Functional encryption for blind external data processing

Adel Hamdi

N°d'ordre NNT : 2021LYSE1144

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée par :

l'Université Claude Bernard Lyon 1

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

Discipline : Informatique

Soutenue publiquement le 09/07/2021, par :

Adel HAMDI

Chiffrement fonctionnel pour le traitement des données externe en aveugle

Functional encryption for blind external data processing

Devant le jury composé de :

CHEVALIER Céline Maître de Conférences, Université Paris 2, CRED, Paris	**Rapporteure**
VERGNAUD Damien Professeur, Sorbonne Université, LIP6, Paris	**Rapporteur**
ABDALLA Michel Directeur de Recherche, CNRS, DIENS, Paris	**Examinateur**
GUERIN LASSOUS Isabelle Professeure, Université Lyon 1, LIP, Lyon	**Examinatrice**
LAGUILLAUMIE Fabien Professeur, Université de Montpellier, LIRMM, Montpellier	**Directeur de thèse**
CANARD Sébastien Ingénieur de recherche, Orange Labs, Caen	**Co-Directeur de thèse**

REMERCIEMENTS

La légende raconte que cette partie est le plus consultée. Fort heureusement, ce n'est qu'une légende. Je dédie ces prochaines lignes à toutes les personnes qui de près, ou même de (très) loin, ont pu contribuer à la réalisation de ce long travail de thèse. Pour ne pas faire de jaloux, je vais choisir un ordre totalement non arbitraire.

D'abord, mes remerciements vont à mes chers directeurs de thèse, Sébastien Canard et Fabien Laguillaumie. C'est un privilège d'avoir travaillé avec vous. Sébastien, j'ai eu l'honneur de découvrir le monde merveilleux du chiffrement fonctionnel à travers un stage. Merci de m'avoir accordé ta confiance pour se lancer dans cette quête de nouveaux usages. Bien évidemment, merci également Fabien pour ta patience et ton encadrement riche en questions précises et constructives. Une thèse se construit sur un temps long et on se souvient surtout des bons moments. Vos encouragements et conseils m'ont toujours permis de retrouver le chemin adéquat quand j'étais coincé. Pour une fois, ce paragraphe n'est pas assez long pour décrire ma reconnaissance.

Un grand merci à Céline Chevalier et Damien Vergnaud d'avoir accepté de rapporter ma thèse. Ce manuscrit a largement bénéficié de vos remarques et suggestions. Merci aussi à Michel Abdalla et Isabelle Guerin Lassous d'avoir accepté de compléter mon jury.

Dans cette contrée normande, au bord d'une impasse et en face d'un stade, je remercie la meilleure équipe qu'on puisse avoir pour un travail de recherche: l'équipe SPI (Security, Privacy and Innovation) d'Orange. Merci particulièrement à Jean-François qui maintient un espace incroyablement bienveillant et chaleureux.

Attends attends attends, qui fait quoi là ? Merci à toi Guillaume, collègue et doctorant dans ce même petit bureau. Merci pour ta patience avec mes idées farfelues dans la construction de schémas cryptographiques de plus en plus tordus, pour ta critique des

3-chocolatines (ou pains aux trois chocolats) de Toulorge et aussi pour tes références culturelles *incroyables*. Je remercie aussi Aïda pour ta bonne humeur et le récit de tes voyages inattendus. Aïda, Guillaume, on peut dire qu'on a eu une pandémie mondiale durant notre thèse. Les nerfs sont détendus maintenant.

Une pensée chaleureuse pour mes autres collègues: Maxime, Paul, Olivier[1], Stéphane, Takoua, Monir, Marie, Solenn, Anaïs, Loïc, Dominique, Jacques, Bastien, Donald, Quentin, Nicolas, Olivier, Yacine... J'en oublie sûrement, mais sachez que j'ai apprécié travailler et discuter de thèmes très variés à vos côtés. Enfin, une pensée particulière et nostalgique pour le regretté Yvan.

Au regard des circonstances, je n'ai pas eu l'occasion d'être souvent présent à Lyon au LIP. J'en garde néanmoins de très bons souvenirs. Je remercie le personnel administratif des différentes entités (Orange, LIP, Université Lyon 1, Université de Montpellier) de m'avoir simplifié mes démarches à distance !

Je remercie également tous les enseignants (de Bordeaux ou d'Oran) qui ont nourri ma curiosité. En particulier, Sœur Agnès et sa belle poésie égyptienne mais aussi M. Tsouria qui a déclenché indirectement ma passion pour les mathématiques.

Arrive la partie avec mes amis et proches d'ici ou d'ailleurs, que j'espère revoir plus souvent. Merci à Mourad pour ta présence indéfectible [2] et pour tous ces moments légendaires, Rafik pour nos improvisations de guitare (et la station Madeleine à 6h30), Amine et Amine pour toutes nos aventures, Azmi pour nos années à l'université, ses soirées mais pour le reste aussi[3]! Nadir 92, Sarah, Sarah et Dimitri depuis presque 10 ans maintenant, Bernard, Livia, Agathe, Abdou, Ahcene, Jean-Louis, Lucette... Ici aussi, vous me pardonnerez si j'en oublie.

Cette thèse n'aurait probablement pas pu aboutir dans de bonnes conditions sans le soutien et la présence de celles qui partagent ma vie. Amandine, merci d'avoir été patiente et encourageante dans ces moments particulièrement stressants. Je suis heureux de continuer à partager de belles nouvelles aventures avec notre nouvelle belle *xérixoutte*[4].

Enfin, je vais conclure cette épopée en remerciant tout naturellement ma famille, mes soutiens de toujours, qui m'aident à avancer sereinement dans ma vie. Ma gratitude est sans limite envers Maman et Papa qui ont tant donné à mes formidables sœurs (bien évidemment les meilleures du monde) et moi. Vous m'apportez à chaque instant le courage et la force de continuer dans mes aventures et je peux toujours compter sur vous. Cette thèse, j'espère, vous rend grâce.

[1] Je ne vais pas faire la blague sur les stagiaires

[2] *bla mangoulek*

[3] *Rak Ghaya sous-place?*

[4] Prononcer chérichoutte

RÉSUMÉ

Le besoin croissant d'externalisation des services numériques et l'augmentation notoire de la quantité d'informations disponibles se heurtent aux différentes questions de la protection des données. Il est devenu crucial de s'appuyer sur des méthodes cryptographiques suffisamment avancées pour concilier fonctionnalité et sécurité. Cette thèse s'articule autour du chiffrement fonctionnel qui permet, grâce à un chiffré et une clef dite fonctionnelle, de déléguer en externe et de manière très fine une fonctionnalité spécifique. En particulier, nous explorons plusieurs pistes et proposons différents modèles formels pour résoudre des problématiques concrètes. D'abord, nous étudions l'interactivité durant la phase de génération de clefs fonctionnelles et identifions des cas d'usages où la protection des spécificités de la fonctionnalité est pertinente. Pour ce faire, nous introduisons une notion d'aveuglement, proposons une transformation générique l'atteignant ainsi qu'une construction efficace ad-hoc pour le produit scalaire. Additionnellement, nous traitons du problème de ré-identification d'individus dans une base de données et établissons des liens entre confidentialité différentielle et sécurité du chiffrement. Partant d'une base chiffrée, nous développons une variante du chiffrement fonctionnel pour des requêtes linéaires aléatoires compatibles avec ce besoin. Notre approche permet de bénéficier en même temps de la délégation fine d'un calcul mais aussi de l'anonymisation du résultat obtenu. Finalement, nous déployons un protocole pratique respectant la vie privée de ses utilisateurs et permettant l'agrégation de données mobiles. Notre proposition modifie une variante du chiffrement fonctionnelle dans un contexte multi-utilisateurs couvrant ainsi notre cas d'usage. En outre, la technique utilisée permet de garantir une tolérance aux pannes, où nous autorisons la défaillance de certains utilisateurs pendant l'exécution du protocole.

ABSTRACT

The increasing need for externalization of digital services as well as the significant rise in the amount of available information brings various data protection issues into play. It has become crucial to rely on advanced cryptographic methods to reconcile functionality and security. This thesis focuses on functional encryption which allows, thanks to an encrypted and a so-called functional key, to externally delegate a specific functionality in a fine-grained manner. In particular, we explore several approaches and propose different formal models to solve concrete problems. First, we study interactivity during the functional key generation phase and identify use cases where protecting the specificities of the functionality is relevant. To this end, we introduce a notion of blindness, propose a generic transformation achieving it as well as an efficient ad-hoc construction for the scalar product. Additionally, we address the problem of individual re-identification in a database and establish links between differential privacy and encryption privacy. Starting from an encrypted database, we develop a variant of functional encryption for random linear queries compatible with this need. Our approach allows to benefit from the fine-grained delegation of a computation but also from the anonymisation of the obtained result. Lastly, we deploy a practical protocol that respects its users' privacy and allows aggregation of mobile data. Our proposal modifies a variant of functional encryption in a multi-user context thereby covering our use case. Moreover, the used technique allows to guarantee a fault-tolerance property, where we allow users drops during the execution of the protocol.

Contents

RÉSUMÉ LONG EN FRANÇAIS

Chaque groupe de personnes est confronté, dans sa gestion structurelle, à la question du maintien, de manière presque inévitable, de la qualité de la transmission de toute information. Avoir la certitude de la véracité de concepts généraux est fondamental pour comprendre des systèmes complexes ou réaliser des objectifs communs. Parallèlement, en réponse à des situations dont les résultats sont inconnus, minimiser et prévoir l'incertitude est un comportement naturel.
La *théorie de l'information* est un champ fondamental qui étudie certains aspects de l'incertitude. Initié par Nyquist [95], Hartley [78] et plus fondamentalement par Shannon [108], ce domaine met en exergue les méthodes mathématiques permettant de comprendre et, plus important encore, de quantifier la façon dont l'information est transmise au cours d'une communication. Un aspect essentiel est la notion d'*encodage* (et de *décodage*). Par opposition à de l'information brute, cette opération transforme un système de langage en un autre, ayant la propriété d'être adaptable efficacement aux ressources disponibles.

L'implication de cette théorie est considérable dans l'ère du développement des ordinateurs, du stockage, de la compression de données ou des télécommunications mobiles; chaque aspect de la technologie moderne puise dans les idées de la théorie de l'information.

Cryptographie ou la science du secret. Disposer de secrets est un processus qui consiste à rendre certaines informations cachées ou indisponibles pour une entité non fiable. Les anciennes méthodes étaient généralement basées sur l'intuition et la ruse : des croyances personnelles ou la *complexité* des systèmes suffisaient à convaincre les

concepteurs (ou les utilisateurs) de l'inviolabilité de leurs mécanismes. La *Cryptologie* fait référence à ces aspects généraux de conception de systèmes secrets et précède initialement l'abstraction mathématique de la théorie de l'information. Considérée d'abord comme un *art*, activement utilisée à des fins militaires, elle a retrouvé un intérêt particulier depuis l'arrivée des premiers ordinateurs à la fin de la seconde guerre mondiale en 1945. La récente conversion de plusieurs secteurs de la société au monde *numérique* propulse la cryptologie au premier plan. Aujourd'hui, les cartes à puce, les transactions commerciales avec les banques, les interactions sociales avec les médias sociaux populaires et les applications de messagerie mobile sont de parfaites illustrations de ses différents usages.

La cryptologie est en fait un terme générique qui couvre plusieurs domaines. Plus populaire, la *Cryptographie* regroupe un ensemble de méthodes visant, en fonction d'un contexte donné, à atteindre des propriétés spécifiques de sécurité. Comme notion complémentaire, la *Cryptanalyse* fait référence à l'analyse du comportement d'un système cryptographique en examinant comment un adversaire *malveillant* peut obtenir des informations non autorisées.

En substance, la sécurité d'un système est une danse perpétuelle entre les concepteurs d'un schéma cryptographique et ses attaquants. Prétendre qu'un schéma est inviolable, c'est être certain qu'il n'existe aucun chemin possible pour obtenir des informations sur le message ciblé sous-jacent. De surcroît, avoir innocemment confiance dans le concepteur du système ne fournit véritablement pas la garantie totale qu'un schéma est effectivement sûr. Une analyse des constructions historiques [58] montre que même les systèmes les plus complexes peuvent être cassés. Par conséquent, il est crucial de mettre en place une notion pertinente modélisant la finalité de tout adversaire malveillant. en définissant les protagonistes, avec leurs pouvoirs respectifs lorsqu'ils interagissent avec un système, un *modèle de sécurité* établit cette base de référence pour analyser les différentes failles potentielles. Ainsi, la sécurité d'un système est par nature un processus *évolutif* qui dépend fortement des ressources disponibles au cours d'une période donnée. En réalité, une simple percée technologique ou une avancée des connaissances en matière de recherche académique conduit à une remise en cause urgente des prétendues garanties de sécurité.

Cryptographie prouvée. Une *preuve de sécurité* en cryptographie a pour objectif de justifier formellement que le pouvoir d'un adversaire est limité lors d'une l'attaque. La *cryptographie prouvée* moderne permet donc de rassembler un ensemble de techniques pour montrer qu'un schéma est effectivement sûr. Dans la recherche actuelle, il est

d'usage de s'intéresser aux aspects *calculatoire* de sécurité. De manière informelle, les modèles mathématiques de calcul sont des idéalisations du comportement d'un ordinateur. L'idée est alors d'étudier les limites possibles d'un adversaire, en analysant ses capacités pour récupérer des informations non autorisées dans ce modèle de calcul.

Motivé par une notion mathématique de *sécurité parfaite*, Shannon [108] a introduit et étudié, dans un résultat fondateur, certains concepts de sécurité au sens de la théorie de l'information. Le désormais populaire *système de chiffrement* connu sous le nom de *masque jetable* (ou *one-time pad*) est l'exemple parfait où un chiffré (qui est l'encodage d'un message) ne contient aucune information sur le message sous-jacent, même pour un adversaire disposant de *ressources illimitées*.

Le résultat de Shannon est important puisqu'il fournit la preuve mathématique et surtout *statistique* qu'un chiffré ne révèle aucune information sur son contenu. Par ailleurs, ce résultat illustre sur la manière dont des outils de mesure de l'incertitude (la probabilité) peuvent fournir des certitudes sur la sécurité d'un schéma. Il existe cependant une limite en termes d'efficacité pour certaines usages modernes. Le chiffrement *one-time pad* sert néanmoins de base à plusieurs autres primitives cryptographiques avancées. Il est donc ainsi possible d'envoyer une information en étant convaincu que le chiffré sous-jacent ne fournit aucun indice utile sur sa nature.

Avec la naissance de la cryptographie à clef publique [47,54,69,100,103], Goldwasser et Micali [69] assouplissent la définition *parfaite* de Shannon en considérant qu'étant donné un certain chiffré, tout ce qui est *efficacement* calculable à propos du message original, est également calculable sans le chiffré. En d'autres termes, il n'est possible d'apprendre et déduire que l'information donnée par le message. Le terme *efficace* est crucial et modélise le pouvoir dont dispose un adversaire lors d'une attaque, en fonction du modèle de calcul sous-jacent. En particulier, alors que la définition de Shannon considère des adversaires non limités en ressources, Goldwasser et Micali [69] considèrent plusieurs types possibles d'adversaires suivant les situations.

La sécurité calculatoire permet une certaine efficacité et flexibilité, mais il est souvent difficile d'analyser directement les actions d'un attaquant. Un compromis intéressant consiste à faire des *hypothèses*. La technique la plus utilisée actuellement consiste à *réduire* la sécurité d'un système à celle d'une attaque sur des problèmes qui sont *supposés être difficiles*, dans un modèle déterminé de calcul.

La robustesse de la cryptographie moderne repose alors sur ces certitudes (mathématiques) et dépend fortement de notre capacité à prévoir les développements éventuels dans la résolution de ces problèmes difficiles. Comme exemple, il est courant [87] de mesurer les ressources disponibles en temps ou en espace.

D'autres propriétés de sécurité sont envisagées par les schémas contemporains au masque jetable. Parmi les plus populaires, on peut mentionner l'intégrité qui protège des altérations ou de l'authentification qui permet de vérifier la validité d'un message. La combinaison de ces propriétés de sécurité est la base même de la conception moderne de systèmes cryptographiques complexes.

Flexibilité avec le chiffrement homomorphe et fonctionnel. Les technologies telles que l'informatique en nuage (ou *cloud computing*) offrent aujourd'hui de nouvelles perspectives de travail aux utilisateurs : stockage de données, délégation de calculs à des serveurs plus puissants ou services d'apprentissage automatique. Pour la cryptographie, il est ainsi primordial d'assurer la continuité des propriétés de sécurité de base lors d'une communication avec le cloud, tout en accordant suffisamment de flexibilité pour proposer des services variés.

Le chiffrement homomorphe est une extension attrayante du chiffrement classique qui offre un algorithme supplémentaire capable de calculer sur des données chiffrés sans avoir besoin de les déchiffrer. Introduit dans les années 1970 par Rivest et al. [102], le chiffrement homomorphe a exploré de nouveaux horizons avec la percée de Gentry en 2009 [62] qui a proposé le premier résultat théorique d'un schéma *entièrement homomorphe* qui permet un système complet d'opérateurs (addition et multiplication), évaluant ainsi théoriquement toute fonction possible sur des données chiffrées.

Plus récemment, une généralisation intéressante du chiffrement classique, le *chiffrement* fonctionnel (ou FE pour *Functional Encryption*) [26,96], apparaît comme un cadre général et très prometteur offrant la flexibilité et la possibilité de conserver le contrôle des informations divulguées. Contrairement aux chiffrements classique et homomorphe, où une seul clef pour chiffrer et déchiffrer est en jeu, avec FE fournit une clef spéciale associée à une certaine fonction. Cette dernière permet avec le chiffré d'un message de révéler l'évaluation de la fonction sur ce message.

Il s'agit de l'une des formes les plus générales de chiffrement puisqu'elle permet théoriquement de déléguer n'importe quel calcul à un tiers tout en révélant le minimum d'informations sur les données sous-jacentes.

Nos contributions

Nous abordons dans la section suivante les principales notions que nous avons traitées au cours de notre thèse. Nous fournissons, pour chacune des trois contributions, un cas d'usage motivé ainsi que les problèmes de sécurité qui en découlent. De plus, nous

présentons un aperçu de la solution que nous proposons dans chaque cas.

Cette thèse expose des résultats qui ont été publiés. Nous présentons cependant deux généralisations des articles [34] (en collaboration avec Sébastien Canard et Fabien Laguillaumie) et [32] (en collaboration avec Sébastien Canard, Nicolas Desmoulins, Sébastien Hallay et Dominique Le Hello) et qui font l'objet respectivement des Chap. 3 et Chap. 5. Le chapitre 4 propose une nouvelle notion et les résultats obtenus sont en cours de soumission pour évaluation (en collaboration avec Sébastien Canard et Fabien Laguillaumie).

Un cadre général pour la protection des fonctions

Avec la croissance des activités en ligne, de multiples données (e-mails confidentiels, contrats de travail, transactions bancaires, etc.) sont transmises et stockées sur différentes plateformes externes. Basés sur ces données, une concurrence acharnée entre plusieurs acteurs se développe afin d'offrir des services particuliers, répondant ainsi positivement à une demande croissante. Par exemple, on peut s'abonner à un *service de détection de logiciels malveillants* (ou un *filtre anti-spam*) qui vise à identifier les motifs malicieux sur certains messages entrants et, au mieux, à les refuser. Dans un autre cas d'usage, une entreprise ou un centre de recherche publique spécialisés dans les algorithmes d'apprentissage automatique pourrait trouver un intérêt à obtenir des données spécifiques auprès d'un détenteur de base de données afin d'améliorer ses algorithmes. Le détenteur correspond à des individus qui possèdent des caractéristiques spécifiques liées, par exemple, aux soins de santé, ou des entreprises avec un certain type de données pour, par exemple, la détection des menaces liées à la navigation sur Intranet/Internet. En même temps, plusieurs préoccupations concernant la sécurité et la confidentialité des données manipulées apportent de nouveaux challenges à ces organisations. Le chiffrement est un instrument qui permet d'atteindre la conformité et la sécurité/protection des données exigées par les législation actuels en matière de sécurité (par exemple, RGPD [5]). Cependant, concilier la confidentialité des données et la fonctionnalité pourrait être une tâche difficile en utilisant l'approche basique *tout ou rien* des schémas de chiffrement traditionnels, où aucun calcul n'est possible, sauf en déchiffrant les données elles-mêmes, ce qui diminue la sécurité obtenue, ou rend la mise en pratique de ces systèmes lourdes en efficacité.

En conséquence, nous considérons un scénario avec une entité qui essaie d'obtenir en clair une fonction sur certaines données chiffrées. Ainsi, nous nous intéressons aux mises

[5] https://ec.europa.eu/info/law/law-topic/data-protection/

en œuvre possibles d'un chiffrement fonctionnel FE pour ces cas d'utilisation pratiques. Ainsi, une étape importante consiste généralement à un protocole interactif de génération de clefs fonctionnelles. En outre, notre sujet concerne un fournisseur de filtres anti-spam qui articule son activité autour d'une fonction sensible. Cela correspond à certaines règles de détection éventuellement protégées [33]. Par conséquent, il est pertinent et crucial de *cacher* la structure sous-jacente au propriétaire de la clef secrète principale, appelée clef maîtresse, servant à générer des clefs fonctionnelles.

Cependant, cacher la fonction à une entité peut avoir différentes significations selon le contexte. En conséquence, il est important d'explorer la possibilité de définir une notion de *aveuglement* (ou blindness) pour un protocole interactif de génération de clef fonctionnelle, où un utilisateur génère une clef fonctionnelle mais ne partage aucune information sur la fonction sous-jacente.

Nos contributions.

Dans cette thèse, nous présentons d'abord une étude systématique et générale de cette problématique en définissant la notion de chiffrement fonctionnel aveugle (ou *blind functional encryption*), qui est donc un FE avec un protocole de génération de clef fonctionnelle interactif et *aveugle*. En particulier, le propriétaire de la clef secrète maîtresse n'apprend rien sur la fonction. Nos contributions pourraient être synthétisées comme suit.

1. **Une définition générale du FE interactif et aveugle.** Nous définissons la notion de *chiffrement fonctionnel interactif* (IFE) avec une considération de sécurité adaptée et générale pour les notions classiques de *message-privacy* (ou confidentialité des messages) et de *function-privacy* (ou confidentialité des fonctions) décrites dans la littérature, dans les cas à clef privée et clef publique. La clef fonctionnelle n'est pas nécessairement conservée par le propriétaire de la fonction : dans une application de filtrage du spam, cette clef peut être installée sur un serveur de messagerie qui appartient à un client de l'éditeur du filtre anti-spam. De plus, nous définissons formellement la propriété *blindness* (ou aveuglement) qui vise à répondre positivement au scénario suivant : un propriétaire de clef maîtresse génère une clef fonctionnelle sans apprendre d'informations supplémentaires sur la fonction (en particulier le choix de l'utilisateur). Nous donnons une notion générale de *blindness* qui pourrait être trouvée dans la terminologie des signatures aveugles [81] et généralisons les constructions existantes dans le contexte de l'IBE aveugle de [30, 75]. Avec notre notion, nous concluons qu'elle est différente (et

complémentaire) de la sécurité bien connue de la *function-privacy*.

2. **Une construction générique à partir de tout FE.** Nous proposons une construction générique de IFE aveugle à partir de n'importe quel schéma de FE (non interactif). Cette construction utilise des techniques provenant d'une calcul à deux parties en deux passes [105] (MPC) et de preuves de connaissance génériques à divulgation nulle de connaissance [66].

3. **Un schéma de produit scalaire spécifique.** De nombreuses applications, telles que la fouille de données ou le calcul statistique, ont besoin comme sous-programmes d'évaluation de produit scalaire. C'est pourquoi plusieurs constructions IPFE (pour *Inner Product Functional Encryption*) ont récemment été proposées [1, 11, 38]. La plupart extraient des clefs fonctionnelles sous la même forme, et se trouve aussi être un produit scalaire. En utilisant la flexibilité du schéma de chiffrement homomorphe linéaire [37], nous concevons une construction efficace pour ce cas spécifique de FE [1, 11] en utilisant un nouveau protocole efficace de calcul de produit scalaire aveugle, à deux parties.

Cette contribution est décrite dans le chapitre 3.

Obtention de la confidentialité différentielle via FE

Considérons maintenant une entreprise détenant certaines données sensibles sur des individus et qui pourrait trouver un intérêt (commercial) à rendre ces données disponibles pour un usage externe, en visant par exemple à devenir un intermédiaire et permettre à d'autres parties d'exécuter certaines statistiques sur les individus concernés. Évidemment, la principale préoccupation est de ne pas agir au détriment de la vie privée des individus. La principale responsabilité d'une telle entreprise doit donc être de proposer des solutions qui minimisent le risque de ré-identification, c'est-à-dire la probabilité de relier chaque individu avec les données qui lui appartiennent. Les statistiques peuvent généralement être représentées comme des fonctions linéaires.

Le *chiffrement* est l'une des premières approches communes. L'idée ici est de protéger les données sensibles/personnelles *en stockage*, en recourant à des techniques de chiffrement standard, ou *en utilisation*, en se servant des primitives cryptographiques avancées telles que FHE ou MPC.

Le choix du système cryptographique avancé pertinent dépend fortement du scénario en question et en particulier de l'architecture (comment gérer les clefs) et de la fonction à exécuter sur les données chiffrées. Mais dans tous les cas, il y a trois parties principales

: (i) le chiffrement des données, (ii) le traitement sur les données chiffrées et (iii) le calcul/déchiffrement du résultat.

Une autre approche courante consiste à protéger la quantité d'informations que l'on peut finalement obtenir à partir de données sensibles/personnelles, *après leur utilisation*. Dans ce cas, une partie autorisée peut déduire certaines informations de ces données, mais la quantité et la qualité de ces résultats sont limitées par une protection technique. Ces techniques sont généralement appelées anonymisation ou pseudonymisation et reposent principalement sur des outils de base tels que le hachage par clef, le masquage, la tokenisation, le brouillage,...etc.

La *confidentialité différentielle* (ou DP pour *differential privacy* [50]) est un framework puissant et bien développé pour protéger les informations recueillies d'utilisateurs dans un certain jeux de données. Tout mécanisme DP donne des garanties par rapport à une certaine notion prédéfinie de confidentialité, qui est concrètement paramétrée par une relation de voisinage entre les entrées. La conséquence est que, pour un ensemble de données d'entrée $\mathbf{x}$, le résultat de sortie relative à $\phi(\mathbf{x})$ pour une certaine requête ϕ, est indépendant de la présence de tout individu au sein de $\mathbf{x}$. Un mécanisme différentiellement privé est généralement obtenu en construisant un algorithme randomisé (ou aléatoire). Le bruit est calibré par une certaine quantité, généralement appelée *sensibilité* [50, 94], et mesure l'impact sur la sortie lors de l'ajout ou de la soustraction d'un individu dans la base de données. Plus précisément, lorsque les données présentent une certaine structure de sortie additive, l'idée est généralement d'ajouter un certain bruit à la sortie $\phi(\mathbf{x})$ comme, par exemple, $f_\phi(\mathbf{x}) = \phi(\mathbf{x}) + e$, où e est une quantité aléatoirement bien choisie. Dans la littérature sur le DP, la valeur e est généralement générée suivant la distribution laplacienne, gaussienne ou géométrique [49, 63].

Nos contributions.

Alors que le chiffrement et la confidentialité différentielle fournissent tous deux certaines garanties de sécurité/confidentialité, il est tentant d'*entremêler* ces deux processus pour obtenir le *meilleur des deux mondes*, c'est-à-dire pour protéger les données sensibles/personnelles *durant le stockage*, *pendant* mais aussi *après leur utilisation*. L'idée principale que nous considérons est alors de :

- chiffrer d'abord les données sensibles/personnelles $\mathbf{x}$ en utilisant comme schéma un MPC/FHE/FE ;
- ensuite exécuter une fonction différentiellement privée dans le domaine chiffré ;

- si le résultat est toujours chiffré (dans les cas MPC/FHE), il doit être (peut-être conjointement dans le cas de MPC) déchiffré. Sinon (dans le cas de FE), il est directement obtenu dans le domaine clair.

Il est maintenant commun que les versions généralisées de MPC/FHE/FE sont conformes à la Turing-complets, de sorte qu'elles peuvent en théorie être utilisées pour évaluer n'importe quelle fonction (même inefficace); y compris notre fonction aléatoire f_ϕ par rapport à toute requête ϕ. Par conséquent, il est théoriquement possible de récupérer génériquement le résultat d'un calcul DP, mais avec une complexité qui rend le système irréalisable en pratique.

À travers cette thèse, nous nous concentrons sur le cas de l'utilisation de FE et nous examinons la possibilité de modéliser précisément la combinaison du FE avec les mécanismes DP. Nos contributions peuvent être résumées comme suit.

1. **Une définition du IPFE aléatoire.** Nous proposons une nouvelle extension de l'IPFE, où nous considérons que la fonctionnalité (représentée comme une fonction linéaire, ou un produit scalaire) doit donner une sortie conforme à DP. Pour cela, nous introduisons la notion de *Randomized* IPFE (ou RIPFE) *pour les mécanismes* DP. De manière globale, en utilisant un schéma de chiffrement fonctionnel, on garantit la confidentialité, et toute personne possédant une clef fonctionnelle spécifique ne peut obtenir que l'évaluation de la fonction autorisée sur les données sous-jacentes. La nouveauté est de fournir une *réponse* DP, c'est-à-dire une sortie qui est conforme aux spécificités de la DP. Pour ce faire, nous donnons un modèle de sécurité formel, inspiré de la littérature sur le FE Randomisé (ou RFE) [13,73], et nous fournissons une adaptation qui s'accorde avec la confidentialité différentielle.

