTO 2012*, *Lecture Notes in Computer Science* 7417, pages 162–179. Springer, Heidelberg, August 2012. 105, 149

[HJPT98] D. Hühnlein, M. J. Jacobson Jr., S. Paulus, and T. Takagi. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In *Advances in Cryptology – EUROCRYPT'98*, *Lecture Notes in Computer Science* 1403, pages 294–307. Springer, Heidelberg, May / June 1998. 66

[HL10] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions.* Springer-Verlag, 1st edition, 2010. 162

[HM89] J. L. Hafner and K. S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American mathematical society*, 2(4):837–850, 1989. 49

[HM00] S. Hamdy and B. Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In *Advances in Cryptology – ASIACRYPT 2000*, *Lecture Notes in Computer Science* 1976, pages 234–247. Springer, Heidelberg, December 2000. 49, 50

[HMRT12] C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft. Efficient RSA key generation and threshold Paillier in the two-party setting. In *Topics in Cryptology – CT-RSA 2012*, *Lecture Notes in Computer Science* 7178, pages 313–331. Springer, Heidelberg, February / March 2012. 46

[HO09] B. Hemenway and R. Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. In *Electronic Colloquium on Computational Complexity, Report*, pages 09–127, 2009. 77

[HO12] B. Hemenway and R. Ostrovsky. Extended-DDH and lossy trapdoor functions. In *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, *Lecture Notes in Computer Science* 7293, pages 627–643. Springer, Heidelberg, May 2012. 69

[HPT99] M. Hartmann, S. Paulus, and T. Takagi. NICE - new ideal coset encryption. In *Cryptographic Hardware and Embedded Systems – CHES'99*, *Lecture Notes in Computer Science* 1717, pages 328–339. Springer, Heidelberg, August 1999. 18, 27

[HS06] S. Hamdy and F. Saidak. Arithmetic properties of class numbers of imaginary quadratic fields. *JP Journal of Algebra, Number Theory and Application*, 6(1):129–148, 2006. 49, 72

[Jac99] M. J. Jacobson. *Subexponential class group computation in quadratic orders.* Shaker Verlag GmbH, Technische Universität Darmstadt, 1999. 49

[JL13] M. Joye and B. Libert. Efficient cryptosystems from 2^k-th power residue symbols. In *Advances in Cryptology – EUROCRYPT 2013*, *Lecture Notes in Computer Science* 7881, pages 76–92. Springer, Heidelberg, May 2013. 42, 208

[JSSS20] J. Jancar, V. Sedlacek, P. Svenda, and M. Sys. Minerva: The curse of ecdsa nonces. IACR-CHES-2020, 2020. 161

[JSW08] M. J. Jacobson Jr., R. Scheidler, and D. Weimer. An adaptation of the NICE cryptosystem to real quadratic orders. In *AFRICACRYPT 08: 1st International Conference on Cryptology in Africa*, *Lecture Notes in Computer Science* 5023, pages 191–208. Springer, Heidelberg, June 2008. 18, 27

[JW09] M. Jacobson and H. Williams. *Solving the Pell Equation.* CMS Books in Mathematics. Springer-Verlag, New York, 2009. 18, 27

[Kap73] P. Kaplan. Divisibilité par 8 du nombr des classes des corps quadratiques dont le 2-groupe des classes est cyclique, et réciprocité biquadratique. *J. Math. Soc. Japan*, 25(4):596–608, 10 1973. 49

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology – EUROCRYPT 2008*, *Lecture Notes in Computer Science* 4965, pages 146–162. Springer, Heidelberg, April 2008. 106, 148

[KY19] S. Katsumata and S. Yamada. Non-zero inner product encryption schemes from various assumptions: LWE, DDH and DCR. In *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 11443, pages 158–188. Springer, Heidelberg, April 2019. 15, 24, 106, 109, 149

[Lag80] J. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *Journal of Algorithms*, 1(2):142 – 186, 1980. 66, 70

[Lin17a] Y. Lindell. Fast secure two-party ECDSA signing. In *Advances in Cryptology – CRYPTO 2017, Part II*, *Lecture Notes in Computer Science* 10402, pages 613–644. Springer, Heidelberg, August 2017. 21, 30, 156, 157, 159, 163, 167, 168, 169, 170, 179, 180, 182, 241

[Lin17b] Y. Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pages 277–346. Springer International Publishing, Cham, 2017. 38

[Lip03] H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *Advances in Cryptology – ASIACRYPT 2003*, *Lecture Notes in Computer Science* 2894, pages 398–415. Springer, Heidelberg, November / December 2003. 60

[Lip12] H. Lipmaa. Secure accumulators from euclidean rings without trusted setup. In *ACNS 12: 10th International Conference on Applied Cryptography and Network Security*, *Lecture Notes in Computer Science* 7341, pages 224–240. Springer, Heidelberg, June 2012. 18, 27, 71

[LM19] R. W. F. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In *Advances in Cryptology – CRYPTO 2019, Part I*, *Lecture Notes in Computer Science* 11692, pages 530–560. Springer, Heidelberg, August 2019. 71, 179, 203

[LN18] Y. Lindell and A. Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1837–1854. ACM Press, October 2018. 156, 158, 182, 201, 202, 203, 209

[LOS$^+$10] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology – EUROCRYPT 2010*, *Lecture Notes in Computer Science* 6110, pages 62–91. Springer, Heidelberg, May / June 2010. 149

[MR01] P. D. MacKenzie and M. K. Reiter. Two-party generation of DSA signatures. In *Advances in Cryptology – CRYPTO 2001*, *Lecture Notes in Computer Science* 2139, pages 137–154. Springer, Heidelberg, August 2001. 21, 30, 156, 163, 182

[MR04a] P. D. MacKenzie and M. K. Reiter. Two-party generation of DSA signatures. *Int. J. Inf. Sec.*, 2(3-4):218–239, 2004. 155

[MR04b] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381. IEEE Computer Society Press, October 2004. 55

[MR07] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. 117

[MY04] P. D. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In *Advances in Cryptology – EUROCRYPT 2004*, *Lecture Notes in Computer Science* 3027, pages 382–400. Springer, Heidelberg, May 2004. 46

[NP15] M. Nandi and T. Pandit. Generic conversions from cpa to cca secure functional encryption. Cryptology ePrint Archive, Report 2015/457, 2015. `https://eprint.iacr.org/2015/457`. 110

[NS98] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *ACM CCS 98: 5th Conference on Computer and Communications Security*, pages 59–66. ACM Press, November 1998. 42

[O'N10] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/2010/556`. 14, 24, 105, 110, 149, 152

[OPJM10] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. Scifi - a system for secure face identification. In *2010 IEEE Symposium on Security and Privacy (SP)*, pages 239–254, Los Alamitos, CA, USA, may 2010. IEEE Computer Society. 42

[Orl11] C. Orlandi. Is multiparty computation any good in practice? In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5848–5851, 2011. 210

[OT10] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Advances in Cryptology – CRYPTO 2010*, *Lecture Notes in Computer Science* 6223, pages 191–208. Springer, Heidelberg, August 2010. 149

[OU98] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology – EUROCRYPT'98*, *Lecture Notes in Computer Science* 1403, pages 308–318. Springer, Heidelberg, May / June 1998. 42

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, *Lecture Notes in Computer Science* 1592, pages 223–238. Springer, Heidelberg, May 1999. 17, 18, 27, 33, 40, 42, 43, 89

[PAR20] PARI Group, Univ. Bordeaux. *PARI/GP version 2.11.4*, 2020. available from `http://pari.math.u-bordeaux.fr/`. 51, 158, 159, 179

[Pie19] K. Pietrzak. Simple verifiable delay functions. In *ITCS 2019: 10th Innovations in Theoretical Computer Science Conference*, pages 60:1–60:15. LIPIcs, January 2019. 71

[PR05] R. Pass and A. Rosen. Concurrent non-malleable commitments. In *46th Annual Symposium on Foundations of Computer Science*, pages 563–572. IEEE Computer Society Press, October 2005. 46

[PS96] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology – EUROCRYPT'96*, *Lecture Notes in Computer Science* 1070, pages 387–398. Springer, Heidelberg, May 1996. 160

[PS00] G. Poupard and J. Stern. Short proofs of knowledge for factoring. In *PKC 2000: 3rd International Workshop on Theory and Practice in Public Key Cryptography*, *Lecture Notes in Computer Science* 1751, pages 147–166. Springer, Heidelberg, January 2000. 202

[PT00] S. Paulus and T. Takagi. A new public-key cryptosystem over a quadratic order with quadratic decryption time. *Journal of Cryptology*, 13(2):263–272, March 2000. 18, 27, 48

[PV96] D. Pointcheval and S. Vaudenay. On provable security for digital signature algorithms, 1996. 161

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008. 209

[SA19] N. P. Smart and Y. T. Alaoui. Distributing any elliptic curve based protocol: With an application to mixnets. *IACR Cryptology ePrint Archive*, 2019:768, 2019. 201

[Sch90] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – CRYPTO'89*, *Lecture Notes in Computer Science* 435, pages 239–252. Springer, Heidelberg, August 1990. 20, 29, 59, 93

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. 16, 25, 156, 160, 167, 188

[SG98] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology – EUROCRYPT'98*, *Lecture Notes in Computer Science* 1403, pages 1–16. Springer, Heidelberg, May / June 1998. 155

[Sha79] A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 15, 25, 35

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO'84*, *Lecture Notes in Computer Science* 196, pages 47–53. Springer, Heidelberg, August 1984. 149

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT'97*, *Lecture Notes in Computer Science* 1233, pages 256–266. Springer, Heidelberg, May 1997. 39

[Sho00] V. Shoup. Practical threshold signatures. In *Advances in Cryptology – EUROCRYPT 2000*, *Lecture Notes in Computer Science* 1807, pages 207–220. Springer, Heidelberg, May 2000. 155

[SS01] D. R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *ACISP 01: 6th Australasian Conference on Information Security and Privacy*, *Lecture Notes in Computer Science* 2119, pages 417–434. Springer, Heidelberg, July 2001. 16, 25, 156, 160

[SS10] A. Sahai and H. Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 463–472. ACM Press, October 2010. 15, 24, 105

[Ste11] D. Stehlé. *Euclidean lattices: algorithms and cryptography.* Habilitation à diriger des recherches, Ecole normale supérieure de lyon - ENS LYON, October 2011. 54

[SW05] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2005*, *Lecture Notes in Computer Science* 3494, pages 457–473. Springer, Heidelberg, May 2005. 105, 149

[Tom19] J. Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In *Advances in Cryptology – ASIACRYPT 2019, Part III*, *Lecture Notes in Computer Science* 11923, pages 459–488. Springer, Heidelberg, December 2019. 106

[Van92] S. Vanstone. Responses to nist's proposal. *Communications of the ACM*, 35:50–52, July 1992. (communicated by John Anderson). 159

[Wee17] H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *TCC 2017: 15th Theory of Cryptography Conference, Part I*, *Lecture Notes in Computer Science* 10677, pages 206–233. Springer, Heidelberg, November 2017. 106, 109, 150, 151, 152

[Wes18] B. Wesolowski. Efficient verifiable delay functions. Cryptology ePrint Archive, Report 2018/623, 2018. `https://eprint.iacr.org/2018/623`. Accepted to *EUROCRYPT 2019*. 18, 27

[Wes19] B. Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology – EUROCRYPT 2019, Part III*, *Lecture Notes in Computer Science* 11478, pages 379–407. Springer, Heidelberg, May 2019. 71

[Yao82] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society Press, November 1982. 16, 25

[ZMY17] S. Zhang, Y. Mu, and G. Yang. Achieving ind-cca security for functional encryption for inner products. In *Information Security and Cryptology*, pages 119–139, Cham, 2017. Springer International Publishing. 106, 112

List of Abbreviations

CCA Chosen Ciphertext Attack. 149

CRHF Collision Resistant Hash Function. 44, 120, 121, 123, 135, 137, 140, 142, 143, 147

DCR Decision Composite Residuosity. 20, 29, 40, 42, 43, 50, 51, 64, 68, 69, 73, 75, 91, 106–109, 115, 129–131, 142, 143, 145–148, 152, 155, 157, 165, 166, 207, 208, 231, 237

DDH Decision Diffie Hellman. 40, 42, 63, 64, 68, 69, 73, 74, 77, 79, 81, 88, 89, 93, 106–108, 113, 115, 125, 128, 130, 131, 142–144, 146, 147, 152, 199–201, 207, 229, 231

DDH-f Extended Decision Diffie Hellman in $< f >$. 10, 19, 28, 63, 64, 69, 70, 73–75, 78, 80, 82, 88–91, 93, 108, 109, 113, 117, 125, 129–131, 133, 134, 142, 143, 145, 152, 229

DDH-CL Decision Diffie Hellman in the CL framework. 10, 63, 69, 73, 89, 93

DE Double Encoding. 165, 166, 170, 175, 177, 195

DH Diffie Hellman. 147

DL Discrete Logarithm. 39, 49, 50, 59, 64, 65, 68–72, 74, 75, 89, 94, 113, 131, 159, 160, 168, 173, 175–177, 183

DPT Deterministic Polynomial Time. 41, 44, 45, 65, 76, 86, 109, 148

DSA Digital Signature Algorithm. 159–161

EC Elliptic Curve. 159, 160, 166, 167, 173, 177, 179, 180, 185, 187, 199, 201, 202

EC-DSA Elliptic Curve Digital Signature Algorithm. 11, 16–18, 21, 22, 25, 26, 28, 30, 31, 39, 43, 44, 155–164, 166–171, 174–178, 180–184, 187, 188, 190–192, 194, 195, 199–202, 207–209, 229, 241

EDDH Extended Decision Diffie Hellman. 69

EPHF Extended Projective Hash Function. 85–87, 112, 119, 125, 137, 147, 235, 236

FE Functional Encryption. 105–111, 125, 128–134, 143, 144, 150, 229

HSM-CL Hard Subroup Membership Problem. 10, 11, 19, 21, 28, 30, 63, 64, 68, 69, 73–75, 78, 79, 82, 88, 89, 91–94, 98, 99, 108, 109, 113, 116, 125, 128–133, 142, 144–148, 152, 157, 158, 164–166, 177, 178, 180–184, 187, 207, 229, 231, 237

IBR Identity Based Revocation. 106, 109, 149

IPFE Inner Product Functional Encryption. 10, 11, 63, 64, 105–109, 112, 114–119, 124–126, 128–131, 135–140, 142–150, 152, 209, 229, 231, 236

lcm Lowest Common Multiple. 64, 96, 97, 179, 181, 187, 201

LO Low Order. 70–73, 98, 99, 101, 102, 179, 183, 184, 192, 194, 199–201

LWE Learning With Errors. 106, 107, 130, 131, 143, 144, 210

MPC Multi-Party Computation. 9, 34

NIPE Non Zero Inner Product Encryption. 106, 148, 149

NIST National Institute of Standards and Technology. 161, 179, 180, 201

NIZK Non Interactive Zero Knowledge. 162

OTS One Time Signature. 135, 142, 143

OWF One Way Function. 164–166

PHF Projective Hash Function. 10, 11, 63, 64, 73, 74, 76–79, 81–85, 87, 89, 91, 92, 107–109, 112, 113, 115, 118, 124–126, 129, 135–137, 139, 140, 147, 150, 152, 157, 163–168, 170, 172, 173, 177, 207, 229, 234–236