2. **Un schéma spécifique pour les requêtes linéaires.** Nous proposons une solution efficace conforme à la DP pour le produit scalaire qui est basée sur une le problème difficile *Diffie-Hellman Décisionnel* dans les groupes cycliques. Notre construction combine et exploite de manière astucieuse des idées puissantes de la variante à deux entrées de l'IPFE.

3. **Une tentative de construction générique.** Enfin, nous esquissons une généralisation étant donné un quelconque IPFE en fournissant notamment une ébauche de preuve de sa sécurité.

Les résultats de cette contribution sont donnés dans le chapitre 4.

Utilisation des données mobiles préservant la vie privée

Le problème du calcul du *fonction somme* sur certaines données chiffrées est un problème bien connu et très étudié en cryptographie. On peut remonter au paradigme général du chiffrement homomorphe, où des schémas pouvaient être utilisés pour obtenir un (chiffré de la) somme de deux données quelconques à partir de la version chiffré chacune d'entre elles. Un système de vote est l'un de ces exemples où il est possible de calculer la somme de tous les choix de certains électeurs pendant une élection, d'une manière qui respecte la vie privée ainsi que les lois en vigueur.

La fonction somme trouve plusieurs applications en cryptographie, du calcul multipartite [114] à la cryptographie à seuil [46]. En fait, selon les parties impliquées, les scénarios d'attaque et les cas d'utilisation pratiques, il est possible d'obtenir plusieurs solutions (pour chacune des architectures considérées). Notre travail se concentre sur une architecture particulière motivée par un cas d'utilisation dédié et expliqué dans le paragraphe suivant.

Collecter l'utilisation des données mobiles tout en préservant la vie privée. Considérons une équipe de chercheurs en sciences sociales qui souhaitent mieux connaître les habitudes des utilisateurs de leurs téléphones. Cela correspond par exemple à étudier l'impact sur une population prédéterminée de l'utilisation, par exemple, d'applications de réseaux sociaux ou à mesurer les causes et les effets de l'addiction aux smartphones. Traditionnellement, ce type d'études est mené à sous la forme d'entretiens, de sondages ou d'enquêtes. Les participants sont invités ainsi à déclarer eux-mêmes la fréquence ou la durée de leur utilisation des applications de réseaux sociaux par le biais de questions précises : "A quelle fréquence ouvrez-vous votre application de réseaux sociaux ? Moins d'un mois ; une fois par mois ; une fois par semaine ; etc. Cette approche est répandue dans la communauté de recherche en sciences sociales. Cependant, elle souffre de plusieurs inconvénients. En effet, les réponses des participants au questionnaire peuvent être biaisées par plusieurs facteurs : la subjectivité (les personnes peuvent sous-estimer ou surestimer leur utilisation des réseaux sociaux), les limites de la mémoire humaine (il est difficile de se rappeler quand ou à quelle fréquence ils ouvrent l'application) ou la volonté (par peur du regard des autres). Par conséquent, les questions doivent être choisies et posées de manière réfléchie afin d'éviter les biais. De plus, les questionnaires et les enquêtes ne couvrent qu'un petit échantillon de la population. Tous ces défauts des outils conventionnels d'étude des sciences sociales conduisent naturellement à la mise en œuvre de moyens lourds, parfois inefficace (bonnes questions à poser), à faible

représentativité (petit échantillon) et à des résultats inexacts (réponses biaisées).

Par conséquent, afin d'obtenir des résultats précis et d'éviter les limites susmentionnées, l'équipe de chercheurs en sciences sociales peut rechercher les informations pertinentes à la source. Les données d'utilisation des téléphones portables regroupent un large ensemble d'informations. Il s'agit par exemple des horodatages d'ouverture et de fermeture des applications, des actions effectuées dans l'application (comme le bouton "j'aime"), des paramètres de configuration...etc. En complément des outils traditionnels, le fait d'avoir une vision fine à grande échelle avec des échantillons évolutifs ne pourrait que réduire les biais éventuels.

Sondage et confidentialité. Ce cas d'utilisation soulève plusieurs problèmes de confidentialité. La nature des informations disponibles doit permettre une certaine vigilance sur le traitement final. Le point de départ évident est de donner aux individus le choix d'appliquer leur consentement à la collecte et au traitement de leurs données pour une analyse statistique. Dans le cas où le consentement est donné, les informations collectées étant sensibles à la vie privée, elles doivent être protégées au regard de la législation en vigueur.

Par ailleurs, un système fournissant un tel service devrait inclure des mécanismes de protection de la vie privée, en particulier conformément au RGPD, le consentement est une première condition essentielle pour collecter et traiter les données des utilisateurs (articles 6, 7 et 8 du RGPD). Ainsi, avant chaque nouvelle étude, il est demandé aux utilisateurs de donner leur consentement (éclairé et explicite) sur la collecte et le traitement de leurs données. En outre, pour respecter le principe de la protection de la vie privée dès la conception, exigé par le RGPD (articles 25 et 32), le présent cas d'usage doit tirer parti de solutions techniques pour la protection des données. L'objectif principal de ces mécanismes techniques est de réduire le risque de ré-identification des données d'un individu.

Pour le spécialiste en sciences sociales, cette situation pose un dilemme. L'algorithme utilisé doit à la fois assurer la confidentialité des données et permettre de dériver les statistiques ci-dessus sur les données agrégées. Par conséquent, l'une des exigences en matière de confidentialité est que la possibilité d'obtenir les données d'un seul utilisateur doit être infaisable. De même, il doit être impossible de calculer les opérations sur un ensemble de données provenant d'un seul utilisateur, sinon, cet utilisateur sera directement ré-identifié. Cela implique que les opérations d'analyses statistiques ne doivent pas être effectuées avant l'agrégation des données provenant de plusieurs sources. Enfin, comme le consentement est donné pour une étude ou un traitement de données, effectuer

d'autres analyses sur les données qui ont été collectées à d'autres fins doit naturellement être irréalisable.

Une autre considération importante est qu'une personne peut prévoir de participer mais décider ensuite de ne pas le faire (ou simplement échoue). Ceci est tout à fait naturel et ne devrait pas poser de problème tant au niveau du résultat final que de la vie privée de cet individu (et de celle des autres).

Nos contributions.

À la lumière de la discussion ci-dessus, notre idée est alors de fournir un nouveau service indépendant, que nous avons appelé *WeStat*, pour l'analyse préservant la vie privée des données d'utilisation des téléphones mobiles auprès des utilisateurs enregistrés.

Nous résumons les principales contributions dans ce qui suit.

1. **Un protocole général pour WeStat.** Nous proposons une architecture interactive générale basée sur trois entités : *utilisateurs*, un *agrégateur* et un *demandeur* d'une étude. L'objectif de ces interactions est d'obtenir des statistiques d'utilisation mobile en préservant la vie privée. Pour cela, nous décrivons un protocole cryptographique ainsi qu'un modèle de sécurité. Pour ce faire, nous définissons les sécurités du demandeur et de l'agrégateur dans le but de modéliser les attaques possibles sur la vie privée de l'utilisateur pour chaque entité impliquée. Nous demandons en outre de restreindre le nombre d'interactions entre tous les acteurs. En effet, nous considérons tout d'abord qu'une solution basée sur la participation active des utilisateurs n'est pas assez pérenne, en se basant sur le fait supplémentaire que ces derniers ne se connaissent pas. Les principales opérations doivent donc être gérées par un agrégateur uniquement. De plus, nous préférons également, dans les différentes étapes, diminuer le rôle du *demandeur*. En particulier, nous pouvons considérer que ce dernier effectue une requête pour une étude au début du protocole, et ne souhaite pas forcément être très actif. Le demandeur ne doit donc participer, de manière non interactive, qu'à (i) la création de l'étude et (ii) au calcul des statistiques.

2. **Une construction générale basée sur une variante de FE.** Nous concevons un protocole WeStat en exploitant une construction bien connue d'une variante multi-client de FE. En particulier, nous partons du schéma FE multi-client, décentralisé, dynamique de Chotard et al. [42] pour le calcul de la fonction somme, mais qui ne correspond pas exactement à notre architecture. Nous montrons comment

une adaptation de ce dernier avec une modification d'une idée d'*arbre binaire* de Chan et al. [39], peuvent être utilisées ensemble pour réaliser notre solution qui est résistante même si des utilisateurs ne participent pas (ou échouent).

Dans ce contexte, l'équipe de chercheurs en sciences sociales contacte (en jouant le rôle d'un demandeur dans notre système) l'agrégateur avec une demande d'analyse spécifique. Elle propose des statistiques (comptage, durée moyenne, régression linéaire...) sur une ou un ensemble d'applications utilisées pendant une période donnée. Les demandes d'analyse spécifient les attributs sur lesquels ils souhaitent que les opérations d'analyse soient effectuées (par exemple, l'âge, le jour de la semaine, le mois, la durée d'utilisation, etc. L'agrégateur effectue alors l'analyse demandée sur les données agrégées de l'individu collectées pendant la période d'observation spécifiée et renvoie les résultats de l'analyse à l'équipe de recherche. Ayant accès à ces résultats, l'équipe de recherche peut en tirer des conclusions sociales sur l'étude. Notre solution WeStat consiste à utiliser des techniques de chiffrement avec FE directement du côté de l'individu. Du point de vue du RGPD, cela correspond à une technique de pseudonymisation, à la différence qu'en cas de compromission, le responsable du traitement (au sens RGPD) n'a pas à informer la personne concernée, mais seulement l'agrégateur.

Nous renvoyons au chapitre 5 pour une présentation de ces résultats.

Autre contribution

L'objectif de ce manuscrit est de mettre en évidence les différentes utilisations de FE pour résoudre des problèmes cryptographiques pratiques. Au cours de cette thèse, nous avons aussi participé à l'élaboration et à la promotion d'un *jeu éducatif* appelé CYBERCRYPT [15]. Nous ne donnerons pas tous les détails mais le but premier est de présenter (par le biais d'ateliers) les principales techniques cryptographiques de manière ludique et récréative. Nous exposons par exemple le chiffré de César et sa cryptanalyse, la machine Enigma, en plus des considérations avancées modernes comme le chiffrement à clef publique ou la signature. Un article contenant les détails techniques a été rédigé à cet effet [15] et a été accepté pour le volet exposition de la conférence HistoCrypt 2019 [104].

Organisation de cette thèse

Les prochaines parties de ce manuscrit sont présentées comme suit. Le chapitre 2 est consacré au rappel du cadre cryptographique existant, y compris les définitions de sécu-

rité pour les primitives utilisées tout au long de cette thèse. Ensuite, dans le chapitre 3, nous donnons notre framework pour la protection des fonctions. Le chapitre 4 fournit notre solution reliant la confidentialité différentielle au chiffrement fonctionnel. Enfin, au chapitre 5, nous exposons notre solution *WeStat* pour l'agrégation de l'utilisation des données mobiles, avant de conclure au chapitre 6.

CHAPTER 1

INTRODUCTION

1.1 General context

Every group of persons is faced in its organizational management with the question of maintaining, in an almost evident manner, the communication or the transmission qualities of any information. Having certainty about the truthiness of general concepts is fundamental in understanding complex systems or realizing common goals. At the same time, minimizing and predicting uncertainty, is a natural behaviour as a response to situations with unknown outcomes.

Information theory is a fundamental field studying some aspects of uncertainty. Initiated by Nyquist [95], Hartley [78] and more fundamentally abstracted by Shannon [108], this area sheds light on the mathematical methods to understand and, may be more importantly, quantify how information is transmitted during a communication. An critical aspect is the notion of *encoding* (and *decoding*). As opposed to raw information, this operation transforms one system of language into another one with the property of being efficiently usable for the available resources. The implication of such theory is considerable in the modern era with the development of computers, data storage, data compression or mobile telecommunication; every aspect of modern technology draws on ideas from information theory.

Cryptography or the science of secret. Having secrets is a general phenomenon of making some information hidden or unavailable to untrusted entity. Early methods were generally based on intuition and guile: personal beliefs or complicated constructions were

good enough to convince the builders (or users) that these mechanisms are inviolable. *Cryptology* refers to these general aspects of making secret schemes and initially predates the mathematical abstraction of information theory. Considered first as an *art*, actively used for military purposes, it has regained a particular interest since the arrival of the first computers at the end of world war in 1945. The recent conversion of different sectors of the society into the *digital* world brings cryptology to the forefront. Today, smartcards, commerce transactions with banking, social interactions with popular social media and mobile message applications are examples of cryptology's usages.

Cryptology is an umbrella that covers different fields. *Cryptography* is the most popular one and regroups a set of general methods that aims to reach specific secrecy properties, depending on a given context. As a complementary notion, *Cryptanalysis* refers to the analysis of a cryptographic system behaviour's by analysing how a *malicious* adversary could obtain unauthorized information.

In essence, the security of a system is a perpetual dance between the designer of a cryptographic construction and adversaries attacking it. Claiming that a scheme is *unbreakable*, is being certain that there is no possible path for obtaining information about some targeted underlying message. Moreover, innocently having trust into the designer of the system does not provide substantially the full guarantee that a scheme is indeed secure. Historical analysis of old constructions [58] shows that even the more complex schemes could be broken. In fact, it is crucial to put forward a meaningful notion of what it is intended by any malicious adversary. A *security model* fixes a baseline for analysing the different potential flaws of a system, by defining the protagonists with their respective powers when interacting with it. Thus, security of a system is by essence an *evolutive* process that highly depends on the available resource during a time period. In fact, a simple technology breakthrough or an advanced research knowledge could lead to a catastrophic questioning about the claimed security properties.

Modern provable cryptography. A *security proof* in cryptography has the purpose to give formal evidence that the power of an adversary is limited during the attack. This is the aim of *provable cryptography* which regroups a set of techniques to prove that a scheme is indeed secure. In modern cryptographic research, it is common to rely on the *computational security* aspects of a scheme. Informally, mathematical models of computation are idealizations of computer's behaviour. Hence, the idea is to analyse the possible limitations of an adversary in this computational model when attacking the scheme, by leveraging its capabilities to retrieve unauthorized information.

Motivated by a mathematical notion of *perfect security*, Shannon [108] introduced

and studied in a popular and seminal work some concepts of secrecy in the information theoretical sense. The now popular *encryption scheme* known as *one-time pad* is the perfect example where a ciphertext (which is the encryption of a message) contains no information about the underlying message, even for an adversary with *unlimited* resources. Shannon's result is powerful since it provides *mathematical* and *statistical* evidence that a ciphertext does not reveal anything about its content. As a side, this is an example of how uncertainty (or probability) measure tools could be used to provide certainty about the security of scheme. There is some caveat in term of efficiency for some modern usages. However, the one-time pad serves as a basis for several modern advanced cryptographic primitives. It is possible to send an information with the conviction that the underlying encoded message does not provide any useful hints about its nature.

With the birth of public-key cryptography [47, 54, 69, 100, 103], Goldwasser and Micali [69] relaxes the *perfect* definition of Shannon by considering that given some ciphertext, whatever is efficiently computable about the original message, is also *efficiently* computable without the ciphertext. In other words, we can only learn, from any ciphertext, the inherent leakage given from the message. The term *efficient* is crucial and models the power that an adversary has during an attack, depending on the underlying computational model. In particular, while Shannon's definition considers unbounded adversaries, Goldwasser and Micali [69] consider several types of adversaries.

Computational security allows some flexibility and efficiency, but it is often difficult to directly analyse the actions of an attacker. An interesting compromise is to make *assumptions*. The most currently used way is to *reduce* security to that of attacking problems which are *supposed to be difficult*, given a determined model of computation.

The solidity of modern cryptography relies then on these (mathematical) beliefs and heavily depends on our ability to predict hypothetical developments in solving these difficult problems. For example, it is classical [87] to measure the available resources (for e.g. time or space memory).

Other security properties are considered by modern schemes. As an example, integrity protects from alterations, or authentication permits to verify the validity of a message. Combining several security properties is the basics for the design of complex cryptographic systems.

Flexibility with Homomorphic and Functional Encryptions. Technologies such as cloud computing offer new working perspectives for remote users today: data storage, delegation of computation to more powerful servers or machine-learning services. For cryptography, it is crucial to ensure the continuity of bringing the basic security prop-

erties during communication with the cloud while giving enough flexibility to propose various services.

Homomorphic encryption is an interesting extension of encryption which allows an extra algorithm that could compute over the underlying encrypted data without the need to decrypt it. Introduced in the 1970s by Rivest et al. [102], homomorphic encryption has explored new horizon with the Gentry's breakthrough in 2009 [62] that proposed the first theoretical result of a *Fully Homomorphic Encryption* scheme that allows a complete system of operators (addition and multiplication), thus evaluating any possible function over some encrypted data.

More recently, an interesting generalization, *Functional Encryption* (or FE) [26, 96], arises as a general and very promising framework giving the flexibility and the possibility to retain control of leaked information. Unlike traditional encryption or homomorphic encryption, where there is one involved key that serves to encrypt and decrypt, FE provides a special key associated to some function that reveals the evaluation of the function over the data, given its encryption.

This is one of the most general form of encryption since it permits theoretically to delegate any computation for a third party while revealing the minimum information about the underlying data.

1.2 Our contributions

We discuss in the following section the main notions that we have treated during our thesis. We provide, for each of the three contributions, a motivated use case as well as the security issues that it brings. Moreover, we present an overview of our proposed solution.

This thesis presents results that have been published. Nevertheless, we present two generalizations of our articles [34] (with Sébastien Canard and Fabien Laguillaumie) and [32] (with Sébastien Canard, Nicolas Desmoulins, Sébastien Hallay and Dominique Le Hello) which are covered in Chap. 3 and Chap. 5 respectively. The chapter 4 proposes a new notion and the obtained results are currently under submission for evaluation (with Sébastien Canard and Fabien Laguillaumie).

1.2.1 A General Framework for Function's Protection

With the growth of online activities, multiple data (confidential emails, employment contracts, bank transactions, etc.) are transmitted and stored over different external

platforms. A ruthless competition between several actors is ongoing in order to offer particular services, based on those data, thus answering positively to an increasing demand. For example, one could subscribe to a *malware detection service* (or a *spam filter*) that aims to identify bad patterns over some incoming messages and, at best, to reject them. In a different use case, a company or an institution specialized in machine learning algorithms could find interest to obtain some specific data from a data owner to improve its algorithms: individuals with specific characteristics related to e.g., healthcare, or companies with some specific kind of data for e.g., threats detection related to Intranet/Internet browsing. At the same time, several concerns about the security and privacy of manipulated data bring new challenges to those organizations. Encryption mechanism is one enabler to achieve the compliance and data security/privacy that is required in today's security interest (for e.g. GDPR [1]). However, conciliate data confidentiality and functionality could be a hard task by using basic *all-or-nothing* approach of traditional encryption schemes, where no computation are possible, except by decrypting the data itself, then decreasing the obtained security. From a higher perspective, we consider a scenario with an entity that tries to get in clear a function over some encrypted data.

Our first scenario covers possible implementations of FE for practical use-cases using a functional interactive key generation protocol. Amongst other concerns, we give interest to a situation where a spam filter provider builds its activity around a sensitive function. This corresponds to some possibly protected detection rules [33]. Hence, it is relevant and crucial to *hide* the underlying structure to the master secret key owner. Hiding the function to an entity can have different meanings depending on the context.

As a consequence, it is important to explore the possibility of defining a notion of *blindness* for an interactive functional key generation protocol, where a user generates a functional key but does not get any information on the underlying function.

Our Contributions.

In this thesis, we first present a systematic and general study of this problematic by defining the notion of *blind functional encryption*, which is a FE with a *blind* interactive functional key generation protocol. In particular, the master secret key owner learns nothing about the function. Our contributions could be resumed as follows.

1. **A general definition of blind interactive FE.** We define the notion of *interactive functional encryption* (IFE) with an adapted and general security consideration

[1] https://ec.europa.eu/info/law/law-topic/data-protection/

for the classical notions of message-privacy and function-privacy presented in the literature in both the private and public key settings. The functional key is not necessarily kept by the owner of the function: in a spam filtering application, this key can be installed on a mail server which belongs to a customer of the spam filter editor. Moreover, we formally define the *blindness* property that aims to positively answer to the following scenario: a master key owner generates a functional key without learning any additional information on the function(in particular user's choice). We give a general notion of *blindness* that could be found in the terminology of blind signature [81] and generalize known constructions in the context of blind IBE of [30, 75]. With our notion, we conclude that it is different (and complementary) from the well known function-privacy security.

2. **A generic construction from any FE.** We propose a generic construction of blind IFE starting from any (non-interactive) FE scheme. This construction uses techniques from a two-move private function evaluation [105] and generic zero-knowledge proofs of knowledge [66].

3. **A specific inner-product scheme.** Many applications, such as data mining or statistical computation need as subroutines inner-product evaluation. That is why several IPFE constructions have recently been proposed [1, 11, 38]. Most of known schemes extract functional keys of the same shape which is also an inner product. Using the flexibility of linearly homomorphic encryption scheme [37], we design an efficient construction for the specific case of the Inner Product FE (or IPFE) [1, 11] by using a new efficient blind two-party inner product protocol which could be of independent interest.

This contribution is described in Chapter 3.

1.2.2 Obtaining Differential-Privacy via FE

Let us now consider a company holding some sensitive data about individuals and which could find (commercial) interest in making such data available for an external use, aiming for example to become a data broker and allowing other parties to execute some statistics about the involved individuals. Obviously, the main concern is to not proceed in the detriment of individual's privacy. As a main responsibility for such company must therefore to propose solutions that minimize the risk of re-identification, i.e. the probability of linking each individual with its belonging data. The statistics could generally be represented as linear functions.

Encryption is one of the first common approach. The idea is here to protect the sensitive/personal data *in storage*, using standard encryption techniques, or *in use*, using advanced cryptographic primitives such as FHE or MPC. The choice of the relevant advanced cryptographic scheme strongly depends on the studied practical scenario and in particular on the architecture (how to manage the keys) and the function to be executed on encrypted data. But in all cases, there are three main parts: (i) data encryption, (ii) treatment over encrypted data, and (iii) result computation/decryption.

Another common approach is to protect the amount of information one can finally obtain from sensitive/personal data, *after its use*. In this case, an authorized party can infer some information from those data, but the quantity and quality of such output is limited by a technical protection. Those techniques are usually referred as anonymization or pseudonymization, and are mainly based on some basic tools such as keyed-hashing, masking, tokenization, blurring, etc.

Differential Privacy (or DP) [50] is now a powerful and well-developed framework for protecting user's recording on some datasets. Any DP mechanism gives guarantees with respect to some predefined notion of privacy, which is concretely parametrized by a neighbouring relation between inputs. The consequence is that, for an input dataset $\mathbf{x}$, the output result related to $\phi(\mathbf{x})$ for some query ϕ, is independent from the presence of any individual within $\mathbf{x}$. A differentially private mechanism is generally obtained by constructing a randomized algorithm whose noise is calibrated by some quantity, usually called *sensitivity* [50, 94], measuring the impact on the output when adding or subtracting an individual in the database. More precisely, where the data has some additive output structure, the idea is generally to add some noise to the output $\phi(\mathbf{x})$ as, for example, $f_\phi(\mathbf{x}) = \phi(\mathbf{x}) + e$, where e is some well-chosen random variable. In the DP literature, the value e is typically drawn from the Laplacian, Gaussian or the Geometric distribution [49, 63].

Our contributions.

While both encryption and differential-privacy provides some security/privacy guarantees, it is tempting to *intertwine* both these process to obtain the *best of both worlds*, i.e. to protect the sensitive/personal data *in storage*, *in use*, and *after its use*. The principal idea we consider is then to:

- first encrypt the sensitive/personal data $\mathbf{x}$ using a MPC/FHE/FE scheme;
- then execute differentially-private function in the encrypted domain;

- if the result is still encrypted (in the MPC/FHE cases), it should be (perhaps jointly in the case of MPC) decrypted. Otherwise (in the FE case), it is directly obtained in the plain domain.

It is now well-known that the generic versions of MPC/FHE/FE are Turing compliant, so that they can in theory be used to evaluate any function (even inefficiently), including our randomized f_ϕ function w.r.t. any query ϕ. Hence, it is theoretically possible to generically recover the result of a differential-private computation, but with a complexity that makes it unrealistic in practice.

Through this thesis, we focus on the case of using FE, as it is relevant in many practical scenarii, and ask the possibility to precisely model the combination of encryption with DP mechanisms. Our contributions could be resumed as follows.

1. **A definition of Randomized IPFE.** We propose a new extension of IPFE, where we consider that the functionality (represented as a linear function) should give a DP-compliant output. For this, we introduce the notion of *Randomized* IPFE (RIPFE) *for* DP *mechanisms*. At a high level, using a functional encryption scheme, data are protected thanks to encryption, and anyone having a specific functional decryption key could only obtain the evaluation of the authorized function over the underlying data, given its encryption. The novelty is to provide a *differential private answer*, i.e an output which is DP-compliant. Moreover, we give a formal security model, inspired from the Randomized FE (or RFE) literature [13, 73], and we provide an adaptation that fits with differential privacy.

2. **A specific scheme for linear queries.** We propose an efficient DP-compliant solution for the inner-product which is based on a the Decisional Diffie-Hellman hardness assumption in cyclic groups. Our construction combines in a clever way and exploits powerful ideas from the two-input IPFE, which is a variant of IPFE.

3. **An attempt for a generic construction.** Finally, we sketch a generalization for our basic solution given any IPFE and we provide evidence about its security.

The results of this contribution are given in Chapter 4.

1.2.3 Privacy-Preserving Mobile Data Usage

The problem of computing the *sum function* over some encrypted data is a well known and old hugely studied problem in cryptography. We could trace back to the general homomorphic encryption paradigm, where schemes could be used to obtain an (encryption

of the) sum of any two data starting from the underlying encrypted version of each one of them. As we will see in our thesis, this primitive allows one to design many practical cryptographic primitives. A voting system is one of this example where it is possible to compute the sum of all some voters' choice during an election, in a privacy-persevering manner.

The basic sum functionality founds several applications in cryptography, from multi-party computation [114] to threshold cryptography [46]. In fact, depending on the involved parties, the attack scenarios and the practical use cases, it is possible to obtain several ways (for each of the considered architecture). Our work focus on a particular architecture motivated by a dedicated use-case and explained as follows.

Collecting mobile data usage while preserving privacy. Consider a team of social scientists that are interested in having a better knowledge of users habits using their phones. This corresponds for example to study the impact on, a predetermined population, the usage of e.g., social network applications or measure the causes and effects of smartphone addiction. Traditionally, this type of studies is conducted through tools such as face-to-face interviews or surveys. Participants would be asked to self-report the frequency or the duration of their use of social networks apps via questions such as "How frequently do you open your social networks application? Less than a month; once a month; once a week; etc.". This approach is spread in the social science research community. However, it suffers from several drawbacks. Indeed, participants' answers to the questionnaire may be biased by several factors: subjectivity (people may underestimate or overestimate their use of social networks), human memory limitations (it is hard to remember when or how often they open the application) or willingness (for fear of other's eyes). Hence, questions must be thoughtfully chosen and asked in order to avoid biases. Additionally, questionnaires and surveys only cover a small sample of the population. All these flaws of conventional tools for social science studies lead to ineffective implementation (right questions to ask), poor representativeness (small sample) and inaccurate results (biased answers).

Therefore, in order to obtain accurate results and avoid the aforementioned limitations, the team of social scientists must search for the relevant information at source. Mobile usage data regroups a large set of information. That consists for example of timestamps of application opening and closing, actions performed in the application (such as "like" button), configuration parameters...etc. As a complement to social scientist's traditional tools, having a fine-grained insights at large scale with scalable samples could only reduce the eventual biases.

Privacy issues This use case raises several privacy concerns. The nature of available information needs to provide a degree of vigilance on the final treatment. The obvious starting point is to provide individuals the choice to give their consent to the collection and processing of their data for a specific study, hence a specific statistical analysis. In case consent is given, as collected information is privacy-sensitive, they should be protected with regard to the current legislation.

As a side, a system providing such service should include mechanisms for privacy protection, in particular in accordance with the GDPR, Consent is a first key condition to collect and process users' data (Articles 6, 7 and 8 of the GDPR). Therefore, before each new study, users are requested to give their (informed and explicit) consent on the collection and processing of their data.. Furthermore, to fulfil the principle of privacy-by-design requested by the GDPR (Articles 25 and 32), the present use case should leverage technical solutions for data protection. The main purpose for those technical mechanisms is to reduce the risk of re-identification of any individual's data.

For the social scientist this poses a dilemma. The used algorithm should both provide data confidentiality and allow to derive the above statistics on the aggregated data. Similarly, it must be impossible to compute the analytics operations on a dataset originating from a single user; otherwise, this user will straightforwardly be re-identified. This implies that analytics must not be performed before multi-source data are aggregated. Finally, as consent is given for one study or one data processing, it must be infeasible to perform any other analytics on the data that have been collected for other purposes.

Another important issue is that an individual may plan to participate but then decide not (or fail) to. This is quite natural and should not pose any problem regarding both the final result and the privacy of this individual (and the one of the others).

Our contributions.

In the light of the above discussion, our idea is then to provide a new independent service, that we have called *WeStat*, for privacy-preserving analytics on mobile usage data collected from registered users.

We resume the main contributions in the following.

1. **A general protocol for WeStat.** We propose a general interactive architecture based on three entities: *users*, an *aggregator* and a *requestor*. The aim of these interactions is to perform data mobile usage in a privacy-preserving manner. For this, we describe a cryptographic protocol as well as a security model. Namely, we

define requestor's and aggregator's securities with the aim to model the possible attacks on user's privacy for each involved entity. We moreover request to restrict the number of interactions between all actors. Indeed, we first consider that a solution based on the active participation of individuals is not enough sustainable, based on the additional fact that individuals do not know each other: the main operations should be managed by an Aggregator solely. Furthermore, we also prefer, in the different steps, to lower the role the *Requestor*, asking for a study. In particular, we can consider that the latter requests a service from the former, and does not necessarily want to be very active. The Requestor should then only participate, in a non-interactive way, in (i) the creation of the study and (ii) the computation of the final statistics.