PKE Public Key Encryption. 10, 41, 42, 69, 79, 83, 85, 89, 91, 92, 105, 108, 114, 118, 119, 125, 136–138, 150, 157, 164, 167, 175, 181–183, 187, 191

PoK Proof of Knowledge. 169, 199, 201

PPT Probabilistic Polynomial Time. 34, 39–46, 56, 59, 69–74, 76, 84, 86, 109–111, 148, 150, 151, 159, 165, 177, 183, 195, 241

PT Polynomial Time. 34, 38, 56, 57, 59, 95, 96, 100, 109, 110, 148, 164, 168

QR Quadratic Residuosity. 69

SD Subgroup Decomposition. 165, 166

SMP Subgroup Membership Problem. 112, 115, 119, 124, 170, 177, 200, 201

SNARK Succinct Non-interactive Argument of Knowledge. 71

SR Strong Root. 70–73, 98, 99, 101, 102, 179, 183, 184, 187, 192, 194, 199–201

UC Universally Composable. 38, 209

VSS Verifiable Secret Sharing. 35, 36, 188

ZK Zero Knowledge. 56, 60, 94, 96, 167, 170, 175, 188, 192, 208

ZKAoK Zero Knowledge Argument of Knowledge. 58, 98–100, 102, 103, 179, 182–184, 187–193, 199, 202, 229

ZKP Zero Knowledge Proof. 169, 179, 180, 182, 187, 200–202

ZKPoK Zero Knowledge Proof of Knowledge. 57, 60, 94–97, 162, 169, 179, 181, 184–190, 192–194, 199–202, 229, 230, 237, 238

List of Figures

List of Tables

Appendices

A Comparing our PHF properties to those of [BBL17]

Chosen plaintext security. We here demonstrate that any PHF which can be used to instantiate the generic construction of [BBL17] for inner product functional encryption secure against chosen plaintext attacks is key homomorphic and vector-smooth.

We refer the reader to [BBL17] for formal definitions of *strong diversity*, *translation indistinguishability*, and *universal translation indistinguishability.*

Lemma A.1. Consider a ring $\mathscr{R}$, an integer $\ell > 0$, a subgroup membership problem $\mathcal{SM} := (\mathcal{X}, \mathcal{L}, \mathcal{W}, \mathsf{R})$, and associated PHF H. Consider a function $\mathsf{hk}_\perp : \mathcal{X}\backslash\mathcal{L} \mapsto K_{\mathsf{hk}}$, an element $\mathfrak{g}_\perp \in \Pi$, and two positive integers n_f and M. For H to meet the properties required in [BBL17] to build $\mathsf{ind\text{-}fe\text{-}cpa}$ inner product functional encryption schemes, it must be key homomorphic, $(\mathsf{hk}_\perp, M, \epsilon_{ti})$-translation-indistinguishable, $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, n_f)$-strongly diverse, and one must chose $M := (n_f/\ell)^{1/2}$. If the above holds, then H is $(\ell \cdot \epsilon_{ti})$-vector-smooth over $\mathcal{X}$ on $\mathfrak{G}_\perp := \langle \mathfrak{g}_\perp \rangle$.

Proof. Consider a PHF H as described in the lemma. For $i \in [\ell]$, sample $\mathsf{hk}_i \leftarrow \mathsf{hashkg}(\mathcal{SM})$ independently, let $\mathbf{hk} := (\mathsf{hk}_1, \ldots, \mathsf{hk}_\ell)$ and $\mathbf{hp} \leftarrow \mathsf{projkg}(\mathbf{hk})$. Consider $\boldsymbol{m}_0 \neq \boldsymbol{m}_1 \in \mathcal{M}$, and denote $\boldsymbol{m} := \boldsymbol{m}_0 - \boldsymbol{m}_1$. Consider vectors $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell)$ as in Definition 4.10. Let $X \hookleftarrow \mathcal{X}\backslash\mathcal{L}$, and $Y \hookleftarrow \mathcal{U}(\mathfrak{G}_\perp)$. Then H is $\delta_{vs}(\ell)$-vector-smooth over $\mathcal{X}$ on $\mathfrak{G}_\perp$ if the distributions induced by $\mathcal{U}$ and $\mathcal{V}$ are $\delta_{vs}(\ell)$-close, where:

$$\mathcal{U} := \{X, \{\mathsf{projkg}(\mathsf{hk}_i)\}_{i\in[\ell]}, \{\langle \mathbf{hk}, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]}, \mathsf{hash}(\langle \mathbf{hk}, \boldsymbol{b}_\ell\rangle, X) \cdot Y\} \text{ and}$$

$$\mathcal{V} := \left\{X, \{\mathsf{projkg}(\mathsf{hk}_i)\}_{i\in[\ell]}, \{\langle \mathbf{hk}, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]}, \mathsf{hash}(\langle \mathbf{hk}, \boldsymbol{b}_\ell\rangle, X)\right\}.$$

We first use the $(\mathsf{hk}_\perp, M, \epsilon_{ti})$-translation-indistinguishability of H, and replace each hk_i by $\mathsf{hk}_i + a_i \cdot \mathsf{hk}_\perp(X)$ for $a_i \hookleftarrow \{-M, \ldots, M\}$ satisfying $\boldsymbol{a} \in \boldsymbol{m}$. By repeated sampling, it holds that $\mathcal{V}'$ is $\ell\epsilon_{ti}$-close to $\mathcal{V}$, where:

$$\begin{aligned}\mathcal{V}' := \{X, &\{\mathsf{projkg}(\mathsf{hk}_i + a_i \cdot \mathsf{hk}_\perp(X))\}_{i\in[\ell]}, \{\langle \mathbf{hk} + \boldsymbol{a} \cdot \mathsf{hk}_\perp(X), \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]},\\ &\mathsf{hash}(\langle \mathbf{hk} + \boldsymbol{a} \cdot \mathsf{hk}_\perp(X), \boldsymbol{b}_\ell\rangle, X)\}\end{aligned}$$

By construction, $\boldsymbol{b}_i \in \boldsymbol{m}^\perp$ for $i \in [\ell - 1]$, furthermore, by the homomorphic properties of H it holds that:

$$\begin{aligned}\mathcal{V}' := \{X, &\{\mathsf{projkg}(\mathsf{hk}_i + a_i \cdot \mathsf{hk}_\perp(X))\}_{i\in[\ell]}, \{\langle \mathbf{hk}, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]},\\ &\mathsf{hash}(\langle \mathbf{hk}, \boldsymbol{b}_\ell\rangle, X) \cdot \mathsf{hash}(\mathsf{hk}_\perp(X), X)^{\langle \boldsymbol{a}, \boldsymbol{b}_\ell\rangle}\}\end{aligned}$$

From the $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, n_f)$-strong diversity of H, we can now write:

$$\mathcal{V}' = \{X, \{\mathsf{projkg}(\mathsf{hk}_i)\}_{i\in[\ell]}, \{\langle \mathbf{hk}, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]}, \mathsf{hash}(\langle \mathbf{hk}, \boldsymbol{b}_\ell\rangle, X) \cdot \mathfrak{g}_\perp^{\langle \boldsymbol{a}, \boldsymbol{b}_\ell\rangle}\}$$