2. **A general construction based on a variant of FE.** We design a WeStat protocol using a well known construction of a multi-client variant of FE. In particular, we exploit the dynamic decentralized multi-client of FE of Chotard et al. [42] for computing the sum function which does not fit exactly with our architecture. We show how an adaptation of this scheme, as well as a modification of a *binary tree* idea of Chan et al. [39] can be used together to achieve our fault-tolerant solution.

In this context, the team of social scientists contacts (playing the role of a Requestor in our system) the Aggregator with a specific analytics request. It will request some statistics (counting, average duration, linear regression...) on one or a set of apps are used for a given period of time. The analytics requests specify the attributes on which they would like the analytics operations to be performed (for instance, age, day of the week, month, duration of use etc.) as well as the operation to be performed on the data. The Aggregator then performs the requested analytics on the individual's aggregated data collected during the specified observation period and sends back the analytics results to the research team. Having access to the result, the research team can derive social conclusions about the study. Our WeStat solution is to make use of encryption techniques directly on the individual's side. For the GDPR's point of view, this corresponds to a pseudonymization technique, with the difference that in case of private data compromise, the data controller does not have to inform the data subject, but only the authority.

We refer to Chapter 5 for a presentation of these results.

1.2.4 Other contribution

The focus of this manuscript is to highlight the different uses of FE for resolving practical cryptographic problems. During this thesis, we participate in the elaboration and promotion of an *educational game* called CYBERCRYPT [15]. We will not give the full details but the primary goal is to present (through workshops) the main cryptographic techniques in a playful and recreational manner. We exhibit for example the basic Caesar cipher and its cryptanalysis, the Enigma machine, in addition to the modern advanced considerations such as public key encryption or signature. An article with the technical details has been written for this purpose [15] and has been accepted for the exposition track of the HistoCrypt 2019 conference [104].

1.3 Organization of this thesis

The next parts of this manuscript are presented as follows. Chapter 2 is dedicated to the relevant cryptographic background, including the security definitions for the used primitives through this thesis. Then, in Chapter 3, we give our framework for function's protection. Chapter 4 provides our solution connecting differential privacy with functional encryption. Finally, in Chapter 5, we exhibit our *WeStat* solution for mobile data usage aggregation, before concluding in Chapter 6.

CHAPTER 2

PRELIMINARIES

In this chapter we collect the necessary background that will be useful in the rest of this thesis. The first section is devoted to fix the terminology as well as the mathematical background. Most part of the next section is dedicated to present the main cryptographic building blocks. Finally, we expose the central tool of our thesis, Functional Encryption and describe the main security properties.

2.1 Mathematical background

Basic notations. For integers $n, m \in \mathbb{N}$ with $n \leq m \in \mathbb{N}$, let $[n, m]$ be the ordered set $\{n, n+1, \ldots, m\}$. If S is a finite set, we denote by $|S|$ its cardinal. The Cartesian product of two sets A and B, denoted by $A \times B$, is the set of all pairs (a, b) where $a \in A$ and $b \in B$. Similarly, for $\ell > 0$ be a natural integer, we define the set $A^\ell := \underbrace{A \times \cdots \times A}_{\ell \text{ times}}$ where each element in $\mathbf{a} \in A^\ell$ is a vector $\mathbf{a} := (a_1, \ldots, a_n)$ with $a_i \in A$ and are written in bold font. We denote $f : A \to B$ to be a function over the set A with co-domain B. For $\mathbf{a} \in A^\ell$, we define the element $f(\mathbf{a}) \in B^\ell$ as the vector satisfying $f(\mathbf{a}) := (f(a_1), \ldots, f(a_\ell))$.

Let $\mathbb{G}$ be a *group*. The *order* of a (finite) group $\mathbb{G}$ is its cardinal $|\mathbb{G}|$. A group $\mathbb{G}$ is cyclic if there exists an element called *generator* g such that every element $x \in \mathbb{G}$ can be written as g^k for a certain integer $k \in \mathbb{Z}$. In particular, cyclic groups are Abelian. We use in our thesis *prime order* groups which are cyclic groups where the order is a prime number.

Probability background. We use the basic terminology of probability theory to describe a random phenomenon in terms of its sample space and the occurrence frequency of events. A discrete random variable X is a function defined over some *randomness* set, generally omitted, and has output some finite set S. The *probability mass function* is a function that assigns a real number in $(0, 1)$ to each possible outcome. Given a discrete random variable X, for all possible $s \in S$ in the output space, it is denoted by $\Pr[X = s]$ and verify $\sum_{s \in S} \Pr[X = s] = 1$. We say sometimes that X is *distributed* according to f if its probability mass function is equal to some function f. The (discrete) uniform distribution over a set S is defined such that $\Pr[X = s] = \frac{1}{|S|}$. The notation $s \leftarrow S$ (resp. over $D(S)$ for a distribution D) means that we sample an element s from S according to the uniform distribution over S (resp. to distribution D over S). We see in this thesis further distributions that will be useful for our results.

Complexity of an algorithm. Evaluating the difficulty of solving a problem using algorithms depends on the computational model one has to consider. As traditionally done in the cryptography literature, We use special *Probabilistic Polynomial Time* (denoted hereafter PPT) Turing Machines (TM) that has the specificity to make some probabilistic choices dictated by a fixed probability distribution. In general, the terminology *random coins* refers to the randomness used during the computation. We talk about PPT *algorithm* referring to the underlying TM machine. Unless specified, we write the PPT algorithms in calligraphic font and in addition for any PPT algorithm $\mathcal{A}$, we write $\mathcal{A}(x)$ to indicate that $\mathcal{A}$ takes some input x. The notation $z \leftarrow \mathcal{A}$ stands for an element which is sampled from the output space of $\mathcal{A}$ according to the distribution defined over the random coins of $\mathcal{A}$. The notation $\mathcal{A}^{\mathcal{B}}$ stands for an *oracle call*, which says that $\mathcal{A}$ has an input/output access to an algorithm $\mathcal{B}$.

A particular class of algorithms important for us when considering differential privacy is a special case of PPT Turing machines called *randomized algorithms*. In order to explain the difference, We detail first how one can view algorithms that makes some probabilistic choices. There is in fact two ways [66] of viewing them.

1. The first possibility is to consider that the algorithm is making random "toss coins" and its output is seen as a random variable. A natural way to manage such random output is to consider the probability $\Pr[\mathcal{A}(\mathbf{x}) = z]$ that an algorithm will give a value z on input $\mathbf{x}$. If we denote by $\mathcal{A}_r(\mathbf{x})$ the output of $\mathcal{A}$ on input $\mathbf{x}$ when r is the outcome of the internal coin tosses, then the probability $\Pr[\mathcal{A}(\mathbf{x}) = z]$ is simply the fraction of r for which $\mathcal{A}_r(\mathbf{x}) = z$, divided by the number of toss coins

made by $\mathcal{A}$. This corresponds to the theory [51] that is usually considered for a differential private mechanism $\mathcal{A}$ since it mainly studies the class of functions f that *implements* $\mathcal{A}$

2. The second one makes use of some *auxiliary input* whose purpose is to manage the underlying randomness. In this case, a randomized algorithm $\mathcal{A}$ is viewed as a *deterministic* function with two inputs: the formal one $\mathbf{x}$, and the randomness r. In this case, the algorithm is denoted by $\mathcal{A}(\mathbf{x}; r)$. In order to evaluate the distribution of the output $\mathcal{A}(\mathbf{x}; r)$, the natural way is to consider uniformly distributed elements r over some randomness space R. This makes it particularly well-suited when considering pseudo-random functions (defined over R).

Randomized algorithms have the particularity to always output a correct answer with some significant probability (say $\geq \frac{1}{2}$). When the answer is not correct, then randomized algorithm always detect it with probability 1. In fact, these algorithms *are* in fact PPT algorithms but PPT are more general in the sense that they accept *non-determinism* in the definition. We do not discuss these concepts in more details but we refer for e.g. to [91] for an exhaustive definition.

Therefore, our own formulation of randomized algorithms uses the first approach in the sequel, considering internal coin tosses and forgets how randomness spaces are generated. More formally, we specify by adapting existing definition of randomized algorithm's notion which is based on the one of *probability simplex*, denoted $\Delta(\mathcal{Y})$ and defined over a discrete set $\mathcal{Y}$ as:

$$\Delta(\mathcal{Y}) := \{p \in \mathbb{R}^{|\mathcal{Y}|} : p_i \geq 0 \text{ for all } i \text{ and } \sum_{i=1}^{|\mathcal{Y}|} p_i = 1\}.$$

Using this probability simplex, we can consider the following definition of randomized algorithm issued from [51].

Definition 2.1.1 (Randomized Algorithm [51])**.** *A* ***randomized algorithm*** $\mathcal{A}$ *defined over a (discrete) domain* $\mathcal{X}$ *and a range* $\mathcal{Y}$ *is associated with a mapping* $p_{\mathcal{A}} : \mathcal{X} \to \Delta(\mathcal{Y})$ *with the following property:*

- *on input* $\mathbf{x} \in \mathcal{X}$*, the algorithm* $\mathcal{A}$ *outputs* $y \in \mathcal{Y}$ *such that if* $(P_{\mathcal{A}}(\mathbf{x})) := (p_1, \ldots, p_{|\mathcal{Y}|})$*, then* $\Pr[\mathcal{A}(\mathbf{x}) = y] := p_y$.

The probability space here is taken over coin tosses of the algorithm $\mathcal{A}$.

We finish this paragraph by providing a notion that will be useful for our consideration in Chapter 3: *feasible entropy distribution.*

Definition 2.1.2 (Feasible entropy distribution)**.** *Let $W_f(\cdot)$ be a universal functional oracle defined as $W_f(x) := f(x)$, let F_λ denote a family of functions, $D(F_\lambda)$ any distribution over F_λ and $U(F_\lambda)$ the uniform distribution over F_λ. We say that D is a feasible entropy distribution, if for all non-uniform polynomial time algorithm $\mathcal{A}$, it holds that:*

$$\left|\mathsf{Pr}_{f\leftarrow D(F_\lambda)}\left[\mathcal{A}^{W_f(\cdot)}(\lambda)=1\right]-\mathsf{Pr}_{f\leftarrow U(F_\lambda)}\left[\mathcal{A}^{W_f(\cdot)}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda).$$

Instead of using the uniform distribution, we can replace D and U by two distributions D_0 and D_1. We can define then the notion of *pair of ensembles of feasible entropy distribution.* An interested reader can refer to [80] for a discussion about this notion, the question of efficient samplable distributions and links to indistinguishability obfuscation [17, 60].

Circuits. Another important model of computation in complexity theory is the class of *circuits* which are more powerful than Turing Machines [14][1]. $\forall n, m \in \mathbb{N}$, a *circuit* C is a directed acyclic graph. It contains n nodes with no incoming edges; called the *input nodes* and m nodes with no outgoing edges, called the *output nodes.* All other nodes are called *gates* and are labelled with boolean operation OR, AND and NOT. The OR and AND nodes have fanin of 2 and the NOT node has fanin 1. Two important measures of complexity are considered in this model. The *size of* C, denoted by $\mathsf{size}(C)$, is the number of nodes in C and the *depth* of C is the length of the longest path in the graph from an input node to the output node. The OR, AND and NOT gates form a *universal basis*, i.e. every function from $\{0,1\}^n$ to $\{0,1\}^m$ can be implemented by a boolean circuit using the AND, OR, NOT gates.

We call *description* of a circuit C: the underlying graph, gates, size and depth of the circuit. We implicitly suppose that there is always a way to encode this information as a vector of binary strings. For every string $u \in \{0,1\}^n$, we denote by $C(u)$ the output of the circuit C on input u. We consider a *poly-sized log-depth* family of circuits, which is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of circuits where each C_n has n inputs, m outputs and $\mathsf{size}(C)$ is at most polynomial with logarithmic depth.

For $n \in \mathbb{N}$, we associate to a function $f : \{0,1\}^* \to \{0,1\}^{l(n)}$ with a family of circuits denoted $\{f_n\}_{n\in\mathbb{N}}$, where f_n represents the restriction of f to a n-bit input, i.e.

[1] Circuits can be considered as *Turing Machines with advices* that has some auxiliary inputs in addition to the classical input.

$f_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ for a certain length-output $l(n)$ function depending only of the input-length n. We represent functions f in some set F by a poly-sized family of circuits $\{f_n\}_{n\in\mathbb{N}}$ and we sometimes use the abuse notation f instead of f_n.

2.2 Cryptographic building blocks

In this section, we review some known techniques that will be useful for our consideration.

2.2.1 Security foundations

A convenient tool when proving security is to use a series of *experiments* or *games* [110]. Informally, it is an interaction between an adversary attacking the system and some *oracles* or *challengers*. The adversary is asked to find a way or a strategy to win the game by having access to a certain amount of information provided by a challenger. In general, measuring this strategy uses probability theory by analysing the possible outcome of the adversary while running the experiments.

When the general analysis of an experiment is complex, another classical technique [66] is to consider transitions and modifications between two (or more) experiments. These methods produce several intermediate games that help to analyse the complexity of the security proof. More importantly, for each step, when a modification occurs in the experiment, one has to measure the potential information that an adversary can obtain during these transitions.

We mention finally the notion of transitivity in a reduction. Suppose that a scheme A is secure and a scheme B is reduced to scheme A. Then scheme B inherits from the underlying security for scheme A. This property serves as a basis for all the constructions that we consider here. For a formal definition of the *Game-Based* approach, we refer to [110]).

Hard problems. The main cryptographic primitives are build around supposed hardness of certain problems in complexity theory.

We consider in particular the **DDH assumption**. In a cyclic group $\mathbb{G}$ of prime order q with generator g, the Decisional Diffie-Hellman Problem (DDH) in $\mathbb{G}$ considers an adversary (called distinguisher) that has the capability of discerning the following two distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), with a, b, c picked uniformly and independently at random in $\mathbb{Z}_q^*$. The DDH assumption in some group $\mathbb{G}$ is the intractability of the DDH problem for any PPT distinguisher.

Another problem that we consider is the general **Hard Subgroup Membership assumption** (HSM). For a finite Abelian group $\mathbb{G}$, consider $\mathbb{G}'$ as a subgroup of $\mathbb{G}$. The HSM assumption states that it is hard to distinguish the elements of $\mathbb{G}'$ from $\mathbb{G}$ (We refer to [112] for a more precise statement).

2.2.2 Cryptographic building blocks

We provide in this following the main cryptographic blocks.

Two-party computation. Some of our protocols are inspired by general techniques from multiparty computation [67, Chap. 7]. In particular, we consider the *two-party case*. There exists some protocol P that can compute a *functionality* between two parties on a pair of inputs. In more details, for some function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$ with $f_i : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ for $i \in \{1,2\}$, there exists a protocol P such that for $x_1, x_2 \in \{0,1\}^n$, the output-pair is a random variable $(f_1(x_1,x_2), f_2(x_1,x_2))$. The first party (which has input x_1) wishes to obtain $f_1(x_1,x_2)$ and the second party (with input x_2) wishes to obtain $f_2(x_1,x_2)$.

The view of the i-th party ($i \in \{1,2\}$) during an execution on (x_1,x_2) is denoted by $\texttt{View}_i(x_1,x_2)$ and is equal to $(x_i, r^i; m_1^i, \ldots, m_t^i)$, where r^i equals the contents of the i-th party's internal random coins, and the m_j^i's represent the j-th message that it received.

Finally, the *output* of the i-th party, denoted by $\texttt{Output}_i(x_1,x_2)$, contains the elements that can be computed from its own view of the execution.

Security is given by *simulation*. In more details, the idea is to consider the capability of generating $\texttt{View}_i(x_1,x_2)$ only from the inputs and the outputs of each party. To formalize this, PPT simulators $\mathcal{S}_1, \mathcal{S}_2$ must exist such that $\mathcal{S}_i(x_1,x_2,f_i(x_1,x_2),f(x_1,x_2))$ has to be indistinguishable from $\texttt{View}_i(x_1,x_2)$ for each $i = 1,2$.

We refer to [67, Chap.7] for an exhaustive presentation of this notion.

Zero-Knowledge Proofs of Knowledge (ZKPoK). They are mechanisms to ensure that some computations are done correctly. In particular, such techniques consider *language* $\mathcal{L}$ containing elements x (named input) and w (named witness) that verify $\mathcal{R}_\mathcal{L}(x,w) = 1$ for some given *relation* $\mathcal{R}_\mathcal{L}$.

Definition 2.2.1. *An n-round interactive (perfect, statistical, computational) Zero Knowledge Proof of Knowledge* (ZKPoK) $(\mathcal{P},\mathcal{V})$ *between a* PPT *prover* $\mathcal{P}$ *and a* PPT *verifier* $\mathcal{V}$*, for a language* $\mathcal{L}$ *with relation* $R_\mathcal{L}$ *is any pair of algorithms such that* $\mathcal{V}$ *and the following conditions hold:*

- *Completeness.* $(\mathcal{P}, \mathcal{V})$ *is complete, if for any* $x \in \mathcal{L}$ *with a membership witness* w*:*

$$\Pr[(\mathcal{P}(w), \mathcal{V})(x) = 1] \geq 2/3;$$

- *Knowledge Extraction.* $(\mathcal{P}, \mathcal{V})$ *is knowledge-extractable with knowledge error* κ*, if there exists an* PPT *algorithm* Ext *and a polynomial* p *such that for any input* x*, for any prover* $\mathcal{P}^*$*, the oracle algorithm* $\mathsf{Ext}^{\mathcal{P}^*}$ *(which is* Ext *that has access to* $\mathcal{P}^*$*) runs in expected polynomial time and satisfies*

$$\Pr[w' \leftarrow \mathsf{Ext}^{\mathcal{P}^*}(\cdot) : R_{\mathcal{L}}(x, w') = 1] \geq \frac{\epsilon - \kappa}{p(|x|)},$$

  where ϵ *denotes the probability that* $\mathcal{V}$ *accepts when interacting with* $\mathcal{P}^*$ *on common input* x*;*

- *Zero-Knowledge.* $(\mathcal{P}, \mathcal{V})$ *is (perfectly, statistically, computationally) zero-knowledge, if for every* PPT $\mathcal{V}^*$ *there exists a probabilistic simulator* $\mathcal{S}im$ *running in expected polynomial time such that for every* $x \in \mathcal{L}$*,*

$$\mathtt{View}^{\mathcal{P}}_{\mathcal{V}^*}(x) \approx \mathcal{S}im(x),$$

  where $\mathtt{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ *consists of the internal random tape of* $\mathcal{V}^*$ *together with the sequence of all messages he received from* $\mathcal{P}$ *on input a witness* w *such that* $\mathcal{R}(x, w) = 1$*.*

We use the notation $\mathsf{ZKPoK}\{(x, w) : \mathcal{R}_{\mathcal{L}}(x, w) = 1\}$ to specify that there exist a prover that uses a witness w and a verifier that use x (with $\mathcal{R}_{\mathcal{L}}(x, w) = 1$), executing a ZKPoK protocol as in the description of the above definition.

Public key Encryption with homomorphism. Public key encryption is one of the fundamental primitive in cryptography. Over a message space M it is given by the following definition.

Definition 2.2.2. *Let* $\lambda \in \mathbb{N}$*. A tuple* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *is an encryption scheme described as follows.*

- $\mathsf{Setup}(1^\lambda)$ *is a* PPT *algorithm which takes as input a security parameter* 1^λ *and outputs a secret key* sk *and a public key* pk*.*

- $\mathsf{Enc}(\mathsf{pk}, m)$ *is a* PPT *algorithm which takes as input a public key* pk *and a message* $m \in M$ *and returns a ciphertext* c*.*

- $\mathsf{Dec}(\mathsf{sk}, c)$ *is a* PPT *algorithm which takes as input a secret key* sk*, a ciphertext* c *and outputs a string* z*.*

For correctness, we require that for all $m \in M$*, given* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c) = m\right] \geq 1 - \mathsf{negl}(\lambda).$$

To model security, the adversary has access to an *oracle* $\mathsf{Enc}_b(\mathsf{pk}, \cdot, \cdot)$ such that for any bit $b \in \{0, 1\}$, it takes as inputs x_0 and x_1 and returns $\mathsf{Enc}(\mathsf{pk}, x_b)$. More formally, The *indistinguishability under chosen-plaintext attack* (hereafter $\mathsf{IND} - \mathsf{CPA}$) consists of the following experiment.

Definition 2.2.3. *Let* $b \in \{0, 1\}$*. An encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* M *is* $\mathsf{IND} - \mathsf{CPA}$ *if for any* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left|\Pr\left[\mathsf{Exp}^{(0)}_{\mathcal{A},\mathsf{IND}-\mathsf{CPA}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{(1)}_{\mathcal{A},\mathsf{IND}-\mathsf{CPA}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Exp}^{(b)}_{\mathcal{A},\mathsf{IND}-\mathsf{CPA}}(\lambda)$ *is defined as*

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $b' \leftarrow \mathcal{A}^{\mathsf{Enc}_b(\mathsf{pk}, \cdot, \cdot)}(1^\lambda, \mathsf{pk})$
3. *output* $b' = b$*.*

The quantity

$$\left|\Pr\left[\mathsf{Exp}^{(0)}_{\mathcal{A},\mathsf{IND}-\mathsf{CPA}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{(1)}_{\mathcal{A},\mathsf{IND}-\mathsf{CPA}}(\lambda) = 1\right]\right|,$$

is called the advantage of $\mathcal{A}$ *and is denoted* $\mathsf{Adv}_{\mathcal{A},\mathsf{IND}-\mathsf{CPA}}(1^\lambda)$*.*

Next, we discuss the notion of FHE (for Fully Homomorphic Encryption) which is informally an encryption mechanism that has the capability of evaluating any function over some encrypted data. The definition of a FHE is given as follows and is adapted from [62].

Definition 2.2.4 (Fully Homomorphic Encryption)**.** *Let* $\mathcal{C} = \{C_\lambda\}$ *be a set of circuits. A tuple of algorithms* $\mathsf{FHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is a homomorphic encryption scheme with respect to* $\mathcal{C}$ *if* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a public key encryption as in Def. 2.2.2 with the following* Eval *algorithm running for all* $C \in \mathcal{C}$*, all* m *ciphertexts* $c_{m_1}, \ldots, c_{m_\ell}$ *encrypting messages* $m_1, \ldots, m_\ell$ *such that*

- $\mathsf{Eval}(\mathsf{pk}, f, c_{m_1}, \ldots, c_{m_l})$ *is a* PPT *algorithm which provide an outputs a ciphertext* c' *that is distributed as a random encryption* $\mathsf{Enc}(f(m_1, \cdots, m_l), \mathsf{pk})$.

We say that a scheme is fully homomorphic if

1. *for every polynomial* $p = p(\lambda)$ *it is homomorphic with respect to the family* $\mathcal{C} = \{C_\lambda\}$ *where* C_λ *is the set of all circuits of size at most* p.

2. *the running time (and the output size) of both the encryption and decryption algorithm is also polynomial in* λ. *If this condition is not satisfied, We say that it is size-dependent.*

The fully homomorphic encryption scheme of Gentry given in [62] realize all the properties described above. Other schemes exists based on various hardness assumptions [29,40,56,113]. If we consider *size-dependent* construction, a possible instantiation under the DDH assumption is given in [18].

Linearly homomorphic scheme. A special class of circuits that we use in the next chapters is the class of linear functions sometimes referred to *linearly homomorphic encryption* [37]. A FHE can compute linear functions, but we can mention the following schemes that support this specific class of linear functions [37, 45, 69, 98]. For our construction in Chapter 3, we use the following Castagnos-Laguillaumie linearly homomorphic scheme (hereafter CL) and presented in [37]. The construction is highlighted in the following paragraph.

CL encryption scheme. The main idea of the CL scheme is to consider a framework that consists of the description of *a DDH group* $\mathbb{G}$ *with an easy discrete logarithm subgroup* in $\mathbb{G}$. In addition, it requires a special hardness membership subgroup assumption which states that it is hard to distinguish between elements of a subgroup from $\mathbb{G}$. This scheme (and its variants) provides $\mathsf{IND}-\mathsf{CPA}$ security, based on various assumption depending on the desired efficiency. The original scheme [37] is based on DDH over the group $\mathbb{G}$ while in [38, 112], it relies on the hard subgroup membership assumption. We provide a high-level description based on the second construction.

The general parameters for the scheme is given by a tuple $(p, \tilde{s}, \mathfrak{g}, \mathfrak{f}, \mathfrak{g}_p, \mathbb{G}, F, G^p)$ where the set $(\mathbb{G}, \cdot)$ is a cyclic group of order ps, for an unknown integer s, p is a prime number such that $\gcd(p, s) = 1$. The only known information on s is an upper bound $\tilde{s}$ of s. The set $G^p = \{\mathfrak{g}^p, \mathfrak{g} \in G\}$ is the subgroup of (unknown) order s of $\mathbb{G}$, and F is

the subgroup of order p of $\mathbb{G}$, so that $\mathbb{G} = F \times G^p$. The elements $\mathfrak{f}$, $\mathfrak{g}_p$ and $\mathfrak{g} = \mathfrak{f} \cdot \mathfrak{g}_p$ are respective generators of F, G^p and $\mathbb{G}$.

The discrete logarithm problem is easy in F, which means that there exists a deterministic polynomial time algorithm Solve that efficiently solves the discrete logarithm problem in F. The message space of CL is $\mathbb{Z}_p$. We refer the reader to [37, 38, 112] for a more precise description and some generalization of this scheme. Informally, it is given by the following algorithms.

The secret key sk is an integer $x \leftarrow \{0, \ldots, \tilde{s}p-1\}$ and the public key is $\mathsf{pk} = \mathfrak{g}_p^x \in G^p$. The encryption procedure returns a ciphertext $c_m = (c_1, c_2)$ where $c_1 \leftarrow \mathfrak{g}_p^r \in G^p$ and $c_2 \leftarrow \mathfrak{f}^m \mathsf{pk}^r \in \mathbb{G}$ for a random r and a message $m \in \mathbb{Z}_p$. The decryption algorithm first computes $M \leftarrow c_2/c_1^x \in F$ and returns m using the Solve algorithm on $M(= \mathfrak{f}^m)$. It is not difficult to see that this scheme is linearly homomorphic.

Non Interactive Key exchange (NIKE). In the following, we describe chameleon hashing and NIKE as primitives needed for our protocol in Chapter 5. Informally, NIKE permits to non interactively share a common secret key between two parties. Moreover, it serves as a basis for the DSum functional encryption presented in Sec. 2.3.4. Hence, we implicitly assume the obtained underlying security properties of both NIKE and chameleon hashing and will only provide instantiations of this primitives on groups.

- **DL Based Chameleon Hashing** The starting point to build NIKE is a certain type of hashing. In a nutshell, a chameleon hashing corresponds to a collision-resistant algebraic hash function with a trapdoor for finding collisions. In their paper [86], Krawczyk and Rabin have introduced such concept (together with the one of chameleon signature schemes) and have proposed several constructions. We here focus on the Discrete Logarithm (DL) based one.

  At a high-level, this DL-based construction considers p and q to be prime numbers such that $p = kq+1$ for some positive integer k. Let g of order q in $\mathbb{Z}_p^*$. The secret key for the chameleon hashing consists of a trapdoor $\mathsf{ck} \in \mathbb{Z}_q^*$ and the corresponding hashing public key is $\mathsf{hk} = g^{\mathsf{ck}} \pmod p$. Given a message $m \in \mathbb{Z}_q^*$, the hashing procedure first chooses at random $r \in \mathbb{Z}_q^*$ and computes $h = g^m \mathsf{hk}^r \pmod p$. To find collision, the secret key ck is used. Indeed, given message m', a value r' such that $g^m \mathsf{hk}^r = g^{m'} \mathsf{hk}^{r'} \pmod p$, it is possible having ck to solve the equation $m + \mathsf{ck} \cdot r = m' + \mathsf{ck} \cdot r' \pmod q$.

  We will use chameleon hashing for our instantiation of the NIKE primitive (see the next paragraph) from the protocol described in Chapter 5.

We focus now on the NIKE given and studied by Freire et al. [57]. It corresponds to a public-key cryptographic primitive which enables two parties to agree on a symmetric shared key without requiring any interaction. Each party owns a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and is able to compute a shared key by using her private key sk_1 (resp. sk_2) and the public key pk_2 (resp. pk_1) of the other party.

For our main construction, we make use of the pairing-based construction, still given in [57]. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ be a bilinear environment, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order prime p, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ and e is a bilinear map. For a more formal definition of bilinear environment, we refer to [59] for an exhaustive presentation.
In order to build NIKE following Freire et al. [57], we take $u, u_1, u_2 \in \mathbb{G}_1^*$ and let hk, ck be as described above for the chameleon hash function. All those values compose the parameters of the NIKE scheme. In fact, we note that in the construction of Freire et al. [57] that we consider, the collision secret key ck is not use explicitly in the construction but rather to prove the security of the scheme. See [57] for details.

We describe next the primitive. To generate a common key, a key generation phase is executed by each party $i = 1, 2$ that consists of choosing at random $x_i \in \mathbb{Z}_p$ and $r_i \in \mathbb{Z}_q^\star$ (as in the chameleon hash function above), then computing $Z_i = g_2^{x_i}$, $t = g^{\mathcal{H}_2(Z_i)}\mathsf{hk}^{r_i} \pmod p$ (this is a chameleon hash), $Y_i = u_0 u_1^{t_i} u_2^{t_i^2}$ and $X_i = Y_i^{x_i}$. The public key pk_i is then (X_i, Z_i, r_i) and the private key sk_i is x_i.

Finally, the computation of the shared key $K_{1,2} = K_{2,1}$ is done as follows: using a public key pk_1 and a private key sk_2, one computes $t_1 = g^{\mathcal{H}_2(Z_1)}\mathsf{hk}^{r_1} \pmod p$ and generates the key $K_{1,2} = e(S^{x_2}, Z_1)$ if and only if $e(X_1, g_2) = e(u_0 u_1^{t_1} u_2^{t_1^2}, Z_1)$.

In fact, this scheme is a modification of the initial description given in [57]. We have first removed the identity part and we have then replaced Z by $\mathcal{H}_2(Z)$ during the computation of t, where $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ is a hash function. This is due to the fact that when considering a practical instance for the NIKE scheme, using the chameleon hash, Z should be in $\mathbb{G}_2$ while the value to be chameleon hashed should be in $\mathbb{Z}_q^*$, as explained in [86]. In addition, the construction given below is secure (as defined in [57]) under the Decisional *Bilinear* Diffie Hellman [59], which is a variant of DDH in bilinear groups (for type 2 pairings).

2.3 Functional Encryption

In this section, we discuss our principal tool for this thesis: *functional encryption.*

The general concept of public key encryption permits to recover, thanks to *some* secret key associated to a public key, the original message being encrypted, and nothing

more. The main novelty behind Functional Encryption (or FE) [26,96] is to give a way to generate *several* keys, called *functional keys*, and attached to some known functions f, that are used to recover related information about the original message. In fact, it is possible to recover the evaluation of the function f over the underlying message. Classical encryption is a special case since there is only *one* such functional key (the classical secret key) corresponding to the identity function, then permitting to obtain $f(m) := m$ for any message m.

Having such general notion allows naturally to obtain many existing primitives in cryptography, by considering several classes of functions [1,16,23,26,71,74,83]. Predicate Encryption [71,83] is an example where a message can be decrypted if and only if the evaluation of some predicate P over the message has a True value.

As FE is general, the classical concepts of security has to be adapted and the $\mathsf{IND}-\mathsf{CPA}$ of classical encryption is not sufficient. For example, the adversary can learn many functional keys for a simple function or many functional keys for possibly different functions.

We recall in the following the definition of Functional Encryption taken from [26]. We fix an arbitrary set of functions F. In addition, we fix a *message* space M, where each $m \in M \subseteq \{0,1\}^*$ is represented by a string input of any $f \in F$.

Public key Functional Encryption. A public key Functional Encryption is defined as follows. [26]

Definition 2.3.1. *Let $\lambda \in \mathbb{N}$. A functional encryption scheme for a set of functions F consists of a tuple* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows.*

- $\mathsf{Setup}(1^\lambda)$ *is a* PPT *algorithm which takes as input a security parameter 1^λ, and outputs a master secret key* msk *and a master public key* mpk.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$ *is a* PPT *algorithm which takes as input a master secret key* msk, *a description of the function $f \in F$ and outputs a functional key sk_f.*
- $\mathsf{Enc}(\mathsf{mpk}, m)$ *is a* PPT *algorithm which takes as input the master public key* mpk *and a message $m \in M$, and returns a ciphertext c.*
- $\mathsf{Dec}(\mathsf{mpk}, sk_f, c)$ *is a* PPT *algorithm which takes as input a master public key* mpk, *a functional key sk_f and a ciphertext c and outputs a string z.*

For correctness, we require that for all $f \in F$ and $m \in M$, given keys $(\mathsf{mpk}, \mathsf{msk}) \leftarrow$

$\mathsf{Setup}(1^\lambda)$, $sk_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ *and* $c \leftarrow \mathsf{Enc}(\mathsf{mpk}, m)$, *we have*

$$\Pr\Big[\mathsf{Dec}(\mathsf{mpk}, sk_f, c) = f(m)\Big] \geq 1 - \mathsf{negl}(\lambda).$$

Private key FE. The above definition can easily be adapted to the *private key* setting where encryption is only possible for the entity knowing msk. In this case, the Setup algorithm outputs msk and some public parameters *params*; the KeyGen algorithm takes as inputs *params*; the encryption algorithm Enc uses the master secret key msk in order to encrypt the message; and Dec algorithm takes as inputs *params* instead of mpk. Note that if it is not specified, *params* will always be known during the invocation of the above algorithms.

2.3.1 Security Definitions for Functional Encryption

We provide in this section the main security properties that an FE should provide.

A note on the terminology. The classical terminology used for security of FE is to consider Indistinguishability under Chosen Plaintext Attack, or $\mathsf{IND-CPA}$. In this thesis, we call it *message-privacy* (MP) since we also deal with a related notion of *function-privacy* for FE. Such choice permits to avoid confusion or heavy notation.

Message-privacy for FE

The basic security consideration for FE is related to the standard notion of message-privacy security with different functional keys [26,70]. As it is usually done, we consider the adaptive form of message-privacy with multiple messages and multiple functional keys.

We specify first some oracles. The adversary has access to a $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle which extracts a functional key when the adversary requests it for a chosen input function f. For any bit $b \in \{0,1\}$, we define $\mathsf{Enc}_b(\mathsf{mpk}, \cdot, \cdot)$ to be an oracle which takes as inputs x_0 and x_1 and returns $\mathsf{Enc}(\mathsf{mpk}, x_b)$. More oracles will be defined all along this manuscript when needed.

Definition 2.3.2 (Message-privacy for FE)**.** *Let $b \in \{0,1\}$. We say that a public key FE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over a message space M and a function space F is* ***message-private*** *(MP) if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that the following difference of two probabilities, called the advantage of $\mathcal{A}$*

and denoted $\mathsf{Adv}_{\mathcal{A},\mathsf{MP}}(1^\lambda)$*, verifies*

$$\left|\Pr\left[\mathsf{Exp}^0_{\mathcal{A},\mathsf{MP}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^1_{\mathcal{A},\mathsf{MP}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda),$$

and the experiment $\mathsf{Exp}^b_{\mathcal{A},\mathsf{MP}}(\lambda)$ *is defined as*

1. $(\mathsf{mpk},\mathsf{msk})\leftarrow\mathsf{Setup}(1^\lambda)$
2. $b'\leftarrow\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{Enc}_b(\mathsf{mpk},\cdot,\cdot)}(1^\lambda,\mathsf{mpk})$
3. *output* $b'=b$

The output b' *depends on some conditions. We require that for all* $f\in F$ *and* (m_0,m_1) *coming from* $\mathcal{A}$*'s calls to the oracles* KeyGen *and* Enc_b *respectively, if* $f(m_0)\neq f(m_1)$ *then set* b' *to be a random bit.*

From such definition, an adversary could ask as many messages as she wants. In the public key setting, it is not difficult to see that many-MP security is equivalent to one-MP security using standard *hybrid* [66, 69] arguments. We then suppose that the adversary is making one single request to the encryption oracle.

In the *private key* setting, the definition is the same (replacing $\mathsf{Enc}_b(\mathsf{mpk},\cdot,\cdot)$ with $\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)$). Moreover, as the adversary cannot naturally encrypt messages of its choice, we additionally provide the oracle $\mathsf{Enc}(\mathsf{msk},\cdot)$ which encrypts messages m of adversary's choice, with the inherent condition that $f(m)=f(m_0)=f(m_1)$ for all requested f (otherwise it could trivially win the game). It is the analogue of the *find-then-guess* security which can be shown to be equivalent to our notion [22].

Function-privacy for FE. Several other security properties have been considered for FE in the literature and we do not review all of them. However, We consider in the sequel the notion of function-privacy which informally states that a functional key sk_f does not give any additional information about the underlying function f, except from what is given by the evaluations over some data being encrypted [8, 24, 28, 84]. More details will be given in Chapter 3, as it is close to our new notion of blindness.

Inner-product FE (IPFE). The particular case of inner-product for FE, or IPFE [1, 11, 38, 112], has been extensively studied as it is one basic functionality for which we can provide very efficient constructions.

The way to define such specific functionality can be done in the following way. The input elements are represented as $\mathbf{x}\in\{0,\ldots,B_{\mathbf{x}}\}^\ell$ and functions are given as

$\mathbf{y} \in \{0, \ldots, B_{\mathbf{y}}\}^{\ell}$ for some integer bounds $B_{\mathbf{x}}, B_{\mathbf{y}}$. In particular, for each input $\mathbf{x} = (x_1, \ldots, x_\ell)$, a *linear* (or inner-product) function is given by

$$\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{i=1}^{\ell} x_i \cdot y_i \in \mathbb{Z}$$

for a certain vector $\mathbf{y} = (y_1, \ldots, y_\ell)$.

An IPFE scheme is an FE scheme for the class of function $F := \{\langle \cdot, \mathbf{y} \rangle, \mathbf{y} \in \{0, \ldots, B_{\mathbf{y}}\}\}$. This important primitive can be instantiated from several assumptions [1,11,38] and we consider it in Chapter 3 and Chapter 4.

2.3.2 DSum Multi-client Functional Encryption

Traditionally, FE considers a single user setting, where there is one master secret key msk that also serves to general functional keys. Naturally, several works extend this primitive into the multi-user, or more traditionally called *multi-client* setting [42]. Informally, in this situation, a coalition (or a group) of users controls the setup, the encryption and the functional key algorithms, and collaborate in order to delegate a computation as in FE. We refer to [42] for more details.

The core of our cryptographic system for the WeStat construction that is presented in Chapter 5, is the DSum scheme which is a special multi-client FE given in [42] that we present in this paragraph. This primitive allows a set of users to agree on the following functionality: encrypting an input to a group of users, then recovering the sum of them with the restrictive condition that all participants of the same group (under some label) indeed sent their contribution. We recall in the following an adapted definition of this notion, where we define the message space over any finite Abelian group $(\mathbb{G}, +)$. We fix a set of users indexed by a subgroup $\mathcal{I}$ of $\mathbb{G}$.

Definition 2.3.3 (DSum [42])**.** *The* DSum *multi-client functional encryption for the sum function over* $\mathbb{G}$ *consists of the following* (Setup, Enc, KeyGen, Dec) *algorithms.*

- $\mathsf{Setup}(1^\lambda)$*: a* PPT *algorithm that outputs public parameters* param *which are included to all the other algorithms.*
- $\mathsf{KeyGen}(i \in \mathcal{I})$*: a* PPT *algorithm that outputs* $(\mathsf{pk}_i, \mathsf{sk}_i)$*, where* pk_i *is party* i*'s public key and* sk_i *is the corresponding secret key.*
- $\mathsf{Enc}((\mathsf{pk}_i, \mathsf{sk}_i), x, \{\mathsf{pk}_j\}_{j \in \mathcal{J}}, \ell)$*: a* PPT *algorithm that takes as inputs a couple* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *generated during* KeyGen*, an input data* x *to encrypt, a set of public keys* $\{\mathsf{pk}_j\}_{j \in \mathcal{J}}$

indexed by some subset $\mathcal{J} \subseteq \mathcal{I}$ and a label ℓ. If $i \notin \mathcal{J}$ returns $\perp$. Otherwise, this algorithm outputs a ciphertext $ct_{i,\ell}$.

- $\mathsf{Dec}(\mathsf{param}, \{ct_{j,\ell}\}_{j\in\mathcal{J}})$*: a deterministic algorithm that takes as inputs* param *and a set of ciphertexts $\{ct_{j,\ell}\}_{j\in\mathcal{J}}$ then outputs a value $y \in \mathbb{G}$.*

The correctness condition states that for all security parameter λ, a label ℓ, for all $\mathcal{J} \subseteq \mathcal{I}$, if $\{\mathsf{pk}_j\}_{j\in\mathcal{J}}$ represents the set of users issued using KeyGen *and all corresponding ciphertexts $\{ct_{j,\ell}\}_{j\in\mathcal{J}}$ under the same set $\mathcal{J}$ and label ℓ, then we have*

$$\Pr[\mathsf{Dec}(\mathsf{param}, \{ct_{j,\ell}\}_{j\in\mathcal{J}}) = \sum_{j\in\mathcal{J}} x_j] = 1.$$

Notice that the correctness condition states that if all users of the *same* group provide the corresponding ciphertexts, then it is possible to recover the sum of all their inputs.

We discuss the main security definition for this primitive. Informally, even if there are some corrupted users, it is difficult to obtain any additional information about the inputs of the remaining honest users.

Definition 2.3.4 $(\mathsf{IND}-\mathsf{DSum})$**.** *For any* PPT *adversary $\mathcal{A}$, consider the following experiment.*

- *Initialization: the experiment starts by generating* $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$*. A random bit $b \in \{0,1\}$ is chosen and* param *is given to $\mathcal{A}$.*
- *Honest User creation: $\mathcal{A}$ has access to a* QHKeygen *oracle, which on input an index i, runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}()$ and returns pk_i to $\mathcal{A}$.*
- *Corrupt User creation: $\mathcal{A}$ has access to a* QCKeyGen *oracle, which on input an index i, gets the corresponding sk_i of any user i of its choice.*
- *Data challenge: $\mathcal{A}$ has an adaptive access to an oracle* QEncrypt*, which on inputs the elements (i, x_i^0, x_i^1, ℓ) and a set $\mathcal{J}$ returns*

$$ct_{i,\ell,b} \leftarrow \mathsf{Enc}((\mathsf{pk}_i, \mathsf{sk}_i), x_i^b, \{\mathsf{pk}_j\}_{j\in\mathcal{J}}, \ell).$$

- *Guessing challenge: $\mathcal{A}$ makes a guess b'.*

The output b' of the game depends on some conditions. Consider $\mathcal{CS}$ the set of corrupted users from QCKeygen *and $\mathcal{HS}$ the set of honest users from* QHKeygen*. If there exists a set $\mathcal{J}$ and $\{(j, x_j^0, x_j^1, \ell)\}_{j\in\mathcal{J}}$ such that $\sum_{j\in\mathcal{J}} x_j^0 \neq \sum_{j\in\mathcal{J}} x_j^1$ with*

- $x_j^0 = x_j^1$ *for all* $j \in \mathcal{CS} \cap \mathcal{J}$*;*
- $\mathsf{QEncrypt}(i, x_0, x_1, \ell)$ *queries have been asked for all* $j \in \mathcal{HS} \cap \mathcal{J}$*,*

then set b' *to be a random bit. Otherwise, the advantage of* $\mathcal{A}$ *is then defined as the quantity*

$$\mathsf{Adv}(\mathcal{A}) = |\Pr[b' = b] - 1/2|.$$

The last condition in the security definition captures the situation when $\mathcal{A}$ could trivially guess the bit b.

Construction and instantiation. In Chotard et al. [42], a generic DSum construction is given and makes use of NIKE [57] with a new concept they have called All-or-Nothing Encapsulation that we will not introduce. For the instantiation, we make use of the discussed construction given in Sec. 2.2.2 and that will fully presented in Chapter 5.

CHAPTER 3

FUNCTION'S PROTECTION IN FUNCTIONAL ENCRYPTION

Overview of our problematic

In functional encryption, the functional key sk_f is derived from a master secret key msk and the function f. The master key owner is then very powerful since it can also decrypt all the ciphertexts. In most use cases, the functional key generation protocol is *interactive* between the manager of the master secret key msk and the user knowing the function to be used. While the natural approach to obtain sk_f is to send f to the master secret key owner, we enlight a situation where the evaluation function f is sensitive. We consider in the sequel some possible implementations of FE motivated by some use cases. In particular, we describe in the following situations where a meaningful function's protection is desirable.

FE for Spam filter. On internet, one desirable feature is the possibility of detecting if some incoming data/packet contains or not a malicious program such as malware or spam. Historically, anti-virus software editors have been able to offer solutions in response to the growing demand of protection against these malicious programs. We will not review all of the solutions but generally speaking, the main feature is to have a local access in order to provide a deep analysis about the incoming traffic. Thanks to this access privilege, the antivirus protects the final subscriber of the service from obtaining malware programs but nothing prevents it from bypassing their initial functions. As a consequence, it is also possible to obtain even more (undesired) data that could potentially compromise the privacy of subscribers.

A way to circumvent this problem using FE is to provide, as a naive solution, a key to the antivirus editor that could permits to evaluate its algorithms over the encrypted data. Eventually, with this solution, the only leaked information in the end of the computation would be the protection of data without compromising the data being encrypted. Hence, with a master secret key msk, it is possible to produce a functional key sk_f corresponding to a function f representing a complex function known by the anti-virus editor, that should lead to the correct malware detection output. Beside, data remains encrypted and the editor could hope to build its activity using these particular detection services while promoting the privacy of its consumers.

Taking a regard back to the beginning of this chapter, recall that the functional key generation is interactive. The antivirus editor could find interest to *blind* the function towards msk owner's. Indeed, the function might be related to some specific spam rules and correspond to the market compliance defined in e.g. [33] which shows the sensitivity of the rules given by the security editor. However, for a possible implementation of FE, any owner of msk needs to somehow obtain the underlying (malware detection) function f to generate the functional key sk_f. A naive implementation that consists of sending f *in clear* would result to a situation where the master secret owner learns all the information about it.

FE for data analytics. In the last decades, there is undeniably a particular growth and interest in massive data analytics algorithms. The most famous ones are issued for the machine learning community. The set of algorithms are used among other concerns to better detect a specific disease or general commercial tendencies. These mechanisms are sometimes linked to some very particular and rare know-how behaviour.

Since these algorithms could manipulate some sensitive data, the growth of privacy-preserving solutions emerge as a natural concept. It is not difficult to imagine that FE could potentially be used in this situation. The data remains encrypted and a master secret key owner could provide a functional key to any complex machine learning algorithm that could safely be executed through the encrypted data. If sometimes these algorithms are seen as *black-box*, there is however some parameters that could potentially be relevant, even crucial, to *blind* for any particular use. As a consequence, a blindness notion seems again necessary anytime the underlying structure of these algorithms needs to be hidden to the master secret key owner.

Generality of our approach. We deliver in two previous examples a view of how FE could be potentially used in some specific cases and where it is crucial to protect the

function. Besides, it is instructive and natural to first question the necessity of considering any notion of *blindness*. Indeed, while our two presentations are *business-oriented* examples of the usefulness of FE for interactive privacy preserving solutions, we argue that treating this intuition for the general case for any function is still meaningful. In fact, a notion of blindness, more generally blind interactive FE, here after blind IFE, is not new. Our work focuses on the initial results of Green and Hohenberger [75] or Camenisch et al. [30]. We remarked that no such study has been done for the more general case of functional encryption and since FE is a generalization of many cryptosystems, it is natural to consider a general comprehensive framework that encompasses these previous works. Our notion of blindness could roughly be resumed in the following sentence: *there is no link between a functional key sk_f and the interactions that help to generate it.*

Having theses examples motivate our study. From a theoretical perspective, a blind IFE notion is interesting on its own. Eventually, it also could potentially be used as a building block for other cryptographic primitives or protocols, for example by considering some specific class of families, as in the previous works for IBE or ABE.

Insufficiency of function-privacy. Several other security properties have been considered for FE in the literature and we will not review all of them. However, we could mention the known [8, 24, 28] notion of *function-privacy* (FP) which informally states that a functional key sk_f does not give any additional information about the underlying function f, except from what is given by the evaluations over some data being encrypted.

Function privacy looks similar to our consideration of blindness. Since FP is a FE related notion, we need first to adapt and propose its generalization in the context of interactive FE. Informally, this is done by adding interactive oracles to the defintion in order to consider potential leakage during the interaction. Comparing our new notion of blindness and the existing one of function privacy is not immediate. Depending on the public or private key setting and the presence or not of the functional key sk_f in the master secret key owner's output, we obtain several disconnections between function-privacy and blindness security properties. Informally, this is due to the *nature* of the considered options. Indeed, the FP security asks *any adversary* which does not have necessarily an access to an encryption oracle, to obtain unwanted information about the function f from sk_f and eventually the interaction. The blindness security game concerns in another context, a *bad* msk's owner with the capability of encrypting arbitrary messages using msk which makes the functional keys sk_f *linked* to the interactions. Our main result in a nutshell says that these two properties are distinct, and then complementary.

Other related works. The notion of interactive functional key generation, without any consideration of blindness, was first considered in the *Accountable Authority* Identity Based Encryption (IBE) in [72] in order to mitigate the inherent key escrow problem in identity-based encryption. *Controlled* functional encryption [93] is also a variant of FE with an interactive behaviour where a fresh functional key is generated in accordance to ciphertext. While similar to our general approach of hiding the function to a master key owner, the model is different from ours. In fact, we only view two parties in our model where the master secret key owner is the only party to provide functional keys. In [93], it is only possible to produce functional keys that depends on the ciphertext and is only used once, while we consider multiple users, functional keys and ciphertexts.

The first closest consideration of *blindness* appears in the work of Green and Hohenberger in [75] followed after by the work of Camenisch et al. [30] for IBE where it was used as a building block for two primitives, respectively a *simulatable oblivious transfer* and a *public key encryption with oblivious keyword search*. In [101], an adaptation is proposed for the case of the Attribute-Based Encryption (ABE) primitive. The notion of blindness considered in these papers are inspired from the terminology of *blind signature*.

Organization of this chapter

We start by providing definitions that capture the situation of a user holding a function f, namely FuncOw and asking the msk's holder MskOw for a corresponding functional key sk_f during an interactive protocol. This leads us to formally introduce the notion of *interactive* FE and study the impact on existing security properties from the classical FE literature. Then, we consider the case where when user FuncOw wants to protect the function f from MskOw, introducing the notion of *blindness* and comparing with the well known function-privacy notion.

Next sections consider a generic construction of IFE from FE and the specific case of IPFE is given.

3.1 Definitions and Security Model

3.1.1 Syntactic Definitions for Interactive FE

The natural starting point is to define a public key *interactive* functional encryption (IFE).

From FE to interactive FE. Syntactically, a definition of an interactive FE is naturally derived from the one of FE with a slight modification on the KeyGen algorithm of Def. 2.3.1. In more details, we develop the notion of Interactive Functional Encryption (IFE) which is mainly adapted from the classical definition of FE, i.e we maintain Setup, Enc, Dec and the correctness condition, except that we replace the KeyGen algorithm by an IKeyGen two-party protocol between two players. For our purpose, the *parties* are modelled by considering the two following entities:

1. a PPT algorithm that represents a *function's holder* of some function f and denoted by FuncOw; and

2. a PPT algorithm that represents a msk*'s holder* of some master secret key msk and denoted by MskOw.

Definition 3.1.1 (Public key IFE)**.** *Let λ be a positive integer. A* public key interactive functional encryption *scheme with some fixed function space F consists of a tuple of algorithms* $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined as*

- $\mathsf{Setup}(1^\lambda)$ *is a* PPT *algorithm which takes as input a security parameter 1^λ, and outputs a master secret key* msk *and a master public key* mpk*.*

- $\mathsf{Enc}(\mathsf{mpk}, m)$ *is a* PPT *algorithm which takes as input a master public key* mpk *and a message $m \in M$, and returns a ciphertext c.*

- $\mathsf{IKeyGen}(\mathsf{MskOw}(\mathsf{msk}), \mathsf{FuncOw}(\mathsf{mpk}, f))$ *is a two-party interactive protocol between* MskOw *which has as input a master secret key* msk *and* FuncOw *which has as inputs a master public key* mpk *(generated using the* Setup *algorithm) and a function $f \in F$. The output of this protocol is, on the* MskOw*'s side an element* $\mathtt{Output}(\mathsf{MskOw})$ *and on the* FuncOw*'s side, a functional key sk_f.*

- $\mathsf{Dec}(\mathsf{mpk}, sk_f, c)$ *is a* PPT *algorithm which takes as input a master public key* mpk*, a functional key sk_f and a ciphertext c and outputs a string z.*

Similarly, we adapt the correctness condition as following. The IFE scheme described above is considered as correct if for all $f \in \mathcal{F}$ and all $m \in M$, if for $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IFE.Setup}(1^\lambda)$, sk_f is the result from the execution of functional key generation protocol $\mathsf{IFE.IKeyGen}(\mathsf{MskOw}(\mathsf{msk}), \mathsf{FuncOw}(\mathsf{mpk}, f))$ and $C_m \leftarrow \mathsf{IFE.Enc}(\mathsf{mpk}, m)$ then

$$\Pr\left[\mathsf{IFE.Dec}(\mathsf{mpk}, sk_f, C_m) = f(m)\right] \geq 1 - \mathsf{negl}(\lambda).$$

We conclude this paragraph by mentioning that the private-key setting can easily be adapted from the definition.

A 2PC formulation. With the two-party computation terminology of sec. 2.2.2, we can reformulate the above IKeyGen protocol and state that from two parties MskOw with input $x :=$ msk and FuncOw with inputs $y :=$ (mpk, f), there exists a (two-party) protocol IKeyGen that can compute the functionality that outputs the pair ($\texttt{Output}$(MskOw), sk_f), where $\texttt{Output}$(MskOw) can be deduced from the $\texttt{View}$ of MskOw (see sec. 2.2.2). In this situation, sk_f is some functional decryption key satisfying the correctness condition of Def. 3.1.1. Notice that sk_f could be the result of a (possibly) randomized evaluation on inputs the master secret key msk, the function f and some randomness used in the interaction. We will discuss about the influence $\texttt{Output}$(MskOw) in the next paragraphs.

3.1.2 A Trivial Example or FE *is* IFE

As already mentioned, for practical implementation of FE use-cases, the functional key generation KeyGen is most of the time interactive. The *non-interactive* case could formally be obtained by letting FuncOw := MskOw. In this situation the master key owner generates the functional key *locally* by its own without any interaction and the IKeyGen protocol is just the execution of KeyGen. A trivial, but still interesting, implementation of a simple IFE is given in the following paragraph.

Trivial IFE from FE. Start from any FE = (Setup, KeyGen, Enc, Dec) following Definition 2.3.1, it is easy to define a *trivial* interactive FE scheme following the above Definition 3.1.1. The Trivial.IFE := (Setup, Trivial.IKeyGen, Enc, Dec) scheme has the same Setup, Enc, Dec algorithms as the initial FE. The protocol Trivial.IKeyGen next uses the KeyGen algorithm of the FE scheme as described in Fig. 3.1. In fact, FuncOw sends f to MskOw in order to execute the KeyGen algorithm over f and obtains the corresponding sk_f. This simple and natural example of IFE will be used in our study for the security properties related to a general interactive FE.

Validity of sk_f. One instructive issue of this trivial example is that MskOw may have sent to FuncOw a functional key that is not generated using the specification of KeyGen. Thus, FuncOw should have a way to verify its validity. One solution was given for interactive blind IBE [30, 75]. In this situation, both the ciphertext and the functional key are associated with an identity. The underlying function permits to recover a message only if it is the same identity, i.e $f_{id'}(id, m) := m \iff id = id'$. The authors [30, 75]

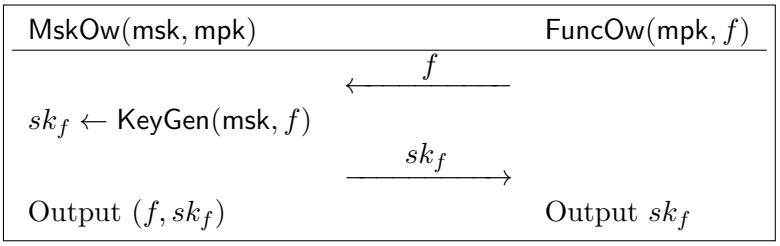

Figure 3.1: Trivial.IKeyGen.

propose to encrypt a polynomial number of random messages with the targeted id, then try to decrypt it using the obtained identity related functional key. Using the correctness of the IBE scheme, the authors conclude that it is sufficient to be convinced about the validity of the key.

A first idea can be to proceed similarly, which works quite well in the public key setting and in the case of (indexed) functions of the form of $f_k(m, y) := m \iff R(k, y) = 1$ where R is a publicly known relation and k is the index's function. However, in the general case, this method may obviously not convince a user of the validity of the functional key, since not all functions are of this form. In addition, this is definitely not possible in the private key setting because it is obviously not possible for FuncOw to encrypt arbitrary messages.

A solution to this verifiability problem consists in using zero-knowledge proof of knowledge mechanisms generated by MskOw to prove that it has correctly computed sk_f, generating a proof $\pi \leftarrow \mathsf{ZKPoK}(\mathsf{msk}) : \{sk_f = \mathsf{KeyGen}(\mathsf{msk}, f)\}$. Using this method, we ensure that sk_f is honestly generated by MskOw.

We stress that considering the validity of sk_f is an *additional* requirement. ZKPoK could be used to provide the validity of sk_f but other approaches are possible. We will not develop more about this notion and leave it as a natural extension of our work. We remark that we will use ZKPoK for MskOw that will ensure that sk_f is well constructed.

3.1.3 High-Level View of Security Properties

Unsurprisingly, an interactive FE should verify a modified version of the message-privacy property by considering some adapted interactive oracles. This will be detailed in Section 3.1.4. We will also consider function-privacy in addition to our new notion of blindness. This is due to the different ways the output of MskOw can be managed.

Output of MskOw. The fact that we want to hide the function to MskOw is at first related to the MskOw's view of the interactive protocol. Indeed, intuitively, the *best* case would be an MskOw which does not learn anything more than what it already knew before the interaction. The view of MskOw consists of $(\mathsf{msk}, r; m_1, \ldots, m_t)$, with r represents some random elements, and m_j the jth message that it received during its interaction.

Intuitively, we want to ensure a notion of *blind* functional key generation algorithm. This would mean that the MskOw cannot obtain from the received messages m_j any information about FuncOw's choice of the function.

A start could be to adapting the definition of blind signature [81] such that MskOw has *to link* a functional key sk_f generated during an interaction with the corresponding function f. We present in the following a similar definition for the case of IFE. To the best of our knowledge, it is new in the general context of functional encryption. We treat this security notion in depth in Section 3.1.7.

1. **Considering f in the output.** If f, or some informations about f is leaked during the execution (i.e. is contained in one of the m_js), it is clear that it is easy to link a function with the interaction. As an example, the construction of Trivial.IFE of sec. 3.1.2) cannot be blind.