Now since $\mathfrak{g}_\perp$ is of order n_f; $\boldsymbol{a}$ is sampled uniformly in $\{-(n_f/\ell)^{1/2}, \ldots, (n_f/\ell)^{1/2}\}$ subject on the condition $\boldsymbol{a} \in \langle \boldsymbol{m} \rangle$, and $\langle \boldsymbol{m}, \boldsymbol{b}_\ell \rangle \neq 0$, the distribution induced by $\mathfrak{g}_\perp^{\langle \boldsymbol{a}, \boldsymbol{b}_\ell\rangle}$ is the uniform distribution in the subgroup $\mathfrak{G}_\perp = \langle \mathfrak{g}_\perp \rangle$. Thus:

$$\mathcal{V}' = \mathcal{U} = \{X, \mathbf{hp}, \{\langle \mathbf{hk}, \boldsymbol{b}_i\rangle\}_{i\in[\ell-1]}, \mathsf{hash}(\langle \mathbf{hk}, \boldsymbol{b}_\ell\rangle, X) \cdot Y | Y \hookleftarrow \mathcal{U}(\mathfrak{G}_\perp)\}$$

□

Chosen ciphertext security. We here demonstrate that any PHF which can be used to instantiate the generic construction of [BBL17] for inner product functional encryption secure against chosen ciphertext attacks is key homomorphic and vector-universal.

We refer the reader to [BBL17] for the definitions of *2-universality* and *universal translation indistinguishability.*

Lemma A.2. Consider a ring $\mathcal{R}$, an integer $\ell > 0$, a subgroup membership problem $\mathcal{SM} := (\mathcal{X}, \mathcal{L}, \mathcal{W}, \mathsf{R})$, and associated EPHF $\mathsf{eH} := (\mathsf{ehashkg}, \mathsf{eprojkg}, \mathsf{ehash}, \mathsf{eprojhash})$. Consider a function $\mathsf{ehashkg}'$, taking as input a security parameter 1^λ and outputting a hashing key ehk in some set $K'_{\mathsf{ehk}} \subseteq K_{\mathsf{ehk}}$, an element $\mathfrak{g}_\perp \in \Pi$, $\epsilon_{uti}, \epsilon_{2u} > 0$, positive integers n_f, M and a subset Σ of $\mathbf{Z}$. For eH to meet the properties required in [BBL17] to build $\mathsf{ind\text{-}fe\text{-}cca}$ inner product functional encryption schemes, it must be key-homomorphic, projection-key-homomorphic, $(\mathsf{ehashkg}', M, \epsilon_{uti})$-universally-translation-indistinguishable and it must hold that for any $t \in \Sigma$, the PHF $(t \cdot \mathsf{ehashkg}', \mathsf{eprojkg}, \mathsf{ehash}, \mathsf{eprojhash})$ is ϵ_{2u}-universal$_2$, where the algorithm $t \cdot \mathsf{ehashkg}'$ runs $\mathsf{ehashkg}'$ and multiplies the output by t. Furthermore one must chose $M := (n_f/\ell)^{1/2}$, and $\Sigma := \{1, \ldots, n_f - 1\}$. If the above holds, then H is $(2\ell \cdot \epsilon_{uti} + \epsilon_{2u})$-vector-universal.

Proof. Consider an EPHF $\mathsf{eH} := (\mathsf{ehashkg}, \mathsf{eprojkg}, \mathsf{ehash}, \mathsf{eprojhash})$ as in the lemma statement. Consider any $\boldsymbol{m} \in \{\boldsymbol{x}_0 - \boldsymbol{x}_1 \mid \boldsymbol{x}_0 \neq \boldsymbol{x}_1 \in \mathcal{M}\}$ and vectors $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell) \in \mathcal{R}^{\ell \times \ell}$ as in Definition 4.15. For $i \in [\ell]$, let $\mathsf{ehk}_i \leftarrow \mathsf{ehashkg}(\mathcal{SM})$, and denote $\mathbf{ehk} := (\mathsf{ehk}_1, \ldots, \mathsf{ehk}_\ell)$. The EPHF eH is $\delta_{vu}(\ell)$-vector-universal if for any $\mathbf{ehp} \in (K_{\widehat{\mathsf{ehp}}})^\ell$; any $\boldsymbol{k} \in \mathcal{K}$ s.t. $\boldsymbol{k} \notin \boldsymbol{m}^\perp$; any $(x^*, e^*) \in \widehat{\mathcal{X}} \times E$, $(x, e) \in \widehat{\mathcal{X}} \backslash \mathcal{L} \times E$, s.t. $(x, e) \neq (x^*, e^*)$, and for any $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\ell-1}) \in (K_{\mathsf{ehk}})^{\ell-1}$; $(\pi_1^*, \ldots, \pi_\ell^*) \in \Pi^\ell$ and $\pi \in \Pi$ it holds that:

$$\begin{aligned}
\Pr\big[\ &\mathsf{ehash}(\langle \mathbf{ehk}, \boldsymbol{k}\rangle, x, e) = \pi \ \wedge\ (\mathsf{ehash}(\mathsf{ehk}_i, x^*, e^*) = \pi_i^* \text{ for } i \in [\ell]) \\
&\qquad \wedge \mathsf{eprojkg}(\mathbf{ehk}) = \mathbf{ehp} \wedge (\boldsymbol{b}_j^T \cdot \mathbf{ehk} = \boldsymbol{v}_j \text{ for } j \in [\ell - 1])\big] \\
&\leqslant \delta_{2vu}(\ell) \cdot \Pr\big[\ (\mathsf{ehash}(\mathsf{ehk}_i, x^*, e^*) = \pi_i^* \text{ for } i \in [\ell]) \\
&\qquad \wedge \mathsf{eprojkg}(\mathbf{ehk}) = \mathbf{ehp} \wedge (\boldsymbol{b}_j^T \cdot \mathbf{ehk} = \boldsymbol{v}_j \text{ for } j \in [\ell - 1])\big].
\end{aligned}$$

Let E_1 denote the event '$\mathsf{ehash}(\mathsf{ehk}_i, x^*, e^*) = \pi_i^*$ for $i \in [\ell]$', E_2 denote the event '$\mathsf{eprojkg}(\mathbf{ehk}) = \mathbf{ehp}$', and E_3 denote the event '$\boldsymbol{b}_j^T \cdot \mathbf{ehk} = \boldsymbol{v}_j$ for $j \in [\ell - 1]$'. Let X be a random variable following the same distribution as $\mathsf{ehash}(\langle \mathbf{ehk}, \boldsymbol{k}\rangle, x, e)$ and denote E_0 the event '$X = \pi$'.

We first use the $(\mathsf{ehashkg}', M, \epsilon_{uti})$-universal-translation-indistinguishability of eH. To this end sample $\mathsf{ehk}' \hookleftarrow \mathsf{ehashkg}'(\mathcal{SM})$. One also samples $\mathsf{ehk}''_i \hookleftarrow \mathsf{ehashkg}(\mathcal{SM})$ and $\alpha_i \hookleftarrow \{-M, \ldots, M\}$ for $i \in [\ell]$, such that $\boldsymbol{\alpha} \in \langle \boldsymbol{m} \rangle$. Since $K_{\mathsf{ehk}'} \subseteq \mathcal{R}^{2a}$, and $\boldsymbol{\alpha} \in \mathcal{R}^\ell$ we denote $\mathsf{ehk}'\boldsymbol{\alpha} := (\mathsf{ehk}'_0 \cdot \boldsymbol{\alpha}, \ldots, \mathsf{ehk}'_{2a-1} \cdot \boldsymbol{\alpha}) \in (\mathcal{R}^\ell)^{2a}$.