2. **Considering sk_f in the output.** The functional key sk_f is used to decrypt ciphertexts c_m of some messages m in order to obtain values $f(m)$. For MskOw, which is in possession of msk, it is possible to encrypt any message m of its choice. If sk_f can be deduced from its view, then MskOw can learn too much information about f by encrypting any message m, thus obtaining $f(m)$ of its choice. This statement remains true even if sk_f have some *hiding* property that does not leak any information about the function f.

   If we use the same blindness definition, it seems difficult to attain the desired security property due to this inherent capability of having access to an unlimited evaluation of the function f.

   In addition, we notice that the same problem arises in the context of *function-private* public key functional encryption [24, 28] where hiding information about f in sk_f gives the same restrictions. In Section 3.1.6, we give a generalization in the context of interactive FE of this known function-privacy framework.

In light of this discussion, we have two different security properties that should be defined for interactive FE: function-privacy (see Section 3.1.6) and blindness (see

Section 3.1.7). But there may also be some relations between both and we will provide a discussion on this point in Section 3.2.

On simulation-based security. These security requirements could of course be defined in terms of *simulatability* which informally enables to design an ideal functionality that captures all the previous discussed properties (message-privacy, function-privacy and blindness) at the same time and consider interdependent executions with other protocols while preserving the main security characteristics.

However, we took the classical approach to provide a natural generalization of the blindness property from the literature, as well as the classical security notions for FE (i.e. message/function privacy) in the presence of an interactive key generation protocol. This has the benefits to only adapt existing definition by adding some interactive oracles, and may avoid some subtle negative results, as in the context of simulation-based blind signature [4]. In addition, our solution fits exactly with the existing constructions for the special cases of blind interactive IBE/ABE [30, 75, 77, 101] presented in some previous works.

3.1.4 Message-Privacy for Interactive FE

There are different approaches for defining the message-privacy of an interactive scheme. We decide to choose the one that could easily integrate the underlying message-privacy of FE. For this purpose, we present generalized notion of *leak-freeness* that was previously considered for the specific case of *blind IBE* [30, 75] or ABE [77]. The *leak-freeness* (see Sec.3.1.4) property is a real/ideal world definition that gives a way to transfer the FE security into IFE. the message-privacy property. For convenience, using leak-freeness provides a reduction to the underlying FE message-privacy property. Notice that this notion is not general as in the simulatability paradigm but we believe that it is sufficient to fill with our requirements.

Syntactic definitions. We here adapt the definition of message-privacy to our interactive setting. The main difference relies on the fact that some information could leak during the interactive key generation. We introduce the following interactive oracle that will serve in our description of the IFE's message-privacy.

$\mathsf{IKeyGen}(\mathcal{O}(\mathsf{msk}), \cdot)$**:** this oracle has msk hardwired in its description and takes as input a function $f \in F$. On every call, the oracle acts as in the interactive protocol by

playing the role of an honest MskOw. The output of this interaction is a functional key sk_f.

We are now ready to give the following definition that extends Definition 2.3.2 in the public key setting.

Definition 3.1.2 (Message-privacy for IFE)**.** *Let* $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public key* IFE*. The message-privacy for* IFE *is the same as the one defined in Def. 2.3.2, except that we replace the oracle* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *with the above oracle* $\mathsf{IKeyGen}(\mathcal{O}(\mathsf{msk}), \cdot)$*. The other oracles and the experiment* $\mathsf{Exp}^b_{\mathcal{A},\mathsf{MP}}(\lambda)$ *are unchanged.*

There are different ways to prove that an interactive FE is message-private. Obviously, one can directly build a scheme and proves that it satisfies the message-privacy of Definition 3.1.2. Another convenient method relies on using a scheme FE for building an IFE with the hope that it inherits the security from FE. We will now focus on the second option. For that, we need to study the message-privacy of the trivial IFE given above.

Message-privacy of Trivial IFE. Consider the IFE with the $\mathsf{Trivial.IKeyGen}$ from Example 3.1.2. Recall that the user sends f and the MskOw generates sk_f using msk. We have the following proposition.

Proposition 1. *The* $\mathsf{Trivial.IFE}$ *of Example 3.1.2 is message-private if the underlying* FE *scheme is message-private. For any adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *such that* $\mathsf{Adv}_{\mathsf{Trivial.IFE},\mathcal{A},\mathsf{MP}}(1^\lambda) = \mathsf{Adv}_{\mathsf{FE},\mathcal{B},\mathsf{MP}}(1^\lambda)$.

Proof. The proof is immediate. Suppose there is an adversary $\mathcal{A}$ attacking the $\mathsf{Trivial.IFE}$ scheme. We will consider the following adversary $\mathcal{B}$ that can break the message-privacy of the FE scheme.

- For the KeyGen's requests, it uses the same function requests that $\mathcal{A}$ makes to the $\mathsf{Trivial.KeyGen}$ oracle.
- It uses the same message (m_0, m_1) that $\mathcal{A}$ makes to the Enc_b oracle.
- Finally, it returns the same bit that $\mathcal{A}$ outputs.

Note that the inputs of $\mathcal{A}$ are well distributed and $\mathcal{B}$ is a valid adversary against the FE scheme because from the interaction in $\mathsf{Trivial.IKeyGen}$, the adversary $\mathcal{A}$ learns exactly the function f and sk_f which are already known after the KeyGen oracle's request. In particular, $\mathcal{B}$ has exactly the same advantage of $\mathcal{A}$ in winning the message-privacy security game. □

For the Trivial IFE, the curious user does not learn any information about the master secret key msk that could help her to break the MP security game of the underlying FE scheme. Considering interactions, we note however that the messages exchanged could potentially leak information about msk which is problematic.

Now, consider the construction of a new IFE and its message-privacy property. In the sequel, we propose to use the message-privacy of the Trivial IFE and the fact that the difference between the proposed interactive key generation and the one of the Trivial IFE does not compromise the message-privacy of the new proposed construction. This is based on the notion of *leak-freeness* that is inspired by the work done for IBE [30, 75].

3.1.5 Obtaining MP secure IFE from MP secure FE: leak-freeness

Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a message-private scheme.

The *leak-freeness* for functional encryption aims at providing a condition to *preserve* from learning any additional information due to the interactive key generation in order to break the message-privacy. Informally, it makes possible to prove that an IFE.IKeyGen protocol executed with an honest MskOw does not leak more information than the Trivial.IKeyGen from Example 3.1.2, with the same honest MskOw. Such notion can then be used to prove that the resulting interactive functional encryption $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is indeed message-private. Informally, this notion says that we cannot obtain additional information other than what is leaked from Trivial.IKeyGen. We provide a generalization of the Leak-Freeness property of [75].

Definition 3.1.3 (**Leak-Freeness**). *We say that an* IKeyGen *protocol is* leak-free *with respect to* KeyGen *of any* FE *scheme if, for all efficient adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for all value λ, no distinguisher $\mathcal{D}$ can determine whether it is playing* GameReal *or* GameIdeal *where*

- GameReal*: Run* $\mathsf{Setup}(1^\lambda)$*. As many times as $\mathcal{D}$ wants, $\mathcal{A}$ chooses a function f and executes the* $\mathsf{IKeyGen}(\mathsf{MskOw}, \cdot)$ *protocol input f with an honest authority* MskOw*. $\mathcal{A}$ produces a view and sends it to$\mathcal{D}$ which returns a bit.*
- GameIdeal*: Run* $\mathsf{Setup}(1^\lambda)$*. As many times as $\mathcal{D}$ wants, $\mathcal{S}$ chooses a function f and asks* $\mathsf{Trivial.IKeyGen}(\mathsf{msk}, \cdot)$ *to obtain a functional key sk_f on input f. Then, $\mathcal{S}$ returns the resulting view to $\mathcal{D}$ which returns a bit.*

The quantity $\mathsf{Adv}_{\mathcal{D},\mathsf{leak-free}}(1^\lambda) := |\mathsf{Pr}[\mathcal{D}^{\mathsf{GameReal}}(1^\lambda) = 1] - \mathsf{Pr}[\mathcal{D}^{\mathsf{GameIdeal}}(1^\lambda) = 1]|$ *is the advantage of $\mathcal{D}$ and* IKeyGen *is* leak-free *w.r.t* KeyGen *if it is negligible.*

We discuss in the following some remarks about the definition.

- We require to start from an FE scheme with some specific KeyGen algorithm in addition to the existence of a simulator (which interacts with a specific oracle Trivial.IKeyGen). This simulator is then asked to produce a consistent view to any distinguisher. As mentioned in previous sections, a two-party protocol would not necessarily offer the blindness property for free. In Example 3.1.2, Trivial.IKeyGen is by definition leak-free w.r.t KeyGen but cannot be blind since f is given to MskOw.

- The adversary in GameIdeal does not appear in the definition. As pointed in [75], the leak-freeness definition implies that the function (for the key being extracted) is *extractable* from the IKeyGen protocol (with all but negligible probability), since for every adversary it must exists a simulator $\mathcal{S}$ that should be able to interact with $\mathcal{A}$, in order to learn which functions to submit to the Trivial.IKeyGen(msk, $\cdot$) oracle.

We can now focus on our main result, which makes the link between leak-freeness and message-privacy. Informally, it states that any IKeyGen protocol with leak-freeness composes with the existing message-privacy of (non-interactive) FE. This result was stated without proof for blind IBE [75]. We provide in the following a proof of this fact for the general case of IFE.

Theorem 3.1.1. *Let* FE $=$ (Setup, Enc, KeyGen, Dec) *be a message-private secure* FE *scheme. Let* IFE $:=$ (Setup, Enc, IKeyGen, Dec)*. If* IFE.IKeyGen *is leak-free with respect to* KeyGen*, then* IFE *is message-private. For any adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{D}$ *such that*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{MP}-\mathsf{IFE}}(1^\lambda) \leq \mathsf{Adv}_{\mathcal{D},\mathsf{leak}-\mathsf{free}}(1^\lambda) + \mathsf{Adv}_{\mathcal{A},\mathsf{MP}-\mathsf{FE}}(1^\lambda).$$

Proof. We will prove this proposition via a sequence of games, where we reduce the MP security of the IFE scheme to the MP security of the initial FE scheme. Suppose there is an adversary $\mathcal{A}$ against the MP security of the IFE scheme as in Def. 3.1.2. By definition of the leak-freeness property, there exists a simulator $\mathcal{S}$ as described in Def. 3.1.3 which interacts with $\mathcal{A}$. Fix a random bit $b \in \{0,1\}$ and for each following game, denote by $\mathsf{Adv}_{\mathcal{A},\texttt{Game}\ i}(1^\lambda)$ the advantage that $\mathcal{A}$ has to win (i.e returns a bit b' such that $b' = b$) the game i, $i \in \{0,1\}$. We will use the public key version of the proof, but it can easily be adapted to the private key setting.

Game 0. This is the original message-privacy game against IFE. More specifically, consider the following phases.

- In the Setup phase, the master secret key msk is generated and the corresponding public key mpk is given to $\mathcal{A}$.
- Whenever the oracle IKeyGen is invoked on input $f \in F$, $\mathcal{A}$ participates in an interactive protocol with the oracle playing the role of an honest MskOw in possession of the master secret key msk. The adversary finally gets the output functional key sk_f. Notice that it also has extra information (messages exchanged) related to the interaction.
- By definition, the oracle Enc_b returns $\mathsf{Enc}(\mathsf{mpk}, x_b)$ on input (x_0, x_1) with $f(x_0) = f(x_1)$, on all the functions f asked in the previous IKeyGen phase.
- Note that $\mathcal{A}$ still has access to the IKeyGen oracle with the above inherent condition on (f, x_0, x_1), and finally returns a bit b'. Notice that

$$\mathsf{Adv}_{\mathcal{A},\mathtt{Game}\ 0}(1^\lambda) = \mathsf{Adv}_{\mathcal{A},\mathsf{MP\text{-}IFE}}(1^\lambda).$$

Game 1. This is the same game as the previous one, except that we change the answers of IFE.KeyGen oracle by exploiting the simulator $\mathcal{S}$ of the leak-freeness property. The Setup and the Enc_b phases remain the same.

We modify the IKeyGen phase in the following way. When $\mathcal{A}$ chooses an input f, the simulator $\mathcal{S}$ uses the same input f and invokes the Trivial.IKeyGen oracle in order to obtain a corresponding functional sk_f. Then, by the leak-freeness property $\mathcal{S}$ uses his simulated view of the interaction and gives it to $\mathcal{A}$. Recall that $\mathcal{S}$ can simulate the message exchanged during the interaction and can give functional keys corresponding to $\mathcal{A}'s$ requests.

In order to prove that the IFE scheme is message-private, we first state the following lemma.

Lemma 3.1.2. *For any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{D}$ *such that we have* $|\mathsf{Adv}_{\mathcal{A},\mathtt{Game}\ 0}(1^\lambda) - \mathsf{Adv}_{\mathcal{A},\mathtt{Game}\ 1}(1^\lambda)| \leq \mathsf{Adv}_{\mathcal{D},\mathsf{leak-free}}(1^\lambda)$.

Before proving this result, first assume that Lemma 3.1.2 is true and consider the adversary $\mathcal{A}$ in Game 1. By Proposition 1, the Trivial.IKeyGen is message-private if the FE with KeyGen (in the non-interactive sense) is message-private.

In Game 1, the adversary obtains no additional information other than what it can learn from Trivial.IKeyGen, i.e. from the simulator $\mathcal{S}$ except with negligible probability. This fact is induced by the leak-freeness property. We can deduce that in Game 1 we have $\mathsf{Adv}_{\mathcal{A},\mathtt{Game}\ 1}(1^\lambda) \leq \mathsf{Adv}_{\mathcal{A},\mathsf{MP\text{-}FE}}(1^\lambda)$. It remains to prove the Lemma 3.1.2.

Proof. (of Lemma 3.1.2) We claim that, if $\mathcal{A}$ can distinguish between Game 0 and Game 1 with non-negligible advantage, then we can build an adversary $\mathcal{D}$ that can distinguish with non-negligible advantage between GameReal and GameIdeal in the leak-free security game (leading to a contradiction). Consider a distinguisher (for the leak-free property) $\mathcal{D}$ that works as follows.

- In GameReal: run $\mathsf{Setup}(1^\lambda)$ then $\mathcal{D}$ uses the same function requests that $\mathcal{A}$ makes to the IKeyGen oracle in Game 0. $\mathcal{D}$ can obtain in the end of each interaction the resulting view (i.e. transcript) from $\mathcal{A}$.
- In GameIdeal: run $\mathsf{Setup}(1^\lambda)$ then $\mathcal{D}$ uses the same function requests that $\mathcal{A}$ makes to the IKeyGen oracle in Game 1. Notice that in this situation, the answers are given by a simulator $\mathcal{S}$ with its access capability to the Trivial.IKeyGen oracle. Here again, $\mathcal{D}$ can also obtain the information that $\mathcal{A}$ obtains after each interaction with $\mathcal{S}$.

From the above requests, we conclude that the probability of success for $\mathcal{D}$ in distinguishing GameReal and GameIdeal is exactly the same as the probability that $\mathcal{A}$ has in distinguishing Game 0 from Game 1. Indeed, to see this, we first remark that the view of $\mathcal{A}$ is consistent (i.e. it provides valid functional keys sk_f) by definition of the simulator $\mathcal{S}$ and the resulting view given by $\mathcal{A}$ to $\mathcal{D}$ is the view given by $\mathcal{S}$ which are again well simulated thanks to the leak-freeness property.

However, we also conclude by this last property that if such $\mathcal{D}$ exists, it has a negligible advantage in distinguishing GameReal from GameIdeal. This leads to a contradiction and we conclude the proof of Lemma 3.1.2 by deducing that $\mathcal{A}$ has a negligible advantage of distinguishing between Game 0 and Game 1. Then, Game 0 and Game 1 are indistinguishable. □

Returning back to the proof of the lemma, we have

$$\mathsf{Adv}_{\mathcal{A},\mathsf{MP\text{-}IFE}}(1^\lambda) \leq \mathsf{Adv}_{\mathcal{D},\mathsf{leak\text{-}free}}(1^\lambda) + \mathsf{Adv}_{\mathcal{A},\mathsf{MP\text{-}FE}}(1^\lambda),$$

which are negligible and where $\mathsf{Adv}_{\mathcal{D},\mathsf{leak\text{-}free}}(1^\lambda)$ is the (negligible) probability resulting from Lemma 3.1.2. We conclude the proof of the Theorem 3.1.1. □

3.1.6 Function-Privacy for Interactive FE

In this section, we present some adapted definitions of function-privacy for FE with interactive key generation. Our plan is not to provide a function-private IFE scheme but,

as we will see, there are some close relations between this existing notion of function-privacy and our new notion of blindness. Informally speaking, function-privacy (FP) security ensures that a functional key sk_f gives no information about the function f. There exist different definitions depending on the situation (public vs. private key setting). We give the adaptations of existing definitions to the case of an interactive FE since the transcript of the interactive protocol could reveal some information about the function f.

Function-privacy for private key IFE. The private key setting follows the *left-or-right* terminology [28, 84]: the adversary guesses which of the two chosen functions is used in a interactive key generation protocol. We then introduce the new following oracle

$\mathsf{IKeyGen}_b(\mathsf{msk}, \cdot, \cdot)$ on input f_0 and f_1, runs (an honest) $\mathsf{IKeyGen}$ protocol on input the function f_b for a bit b, then generates trans_b which contains the messages exchanged during the protocol and the private user's output sk_{f_b}. The oracle finally sends trans_b and sk_{f_b} to the adversary.

We can now provide an adapted definition of function-privacy or the IFE case, in the private-key setting. In particular, we consider a Left-Or-Right definition.

Definition 3.1.4 (LoR function-privacy)**.** *We say that a private key* IFE *scheme* $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* M *and a function space* F *is* ***function-private*** *if for any adversary* $\mathcal{A}$*, there exists a negligible function* negl *such the following difference of two probabilities, called the advantage of* $\mathcal{A}$ *and denoted* $\mathsf{Adv}_{\mathcal{A},\mathsf{LOR}-\mathsf{FP}}(1^\lambda)$*, verifies*

$$\left|\Pr\left[\mathsf{Exp}^0_{\mathcal{A},\mathsf{FP}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^1_{\mathcal{A},\mathsf{FP}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^b_{\mathcal{A},\mathsf{FP}}(\lambda)$ *is defined as*

1. $(params, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $b' \leftarrow \mathcal{A}^{\mathsf{IKeyGen}_b(\mathsf{msk},\cdot,\cdot)}(1^\lambda, params)$
3. *output* $b = b'$

with the descriptions of f_0 *and* f_1 *that* $\mathcal{A}$ *queries have the same length*[1]*.*

[1] This restriction is asked to avoid trivial attacks.

If a function-private FE is implemented for real-world applications, an eavesdropper could try to get some information about f starting from the transcript. Note that we do not give the adversary an access to an encryption oracle since it permits to easily win the game (see the public key case below). Such limits can be easily surrounded by establishing a secure channel first.

Example. *The* Trivial.FE *of Example 3.1.2 is obviously* not *FP in the sense of* IFE *according to the above definition since the transcript contains the function itself. This argument holds even if the original* FE *is function-private. Fortunately, we could easily modify it by using a secure channel (for example using one-time encryption) to transmit the function and the functional key. Note that the* MskOw *would still have access to the function f.*

Function-privacy for public key IFE. If an adversary can encrypt messages of its choice (in the public key setting or in the case of a malicious MskOw having access to msk), the previous LoR definition 3.1.4 fails since a simple attack consists in encrypting a message m and get $c = \mathsf{Enc}(\mathsf{mpk}, m)$ (or $\mathsf{Enc}(\mathsf{msk}, m)$ respectively), such that $f_0(m) \neq f_1(m)$. Then, the adversary can use its challenge functional key sk_{f_b} in order to decrypt c and get $f_b(m)$. The above Left-Or-Right definition is thus not possible in the public-key setting.

To overcome this issue, the *real-or-random* (RoR) approach was proposed in [8,25] by adding some hypothesis about the entropy of the function space. The RoR-FP security below informally states that as long as the adversary asks a functional key for a function sampled from a *feasible entropy distribution* (see Definition.2.1.2), it can not decide if this key is actually coming from his distribution or from a uniform one.

We will consider both a passive adversary (not involved in the protocol but having access to exchange data) and an active one (acting as the involved MskOw). Let $\mathtt{mode} \in \{\mathtt{real}, \mathtt{random}\}$ and $\mathtt{case} \in \{\mathtt{weak}, \mathtt{strong}\}$ be formal variables. We first define the real-or-random function-private oracle as follows.

Definition 3.1.5 (Real-or-random key generation oracle)**.** *The real-or-random* IKeyGen *oracle, denoted* $\mathsf{RoRIKeyGen}(\mathtt{case}, \cdot, \cdot, \cdot)$*, takes as input triplets of the form* $(\mathtt{mode}, \mathsf{msk}, D)$*, where* $\mathtt{mode} \in \{\mathtt{real}, \mathtt{random}\}$*,* msk *is the master secret key and D is a feasible entropy distribution over F (if not, the oracle aborts). If* $\mathtt{mode} = \mathtt{real}$ *then the oracle samples* $f \leftarrow D$*. If* $\mathtt{mode} = \mathtt{random}$ *then the oracle samples* $f \leftarrow U(F)$*. There are then two cases.*

- *If* $\mathtt{case} = \mathtt{weak}$*, the oracle runs both parties of the* IKeyGen *protocol, which outputs* trans*,* $\mathtt{Output}(\mathcal{O})$ *and* sk_f*. It returns* trans *and* sk_f *to* $\mathcal{A}$*.*

- *If* $\texttt{case} = \texttt{strong}$*, the oracle (playing the role of the* FuncOw*) interacts with the adversary (playing the role of the* MskOw*) in the* IKeyGen *protocol. At the end of the protocol, the* FuncOw *output* sk_f *is given to* $\mathcal{A}$*.*

The RoR-FP security game follows by using such RoRIKeyGen oracle, where the adversary is asked to distinguish between the real and the random mode, and is restricted to distributions with feasible entropy.

Definition 3.1.6 (RoR function-privacy)**.** *We say that an* IFE *scheme over a message space* M *and a function space* F *is RoR-FP secure if for any adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left|\mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\texttt{real},\mathsf{FP}}(1^\lambda)=1\right]-\mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\texttt{random},\mathsf{FP}}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}_{\mathcal{A},\texttt{mode},\texttt{case},\mathsf{FP}}(\lambda)$ *is defined for* $\texttt{mode}\in\{\texttt{real},\texttt{random}\}$ *as*

1. $(\mathsf{mpk},\mathsf{msk})\leftarrow\mathsf{Setup}(1^\lambda)$*;*
2. *if* $\texttt{case}=\texttt{weak}$*,* $\texttt{mode}'\leftarrow\mathcal{A}^{\mathsf{RoRIKeyGen}(\texttt{weak},\texttt{mode},\mathsf{msk},\cdot),\mathsf{IKeyGen}}(\lambda,\mathsf{mpk})$*;*
3. *if* $\texttt{case}=\texttt{strong}$*,* $\texttt{mode}'\leftarrow\mathcal{A}^{\mathsf{RoRIKeyGen}(\texttt{strong},\texttt{mode},\mathsf{msk},\cdot)}(\lambda,\mathsf{msk})$*;*
4. *output* 1 *if* $\texttt{mode}'=\texttt{mode}$*,* 0 *otherwise.*

In the weak case (without any reference to a transcript since it is not interactive), our notion exactly meets the classical FP defined in [28]. To the best of our knowledge, the strong case has never been proposed in the literature and is the strongest requirement one can get for function-privacy in public key IFE. Notice that since any public key IFE can be converted into a private key IFE scheme, we can also use Definition 3.1.6 in the private key setting.

Thus, a private key IFE can be proved to be RoR-FP or LoR-FP secure, while a public key IFE can only be proved to be RoR-FP secure.

On the feasible entropy condition. A natural question is whether it is possible to avoid the restriction on the distributions with feasible entropy. We can allow the adversary to choose a function f and get either the functional key sk_f related to f or a functional key sk_g related to a function g randomly chosen in the set of all functions with the same length description as f. However, in the strong case, as in the context of the public key setting, the adversary controls the master secret key and could find

a message m such that $f(m) \neq g(m)$. We cannot control the evaluation of $g(m)$ since it is chosen randomly and could be different from $f(m)$. In the private key setting and for the special weak case, it is possible to control the encryption oracle. We trace all the requests m and f that an adversary can ask, and get a uniform sample g such that $g(m) = f(m)$, but it seems to be a strong restriction and we will not consider it here.

3.1.7 Blindness for Interactive FE

In this section we formally define our new *blindness* property. Intuitively, following the usual definition for blind signatures [81], blindness means that the MskOw cannot link a functional key to an interaction it had with an honest user. This is clearly related to the information that the MskOw has at the end of the key generation protocol, namely $\mathtt{Output}(\mathsf{MskOw})$.

It is possible to define a unique notion of blindness independently for both the private and public key settings. Our objective is to simulate an adversary who can choose maliciously the parameters but follows the specification of the protocol. Its aim is to decide which of two chosen functions f_0, f_1 has been used to generate the functional keys sk_{f_0} and sk_{f_1} in two sequential executions with an honest user FuncOw. This notion corresponds to a variant of the *selective-failure blindness* security considered in [30,75] for IBE. This additional security requirement was used in order to build oblivious transfer [75] or searchable encryption [30]. Here, we considered the basic definition and leave the additional requirement for possible applications.

We introduce the interactive oracle $\mathsf{IKeyGen}(\cdot, \mathcal{O}(\mathsf{mpk}, f))$ in which the adversary plays the role of the MskOw and only obtains his own output. In the game below, we write $\mathcal{A}^{\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(f_0))/\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(f_1))}$, which means that $\mathcal{A}$ can query each oracle only once (hence the notation $\mathsf{IKeyGen}^{(1)}$) and that the two oracles can be invoked in an arbitrary order but in a *sequential* manner[2].

Definition 3.1.7 (Blindness). *Let $b \in \{0,1\}$. An IFE is **blind**, if every adversary $\mathcal{A}$ has a negligible advantage in the following experiment*

1. $(\mathsf{mpk}, f_0, f_1, st_{find}) \leftarrow \mathcal{A}^{\mathsf{Setup}(\cdot)}(find, 1^\lambda)$
2. $st_{issue} \leftarrow \mathcal{A}^{\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(\mathsf{mpk},f_b))/\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(\mathsf{mpk},f_{1-b}))}(issue, st_{find})$, *at the end of the executions, this step produces local outputs (possibly undefined) sk_{f_b} and $sk_{f_{1-b}}$ respectively*

[2] By standard hybrid arguments, it is possible to show that it is equivalent to multiple session, as done in [81] for blind signatures.

3. If $sk_{f_0} = \perp$ *or* $sk_{f_1} = \perp$*, set* $(sk_{f_0}, sk_{f_1}) = (\perp, \perp)$

4. $b' \leftarrow \mathcal{A}(guess, sk_{f_0}, sk_{f_1}, st_{issue})$

The advantage of $\mathcal{A}$ *in this game is* $\mathsf{Adv}_{\mathcal{A},\mathsf{Blind}} := |\mathsf{Pr}[b' = b] - 1/2|$.

This definition can easily be adapted to the private key setting.

It is important to notice, as in the context of blind signature, that any information about sk_f that can be deduced during the interaction from $\mathtt{Output}(\mathsf{MskOw})$ lead our definition to fail. Indeed, for example if $\mathcal{A}$ gets sk_f in the end of the interaction, it will obviously win the game by just interacting with one of the two oracles. In fact, any *Left-Or-Right* definition would fail, since during the interaction there is always a way to distinguish between two keys/interactions. This difficulty comes from the inherent capabilities of the FE scheme. From the encryption of a certain message m such that $f_0(m) \neq f_1(m)$ and an interaction giving sk_{f_b} at the end of one of the two interactions, it is always possible to decrypt and get $f_b(m)$. Since f_0 and f_1 are chosen by $\mathcal{A}$, it seems clear that the blindness implies in particular that the malicious MskOw does not get informations about sk_f and f during (or in the end of) the interaction. Notice the similarities with the function-privacy notion in the public key case. We will discuss in the next section in further details the relationship between these notions.

3.2 On the Relationship between Blindness and Function Privacy

Depending on the public or private key setting and the presence or not of sk_f in the MskOw's output, we obtain several (dis)connections between function-privacy and blindness security properties. Informally, this is due to the *nature* of the considered options. Indeed, FP security is not specific to any entity and asks an adversary to obtain unwanted information about the function f from sk_f and eventually the interaction. The blindness security game only concerns MskOw with the capability of encrypting arbitrary messages using the master secret key msk. We deduce that in the private key setting, we can compare both the RoR-FP and the LoR-FP properties to the blindness one. In the public key setting, we can only compare the RoR-FP security property to the blindness one. We now give our main theorem which, in a nutshell, says that these two properties are distinct, and then complementary.

Theorem 3.2.1. *Function-Privacy and Blindness properties are different, for both private-key and public-key IFE. There exists a set of functions with a family of computationally*

secure constructions, based on hardness assumptions and satisfying the following relations:

- blind *secure* IFE *scheme that is not* (weak/strong)-RoR-FP *secure;*
- (weak/strong)-RoR-FP *secure* IFE *scheme that is not* blind*;*
- blind *secure* IFE *scheme that is not* LoR-FP *secure;*
- LoR-FP *secure* IFE *scheme that is not* blind*.*

In the next section, we give the proof of this theorem.

Proof of Theorem 3.2.1. First, we start with the obvious implications. Recall that Strong RoR-FP implies Weak RoR-FP by definition, and the converse is not true in general because we give more power to the adversary. The argument works for both public/private key IFE.

Next, for each of the remaining cases, we will exhibit an IFE, and in particular a set of function F, such that it verifies one security property but fails to verify the other one.

A blind scheme that is not (`strong`/`weak`)-RoR-FP. In this section we will build a scheme that is blind and not `strong`/`weak`-RoR-FP. Fix a finite field $\mathbb{F}_p$, an integer $\ell \geq 1$ and consider the following set of functionalities $F := \{\langle \cdot, \mathbf{y} \rangle,\ \mathbf{y} \in \mathbb{F}_p^\ell\}$, where for $\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{i=1}^{\ell} x_i \cdot y_i$. Remark that if we note $e_i := (0, \ldots, 1, \ldots, 0)$, where the 1 is at position i, we have trivially for all $\mathbf{y} = (y_1, \cdots, y_\ell)$ the result $\langle y, e_i \rangle = y_i$. We can deduce that for a given functional key sk_y for a certain vector $\mathbf{y}$ corresponding to a function $\langle \cdot, \mathbf{y} \rangle \in F$, if one can encrypt the vector e_i (in the public key case for example), then it could get the i-th component of the vector $\mathbf{y}$ which is y_i. This is inherent to any IPFE supporting this family F. In the public key setting, or in the case of a curious MskOw (knowing msk), it is possible to encrypt this kind of vectors. We anticipate a little and consider our blind construction of sec. 3.3.4 for this set of functionalities. Recall that depending on the case, the functional keys enjoy the same structure (i.e an inner-product over the integers). Indeed, in most of the known schemes [1, 11, 38, 111], the functional key has the form of $sk_{\mathbf{y}} := (\mathbf{y}, \langle \mathbf{s}, \mathbf{y} \rangle) \mod q$ for a certain vector $\mathbf{s} \in \mathbb{F}_p^\ell$. Notice that the *key* y is given in the functional key. Intuitively, we could rapidly conclude that every IFE for this set F is not function private. However, there is a subtlety in the definition of the RoR-FP security game.
The adversary is asked to give a distribution D with a *feasible entropy* condition, which means that the adversary is restricted to some *unpredictable* set of vectors y. Given

sk_y, its goal is to distinguish if y was chosen in D or from a uniform distribution. In particular, in our case it corresponds to the situation where y is *in* the description of sk_y and is coming from a unpredictable set.

At first glance, this view seems to contradict the intuition that the scheme is not function private. However, notice that the feasible entropy condition does not mean that D *is* the uniform distribution, it is classical to approximate it using for example rejection sampling. The adversary has the capability to find some elements that could distinguish this two distributions but makes the distribution D with enough entropy to fit the feasible entropy condition.

In fact, we will see how we can build an adversary that can break the RoR-FP security in our IFE scheme for the inner-product construction of sec. 3.3.4. We have the following Lemma.

Lemma 3.2.2. *The* IIPFE *scheme of sec. 3.3.4 is not RoR-FP secure. In particular, there exists an* IFE *scheme that is blind and not RoR-FP secure.*

Proof. Fix the parameters for the construction of sec. 3.3.4. Consider the following RoR-FP privacy adversary $\mathcal{A}$.

- $\mathcal{A}$ computes the distribution D that samples a uniformly distributed $\mathbf{y} \in \mathbb{F}_p^\ell$ for which the most significant bit of y (represented as an integer) is equal to 0. Note that this step could be done by using rejection sampling to obtain a sufficiently good approximation of the uniform distribution over $\mathbb{F}_p^\ell$.

- Then, it asks the $\mathsf{RoR} - \mathsf{IKeyGen}(\texttt{Case}, \texttt{mode})$ oracle in order to get a key sk_y no matter the `Case` is `strong`/`weak`.

- It parses the received functional key as $sk_y = (y, \langle s, y \rangle)$ and returns 1 if the most significant bit of y is equal to 0. Otherwise it returns 0.

Next, we remark that the above adversary has an advantage $1 - \frac{1}{2}$ in distinguishing the `real` mode from the `random` mode, thereby breaking the function-privacy. To see this, note that when $\texttt{mode} = \texttt{real}$ the adversary wins the game with probability 1 and when $\texttt{mode} = \texttt{random}$, it returns 1 when the most significant bit of y is equal to 0 which happens with probability exactly $\frac{1}{2}$.

We deduce that the adversary has advantage $1 - 1/2 = 1/2$ in distinguishing the `real` mode from the `random` mode, which is a non-negligible advantage. Notice however that we will prove in sec. 3.3.4 that this scheme is blind. □

Notice that our result also holds for any blind construction of the inner-product case and is not specific to our construction.

A (`weak`/`strong`)-RoR-FP scheme that is not blind. In this section, we build an IFE scheme that is RoR-FP and not blind. The intuition behind is related to the nature of the security games.[3] For any IFE scheme denoted by $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$, let the following $\mathsf{IFE}' := (\mathsf{Setup}, \mathsf{IKeyGen}', \mathsf{Enc}, \mathsf{Dec})$, where we only modify the $\mathsf{IKeyGen}$ protocol and does not change the other algorithms. The $\mathsf{IKeyGen}'$ protocol is described as follows.

$\mathsf{IKeyGen}'$**:** MskOw holds a master secret key msk and the user FuncOw holds an input (f, mpk), and they run protocol $\mathsf{IKeyGen}(\mathsf{MskOw}(\mathsf{msk}), \mathsf{FuncOw}(\mathsf{mpk}, f))$ protocol. In the the end of the interaction PPf has a functional key sk_f. The modification consists of FuncOw sending sk_f to MskOw.

The resulting IFE' scheme is clearly correct and we have the following lemma.

Lemma 3.2.3. *Suppose that* IFE *scheme is* `weak`/`strong`*-RoR-FP secure. Then* IFE' *is* `weak`/`strong`*-RoR-FP secure and not blind.*

Proof. We start with the `weak`/`strong` RoR-FP security. Recall that an adversary $\mathcal{A}$ interacts with the $\mathsf{RoRIKeyGen}$ oracle in order to get a functional key sk_f where f is chosen from a feasible entropy distribution or a uniform one. The task for the adversary is to distinguish these two cases. In our situation, remark that last message of the interaction for $\mathsf{IKeyGen}'$, FuncOw sends the functional key to the MskOw. We deduce that when an adversary is using $\mathsf{RoRIKeyGen}'$ oracle, it receives twice the function sk_f from an honest party and this final interaction does not give additional information. We deduce that this is exactly the same as if the adversary is using the $\mathsf{RoRIKeyGen}$ oracle.

Consider an adversary $\mathcal{B}$ attacking the RoR-FP security for the IFE' scheme. We build an adversary $\mathcal{A}$ that breaks the RoR-FP security of the IFE scheme using the adversary $\mathcal{B}$ as follows

- $\mathcal{A}$ runs $\mathcal{B}$ and gets a distribution D with the feasible entropy condition. In addition, $\mathcal{A}$ eventually records the transcript that $\mathcal{B}$ generates when it calls the $\mathsf{RoRIKeyGen}$ oracle on input D. This transcript contains the elements that $\mathcal{B}$ used in the $\mathsf{IKeyGen}$ protocol with an honest user $\mathcal{O}$.

[3] i.e *linking a functional key to a function* versus *a functional key does not give information about the function*

- $\mathcal{A}$ asks for each $\texttt{case} \in \{\texttt{weak}, \texttt{strong}\}$ and $\texttt{mode} \in \{\texttt{real}, \texttt{random}\}$, the above $\mathsf{RoRIKeyGen}'(\texttt{case}, \texttt{mode}, \mathsf{msk}, \cdot)$ oracle, on input the distribution D and the corresponding transcript. $\mathcal{A}$ receives a functional key sk_f where $f \in D$ or chosen uniformly from the set of functions F.
- $\mathcal{A}$ forwards the functional key sk_f to the adversary $\mathcal{B}$ attacking the RoR-FP security of IFE' scheme and returns the same bit output as $\mathcal{B}$.

Note that all the values are well-distributed and we can conclude that the adversary $\mathcal{A}$ has exactly the same advantage as $\mathcal{B}$ in breaking the IFE scheme The IFE' scheme is then RoR-FP.

The scheme is not blind. Consider the following adversary $\mathcal{A}$ against the blindness security game. It chooses two functions f_0, f_1 and a message m such that $f_0(m) \neq f_1(m)$. A random bit $b \in \{0, 1\}$ is chosen and $\mathcal{A}$ runs honestly the protocol $\mathsf{IKeyGen}'$ with an oracle user $\mathsf{FuncOw}_b(f_b)$. Note that at the end of the interaction, it receives sk_{f_b}. It can uses an encryption $c_m \leftarrow \mathsf{Enc}(\mathsf{msk}, m)$ of the message m. If $\mathsf{Dec}(sk_{f_b}, c_m) = f_0(m)$, $\mathcal{A}$ returns 0, else, it returns 1. The adversary has probability 1 to find the bit b and the scheme is not blind. □

A blind scheme that is not LoR-FP. We anticipate and we argue that the IPFE in sec. 3.3.4 is blind but not LoR-FP since $\mathbf{y}$ is in the description of $sk_{\mathbf{y}}$. We can build an adversary $\mathcal{A}$ that can win the LoR-FP security game as follows. It chooses two functions $(\mathbf{y}^0, \mathbf{y}^1)$ with the same length in $\mathbb{Z}_p^{2\ell}$ with $\mathbf{y}^0 \neq \mathbf{y}^1$. After receiving the functional key $sk_{y^b} := (\mathbf{y}^b, \langle \mathbf{y}^b, \mathbf{s} \rangle)$ for a certain bit b corresponding to one of the two keys $\mathbf{y}_0, \mathbf{y}_1$, it parses $sk_{\mathbf{y}^b}$ and returns 0 if the first coordinate $\mathbf{y}^b$ is equal to $\mathbf{y}^0$, and 1 otherwise. It is easy to see that $\mathcal{A}$ will win the game with probability 1. In addition, we prove in sec. 3.3.4 that the underlying interactive IPFE is blind.

LoR-FP secure IFE scheme that is not Blind. Consider any non-interactive LoR-FP FE implemented following the $\mathsf{Trivial.FE}$ example. Fix any secure private-key encryption scheme $\mathsf{SE} := (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ with the standard CPA security[4]. We suppose that MskOw and FuncOw shares the same private key K for encryption and decryption. We modify the $\mathsf{Trivial.FE}$ as follows. The user and the MskOw generates the parameters for the secure encryption. The user then encrypts, using the algorithm E and the key K the description of the function f. The MskOw decrypts the received ciphertext using D and the key K and obtain f. It generates the functional key sk_f then returns an encryption

[4] or any secure channel.

of sk_f using E and the key K. Finally, FuncOw decrypt the received ciphertext and return sk_f.

This modified scheme is clearly LoR-FP since a passive adversary does not learn anything about the function f thanks to the security of the SE scheme. Notice however that the MskOw learns the function f during the interaction and the scheme could not be blind.

3.3 IFE from non-interactive FE

In this section, we develop our general construction from any set of functions that transforms an FE to an IFE. On our previous published work, Canard et al. [34], we build a generic IFE scheme from any FE using fully homomorphic encryption. The idea is to encrypt a function f using FHE, then homomorphically evaluates $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ over the encrypted data. The resulted ciphertext is an encryption of the functional key. Moreover, we provide ZKPoK to prove the validity of the computations. The way we used this primitive is a special case of a more general setting of the notion of *Private Function Evaluation* (hereafter PFE). In a nutshell, it is a special case of a two-party computation, where a Party 1 has input x and Party 2 inputs (y, g) for a function g such that Party 2 obtains in the end of the interaction $g(x, y)$ while Party 1 obtains nothing.

The problem of Private Function Evaluation (PFE) can be reduced to the problem of secure computation using 2PC. [5] It is also believed [97] that using (circuit-private) fully homomorphic encryption scheme with adapted zero knowledge proofs gives some feasibility result and can be used to achieve PFE for all functions.

An interesting path is to generalize our previous published work and eventually, in some cases, obtain more efficient consideration. For this purpose, we point that several other works [82, 89, 90] aim at improving the efficiency (communication cost, round complexity) or at reducing the assumption in order to get practical instantiation by using universal circuit, homomorphic encryption or secret sharing respectively.

The novelty of this thesis is to extract the needed properties to propose, using PFE, a modular approach for building IFE from FE.

Intuition. FuncOw and MskOw agree on a two-round secure private function evaluation protocol on inputs a master secret key msk and a function f and eventually the

[5] by using a *universal machine/circuit* U defined by $U(x, C_g) := g(x)$ for every circuit C_g *implementing* the function g. Than PFE can be solved by having the parties run a standard general two-party computation for U.

user obtains the evaluation of the circuit $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ on input f, which is a valid functional key sk_f. The protocol has the following properties: the FuncOw sends the first message, the MskOw replies and the outputs stay on user's side. Thereby, the PFE hides to the MskOw both the function f and the key sk_f and hence the pair (f, sk_f) cannot be deduced from the interaction or $\mathtt{Output}(\mathsf{MskOw})$.

We now highlight the requirement and the limitation of this basic idea and a brief comparison with our FHE-based construction.

1. In order to reduce the message privacy security game to the security of the PFE scheme, we need to extract the underlying function. However, after seeing the (possibly) first message of the PFE protocol, it is not clear how a simulator can generate the second message that should be consistent with the specification of the PFE scheme. In comparison with the FHE based construction presented in our article [34], the first message consists of an encryption of f. In the proof, we used ZKPoK in order to extract the function in addition to a *weak function indistinguishablity* property in order to give provide well formed distributions. Having this, we exploited the homomorphic structure of the underlying FHE scheme in order to create consistent elements during the interaction. We adapt this requirement for the PFE situation since we are not necessarily in the presence of homomorphic properties.

2. Concerning blindness, even if MskOw does not learn the FuncOw's inputs, it could cheat to make the output within two interactions depend on the function in different manners. As an illustration, the adversary could use during two interactions, two master secret keys (compatible with the scheme) that makes it deduce the FuncOw's choice. For our FHE-based construction, we exploit the indistinguishability of two ciphertexts (with ZKPoK) to argue that each interaction is independent from any function choice. We propose an adapted property for the case of PFE.

Inspired by the work of [61] in the context of blind signature, we can fix this limitations and produce a generic construction using any PFE scheme.

3.3.1 Definition of PFE

We will define formally our need of a two-round PFE. Consider two parties (P_1, P_2). We suppose that P_1 holds a circuit C and P_2 holds an input x. In our situation, the P_1 holds $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ and P_2 the description of circuit computing a function f.

Experiment 1	Experiment 2
$(msg_1, C, st) \leftarrow \mathcal{A}(1^\lambda)$	$(msg_1, C, st_1) \leftarrow \mathcal{A}(1^\lambda)$
	$x \leftarrow \mathsf{PFEExt}(1^\lambda, msg_1)$
$msg_2 \leftarrow \mathsf{PFE}_2(1^\lambda, msg_1, C)$	$msg_2 \leftarrow \mathsf{PFEFake}_2(1^\lambda, msg_1, C(x))$
$b \leftarrow \mathcal{A}(msg_2, st)$	$b \leftarrow \mathcal{A}(msg_2, st)$

Figure 3.2: $P2$'s privacy experiment.

Definition 3.3.1 (from [61])**.** *A two-move private function evaluation protocol is described by three* PPT *algorithms* $(\mathsf{PFE}_1, \mathsf{PFE}_2, \mathsf{PFE}_3)$ *such that*

- *the algorithm* PFE_1 *is executed by* P_2 *on input* x *and outputs* (msg_1, st_1) *where* msg_1 *is sent to* P_1 *and* st_1 *represents the state of party* P_2*.*
- *the algorithm* PFE_2 *is executed by* P_1 *on input* $C \in \mathcal{C}$ *and outputs* (msg_2, st_2)*, where* msg_2 *is sent to* P_2*.*
- *the algorithm* PFE_3 *is executed by* P_2 *on inputs* msg_2, st_1 *and outputs a quantity* msg_3*.*

In addition, for the construction, we require the following security properties.

- **Perfect correctness**. With probability 1, we have that $msg_3 = C(x)$.
- P_2**'s privacy**. The party P_1 cannot distinguish between two different P_2's incoming messages. More formally, for a random bit $b \leftarrow \{0,1\}$, for any PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A},P_2}(1^\lambda) := |\Pr[b = b'] - 1/2|$ in the following game is negligible
  - $(x_0, x_1, st_1) \leftarrow \mathcal{A}(1^\lambda)$
  - $(msg, st_2) \leftarrow \mathsf{PFE}_1(x_b)$
  - $b' \leftarrow \mathcal{A}(msg_2, st)$
- P_1**'s privacy.** If P_1 knows x (simulated by an extraction algorithm PFEExt.), then instead of applying PFE_2 with circuit C, it can compute $C(x)$ and uses it in the protocol via some function $\mathsf{PFEFake}_2$. More formally, suppose there exists a PPT algorithm PFEExt that extracts $P_2's$ input x[6] on every message msg_1, then there exists a PPT algorithm $\mathsf{PFEFake}_2$ such that the following holds: for any adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A},P_2}(1^\lambda) := |\Pr[b = b'] - 1/2|$ in Fig. 3.2 is negligible.

[6] i.e. returns x with non-negligble probability.

- $\mathsf{IFE.Setup}(1^\lambda)$: Output $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- $\mathsf{IFE.IKeyGen}(\mathsf{MskOw}(\mathsf{msk}), \mathsf{FuncOw}(\mathsf{mpk}, f))$ is described in Fig. 3.4
- $\mathsf{IFE.Enc} = \mathsf{FE.Enc}$
- $\mathsf{IFE.Dec} = \mathsf{FE.Dec}$

Figure 3.3: Generic blind IFE from PFE

We will consider in the following *deterministic* circuits. The condition simplifies our security definition for PFE in this setting. Indeed, all that the party learns from the execution of the protocol is essentially implied by the output itself (in the deterministic case) and as noted in [67, Sec.7.2.2], it suffices to consider the views of the parties separately. Otherwise in the randomized case, since the output could be a random variable, one has to take into account the joint distribution of party's output (We refer to [67] for a more detailed discussion). While it seems as a strong restriction, we argue that if we consider PPT algorithms represented by probabilistic poly-sized circuit, then a classical result [14] (Adleman's theorem) on complexity theory states that it is possible to *derandomize* these circuits, in a sense that it is possible to consider poly-sized deterministic circuits that represent them. Of course, when derandomizing, additional cares should be taken and verification about the proper output distribution of the computation for each step need to be verified. From that observations, we consider in the following PFE for deterministic circuits.

3.3.2 The scheme

We suppose that FE.KeyGen is a *deterministic* algorithm that is described by a circuit of depth $d(\lambda)$. Let $\lambda > 0$ be a security parameter and consider a family of functions $F = \{F_\lambda\}_\lambda$ whose input size $n(\lambda)$ which is polynomial in λ. Suppose that all functions $f \in F$ can be encoded as a $P(\lambda)$-bit string (for a polynomial P). Consider a FE scheme for this family F. Finally, consider a two move PFE computing the FE.KeyGen algorithm. Our interactive blind functional encryption for the class of function F is described in Fig 3.3 and 3.4.

We have the following theorem.

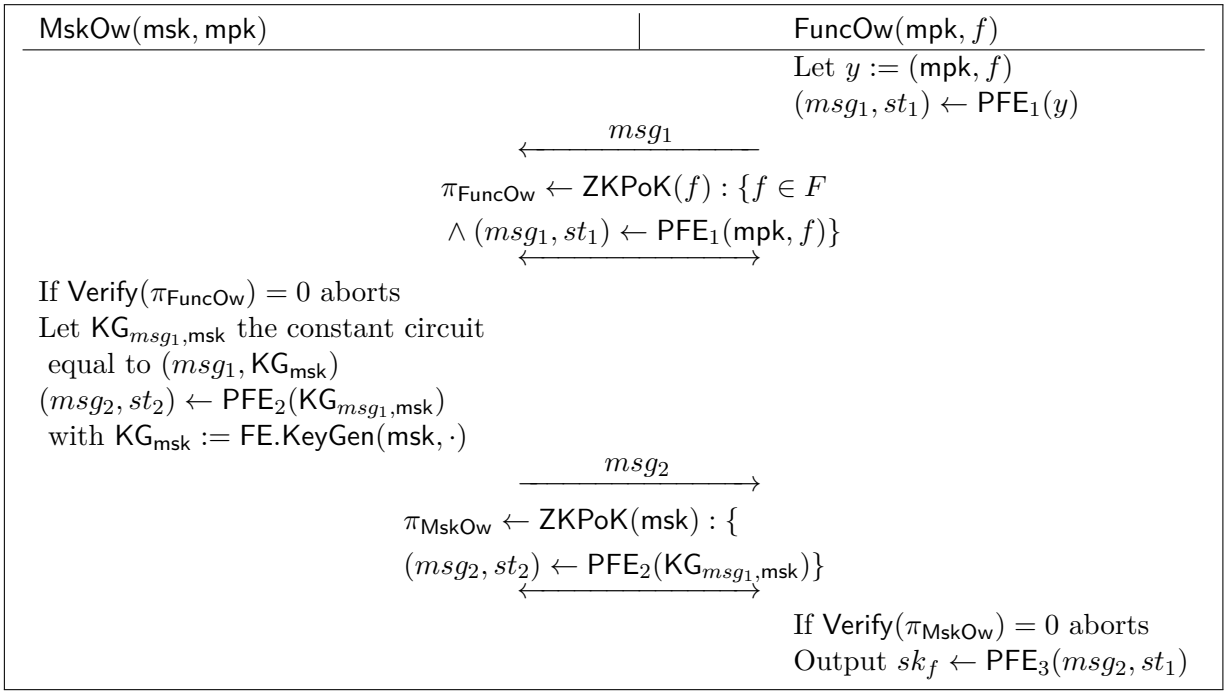

Figure 3.4: Interactive Key Generation IFE.IKeyGen.

Theorem 3.3.1. *The* IFE *scheme described in Fig. 3.3 is a blind* IFE *if* PFE *is a secure private function evaluation as in def. 3.3.1 and proofs* π_{FuncOw} *and* π_{MskOw} *are zero-knowledge proofs of knowledge.*

We begin with the message-privacy.

Proposition 2. *The* IFE *scheme described in Fig. 3.3 is message-private if* π_{FuncOw} *and* π_{MskOw} *are zero-knowledge proofs of knowledge and* PFE *is secure as in def. 3.3.1.*

Proof. We prove that the IFE.KeyGen protocol is *leak-free* with respect to this FE.KeyGen and by Prop. 3.1.1, it implies that the IFE is message-private. From the leak-freeness definition 3.1.3, we have to show that for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that no distinguisher $\mathcal{D}$ can distinguish between the GameReal experiment (where $\mathcal{A}$ is allowed to interact with an honest $\mathcal{AUT}$) and the GameIdeal experiment (where $\mathcal{S}$ interacts with a Trivial.IKeyGen).

Informally, to achieve this property, we first consider any adversary $\mathcal{A}$ interacting as in the GameReal experiment. Then we describe the ideal simulator $\mathcal{S}$ in GameIdeal using the information obtained from $\mathcal{A}$. Considering any distinguisher $\mathcal{D}$ playing the two above games, we have to show that it has a negligible advantage to distinguish between

both experiments. We fix an adversary $\mathcal{A}$ that will interact with an honest authority $\mathcal{AUT}$ and consider a potential distinguisher $\mathcal{D}$.

Recall that for any adversary $\mathcal{A}$ (that interacts with an honest MskOw denoted by $\mathcal{O}$), we need to build a simulator $\mathcal{S}$ (which has access to the $\mathsf{FE.Trivial.KeyGen}$ oracle) that simulates the view of $\mathcal{A}$. Intuitively, we use the extractor simulator of the$\mathsf{ZKPoK}$and its rewinding capability in order to extract the corresponding function. Then, we use the $\mathsf{FE.Trivial.KeyGen}$ oracle and generate a valid functional key for this extracted value. Finally, instead of computing the second message of the PFE protocol honestly, we exploit the $\mathsf{PFEFake}_2$ algorithm. By the security property of the PFE protocol (i.e P_1's privacy) this modification does not affect the view of the adversary. More formally, for any adversary $\mathcal{A}$, we describe the ideal simulator $\mathcal{S}$ in $\mathsf{GameIdeal}$ as follows.

- The simulator $\mathcal{S}$ has the capability of rewinding an instance of the adversary $\mathcal{A}$ that he runs internally. In order to achieve this, $\mathcal{S}$ simulates the communication between any possible PPT algorithm $\mathcal{D}$ and $\mathcal{A}$ by passing $\mathcal{D}$'s input to $\mathcal{A}$ and $\mathcal{A}$'s output to $\mathcal{D}$.

- By definition of $\mathsf{GameReal}$, the adversary $\mathcal{A}$ chooses a function $f \in F$ and runs the $\mathsf{IFE.IKeyGen}(\mathcal{O}(\mathsf{msk}), \cdot)$ protocol with an honest MskOw. In the first message of the protocol, the adversary runs $(msg_1, st_1) \leftarrow \mathsf{PFE}_1(\mathsf{mpk}, f)$ and sends msg_1 to the honest $\mathcal{O}(\mathsf{msk})$. $\mathcal{A}$ must generate a valid proof ZKPoK $\pi_{\mathcal{A}}$ corresponding to the correct evaluation of the PFE_1 algorithm.

- After this interaction, $\mathcal{S}$ can check the validity of the proof and if there is a failure, then could abort. $\mathcal{S}$ needs to produce a second message msg_2 corresponding to PFE_2. Recall that $\mathcal{S}$ does not know msk but has an access to an ideal $\mathsf{Trivial.IKeyGen}(\mathsf{msk}, \cdot; \cdot)$ oracle which produces on input f a functional key $sk_f := \mathsf{FE.KeyGen}(\mathsf{msk}, f)$. As mentioned in a previous remark Rem.3.1.5, $\mathcal{S}$ needs to extract the function f in order to obtain a corresponding functional keys sk_f. This is possible in this situation thanks to the extractability condition of the ZKPoK. Indeed, there exists an extractor Ext that $\mathcal{S}$ can use, such that for $x := msg_1$, Ext returns a valid witness $w_{\mathcal{A}} := f$ with all but negligible probability. $\mathcal{S}$ is now capable of calling the $\mathsf{Trivial.IKeyGen}(\mathsf{msk}, \cdot; \cdot)$ oracle on input f to obtain sk_f.

- The next step is to build a consistent msg_2. Consider the $\mathsf{PFEFake}_2$ algorithm

from def. 3.3.1. $\mathcal{S}$ can use it (instead of PFE_2) and compute

$$msg_2 := \mathsf{PFEFake}_2(1^\lambda, msg_2, sk_f).$$

- Finally, $\mathcal{S}$ needs to generate two valid views corresponding to the ZKPoK, i.e. $\pi_{\mathcal{A}}, \pi_{\mathsf{MskOw}}$ without having access to the master secret key msk. Thanks to the rewinding capability of $\mathcal{S}$ and the zero-knowledge property of the ZKPoK, there exists two probabilistic simulators $\mathcal{S}im, \mathcal{S}im'$ that $\mathcal{S}$ can use in order to simulate the view of eachZKPoKinteraction between $\mathcal{A}$ and an honest MskOw in the GameReal experiment for any input. In particular, $\mathcal{S}im(msg_1)$ is indistinguishable from $\mathsf{View}_{\mathcal{A}}^{\mathsf{MskOw}}(msg_1)$ for $\pi_{\mathcal{A}}$ and $\mathcal{S}im'(msg_2)$ is indistinguishable from $\mathsf{View}_{\mathcal{A}}^{\mathsf{MskOw}}(msg_2)$ for π_{MskOw}. Finally, the simulator $\mathcal{S}$ returns

$$(msg_1, msg_2, \mathcal{S}im(msg_1), \mathcal{S}im'(msg_2)).$$

Analysis. We argue that our simulator $\mathcal{S}$ correctly simulates the view of any adversary $\mathcal{A}$, hence any $\mathcal{D}$ which plays the leak-freeness game cannot distinguish between GameReal and GameIdeal (leading to a contradiction). Assume that such a distinguisher $\mathcal{D}$ exists with non negligible advantage in distinguishing between these two games. The extractability property of ZKPoK and the rewinding capability of $\mathcal{S}$ to extract the elements f permits to use the algorithm $\mathsf{PFEFake}_2$ (see definition 3.3.1).

In addition, the ideal Trivial.IKeyGen(msk, $\cdot;\cdot$) oracle produces a valid functional key sk_f. The Zero-Knowledge property of the proofs ensures that $\mathcal{S}im$ and $\mathcal{S}im'$ produce consistent views. We next analyse the distribution of msg_2 produced by $\mathcal{S}$.

First notice that msg_2 is not produced as in the description of the scheme. In particular, the simulator used the algorithm $\mathsf{PFEFake}_2$. Recall however, that the PFE scheme verifies the P_2's privacy property in def. 3.3.1, so by definition, the adversary does not notice the difference when receiving the value msg_2 from an honest evaluation. Hence, $\mathcal{D}$ has a negligible advantage of distinguishing GameReal and GameIdeal.

We deduce that The IFE.IKeyGen does not leak any additional information than FE.Trivial.KeyGen and we conclude that the protocol is leak-free with respect to FE.KeyGen algorithm. Assuming the above properties ofZKPoKand the PFE scheme, we deduce that the IFE scheme in 3.3 is message private by Prop. 3.1.1. □

Next, we propose a proof that our scheme verifies the blindness property.

Proposition 3. *The* IFE *scheme described in Fig. 3.3 is blind if* π_{FuncOw} *and* π_{MskOw} *are zero-knowledge proofs of knowledge and* PFE *is secure as in def. 3.3.1.*

Proof. Suppose having an adversary $\mathcal{A}$ attacking the blindness game. Recall that it chooses the public parameters $(\mathsf{mpk}, \mathsf{msk})$ and two functions f_0 and f_1 and runs two sequential interactions with honest user $\mathsf{FuncOw}(\mathsf{mpk}, f_b)$ and $\mathsf{FuncOw}(\mathsf{mpk}, f_{1-b})$ respectively where b is a random bit. At the end of the interactions, if not defined, $\mathcal{A}$ received the two functional keys (sk_{f_0}, sk_{f_1}) or $(\perp, \perp)$ corresponding to (f_0, f_1). The goal for $\mathcal{A}$ is to find the bit b with non-negligible probability. We prove the blindness property via a sequence of games. The proof will use the following path.