Consequently:

1. Consider the random variable

$$X' := \mathsf{ehash}(\boldsymbol{k}^T \cdot (\mathbf{ehk}'' + \mathsf{ehk}' \cdot \boldsymbol{\alpha}), x, e).$$

By the key homomorphism of eH, and denoting $t := \langle \boldsymbol{\alpha}, \boldsymbol{k} \rangle \in \mathcal{R}$, it holds that

$$X' = \mathsf{ehash}(\boldsymbol{k}^T \cdot \mathbf{ehk}'', x, e) \cdot \mathsf{ehash}(t \cdot \mathsf{ehk}', x, e).$$

Let E'_0 denote the event $X' = \pi$. If we consider random variables $Y'' := \mathsf{ehash}(\boldsymbol{k}^T \cdot \mathbf{ehk}'', x, e)$ and $Y' := \mathsf{ehash}(t \cdot \mathsf{ehk}', x, e)$

2. Event E'_1 denotes '$\mathsf{ehash}(\mathsf{ehk}''_i + \alpha_i \mathsf{ehk}', x^*, e^*) = \pi_i^*$ for $i \in [\ell]$'. Or equivalently $\mathsf{ehash}(\mathsf{ehk}''_i, x^*, e^*) \cdot \mathsf{ehash}(\mathsf{ehk}', x^*, e^*)^{\alpha_i} \pi_i^*$.

3. From the projection key homomorphism of eH, event E_2' is:

$$\mathbf{ehp} = \mathsf{eprojkg}(\mathbf{ehk}'' + \mathsf{ehk}' \cdot \boldsymbol{\alpha}) = \mathsf{eprojkg}(\mathbf{ehk}'') \cdot \mathsf{eprojkg}(\mathsf{ehk}')^{\boldsymbol{\alpha}}$$

4. Since $\boldsymbol{\alpha} \in \langle \boldsymbol{m} \rangle$ and by construction, $\boldsymbol{b}_i \in \boldsymbol{m}^{\perp}$ for $i \in [\ell - 1]$, event E_3' is

$$\boldsymbol{b}_j^T \cdot (\mathbf{ehk}'' + \boldsymbol{\alpha} \cdot \mathsf{ehk}') = \boldsymbol{b}_j^T \cdot \mathbf{ehk}'' = \boldsymbol{v}_j \text{ for } j \in [\ell - 1].$$

From the $(\mathsf{ehashkg}', M, \epsilon_{uti})$-universal-translation-indistinguishability of eH:

$$|\Pr[E_0 \wedge E_1 \wedge E_2 \wedge E_3] - \Pr[E_0' \wedge E_1' \wedge E_2' \wedge E_3']| \leqslant \ell \cdot \epsilon_{uti}.$$

Let us now consider the probability $\Pr[E_0' \wedge E_1' \wedge E_2' \wedge E_3'] \leqslant \Pr[E_0' | E_1' \wedge E_2' \wedge E_3']$. We denote $\mathfrak{p} = \Pr[E_0' | E_1' \wedge E_2' \wedge E_3']$.

We first observe that event E_3' is independent of ehk'. So the only fixed information on ehk' comes from event E_2' which, at most, fixes the value of $\mathsf{ehp}' := \mathsf{eprojkg}(\mathsf{ehk}')$; and E_1' fixing the value of $\mu^* := \mathsf{ehash}(\mathsf{ehk}', x^*, e^*)$.

Since, from the norm bounds on $\boldsymbol{k} \in \mathcal{K}$ and $\boldsymbol{\alpha}$, it holds that $t = \langle \boldsymbol{\alpha}, \boldsymbol{k} \rangle \in \{1, \dots, n_f - 1\}$ – the PHF $(t \cdot \mathsf{ehashkg}', \mathsf{eprojkg}, \mathsf{ehash}, \mathsf{eprojhash})$ is ϵ_{2u}-universal$_2$. Thus, for any $\pi' \in \Pi$, it holds that:

$$\begin{aligned}\Pr[\pi' = &\mathsf{ehash}(t\mathsf{ehk}', x, e) \wedge \mu^* = \mathsf{ehash}(\mathsf{ehk}', x^*, e^*) \wedge \mathsf{ehp}' = \mathsf{eprojkg}(\mathsf{ehk}')] \\ &\leqslant \epsilon_{2u} \cdot \Pr[\mu^* = \mathsf{ehash}(\mathsf{ehk}', x^*, e^*) \wedge \mathsf{ehp}' = \mathsf{eprojkg}(\mathsf{ehk}')].\end{aligned}$$

Since $\mathbf{ehk}''$ and ehk' are sampled independently, we have: $\Pr[E_0' \wedge E_1' \wedge E_2' \wedge E_3'] \leqslant \epsilon_{2u} \Pr[E_1' \wedge E_2' \wedge E_3']$ Which allows us to conclude: $\Pr[E_0 \wedge E_1 \wedge E_2 \wedge E_3] \leqslant (\epsilon_{2u} + 2\ell \cdot \epsilon_{uti}) \Pr[E_1 \wedge E_2 \wedge E_3]$, or equivalently, eH is $(\epsilon_{2u} + 2\ell \cdot \epsilon_{uti})$-vector universal. □

Comparing tightness of security reductions. We here compare the quality of security reductions obtained in [BBL17] to ours.

Consider a ring $\mathcal{R}$, an instance of a $\delta_{\mathcal{L}}$-hard subgroup membership problem $\mathcal{SM} := (\widehat{\mathcal{X}}, \mathcal{X}, \widehat{\mathcal{L}}, \mathcal{W}, \mathsf{R})$, and associated $(\mathcal{R}, f, n_f, \mathcal{M}, \mathcal{K})$ -ipfe-compatible PHF H. Further consider the resulting EPHF eH, obtained via the generic construction of [CS02] (detailed in Section 3.4.3).

As explained in the previous two lemmas, in [BBL17], to build ind-fe-cpa-secure IPFE, H must be translation indistinguishable (parametrised by ϵ_{ti}), moreover to build ind-fe-cca-secure IPFE, eH must be universal translation indistinguishable (parametrised by ϵ_{uti}) and a slight variant of H must be universal$_2$ (parametrised by ϵ_{2u}). These properties imply δ_{vs}-vector smoothness for H and δ_{vu}-vector universality for eH where $\delta_{vs} = \ell \cdot \epsilon_{ti}$, and $\delta_{vu} = 2\ell \cdot \epsilon_{uti} + \epsilon_{2u}$.

Chosen plaintext attacks. The [BBL17] proof technique bounds adversarial advantage by:

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{BBL17,\mathsf{fe\text{-}cpa}} \leqslant 2 \cdot \delta_{\mathcal{L}} + \ell \cdot |\Delta \mathcal{M}| \cdot \epsilon_{ti}$$

where $|\Delta \mathcal{M}| \leqslant (4 \cdot (\frac{n_f}{2\ell})^{1/2})^{\ell}$. From our security proof this advantage is bound by:

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\mathsf{fe\text{-}cpa}} \leqslant \delta_{\mathcal{L}} + \delta_{vu}$$

We thus gain a factor $|\Delta \mathcal{M}|$. We note that for projective hash functions where hash keys are sampled uniformly from K_{hk} this term disappears. Since such PHF's are 0-vector smooth, in this particular case the quality of our security reduction and that of [BBL17] coincide.

Chosen ciphertext attacks. The [BBL17] proof technique bounds adversarial advantage by:

$$\mathsf{Adv}^{BBL17,\mathsf{fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}} \leqslant 2\delta_{\mathcal{L}} + \ell|\Delta\mathcal{m}|(\epsilon_{ti} + 2\epsilon_{uti}) + 2q_{\mathsf{dec}}|\Delta\mathcal{m}|(\epsilon_{2u} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}),$$

where $|\Delta\mathcal{m}| \leqslant (4(\frac{n_f}{2\ell})^{1/2})^{\ell}$. From our security proof this advantage is bound by:

$$\mathsf{Adv}^{\mathsf{us,fe\text{-}cca}}_{\mathsf{FE},\mathcal{A}} \leqslant \delta_{\mathcal{L}} + \ell(\frac{q_{\mathsf{dec}}n_f}{n_f - q_{\mathsf{dec}} + 1}2\epsilon_{uti} + \epsilon_{ti}) + q_{\mathsf{dec}}(\frac{n_f}{n_f - q_{\mathsf{dec}} + 1}\epsilon_{2u} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$$

For a message space of order n_f of 128 bits, and allowing the adversary to make $q_{\mathsf{dec}} = 2^{20}$ decryption queries this yields:

$$\mathsf{Adv}^{\mathsf{fe\text{-}cca}}_{BBL17} \leqslant 2 \cdot \delta_{\mathcal{L}} + 2^{66\ell}\ell^{1-\ell/2}(\epsilon_{ti} + 2 \cdot \epsilon_{uti}) + 2^{66\ell+21}\ell^{-\ell/2}(\epsilon_{2u} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}),$$

whereas in this work:

$$\mathsf{Adv}^{\mathsf{us,fe\text{-}cpa}}_{\mathsf{FE},\mathcal{A}} < \delta_{\mathcal{L}} + \ell \cdot (2^{21}\epsilon_{uti} + \epsilon_{ti}) + 2^{20}(\epsilon_{2u} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$$

Finally for vectors of length $\ell = 100$:

$$\mathsf{Adv}^{\mathsf{fe\text{-}cca}}_{BBL17} < 2\delta_{\mathcal{L}} + 2^{6224}(\epsilon_{ti} + 2\epsilon_{uti}) + 2^{6238}(\epsilon_{2u} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}}),$$

whereas in this work:

$$\mathsf{Adv}_{\mathrm{us}} < \delta_{\mathcal{L}} + 2^{27}(\epsilon_{ti} + 2\epsilon_{uti}) + 2^{20}(\epsilon_{2u} + \delta_{\mathsf{cr}} + \delta_{\mathsf{OTS}})$$

We note that even if hashing keys are sampled uniformly, which sets $\epsilon_{ti} = \epsilon_{uti} = 0$, our security proof significantly reduces $\mathcal{A}$'s advantage (we do not have the $|\Delta\mathcal{m}|$ term), which allows us to use smaller keys, and significantly gain in efficiency. We demonstrate this in Section 4.5, by comparing their instantiation from DCR to our instantiation from HSM-CL.

B Zero Knowledge Property of the ZKPoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$

We here prove the zero-knowledge property of the protocol of Fig. 3.7, which is a ZKPoK for the following relation:

$$\mathsf{R}_{\mathsf{cl\text{-}dl}} := \{(\mathsf{hp}, (c_1, c_2), Q); (x, r) \;\;|\;\; c_1 = g_q^r \wedge c_2 = f^x\mathsf{hp}^r \wedge Q = xG\}.$$

Our analysis follows the lines of that for the Girault-Poupard-Stern statistically zero-knowledge identification scheme [GPS06], which we turn into a ZKPoK of the randomness used for encryption and of the discrete logarithm of an element of **G**, using binary challenges. Our proof is partly performed in a group of unknown order.

Theorem (Zero-Knowledge)**.** The protocol described in Fig. 3.7 is statistically zero-knowledge if ℓ is polynomial and $\ell S/A$ is negligible.

Proof. We describe an expected polynomial time simulation of the communication between a prover P and a malicious verifier V^*. The verifier V^* may use an adaptive strategy to bias the choice of the challenges to learn information about (x, r). This implies that challenges may not be randomly chosen, which must be taken into account in the security proof. If we focus on the $i-$th round of identification, V^* has already obtained data, denoted by hist, from previous interactions with P. Then P sends the commitments $t_1^{(i)}, T_2^{(i)}, t_3^{(i)}$ and $\mathcal{A}_2$ chooses – possibly using hist and $t_1^{(i)}, T_2^{(i)}, t_3^{(i)}$ – the challenge $k^{(i)}(t_1^{(i)}, T_2^{(i)}, t_3^{(i)}, \mathsf{hist})$.

Input : (r, x) and $(\mathsf{hp}, c_1, c_2, Q, P)$		Input : $(\mathsf{hp}, c_1, c_2, Q, P)$
`Repeat` ℓ `times`		
$r_1 \hookleftarrow \{0, \ldots, A-1\}$; $r_2 \hookleftarrow \mathbf{Z}/q\mathbf{Z}$		
$t_1 \leftarrow \mathsf{hp}^{r_1} f^{r_2}$; $T_2 \leftarrow r_2 P$; $t_3 \leftarrow g_q^{r_1}$	$\xrightarrow{t_1, T_2, t_3}$	
	$\xleftarrow{k}$	$k \hookleftarrow \{0, 1\}$
$u_1 \leftarrow r_1 + kr$ in $\mathbf{Z}$		
$u_2 \leftarrow r_2 + kx \bmod q$	$\xrightarrow{u_1, u_2}$	`Check` $u_1 \in \{0, \ldots, A+S\}$ $t_1 \cdot c_2^k = \mathsf{hp}^{u_1} \cdot f^{u_2}$ $T_2 + k \cdot Q = u_2 \cdot P$ $t_3 \cdot c_1^k = g_q^{u_1}$

Figure .1: The ZKPoK for $\mathsf{R}_{\mathsf{cl\text{-}dl}}$

Description of the simulator: Consider the simulator $\mathcal{S}_i$ which simulates the $i-$th round of identification as follows:

1. Choose random values $\bar{k}^{(i)} \in \{0, 1\}$, $\bar{u}_1^{(i)} \in \{S-1, \ldots, A-1\}$ and $\bar{u}_2^{(i)} \in \mathbf{Z}/q\mathbf{Z}$.
2. Compute $\bar{t_1}^{(i)} = \mathsf{hp}^{\bar{u}_1^{(i)}} f^{\bar{u}_2^{(i)}} / c_2^{\bar{k}^{(i)}}$; $\bar{T_2}^{(i)} = \bar{u}_2^{(i)} P - \bar{k}^{(i)} Q$ and $\bar{t_3}^{(i)} = g_q^{\bar{u}_1^{(i)}} / c_1^{\bar{k}^{(i)}}$.
3. If $k^{(i)}(\bar{t_1}^{(i)}, \bar{T_2}^{(i)}, \bar{t_3}^{(i)}, \mathsf{hist}) \neq \bar{k}^{(i)}$ then go to step 1, else return $(\bar{t_1}^{(i)}, \bar{T_2}^{(i)}, \bar{t_3}^{(i)}, \bar{k}^{(i)}, \bar{u}_1^{(i)}, \bar{u}_2^{(i)})$

We now show that, as long as $(S-1) \ll A$, the simulation outputs tuples which are statistically indistinguishable of those output by a real execution of the protocol. The main task is in demonstrating that the distribution of $\bar{t_1}^{(i)} = \mathsf{hp}^{\bar{u}_1^{(i)}} f^{\bar{u}_2^{(i)}} / c_2^{\bar{k}^{(i)}}$; $\bar{T_2}^{(i)} = \bar{u}_2^{(i)} P - \bar{k}^{(i)} Q$ and $\bar{t_3}^{(i)} = g_q^{\bar{u}_1^{(i)}} / c_1^{\bar{k}^{(i)}}$ as computed by $\mathcal{S}_i$ are statistically close to the distributions of $t_1 = \mathsf{hp}^{r_1} f^{r_2}$, $T_2 = r_2 P$ and $t_3 = g_q^{r_1}$ in the real execution.