- We invoke the extractor simulator in order to extract the witness from the ZKPoK of msk given by the adversary.
- We will use this key msk to generate a functional key sk_f locally and answer the adversary with this keys (instead of sending the first message using PFE_1).
- We modify the first message concerning f and the proof such that the transcript is independent of f.

We note $\bar{b} := 1 - b$.

Game 0. This is the original game as in Def. 3.1.7. We give more details about each interaction in Fig. 3.5. We describe the interaction of the adversary with each oracle user FuncOw_b and $\mathsf{FuncOw}_{\bar{b}}$. Lines 1,6,10-11 describe the behaviour of $\mathcal{A}$ during the blindness game and the remaining lines the users behaviour.

Game 1. We modify Game 0 in the following sense. In this game, thanks to the ZKPoK, we know that there exists extractors $\mathsf{Ext}_{\mathsf{b}}$ and $\mathsf{Ext}_{\bar{\mathsf{b}}}$ that can extract the witnesses from π'_b and $\pi'_{\bar{b}}$ (the second proofs in line 7) providing $w^*_b = \mathsf{msk}^*_b$ for each bit $b \in \{0, 1\}$. We add the following quantities for each user in line 8

$$w^*_b := \mathsf{msk}^*_b \quad w^*_{\bar{b}} := \mathsf{msk}^*_{\bar{b}}.$$

Game 2. We modify the Game 1 as follows. If the master secret keys does not match, i.e $\mathsf{msk}^*_b \neq \mathsf{msk}^*_{\bar{b}}$, the user oracles in the two interactions aborts and we set $(sk_{f_0}, sk_{f_1}) = (\perp, \perp)$. Otherwise, we set $\mathsf{msk} := \mathsf{msk}^*_b$ and instead of decrypting executing PFE_3 in line 9, we exploit the extracted value msk and the $\mathsf{FE.KeyGen}(\mathsf{msk}, \cdot)$ algorithm on input f_b

Game 0	
1. $(\mathsf{mpk}, f_0, f_1) \leftarrow \mathcal{A}^{\mathsf{Setup}(\cdot)}(1\lambda)$	
2. $(msg_{1,b}, st_{1,b}) \leftarrow \mathsf{PFE}_1(\mathsf{mpk}, f_b)$ 3. $w_b := f_b$ 4. $m_b := msg_{1,b}$ 5. $\pi_b \leftarrow \mathsf{ZKPoK}(\mathcal{O}(w_b), \mathcal{A}(m_b))$	$(msg_{1,\bar{b}}, st_{1,\bar{b}}) \leftarrow \mathsf{PFE}_1(\mathsf{mpk}, f_{\bar{b}})$ $w_{\bar{b}} := f_{\bar{b}}$ $m_{\bar{b}} := msg_{1,\bar{b}}$ $\pi_{\bar{b}} \leftarrow \mathsf{ZKPoK}(\mathcal{O}(w_{\bar{b}}), \mathcal{A}(m_{\bar{b}}))$
6. $(msg_{2,b}, msg_{2,\bar{b}}) \leftarrow \mathcal{A}((\pi_b, m_b), (\pi_{\bar{b}}, m_{\bar{b}}))$	
7. $\pi'_b \leftarrow \mathsf{ZKPoK}(\mathcal{A}(w'_b), \mathcal{O}(msg_{2,b}))$ 8. If $\mathsf{Verify}(\pi'_b) = 1$ 9. $sk_{f_b} \leftarrow \mathsf{PFE}_3(msg_{2,b}, st_{1,b})$ else $sk_{f_b} \leftarrow \perp$	$\pi'_{\bar{b}} \leftarrow \mathsf{ZKPoK}(\mathcal{A}(w'_{\bar{b}}), \mathcal{O}(msg_{2,\bar{b}}))$ If $\mathsf{Verify}(\pi'_{\bar{b}}) = 1$ $sk_{f_{\bar{b}}} \leftarrow \mathsf{PFE}_3(msg_{2,\bar{b}}, st_{1,\bar{b}})$ else $sk_{f_{\bar{b}}} \leftarrow \perp$
10 $b' \leftarrow \mathcal{A}(sk_{f_0}, sk_{f_1})$ 11. returns 1 iff $b' = b$	

Figure 3.5: Blindness experiment.

(resp. $(f_{\bar{b}})$ to obtain valid functional key(s). We replace line 9 by the new line (depending on the bit b)

$$sk_{f_b} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f_b), \quad sk_{f_{\bar{b}}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f_{\bar{b}}).$$

If the proof does not fail, the oracles return (locally) sk_{f_b} (resp. $sk_{f_{\bar{b}}}$). Otherwise, they returns $(sk_{f_0}, sk_{f_1}) = (\perp, \perp)$. Finally, we give as in line 10. (sk_{f_0}, sk_{f_1}) to $\mathcal{A}$.

Game 3. We change the behaviour of user FuncOw_1 while maintaining unchanged the one of FuncOw_0. Consider the zero function $\mathbf{0}$ (equal to zero in all point) with size description equals to f_1 with a modified proof π_1 in the first message. In more details, there exists a zero-knowledge simulator $\mathsf{Sim_1}$ for π that can simulate the proof of knowledge without knowing the underlying witness. We replace the corresponding term in line 3. for FuncOw_1 with

$$msg_{1,1} \leftarrow \mathsf{PFE}_1(\mathsf{mpk}, \mathbf{0}).$$

Next, we simulate the corresponding term line 5. with

$$\pi_1^* \leftarrow \mathsf{Sim_1}(\pi_1).$$

In addition, there exists a simulator Sim'_1 for π'_1 such that the line 7. becomes

$$\pi_1^{**} \leftarrow \mathsf{Sim}'_1(\pi'_1).$$

Game 4 We change the behaviour of FuncOw_0 as in the previous Game 3 by considering the zero function $\mathbf{0}$ (equal to zero in all points) with size description equals to f_0 with a modified proof π_0 in the first message. In more details, there exists a zero-knowledge simulator Sim_0 for π that can simulates the proof of knowledge without knowing the underlying witness. We replace the corresponding term in line 2. for FuncOw_0 with

$$msg_{2,0} \leftarrow \mathsf{PFE}_1(\mathsf{mpk}, \mathbf{0}).$$

Next, we simulate the corresponding term in line 5. with

$$\pi_0^* \leftarrow \mathsf{Sim}_0(\pi_0).$$

In addition, there exists a simulator Sim_0' for π_0' such that the line 7. becomes

$$\pi_0^{**} \leftarrow \mathsf{Sim}_0'(\pi_0').$$

The proof of the blindness property is a corollary of the following lemma.

Lemma 3.3.2. *Assuming that the proofs π' are proofs of knowledge, then we have that Game 0 is indistinguishable from Game 1.*

Proof. The matching condition prevents the adversary to use two different master secret keys. Thanks to the extractability condition, the rewinding techniques of the ZKPoK, it is possible to efficiently extract the corresponding witness, and for the adversary, the success probability remains the same (except with negligible probability). So, Game 0 is indistinguishable from Game 1. □

Lemma 3.3.3. *Assuming that the* PFE *scheme is correct and the underlying* FE *is correct, then Game 1 is indistinguishable from Game 2.*

Proof. Since the PFE scheme is perfectly correct, applying the following algorithm $\mathsf{PFE}_3(msg_{2,b})$ or generating directly the functional key with $\mathsf{FE.KeyGen}(\mathsf{msk}, f_b), b \in \{0,1\}$ yields to the same result. Notice that theZKPoKprevents the adversary from using another circuit. We can deduce that Game 1 is indistinguishable from Game 2. □

Lemma 3.3.4. *Assuming that the* PFE *scheme is secure and the proofs π, π' are zero-knowledge, then Game 2 is indistinguishable from Game 3.*

Proof. The adversary cannot guess whether the message received corresponds to the the function $\mathbf{0}$ thanks to the privacy of user 1 (P_2's privacy in def. 3.3.1). Also, the view

of $\mathcal{A}$ is correctly distributed. Indeed, suppose there is an attacker that can distinguish between Game 2 and Game 3 with non-negligible advantage, then we show how to build an adversary $\mathcal{B}$ that breaks the P_2's privacy in the PFE security game as in Def. 3.3.1. $\mathcal{B}$ has the following behaviour. It runs $\mathcal{A}$ in order to get f_0, f_1, mpk and uses them for the PFE security game by choosing the messages $(\mathbf{0}, f_1)$. It forms $msg_{1,0} := \mathsf{PFE}_1(\mathsf{mpk}, f_0)$ and receives $msg^\star$ corresponding to the execution of PFE_1 over one of the two functions $\{\mathbf{0}, f_1\}$ from the PFE challenger.

Then, it uses $\mathcal{A}$ in the following way by interacting as a legitimate user. It simulates the first messages of FuncOw_0 with $(m_0 := msg_{1,0})$ and FuncOw_1 with $(m_1 := msg^\star)$. Up to this point, it could use the zero knowledge property to simulate the corresponding proofs π_1^* and π_1^{**}. Finally, $\mathcal{B}$ returns the same output of $\mathcal{A}$ (the same bit).

Now, returning to the user's privacy security game, if $msg^\star$ corresponds to the function f_1, then this situation corresponds to Game 2 experiment. Otherwise, $msg^\star$ represents the function $\mathbf{0}$ then it corresponds to Game 3 by construction.

Unless the proofs are not zero-knowledge, the advantage of $\mathcal{B}$ for breaking P_2's privacy in the PFE scheme is the same as the advantage of $\mathcal{A}$ in distinguishing between Game 2 and Game 3. We deduce that Game 2 is indistinguishable from Game 3.

□

Lemma 3.3.5. *Assuming that the* PFE *scheme is secure and the proofs π, π' are zero-knowledge, then Game 3 is indistinguishable from Game 4.*

Proof. This is similar to the previous lemma 3.3.4. Suppose there is an attacker that can distinguish between Game 3 and Game 4 with non-negligible advantage, then we show how to build an adversary $\mathcal{B}$ that breaks the $P2's$ security of the PFE scheme. $\mathcal{B}$ has the following behaviour. $\mathcal{B}$ runs $\mathcal{A}$ in order to get f_0, f_1, mpk and uses them for the PFE security game by choosing the messages $(f_0, \mathbf{0})$. It forms $msg_{1,1} := \mathsf{PFE}_1(\mathsf{mpk}, f_0)$ and receives $msg^\star$ corresponding to the execution of PFE_1 over one of the two functions $\{f_0, \mathbf{0}\}$ from the PFE challenger.

Then, it uses $\mathcal{A}$ in the following way by interacting as a legitimate user. It simulates the first messages of FuncOw_0 with $(m_0 := msg^\star)$ and FuncOw_1 with $(m_1 := msg_{1,1})$. Up to this point, it could use the zero knowledge property to simulate the corresponding proofs π_0^* and π_0^{**}. Finally, $\mathcal{B}$ returns the same output of $\mathcal{A}$ (the same bit).

Now, returning to P_2's privacy security game, if $msg^\star$ corresponds to the function f_0, then this situation corresponds to Game 3 experiment. Otherwise, $msg^\star$ represents the function $\mathbf{0}$ then it corresponds to Game 3 by construction. Unless the proofs are not zero-knowledge, the advantage of $\mathcal{B}$ breaking $P_2's$ privacy in the PFE scheme is the same

as the advantage of $\mathcal{A}$ in distinguishing between Game 2 and Game 3. We deduce that Game 2 is indistinguishable from Game 3. □

We conclude that Game 0 is indistinguishable from Game 4. In Game 4, the view of $\mathcal{A}$ is independent of b: the functional keys $sk_{f_b}, sk_{f_{\bar{b}}}$ do not depend on the values sent by $\mathcal{A}$ by construction. Thus, the probability of guessing the bit b is exactly $1/2$. Hence, by combining all the above lemma, we conclude that this scheme satisfies the blindness property.

This ends the proof of the proposition. □

3.3.3 Using FHE: a special case

As it was already mentioned, in our published work [34], we used the power of fully homomorphic encryption FHE with adapted ZKPoK in order to obtain a blind IFE from any FE. We will see that this construction could be considered in the language of PFE. We will recall the result from our paper [34] in the following.

Our approach starts from an existing FE scheme for a class of function F and upgrades it to a blind IFE scheme from the same class F, by only modifying the KeyGen algorithm. FuncOw starts by *encrypting* an encoded version of some function f with a *fully homomorphic scheme* [62] under her own key and sends the ciphertext C_f to MskOw. With msk, the party MskOw homomorphically evaluates the circuit $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ using the FHE.Eval algorithm on C_f, then sends back a ciphertext C_{sk_f} of the corresponding functional key sk_f. FuncOw can now decrypt with her (FHE) secret key the received ciphertext and recover sk_f. Thereby, the FHE *blinds* to MskOw both the function f and the key sk_f.In addition, we used ZKPoK as in the PFE construction.

In the langage of PFE, we have the following description that we will present for completeness. Let $\lambda > 0$ be a security parameter and consider a family of functions $F = \{F_\lambda\}_\lambda$ whose input size $n(\lambda)$ which is polynomial in λ. Suppose that all functions $f \in F$ can be encoded as a $P(\lambda)$-bit string (for a polynomial P). Consider a FE scheme for this family F. We can suppose in our case that FE.KeyGen is a randomized or deterministic as in the PFE construction which was discussed in the last paragraph of Sec.3.3.1. algorithm that is described by a circuit of logarithmic depth $d(\lambda)$. Consider (a possibly size-dependent) $\mathsf{FHE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a fully homomorphic encryption scheme, where the input of the encryption algorithm is a bit string with size at least $P(\lambda)$ and supports evaluation of circuits of depth at least $d(\lambda)$. Our interactive blind functional encryption for the class of function F is described in Figures 3.6 and 3.7.

- $\mathsf{IFE.Setup}(1^\lambda)$: Output

$$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda).$$

- $\mathsf{IFE.IKeyGen}(\mathsf{MskOw}(\mathsf{msk}), \mathsf{FuncOw}(\mathsf{mpk}, f))$ is described in Fig. 3.7
- $\mathsf{IFE.Enc} = \mathsf{FE.Enc}$
- $\mathsf{IFE.Dec} = \mathsf{FE.Dec}$

Figure 3.6: Our generic blind IFE from FHE

$\mathsf{MskOw}(\mathsf{msk}, \mathsf{mpk})$		$\mathsf{FuncOw}(\mathsf{mpk}, f)$
		$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$
		$ct_f \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, f; R)$
	$\mathsf{pk}, ct_f \longleftarrow$	
	$\pi_{\mathsf{FuncOw}} \leftarrow \mathsf{ZKPoK}(\mathsf{sk}, f, R) : \{f \in F$	
	$\wedge\ (\mathsf{sk}, \mathsf{pk}) = \mathsf{FHE.Setup}(1^\lambda)$	
	$\wedge\ ct_f = \mathsf{FHE.Enc}(\mathsf{pk}, f; R)\}$	
	$\longrightarrow$	
If $\mathsf{Verify}(\pi_{\mathsf{FuncOw}}) = 0$ aborts		
Select random R', R''		
$ct_{sk_f} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, \mathsf{KG}_{\mathsf{msk},R'}, ct_f; R'')$		
with $\mathsf{KG}_{\mathsf{msk},R'} := \mathsf{FE.KeyGen}(\cdot, \mathsf{msk}; R')$		
	$ct_{sk_f} \longrightarrow$	
	$\pi_{\mathsf{MskOw}} \leftarrow \mathsf{ZKPoK}(\mathsf{msk}, R', R'') : \{$	
	$ct_{sk_f} = \mathsf{FHE.Eval}(\mathsf{pk}, \mathsf{KG}_{\mathsf{msk},R'}, ct_f; R'')\}$	
	$\longleftarrow$	
		If $\mathsf{Verify}(\pi_{\mathsf{MskOw}}) = 0$ aborts
		Output $sk_f \leftarrow \mathsf{FHE.Dec}(\mathsf{sk}, ct_{sk_f})$

Figure 3.7: Our interactive key generation IFE.IKeyGen from FHE

As we can see this construction is a special case of generic construction from any PFE. Informally, in the language of PFE, PFE_1 consists of running the FHE.Setup algorithm, then encrypt the function f using FHE.Enc. In addition, PFE_2 consists of homomorphically evaluating the circuit KG. (equal to $\mathsf{FE.KeyGen}(\cdot)$). Finally, PFE_3 is the decryption algorithm FHE.Dec.

In fact, our generic proof for the PFE case is inspired from our previous work for the FHE case. P_1's privacy in the definition 3.3.1 thanks is by definition the analogue of CPA security of FHE scheme and P_2's privacy is similar to the extractability condition of the ZKPoK in combination with an specific property of weak-function indistinguishability [18] (for FHE) that is also considered in our paper [34]. [7]

From two-party computation protocol. A possible approach to derive generically an interactive FE would be to use a secure two-party computation of the IKeyGen protocol. We insist that such an approach *does not* achieve the blindness property we are interested in. Indeed, although the authority does not learn the user's input with 2PC, it could make the functional keys output by two users in the blindness security game depend on the function in different ways. This is possible by using for example two different master keys.

Our notion of PFE is inspired by the work that was introduced in the context of secure two-move blind signature [61]. Generically, it is a two-party protocol with some additional properties that are sufficient in order to obtain a blind signature scheme. It seems possible, similarly to [61] (see also [18]), to use Yao's Garbled circuit [115] combined with any two-messages oblivious transfer [92] (which leads to a semi-honest 2PC), in addition to ZKPoK, to finally get the properties needed to instantiate a two-round secure PFE.

3.3.4 Efficient Blind Interactive Inner-Product FE

Many applications, such as data mining or statistical computation need as subroutines inner-product evaluations. We propose in this section a blind interactive functional encryption for *inner-product*, which is inspired by our construction from the fully homomorphic case. Notice that, again, generically, it is possible to obtain general PFE or FHE-based protocol that computes the inner-product with adapted ZKPoK.

[7] Informally, this notion guarantees that any adversary (even if it knows the secret key of the FHE) cannot produce any FHE ciphertext ct_y, for input y, two circuits C_0, C_1 with $C_0(y) = C_1(y)$ such that it could distinguish between $\mathsf{FHE.Eval}(\mathsf{pk}, C_0, \mathsf{ct}_y)$ and $\mathsf{FHE.Eval}(\mathsf{pk}, C_1, \mathsf{ct}_y)$ (we refer to [18] for a definition).

For most of the known IPFEs, we show that it is possible to provide an instantiation using the power of linearely homomorphic encryption scheme [37]. Indeed, the structure of the underlying KeyGen for IPFE is a linear function. Thus, it is natural to choose this primitive and which will be sufficient for our purpose as we will describe in the following paragraphs.

On IPFE's.KeyGen. Several IPFE constructions have recently been proposed [1, 11, 38, 111]. Most of these schemes extract functional keys following the same shape: $(\mathbf{y}, \langle \mathsf{msk}, \mathbf{y} \rangle)$ where $\mathsf{msk}, \mathbf{y} \in \mathbb{Z}_p^\ell$ for a (large) prime p, the master secret key msk and the functional key represented by the vector $\mathbf{y}$.

In fact, the choice of p depends on the scheme. Before, remark that the classical IPFE functionality computes inner-products modulo some prime q, while the functional key, which is in these schemes also an inner-product, is considered eventually modulo a different prime p. The ElGamal-based schemes of [1,11,111] consider msk, $\mathbf{y}$ as well as the functional keys for the modulo the same prime integer q used for the functionality, i.e. $p = q$.

Moreover, there is some schemes [11,38] that computes the functional keys as inner-product *over the integers* (which is important for security). This means that we have to consider the size of the eventual inner-product. Then, we can always choose a large prime p bigger than maximum size of the functional keys and view them modulo p. In particular, this prime p is eventually different from q the module of the functionality.

Our contribution is to give an efficient two-party protocol computing these functional keys with the blindness property, and which can be used in the constructions whose functional key is an inner-product. Following our FHE-based construction, we show how to adapt our construction.

On the choice of the linearly homomorphic encryption scheme. As we mentioned, the functional key is an inner-product between msk and a vector $\mathbf{y}$ that imposes a size on the functional keys. In general, this is a relatively large integer in $\mathbb{Z}$ and the choice of the linearly homomorphic scheme is impacted.

While impossibility results hold for unconditionally secure two party inner-product computation [99], there exist protocols that rely on linear homomorphic encryption schemes to securely compute inner-products as in [65].

A direct option would have been to use the additive variant of ElGamal [53] modulo p, but this would imply to compute a final discrete logarithm which is not possible for large p. The Paillier encryption scheme [98] is also a possibility as an alternative to this

size problem, but we remarked that proofs of knowledge are less efficient for the purpose of our protocol.

Hereafter, we modify the former construction [65] by using the Castagnos-Laguillaumie (CL) linear homomorphic encryption [37] since we need inner-product computed in $\mathbb{Z}_p$. The scheme provided in [37] puts forward a general framework for linearly homomorphic encryption schemes, by considering similarly to the Paillier encryption, a group that has some subgroup where the Discrete Logarithm problem is easy to solve.

The choice of CL scheme has positive consequences: it allows to tune the size of the prime p according to the applications and it is asymptotically more efficient (since best known attacks have a subexponential complexity of parameter 1/2 instead of 1/3 for factoring). We refer to [37] for a more detailed definition of this scheme.

On zero-knowledge proofs. For our consideration, ZKPoK are proofs for classical discrete logarithm-based expressions. We choose to focus on the ElGamal-based IPFE that has the particularity of considering all elements $\mathsf{msk}, \mathbf{y}, \mathsf{sk}_{\mathbf{y}}$ modulo the same prime q. The proofs that concerns the group coming from the IPFE setup could be computed using a standard Schnorr proof [106].

The main subtleties concern the CL part since it uses a group of unknown order. As in [36], the solution is to use repeated GPS proofs [64] with binary challenges for exponents sampled from some bounded set and used as randomness (or private key). We refer to [36] or [112] for a more detailed presentation .

Taking a step back, using Paillier encryption instead of CL prevents the necessity to repeat a GPS proof with binary challenge. It however necessitates to add (i) a proof that the Paillier modulus N has truly been computed as the multiplication of two primes [31], (ii) a proof of knowledge of a plaintext m and its randomness r composing the given Paillier ciphertext $c = (1+n)^m r^N \pmod{N^2}$, which can be done using techniques given in [44] and (iii) a proof that $m < p$ in a group of composite order [27]. We argue that this implies a heavier global zero-knowledge proof than what we propose using CL encryption.

Using the CL scheme, we can then directly embed our protocol into secure DDH-based IPFE schemes like those of [1, 11].

Description of our IFE for known IPFE. For most of the known IPFE scheme [1,11], the Setup algorithm consists of a description of a cyclic group $\mathbb{G}$ of large prime order $p > 2^\lambda$ with generator $g \leftarrow \mathbb{G}$. For each $i \in \{1, \ldots, \ell\}$, it samples $s_i \leftarrow \mathbb{Z}_p$ and compute $h_i = g^{s_i}$. Finally, define the master secret key as $\mathsf{msk} := (s_i)_{i=1}^{\ell}$ and the master public

key as $\mathsf{mpk} := \left(\mathbb{G}, g, \{h_i\}_{i=1}^{\ell}\right)$.

We use the same prime p for both the CL scheme 2.2.2 and IPFE. This is possible thanks to the flexibility of the CL key generation. The interactive key generation $\mathsf{IKeyGen}(\mathsf{MskOw}(\mathsf{s} \in \mathbb{Z}_p^{\ell}), \mathsf{FuncOw}(\mathsf{mpk}, \mathbf{y} \in \mathbb{Z}_p^{\ell}))$, consisting of the two party computation of an inner-product is then as follows.

- The user FuncOw generates a pair of keys $\mathsf{pk} = \mathfrak{g}_p^x$ and $\mathsf{sk} = x$ for the CL scheme over the message space $\mathbb{Z}_p$. It encrypts each coordinate y_i for $i \in \{0, \ldots, \ell\}$ as $c_i = (c_{1,i}, c_{2,i}) = (\mathfrak{g}_p^{r_i}, \mathfrak{f}^{y_i}\mathfrak{h}^{r_i})$ for uniformly random r_i in a set S included in $\mathbb{Z}$ [8]. Let $c_{\mathbf{y}} := (c_i)_{i=1,\ldots,\ell}$, it sends $\mathsf{pk}, c_{\mathbf{y}}$ to MskOw and a zero-knowledge proof of knowledge π_{FuncOw} such that

$$\{(x, r_i, y_i) : \ \mathfrak{h} = \mathfrak{g}_p^x \wedge c_{1,i} = \mathfrak{g}_p^{r_i} \wedge c_{2,i} = \mathfrak{f}^{y_i}\mathfrak{h}^{r_i} \text{ for } i \in \{1, \ldots, \ell\}\}.$$

- If the proof fails, MskOw aborts. Otherwise, it homomorphically computes $c_{sk_y} := (c_{1,sk_y}, c_{2,sk_y}) \longleftarrow \left(\left(\prod_{i=1}^{\ell} c_{1,i}^{s_i}\right)\mathfrak{g}_p^{r'}, \left(\prod_{i=1}^{\ell} c_{2,i}^{s_i}\right)\mathfrak{h}^{r'}\right)$ for some random r' in S that it sends to FuncOw as well as a proof π_{MskOw} that:

$$\{(r', s_i) : \ \{g^{s_i} = h_i\}_{i=1}^{\ell} \wedge c_{sk_y} = \left(\left(\prod_{i=1}^{\ell} c_{1,i}^{s_i}\right)\mathfrak{g}^{r'}, \left(\prod_{i=1}^{\ell} c_{2,i}^{s_i}\right)\mathfrak{h}^{r'}\right)\}.$$

- If π_{MskOw} fails, FuncOw aborts. Otherwise, it decrypts c_{sk_y} and gets $\mathsf{sk}_y := (\mathbf{y}, \langle \mathbf{s}, \mathbf{y} \rangle) \in \mathbb{Z}_p^{\ell} \times \mathbb{Z}_p$.

Description of the proofs. We will discuss in the following how the proofs could be performed. Recall that a Schnorr proof of knowledge $h = g^s$ consists of sending a commitment $T = g^t \pmod p$ for a random $t \in \mathbb{Z}_p^{\star}$, receiving a uniformly random challenge c in a set $[0, B-1]$ then answering $u := t + sc \pmod p$ that should verify $g^u = T \cdot h^c$. This proof may be eventually repeated ℓ times in order to obtain a soundness of $\frac{1}{B^{\ell}}$. For our purpose, we use this proof in π_{MskOw} several times in the first part, one for every i, to prove that $h_i = g^{s_i}$.

The GPS proof [64] consists of a proof of knowledge $h = g^s$ for an $s \in [0, S-1]$ and g is a generator of group of unknown order. Then, the proof starts with a commitment $T = g^t$ for a uniformly random $t \in [0, B-1]$. For a binary challenge $c \in \{0, 1\}$, the

[8] See [112] for a more detailed justification

answer consists of $u := t + cs \in \mathbb{Z}$ as in Schnorr. The equation $g^u = T \cdot h^c$ should also hold as well as an additional check that $u \in [0, B + S]$.

The resulted ZKPoK provides soundness of $2^{-\ell}$ as long as ℓ is polynomial and $\ell S/B$ is negligible. For our purpose, we use this proof in π_{FuncOw} for proving that $\mathfrak{h} = \mathfrak{g}_p^x$ (for π_{FuncOw})

For the encryption part concerning the group of unknown order (in both π_{MskOw} and π_{FuncOw}), we have as a high-level to prove that $(c_1 = \mathfrak{g}_p^r, c_2 = \mathfrak{f}^m \mathfrak{h}^r)$ is indeed an encryption of some message $m \in \mathbb{Z}_p$ under the public key $\mathfrak{h} = \mathfrak{g}_p^x$ for $x \in [0, S]$, we consider an adaptation of GPS proof. Let as a commitment an encryption of a random element $r_2 \in \mathbb{Z}_p$ using randomness $r_1 \in [0, B - 1]$, i.e. computes $t_1 := \mathfrak{g}_p^{r_1}, t_2 = \mathfrak{f}^{r_2} \mathfrak{h}^{r_1}$. For a binary challenge $c \in \{0, 1\}$, compute $u_1 := r_1 + cr \in \mathbb{Z}, u_2 := r_2 + cx \in \mathbb{Z}_p$. The verification consists of checking if $u_1 \in [0, B + S]$ and $t_2 \cdot c_2^c = \mathfrak{h}^{u_1} \mathfrak{f}^{u_2}$ as well as $t_1 \cdot c_1^c = \mathfrak{g}_p^{u_1}$. Here, we have similarly a soundness $2^{-\ell}$ as long as ℓ is polynomial and $\ell S/B$ is negligible.

Concrete efficiency. A precise efficiency analysis of GPS-like proof in the context of CL encryption has been performed in [36] or [112]. It is implemented within class groups of some imaginary quadratic fields. The cost of such a proof is dominated by the computation of exponentiations in the class group. [36, Fig. 9] gives some measurements: on their architecture, an exponentiation takes 55ms for a 128 bit security. For the proof described in Eq. 3.3.4 with $\ell = 1$, there are essentially 4 exponentiations in the class groups. This protocol has to be repeated say 40 times to get a soundness error of 2^{-40}, which means that such a proof costs less than 10 second (with $\ell = 1$). The overall cost is then linear in ℓ, which means that our interactive blind IFE has a reasonable practical cost of ℓ times tens of seconds. This is even more reasonable that this extraction is done only each time that a functional key is necessary, which most of the time happens once for all.

Analysis of Our Inner Product IFE. The following result is a corollary our previous result and a proof is given next.