Statistical indistinguishability of real and simulated triplets: To prove the distribution of simulated triplets is *statistically indistinguishable* of the distribution of triplets in a real execution we need to prove that Σ is negligible where

$$\Sigma = \sum_{\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2} \big| \Pr[(t_1, T_2, t_3, k, u_1, u_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)] \\ - \Pr[(\bar{t}_1, \bar{t}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)] \big| . \quad (.1)$$

Let $(\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)$ be a fixed tuple, we consider the probability of obtaining this tuple (1) during one round of a real execution of the protocol and (2) during a simulation.

Distribution in a real execution: We assume P is honest, therefore:

$$\Pr\Big[(t_1, T_2, t_3, k, u_1, u_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)\Big] \\ = \Pr_{r_1 \in [0, A-1]; r_2 \in \mathbf{Z}/q\mathbf{Z}} \Big[\alpha_1 = \mathsf{hp}^{r_1} f^{r_2} \wedge \alpha_2 = r_2 P \wedge \alpha_3 = g_q^{r_1} \wedge \beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})$$

$$\wedge\, \gamma_1 = r_1 + \beta \cdot r \wedge \gamma_2 = r_2 + \beta \cdot x \bmod q\Big]$$

$$= \sum_{r_1 \in [0, A-1]} \frac{1}{A} \sum_{r_2 \in \mathbf{Z}/q\mathbf{Z}} \frac{1}{q} \delta\Big(\alpha_1 = \mathsf{hp}^{\gamma_1 - \beta \cdot r} f^{\gamma_2 - \beta \cdot x} \wedge \alpha_2 = (\gamma_2 - \beta \cdot x)P \wedge \alpha_3 = g_q^{\gamma_1 - \beta \cdot r}$$

$$\wedge\, \beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist}) \wedge r_1 = \gamma_1 - \beta \cdot r \wedge r_2 = \gamma_2 - \beta \cdot x \bmod q\Big)$$

$$= \frac{1}{qA} \delta\Big(\alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta} \wedge \alpha_2 = \gamma_2 P - \beta Q \wedge \alpha_3 = g_q^{\gamma_1}/c_1^{\beta} \wedge \beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})$$

$$\wedge\, \gamma_1 - \beta \cdot r \in [0, A-1] \wedge \gamma_2 - \beta \cdot x \in \mathbf{Z}/q\mathbf{Z} \bmod q\Big)$$

$$= \frac{1}{qA} \delta(\alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta}) \delta(\alpha_2 = \gamma_2 P - \beta Q) \times \delta(\alpha_3 = g_q^{\gamma_1}/c_1^{\beta}) \times \delta(\beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist}))$$

$$\times\, \delta(\gamma_1 - \beta \cdot r \in [0, A-1]) \delta(\gamma_2 - \beta \cdot x \in \mathbf{Z}/q\mathbf{Z} \bmod q) \qquad (*)$$

Distribution in a simulated execution: We now consider the probability of obtaining the tuple $(\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)$ from the simulation described above, ie.

$$\Pr[(\bar{t}_1, \bar{t}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)].$$

This is a conditional probability given by:

$$\Pr_{\substack{\bar{u}_1 \in [S-1, A-1] \\ \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}; \bar{k} \in \{0,1\}}} \Big[\alpha_1 = \mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}} \wedge \alpha_2 = \bar{u}_2 P - \bar{k} Q \wedge \alpha_3 = g_q^{\bar{u}_1}/c_1^{\bar{k}} \wedge \beta = \bar{k}$$

$$\wedge\, \gamma_1 = \bar{u}_1 \wedge \gamma_2 = \bar{u}_2 \bmod q | \bar{k} = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})\Big].$$

Using the definition of conditional probabilities this can be written as:

$$\frac{\Pr_{\bar{u}_1 \in [S-1, A-1]; \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}; \bar{k} \in \{0,1\}} \begin{bmatrix} \alpha_1 = \mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}} \wedge \alpha_2 = \bar{u}_2 P - \bar{k} Q \wedge \alpha_3 = g_q^{\bar{u}_1}/c_1^{\bar{k}} \wedge \\ \beta = \bar{k} = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist}) \wedge \gamma_1 = \bar{u}_1 \wedge \gamma_2 = \bar{u}_2 \bmod q \end{bmatrix}}{\Pr_{\bar{u}_1 \in [S-1, A-1]; \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}; \bar{k} \in \{0,1\}} [\bar{k} = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})]} \qquad (.2)$$

We denote $Q = \sum_{\bar{u}_1 \in [S-1, A[; \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}; \bar{k} \in \{0,1\}} \delta(\bar{k} = k(\mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}}, \bar{u}_2 P - \bar{k} Q, g_q^{\bar{u}_1}/c_1^{\bar{k}}, \mathsf{hist}))$, and can thus re-write the above denominator as:

$$\frac{Q}{(A - (S-1)) \times 2 \times q}.$$

So returning to the initial probability of obtaining $(\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)$ as output from $\mathcal{S}$,

$$\Pr[(\bar{t}_1, \bar{t}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)]$$

$$= \Pr_{\substack{\bar{u}_1 \in [S-1, A-1]; \\ \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}; \bar{k} \in \{0,1\}}} \Big[\alpha_1 = \mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}} \wedge \alpha_2 = \bar{u}_2 P - \bar{k} Q \wedge \alpha_3 = g_q^{\bar{u}_1}/c_1^{\bar{k}}$$

$$\wedge\, \beta = \bar{k} = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist}) \wedge \gamma_1 = \bar{u}_1 \wedge \gamma_2 = \bar{u}_2 \bmod q\Big] \cdot \frac{(A - (S-1)) \times 2 \times q}{Q}$$

$$= \sum_{\bar{k} \in \{0,1\}} \frac{1}{2} \Pr_{\substack{\bar{u}_1 \in [S-1, A-1]; \\ \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}}} \Big[\alpha_1 = \mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}} \wedge \alpha_2 = \bar{u}_2 P - \bar{k} Q \wedge \alpha_3 = g_q^{\bar{u}_1}/c_1^{\bar{k}}$$

$$\wedge\, \beta = \bar{k} = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist}) \wedge \gamma_1 = \bar{u}_1 \wedge \gamma_2 = \bar{u}_2 \bmod q\Big] \cdot \frac{(A - (S-1)) \times 2 \times q}{Q}$$

$$
\begin{aligned}
&= \Pr_{\substack{\bar{u}_1 \in [S-1, A-1];\\ \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}}} \Big[\alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta} \wedge \alpha_2 = \gamma_2 P - \beta Q \wedge \alpha_3 = g_q^{\gamma_1}/c_1^{\beta} \\
&\qquad \wedge \beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})\Big] \cdot \frac{(A-(S-1)) \times q}{Q} \\
&= \sum_{\bar{u}_1 \in [S-1, A-1]; \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}} \frac{1}{q \times (A-(S-1))} \times \delta(\alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta} \wedge \alpha_2 = \gamma_2 P - \beta Q \wedge \alpha_3 = g_q^{\gamma_1}/c_1^{\beta} \\
&\qquad \wedge \beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})) \cdot \frac{(A-(S-1)) \times q}{Q} \\
&= \frac{1}{Q} \times \delta(\alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta}) \times \delta(\alpha_2 = \gamma_2 P - \beta Q) \times \delta(\alpha_3 = g_q^{\gamma_1}/c_1^{\beta}) \\
&\quad \times \delta(\beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})) \times \delta(\gamma_1 \in [S-1, A-1]) \times \delta(\gamma_2 \in \mathbf{Z}/q\mathbf{Z}) \qquad (**)
\end{aligned}
$$

Comparison of obtained distributions: Comparing $(*)$ and $(**)$ we can see that Q must be close to A in order for both distributions to be indistinguishable. Lemma B.1 allows us to count how many triples $(\bar{k}, \bar{u}_1, \bar{u}_2) \in \{0,1\} \times [S-1, A-1] \times [0, q-1]$ satisfy $\bar{k} = k(\mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}}, \bar{u}_2 P - \bar{k}Q, g_q^{\bar{u}_1}/c_1^{\bar{k}}, \mathsf{hist})$.

Lemma B.1. If f is a function from $G \times \mathbf{G} \times G$ to $\{0,1\}$ and $c_1 \in \{g_q^r; r \in [0, S-1]\}$, $Q \in \{x \cdot P; x \in \mathbf{Z}/p\mathbf{Z}\}$ and $x_2 = f^x \mathsf{hp}^r$ then the total number N of solutions $(k, u_1, u_2) \in \{0,1\} \times [S-1, A-1] \times [0, q-1]$ of the equation $k = \mathsf{f}(\mathsf{hp}^{u_1} f^{u_2}/c_2^k, u_2 P - kQ, g_q^{u_1}/c_1^k)$ satisfies $q(A-2)(S-1) \leqslant N \leqslant qA$.

The proof of Lemma B.1 is essentially that of [GPS06, Lemma 3]. Hence we do not provide it here. Now from Lemma B.1 it holds that $q(A - 2(S-1)) \leqslant Q \leqslant qA$. We can now bound the distance between the distribution of simulated triplets and the distribution of triplets in a real execution:

$$
\begin{aligned}
\Sigma &= \sum_{\alpha_1,\alpha_2,\alpha_3,\beta,\gamma_1,\gamma_2} \big|\Pr[(t_1, T_2, t_3, k, u_1, u_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)] \\
&\qquad - \Pr[(\bar{t}_1, \bar{t}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)]\big| \\
&= \sum_{\alpha_1,\alpha_2,\alpha_3,\beta,\gamma_2,\gamma_1 \in [S-1,A-1]} \big|\Pr[(t_1, T_2, t_3, k, u_1, u_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)] \\
&\qquad - \Pr[(\bar{t}_1, \bar{t}_2, \bar{t}_3, \bar{k}, \bar{u}_1, \bar{u}_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)]\big| \\
&\quad + \sum_{\alpha_1,\alpha_2,\alpha_3,\beta,\gamma_2,\gamma_1 \notin [S-1,A-1]} \Pr[(t_1, T_2, t_3, k, u_1, u_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)] \\
&= \sum_{\substack{\gamma_1 \in [S-1,A[, \gamma_2 \in [0,q[, \beta \in \{0,1\},\\ \alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta}, \alpha_2 = \gamma_2 P - \beta Q, \alpha_3 = g_q^{\gamma_1}/c_1^{\beta}}} \left| \begin{array}{c} 1/(qA) \times \delta(\beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})) \\ -1/Q \times \delta(\beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})) \end{array} \right| \\
&\quad + \left(1 - \sum_{\alpha_1,\alpha_2,\alpha_3,\beta,\gamma_2,\gamma_1 \in [S-1,A-1]} \Pr[(t_1, T_2, t_3, k, u_1, u_2) = (\alpha_1, \alpha_2, \alpha_3, \beta, \gamma_1, \gamma_2)]\right) \\
&= (|1/(qA) - 1/Q| \times Q) + 1 - \sum_{\substack{\gamma_1 \in [S-1,A[, \gamma_2 \in [0,q[, \beta \in \{0,1\},\\ \alpha_1 = \mathsf{hp}^{\gamma_1} f^{\gamma_2}/c_2^{\beta}, \alpha_2 = \gamma_2 P - \beta Q, \alpha_3 = g_q^{\gamma_1}/c_1^{\beta}}} \frac{1}{qA} \delta(\beta = k(\alpha_1, \alpha_2, \alpha_3, \mathsf{hist})) \\
&\qquad = \frac{|Q - qA|}{qA} + 1 - \frac{Q}{qA} \leqslant 2\frac{|Q - qA|}{qA}
\end{aligned}
$$

But from Lemma B.1 we know that $q(A-2(S-1)) \leqslant Q \leqslant qA$ so $2|Q-qA| \leqslant 4q(S-1)$ and

$$\Sigma \leqslant \frac{4(S-1)}{A} < \frac{8S}{A}.$$

This proves the real and simulated distributions are statistically indistinguishable if $2S/A$ is negligible.

Running time of the Simulator: Step 3 outputs a tuple $(\bar{t_1}^{(i)}, \bar{T_2}^{(i)}, \bar{t_3}^{(i)}, \bar{k}^{(i)}, \bar{u}_1^{(i)}, \bar{u}_2^{(i)})$ if

$$k^{(i)}(\bar{t_1}^{(i)}, \bar{T_2}^{(i)}, \bar{t_3}^{(i)}, \mathsf{hist}) = \bar{k}^{(i)}.$$

We proved earlier that

$$\Pr_{\bar{u}_1 \in [S-1, A-1]; \bar{u}_2 \in \mathbf{Z}/q\mathbf{Z}; \bar{k} \in \{0,1\}} [\bar{k} = k(\mathsf{hp}^{\bar{u}_1} f^{\bar{u}_2}/c_2^{\bar{k}}, \bar{u}_2 P - \bar{k} Q, g_q^{\bar{u}_1}/c_1^{\bar{k}}, \mathsf{hist})] = \frac{Q}{(A-(S-1)) \times 2 \times q},$$

and that $q(A-2(S-1)) \leqslant Q \leqslant qA$ so the probability of success at step 3 is bounded between

$$\frac{1}{2}\left(1 - \frac{(S-1)/A}{1-(S-1)/A}\right) \text{ and } \frac{1}{2}\left(\frac{1}{1-(S-1)/A}\right).$$

Since $2S/A$ is negligible, this quantity is essentially $1/2$, and so the expected number of iterations of the loop is 2. Therefore the complexity of the simulation of all ℓ rounds is $O(\ell)$.

Conclusion: If $\ell S/A$ is negligible and ℓ is polynomial, the protocol is statistically zero-knowledge. □

C Lindell's interactive assumption on Paillier's cryptosystem

In order to prove the security of his 2-party EC-DSA, Lindell introduced in [Lin17a] the following ad hoc interactive assumption, called Paillier-EC assumption. It is defined via the following random experiment.

```
Experiment  $\mathsf{Exp}_{\mathcal{A}}(1^\lambda)$
  $(pk, sk) \leftarrow \mathsf{Paillier.KeyGen}(1^\lambda)$
  $(\omega_0, \omega_1) \hookleftarrow \mathbf{Z}/q\mathbf{Z}$, $Q \leftarrow \omega_0 P$
  $b^\star \hookleftarrow \{0,1\}$,
  $c^\star \leftarrow \mathsf{Paillier.Enc}(1^\lambda, pk, \omega_{b^\star})$
  $b \leftarrow \mathcal{A}^{\mathcal{O}_{c^\star}(\cdot,\cdot,\cdot)}(pk, c^\star, Q)$
  if $b = b^\star$ then return 1
    else return 0
```

where $1 \leftarrow \mathcal{O}_{c^\star}(c', \alpha, \beta)$ if and only if $\mathsf{Paillier.Dec}(1^\lambda, sk, c') = \alpha + \beta\omega_{b^\star} \mod q$ and $\mathcal{O}$ stops after the first time it returns 0. The Paillier-EC assumption is hard if for every PPT adversary $\mathcal{A}$ $\Pr[\mathsf{Exp}_{\mathcal{A}}(1^\lambda) = 1] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda)$.