Theorem 3.3.6. *The scheme described above is message-private and blind assuming that* CL *scheme is CPA-secure and the* $\pi_{\mathsf{FuncOw}}, \pi_{\mathsf{MskOw}}$ *are zero-knowledge proofs of knowledge.*

This theorem is a corollary of our previous construction. The only difference with our FHE-based construction [34] is that we suppose that the FHE is weak-function indis-

tinguishable. And this notion was used in the leak-freeness security game. The blindness proof can easily be adapted since it is the same as in our construction for FHE and we will omit to describe it.

Proof. We prove that the IFE.KeyGen protocol is **leak-free** with respect to this IPFE.KeyGen. By Prop. 3.1.1, it implies that the IFE scheme is message-private.

More formally, for any adversary $\mathcal{A}$, we describe the ideal simulator $\mathcal{S}$ in GameIdeal as follows.

- First in the leak freeness security game, there is a distinguisher $\mathcal{D}$ that interacts with $\mathcal{A}$. Note that the simulator $\mathcal{S}$ has the capability of rewinding an instance of the adversary $\mathcal{A}$ that he runs internally. In order to achieve this, $\mathcal{S}$ simulates the communication between $\mathcal{D}$ and $\mathcal{A}$ by passing $\mathcal{D}$'s input to $\mathcal{A}$ and $\mathcal{A}$'s output to $\mathcal{D}$.

- By definition of GameReal, the adversary $\mathcal{A}$ chooses a vector $\mathbf{y} \in \mathbb{Z}_p^\ell$ and runs the IFE.IKeyGen$(\mathcal{O}(\mathsf{msk}), \cdot)$ protocol with an honest MskOw. In the first message of the protocol, the adversary runs $(\mathsf{pk} := \mathfrak{g}_p^x, \mathsf{sk} := x) \leftarrow \mathsf{CL.Setup}(1^\lambda)$ for a certain x and must send to the honest oracle representing MskOw, and denoted by $\mathcal{O}(\mathsf{msk})$ the parameters for the CL scheme, i.e the public key pk. In addition, for each $i \in \{1, \cdots, \ell\}$, $\mathcal{A}$ generates a ciphertext corresponding to one encryption of his choosing vector $(c_{1,i}, c_{2,i}) = (\mathfrak{g}_p^{r_i}, \mathfrak{f}^{y_i}\mathfrak{h}^{r_i})$, from some randomness r_i and a ZKPoKπ_{FuncOw} that all the values were correctly generated.

  After this first interaction, if the proof π_{FuncOw} fails, $\mathcal{S}$ aborts. Otherwise, thanks to the extractability condition of the ZKPoK, we find an extractor Ext that $\mathcal{S}$ can use to extract a valid witness $w_{\mathsf{FuncOw}} = (x, y, r)$ with probability $2^{-\ell}$ (see [36]) such that $x = (\mathsf{pk}, (c_{1,i}, c_{2,i}))$.

- $\mathcal{S}$ generates a randomness $r'[0, S]$ where S is an integer from the CL specification. It then invokes the oracle Trivial.KeyGen$(\mathsf{msk}, \cdot)$ on input $\mathbf{y}$ which means it gets back a corresponding functional key $sk_{\mathbf{y}} := (\mathbf{y}, \langle \mathbf{s}, \mathbf{y} \rangle) \in \mathbb{Z}_p^\ell \times \mathbb{Z}_p$ for the vector $\mathbf{y}$.

- $\mathcal{S}$ computes an encryption of $sk_{\mathbf{y}}$ as $c_{sk_{\mathbf{y}}} := (\mathfrak{g}_p^{r'}, \mathfrak{f}^{sk_{\mathbf{y}}}\mathfrak{h}^{r'})$, for randomness r' in $[0, S]$ that it sends to FuncOw. To be consistent, $\mathcal{S}$ needs to generate a proof π_{MskOw} that these values were correctly generated. Note that it does not have access to the master secret key $\mathbf{s}$. Thanks to the rewinding capability of $\mathcal{S}$ and the zero knowledge property, there exists a probabilistic simulator $\mathcal{S}im$ that $\mathcal{S}$ can use in order to simulate the view of the ZKPoK interaction between $\mathcal{A}$ and an honest $\mathcal{O}(\mathsf{msk})$ in the GameReal experiment for any input and this without knowing $\mathbf{s}$.

We deduce the result from the following remarks.

- Thanks to the soundness property of the proof of knowledge, such extractor Ext exists, and $\mathcal{S}$ can extract the input y from the ZKPoK in order to generate the corresponding functional keys.
- The CL ciphertext is correctly distributed, so that the adversary does not notice the difference when receiving the value $ct_{sk_{\mathbf{y}}}$ from an honest evaluation (thanks to the CPA security of [37,38]) and remark that other values are correctly distributed.
- Finally, with the zero-knowledge property, the verifier $\mathcal{A}$ is convinced about the validity of the proofs.

Now suppose that there exists a distinguisher $\mathcal{D}$ that have a black-box access to $\mathcal{A}$. Clearly, the simulator $\mathcal{S}$ acts as a valid honest MskOw and simulates, as mentioned before, any view that $\mathcal{D}$ wish to obtain on any inputs $\mathbf{y} \in \mathbb{Z}_p$.

We deduce that the IFE.IKeyGen does not leak any additional information than IPFE.Trivial.KeyGen and we conclude that the protocol is leak free with respect to FE.KeyGen algorithm. Assuming the above properties of ZKPoK and the CPA security of the CL scheme, we deduce that the IFE scheme is message private by Prop. 3.1.1. □

3.4 Conclusion

In this chapter, we tackle the problem of providing a precise framework for a blindness security in a scenario where protecting some function towards an entity is meaningful. Functional encryption is classically consider as a non-interactive primitive. Our work leads to rethink the main security properties by carefully analysing the impact of adding interactivity when implementing FE for practical use-cases. We show that a notion of blindness could emerge and is meaningful in several situations.

From a theoretical perspective, we show that a secure generic construction is possible from any non-interactive FE supporting some set of functions. Generalizing our previous construction from FHE, we provide a more general construction from private function evaluation equipped with the corresponding ZKPoK.

Having a generic solution comes with a significant computational cost. Nevertheless, keeping in mind the developed techniques, we could exploit the linearity of the particular inner product function and present an efficient solution. We compared existing solution

to tackle this problem, and thanks to the flexibility of some linear homomorphic encryption schemes such as the CL scheme with the adapted ZKPoK, we proposed a secure blind interactive IPFE.

CHAPTER 4

USER'S PROTECTION VIA DIFFERENTIAL-PRIVATE MECHANISMS

Overview of our contribution. In this chapter, we provide a construction bringing differential privacy and FE in a common study. More precisely, we show how it is possible to provide a new DP mechanism by considering a functional encryption primitive as a building block. The main contribution consists of designing a *differential private mechanism for linear queries in encrypted databases*, based on a *randomized* variant of IPFE, named RIPFE.

It is well-known that linear queries can ben implemented (without DP) using IPFE. Our solution combines in a new way ideas from variants of IPFE. The randomized algorithm enjoys from both the confidentiality property inherited from encryption and the required DP noise from privacy, obtaining the *best of both worlds*.

We first present and express a general definition as well as a security model adapted to suit the particular use with DP mechanisms. We focus on the *private-key* setting, since it is sufficient in a lot of scenarii, including ours. We then provide a construction based on the DDH assumption by carefully introducing the required DP *noise* into a modified version of classical IPFE. Finally, we sketch some possible generalizations of our approach using a general variant of IPFE, namely *two-input* IPFE.

Confidentiality against privacy. Similarly to the case discussed in the introduction, consider a scenario where an entity stores its dataset on a public domain (for example a cloud). Moreover, it could be interested to give the possibility of performing computation over the dataset such as statistics. A common requirement is to ensure that the

manipulated data nor the output of a computation do not permits to leak additional information about the content of the dataset.

Thanks to some advanced cryptographic primitives, there are multiple possibilities of protecting the underlying data. With FHE for example, it is theoretically possible to compute any function over the underlying dataset before being revealed. Hence, regarding our scenario, it is possible to publish an encrypted dataset on a cloud and perform any statistics on it. Then, an interaction with the owner(s) of the FHE secret key is needed to recover the content.

Another consideration when releasing sensitive dataset consists of avoiding *re-identification* attacks. Indeed, *reconstruction* or *tracing* attacks [52] are powerful statistical methods that identify each individual entry on a dataset, based only on the published information.

Techniques to prevent from these attacks are one of the topics of Differential Privacy [51]. This notion gives a general *anonymization* framework for *privately* sharing dataset. In particular, if some information are subject to manipulation (such as statistics), then DP provides a modification on the dataset to avoid these re-identification attacks by relying on randomized algorithms [51].

Considering the FHE example, it is possible to make adjustments on the encrypted data by simply evaluating the required (randomized) algorithm. However, an extra interaction for FHE decryption is needed to recover the final statistics.

While in the end, there is a need of protecting the information from disclosure to unauthorized parties (encryption) as well as from statistical attacks (DP), it is challenging to find a cryptographic primitive that suits for this purpose.

Confidentiality with privacy. In this work, we consider the possibility of reconciling these two frameworks by bringing a new primitive using FE that guarantees confidentiality and (DP) privacy.

More precisely, consider a protocol for providing statistics as in the discussed use cases. We put several requirements in its architecture:

- the dataset is stored in an external platform and the data confidentiality is achieved thanks to encryption;
- several functions based on *linear queries* (or inner product) can be evaluated over the underlying data, while maintaining a differential private output;
- a coalition of entities should not be able to learn any additional information about the input data even when using several queries (unless from what is leaked inherently);

- the output of the system should provide significant useful statistics.

As discussed, FHE could be a solution but requires the owner(s) of the secret key to be available for decryption. This leads us to consider how FE could be helpful for providing such model. Recall that FE permits to delegate several evaluation of functions to a third party. The natural starting point is then to consider FE that supports functions which are adjusted according to the DP specifications.

However, classical definitions of FE consider *deterministic functions*. In particular, since DP mechanisms are *randomized* functions by essence [51], it is natural to consider a variant for our particular setting. Goyal et al. [73] study the *randomized* Functional Encryption for general class of randomized functions. Syntactically, it consists of the same algorithm of Def. 2.3.1 with a difference in the correctness condition [73]. In RFE, the randomness used for the underlying randomized algorithm is completely unknown to the decryptor.

We will exploit this variant of FE but we identify that the current literature does not consider the specificities of differential privacy, hence cannot be used directly.

Main strategy. We consider the classical *output perturbation* technique for releasing statistics [51]. Let $\mathbf{x}$ be an input dataset and ϕ a function query. A differential private mechanism is a randomized function represented as $f(\mathbf{x}) = \phi(\mathbf{x}) + e$ for a *noise* e. The DP's literature tells us that giving this *noisy* version of the output $\phi(x)$ allows to immune against known reconstruction attacks.

As in existing generic constructions of RFE, our idea is to start from a sufficiently expressive FE scheme that supports a (deterministic) *derandomized* version of the underlying (randomized) DP function. More precisely, we start from an FE scheme that allows to manage the query ϕ and to transform it into the function $f_\phi(x) = \phi(\mathbf{x}) + e$ with e being a DP-*compatible noise*. The important point is that e must be freshly sampled and be kept secret (otherwise DP is compromised).

From a security point of view, we require that the input elements remain protected, even if an adversary has access to several (say polynomially bounded number) queries functions ϕ. The only information that should be revealed is $f_\phi(\mathbf{x})$ for possibly multiple f_ϕ. We define the security of a DP-RFE using the simulation-security framework, adapting the work that has been done by Agrawal and Wu in [13]. It is in particular important to restrict the adversary from learning the underlying randomness used in the DP mechanism in order to ensure the semantic security of the input data as well as the DP requirement.

The linear query case. Several statistics can be represented as an inner-product function. To have a DP solution, we can consider generic constructions of RFE [13, 73] but they are inefficient.

Just as in FE, considering specific functions (such as IPFE) has some benefits in term of efficiency. Another contribution for this thesis is then to design an efficient scheme for inner-product, or *linear query* as in DP terminology [51]. The function ϕ is given by an inner-product $\langle \cdot, \mathbf{y} \rangle$ for a vector $\mathbf{y}$ and the related randomized function is $f_{\langle \cdot, \mathbf{y} \rangle}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle + e$ as before. For our situation, we provide a *derandomized* version of $f_{\langle \cdot, \mathbf{y} \rangle}$ by considering a noise e that can also be expressed as an inner-product.

Discussion and related works. Traditionally, differential privacy [50] is studied in a setting there is a central data owner that collects raw entries for a dataset, then releases an output of any query, in a differential-private manner.

There is a distinction between a *interactive vs. non-interactive differential-privacy*. In the former [20], a relatively small set of (possibly online) queries are made to the dataset owner that must outputs DP answers for each query. In the latter [50], the idea is to release a *summary* of the dataset that must be generated without knowing the set of queries in advance.

Our construction could be seen as a specific differential-private mechanism where the dataset owner encrypts once for all the data, then share them in a public domain (this corresponds to a summary). However, to make statistics, a statistician must specify the set of queries to be able to decrypt the received ciphertexts. In particular, our work lies in between the interactive and the non-interactive cases and has the benefit to be flexible enough.

Another line of work considers the notion of *structured encrypted database* that support differential private queries [5]. While it looks similar to our use case, we explain the main differences. Similarly to the result of this chapter, a dataset is encrypted in [5]. Moreover, a differential private noise is added anytime an operation is queried. The supported functions are search functions and Agarwal et al. [5] uses homomorphic encryption to obtain a secure solution. In particular, their works corresponds to the interactive setting discussed in the last paragraph. Our approach gives the ability to evaluate more interesting functions such as inner-product while retaining the control of leaked information and as discussed,using functional encryption.

Randomized Functional Encryption. As a generalization of FE, randomized FE has been introduced in [73] and studied in the general setting in both [73] and [13].

In [73], the authors focus on the public key setting and provide a generic construction from indistinguishable obfuscation. Another generic construction, using standard assumptions, is then given in [13]. In the private key setting, the idea of the generic construction from (expressive) Function-Private FE is proposed in [85]. As mentioned above, we do consider a generic construction in this work, but focus on the inner-product. Our security is given in the private-key setting, using standard assumptions, namely the DDH assumption.

Relation to NoisyLinFE from [6, 7, 12] . The related work that is closest to ours is the notion of Noisy linear FE (NoisyLinFE). It has been introduced by Agrawal [7] in the context of building indistinguishable obfuscation (iO). NoisyLinFE is like a regular FE for the class of inner-product functionalities, except that this quantity is recovered only up to some bounded error term. From a ciphertext $c_{\mathbf{x}}$ and a functional key $\mathsf{sk}_{\mathbf{y}}$, it is possible to retrieve the value $\langle \mathbf{x}, \mathbf{y} \rangle + \mathit{noise}$ for some vectors $\mathbf{x}, \mathbf{y}$ and a bounded error term noise. This noise term needs to satisfy some weak pseudorandomness property in order to be useful for building iO [7]. Indistinguishability should take into account such noise, and is then guaranteed as long as two ciphertexts approximately lead to the same inner-product (up to some polynomial bounded term). As far as we know, there are now two different NoisyLinFE constructions in the literature [6, 7]:

1. a non-succinct[1] construction using an IPFE as a building block. In fact, in the above sketch of construction, if we consider $\mathbf{e_y} := (\gamma, \mathbf{1})$ and $\mathbf{e_x} := (\mu, \delta)$ [2], the decryption algorithm given in [6] recovers $\langle \mathbf{x}, \mathbf{y} \rangle + \mu \cdot \gamma + \delta$. Note that this work only considers a stateful KeyGen, in order to trace the used noises. It is interesting to notice that since Agrawal considers a public-key variant, the adversary has the ability to encrypt the messages of its choice and could also learn in the plain some partial noises. However, by combining the noise flooding techniques (with a stateful KeyGen), the result is proved secure [6] in an indistinguishable-like definition;

2. a succinct construction, useful for building iO, using new non-standard assumptions from lattices. Such construction adapted the same kind of techniques to compute the desired noise as a structured linear combination of several carefully chosen ones.

[1] In the sense that the ciphertext size is sublinear in the number of requested functional keys.

[2] where γ is distributed according to some discrete Gaussian distribution of width large enough to flood some bounded error B, and where μ is taken from a distribution of width large enough so that μ is indistinguishable from $\mu + 1$ and δ is such that $\delta + \gamma$ floods γ.

While the aim of [7] is to provide a specific output noise in order to build iO, a careful cryptanalysis study of this scheme, given in [12], shows that the second construction is in fact insecure when there are more requested ciphertexts.

Coming back to our need, it is mentioned in [7] that NoisyLinFE could be used to provide DP mechanism, but without any additional discussion. Indeed, an adaptation of the above NoisyLinFE non-succinct construction seems compatible with our consideration. However, we remark that we do not not need the full power of this primitive, especially the noise flooding techniques. In fact, our construction from two-input FE provides new techniques for building a simplified version of NoisyLinFE where the error terms are only required to be compatible with the DP mechanisms, without any pseudorandomness restriction on the desired output (which is necessary to build an iO). In addition, while [6, 7, 12] consider an indistinguishable security, we provide in this thesis a stronger simulation security and we leave as an interesting open question to provide connections between those notions.

Organization We now organize the following chapter as follows. We first recall the main technical ingredients related to differential privacy. Section 4.2 introduces the formal definition and security requirements for private-key randomized functional encryption schemes for differential-private functionalities. Our construction and its security analysis are given in Section 4.3 and we finally conclude in Section 4.4.

4.1 Differential Privacy

In this section, we present the needed background and tools for our study. We first start by a presentation of basic differential privacy definitions.

4.1.1 Preliminaries

Differential privacy is a mathematical definition of privacy for *datasets* which contain some sensitive information about individuals. In a nutshell, an algorithm is said to be differentially private if by looking at the output of its execution, one cannot tell the presence of any individual's input on it. The traditional way to model such intuition is to view the effect of a computation (by considering the output distribution) on two datasets that differs by one entry. Today, this notion is coupled with a set of different probability tools to provide such differential private algorithms. We give below an overview of the main concepts behind this theory and refer to e.g. [50, 51] for more details.

Notation and terminology of this chapter. Let $\ell \geq 1$ be an integer, we define a *dataset* $\mathbf{x} = (x_1, \ldots, , x_\ell) \in \mathcal{D}^\ell$ as an ordered tuple of ℓ rows, where each row x_i is drawn from a discrete *data universe* $\mathcal{D}$. The ℓ_1-norm of a dataset $\mathbf{x} \in \mathcal{D}^\ell$ is defined as $\|\mathbf{x}\|_1 = \sum_{i=1}^{\ell} |x_i|$. For two datasets $\mathbf{x}$ and $\mathbf{x}'$, the ℓ_1-distance between $\mathbf{x}$ and $\mathbf{x}'$ is defined as $\|\mathbf{x} - \mathbf{x}'\|_1$. Note that $\|\mathbf{x}\|_1$ gives the number of records the dataset contains, while $\|\mathbf{x} - \mathbf{x}'\|_1$ is a measure of how many records differ between $\mathbf{x}$ and $\mathbf{x}'$: it gives a way to compare those two datasets. In the DP literature, the notion of *neighbours* is sometimes used when $\|\mathbf{x} - \mathbf{x}'\|_1 \leq 1$.

Randomized algorithms. The concept of differential privacy is strongly related to the notion of randomized algorithm and in this thesis, our purpose is to combine such a concept with functional encryption. Recall that according to [66], there are two equivalent ways to define a randomized algorithm that are denoted by the letter $\mathcal{M}$ (for Mechanism) and which are described in section 2.1.

Basic definitions. We now recall some definitions from the differential privacy literature. Let $\mathbf{x} \in \mathcal{D}^\ell$ be a dataset. Let $\mathcal{F}$ be a set of queries on datasets and let $\phi \in \mathcal{F}$ be a query done on $\mathbf{x}$.

Definition 4.1.1 (Differential private mechanism [50, 51])**.** *A randomized algorithm* $\mathcal{M} : \mathcal{D}^\ell \times \mathcal{F} \to \mathcal{Y}$ *is* (ϵ, δ)***-differential private*** *if for all* $S \subseteq \mathcal{Y}$*, every* $\phi \in \mathcal{F}$ *and every adjacent datasets* $\mathbf{x}, \mathbf{x}' \in \mathcal{D}^\ell$

$$\Pr[\mathcal{M}(\mathbf{x}, \phi) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(\mathbf{x}', \phi) \in S] + \delta.$$

Note that the smaller ϵ, the better the privacy. We refer to [51] for a complete background on the impact of this parameter. What is important here is that the value ϵ gives the quantity of noise that is added to $\phi(\mathbf{x})$. In fact, there are two ways to proceed: the noise is either (i) generically chosen so as to manage any function and any input, but the risk is to add too much noise, and thus obtain a less suitable answer, or (ii) adapted and calibrated with a less noise, but for a specific set of functions.

Adapting the noise to the function is precisely the purpose of the so-called sensitivity: calibrate the standard deviation of the noise according to the sensitivity of the function ϕ. In a nutshell, the ℓ_1-sensitivity of a query ϕ captures the magnitude by which a single individual's data can change the function ϕ in the worst case, and therefore, intuitively, the uncertainty in the response that we must introduce in order to hide the participation of a single individual.

Definition 4.1.2 (Sensitivity [50, 51])**.** *Let* $(\mathbf{x}, \mathbf{x}')$ *be two datasets in* $\mathcal{D}^\ell \times \mathcal{D}^\ell$*. The* ℓ_1***-sensitivity*** *of a function (or query)* ϕ *is the quantity*

$$\Delta_\phi := \max_{(\mathbf{x},\mathbf{x}') \in \mathcal{D}^\ell \times \mathcal{D}^\ell, \|\mathbf{x}-\mathbf{x}'\|_1 \leq 1} ||\phi(\mathbf{x}) - \phi(\mathbf{x})'||_1.$$

In our case, as the dataset is encrypted, one may wonder whether it will still be possible to use this ℓ_1-sensitivity. In fact, the entity who encrypts the dataset is the one that will also give its agreement for a query ϕ. Thus, the data owner will be the entity that will add the noise during both the encryption and the functional key generation phases, so that it can easily manage such ℓ_1-sensitivity.

Statistical/Computational DP. The definition of differential privacy above is given as a *statistical* one. There exists in the literature [88] some adaptations in the *computational* setting. For example, the indistinguishable-DP (IND-DP) is defined for every possible PPT distinguisher $\mathcal{A}$ as

$$\Pr[\mathcal{A}(\mathcal{M}(\mathbf{x})) = 1] \leq e^\epsilon \cdot \Pr[\mathcal{A}(\mathcal{M}(\mathbf{x}')) = 1] + negl(\lambda),$$

for some security parameter λ.

Our construction in Sec. 4.3 satisfies computational DP but as treated in [76], it is still possible to convert this scheme to get statistical DP.

4.1.2 DP-compliant noise distributions

Our main goal is to use of a noise that is both compatible with the DP principles, and does not compromise the security (see Sec. 4.2.3) of our randomized functional encryption scheme. We should also take into account, for those two aspects, that the noise is generated both during the encryption and the functional key generations steps.

Usual distributions. There are several important distributions that have been studied in the DP literature, depending on the level of differential privacy one want to consider. The most explored one is the *Laplacian distribution* (centered on 0) with scale b, given by its probability mass function on any (real) point $\mathbf{x}$:

$$\mathsf{Lap}(\mathbf{x}|b) = \frac{1}{2b} e^{-\frac{|\mathbf{x}|}{b}}.$$

Another example could be found on the traditional *Gaussian distribution* denoted by $\mathcal{N}(\mu, \sigma^2)$.

A discrete variant of Lap**.** Since we work with discrete groups to design our protocols, the most suited distribution is the *two-sided Geometric distribution*, which can be seen as a discrete version of the Laplacian one [63, 79].

Definition 4.1.3 ([63, 79])**.** *Let* $\alpha \in (0, 1)$ *be a parameter.*

- *A random variable* X *is distributed according to the (discrete) one-sided geometric distribution denoted by* $\mathsf{Geo}^+(\alpha)$ *if it has a mass function defined for every positive integer* $k \in \mathbb{Z}$ *as*

$$\Pr[X = k] = \alpha^{k-1}(1 - \alpha)$$

- *Similarly a random variable* X *is distributed according to* $\mathsf{Geo}(\alpha)$*, called a* two-sided geometric distribution *if for every integer* k

$$\Pr[X = k] = \frac{1 - \alpha}{1 + \alpha} \cdot \alpha^{|k|}.$$

We will simply refer to the geometric distribution when talking about Geo*.*

Moreover, based on the fact that in our construction (see Section 4.3), the noise that should follow such a distribution is computed in two steps, using first $e_{\mathbf{x}}$ during encryption and using then $e_{\mathbf{y}}$ during functional key generation, we need to find their right linear combination to eventually obtain a noise following the two-sided geometric distribution. For this purpose, we base our choice on the following result using the *one-sided* geometric distribution (see [79] for a proof).

Lemma 4.1.1 (Proposition 3.1 in [79])**.** *Let* $\alpha \in (0, 1)$*. A random variable* Y *is distributed according to* $\mathsf{Geo}(\alpha)$ *if and only if* $Y = X_1 - X_2$*, where* X_1, X_2 *are two independent and identically distributed random variables distributed according to* $\mathsf{Geo}^+(\alpha)$ *with* $i = 1, 2$*.*

This lemma gives us the insurance that if both $e_{\mathbf{x}}$ and $e_{\mathbf{y}}$ are sampled from the one-sided geometric distribution, then the final noise $e := e_{\mathbf{x}} - e_{\mathbf{y}}$ will have the desired output distribution.

Noise distribution and DP. Given any of the above distributions, and assuming that the query function ϕ has some additive structure (as it is the case in our whole study), we can now define a DP mechanism $\mathcal{M}$ verifying the definition 4.1.1.

Lemma 4.1.2 (DP mechanism)**.** *Fix some $\epsilon > 0$ (resp. $\delta \geq 0$). Let $\mathcal{F}$ be a set of queries on datasets and $\phi \in \mathcal{F}$. Consider E be a random variable distributed according to the geometric (resp. Laplacian, Gaussian) distribution, whose parameter depends on Δ_ϕ, the ℓ_1-sensitivity of ϕ. The geometric (resp. Laplacian, Gaussian) mechanism defined as*

$$\mathcal{M}(\mathbf{x}, \phi, \epsilon) := \phi(\mathbf{x}) + E,$$

is $(\epsilon, 0)$ (resp. (ϵ, δ)) differential private.

For the geometric distribution, we need to set $\alpha = exp(\frac{\epsilon}{\Delta_\phi})$ to reach the $(\epsilon, 0)$-DP with the ℓ_1-sensitivity. This result can be found in [63,109], while the ones for Laplacian and Gaussian are given in [51].

We finish this section by illustrating how our definition of *randomized algorithm* fits in this situation. To satisfy Def. 2.1.1 with the geometric distribution, we consider the associated mapping $P_{\mathcal{M}}$, defined for a mechanism $\mathcal{M}$, that takes as inputs $\mathbf{x}, \phi, \epsilon$ and that outputs the probabilities (centered on $\phi(\mathbf{x})$) defined by $p_k := \frac{1-\alpha}{1+\alpha} \cdot \alpha^{|k|}$ for $k \in \mathbb{Z}$ and α defined as above. In this case, it is obvious that Def. 2.1.1 is satisfied since $\Pr[\mathcal{M}(\mathbf{x}, \phi, \epsilon) = k] = p_k$.

4.2 Randomized FE for DP Functionalities

In the previous section, we have seen in Lemma 4.1.2 that a way to achieve differential-privacy for linear queries is to compute, from a dataset $\mathbf{x}$ and a query $\mathbf{y}$, the value $\langle \mathbf{x}, \mathbf{y} \rangle + e$ where e is a random element drawn from the geometric distribution. Given that $\mathbf{x}$ is the vector to be encrypted, and $\mathbf{y}$ is assimilated to the function behind the functional key, we have to design a (randomized) inner-product functional encryption, for a specific class that makes the output DP-compliant.

The natural choice for a security model is then to consider the existing definitions for RFE [13,73] and while we do not need this general definition, we present in the sequel an adaptation for linear queries and in the case of differential privacy.

Remark. *As mentioned in [13,73], while a randomized algorithm permits, each time it is executed, to obtain different fresh outputs, a party executing the* FE *decryption procedure on input* one *given ciphertext for a message* $\mathbf{x}$ *and* one *given functional key for a vector*

$\mathbf{y}$ *should always get the same output. In addition, given two different ciphertexts and one functional key, such party should get two independent outputs. The idea behind is that the latter cannot repeatedly sample the functionality to obtain multiple outputs for different random coins. In our situation, this means that when a ciphertext and a functional key are generated, running the decryption algorithm leads to the same value* $\langle \mathbf{x}, \mathbf{y} \rangle + e$.

4.2.1 DP Randomized FE for Linear Queries

In this section, we present our version of Randomized Inner-Product Functional Encryption, hereafter RIPFE, in which we first consider *linear queries*, reducing the functionality to an inner-product, and then integrate the geometric distribution to manage randomness.

Definitions and Security Model for randomized inner-product. For $\alpha \in (0, 1)$, we define the randomized version of the above inner-product, according to the geometric distribution, as

$$f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle + e = \sum_{i=1}^{\ell} x_i \cdot y_i + e \pmod{q},$$

where $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$, $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathcal{Y}^\ell$, and where e is sampled according to the geometric distribution $\mathsf{Geo}(\alpha)$ (see Section 4.1.1).

We finally define the family of differential-private inner-product functions, according to the geometric distribution, as

$$\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q} := \{f_{\mathbf{y}} : \mathcal{X}^\ell \to \mathbb{Z}_q\}.$$

Given such family of DP-friendly functions, we need to come back to the notion of ℓ_1-sensitivity. In fact, it is easy to see that the magnitude of the inner-product function, denoted $\Delta_{f_{\mathbf{y}}}$, is given in this case by $\Delta_{f_{\mathbf{y}}} := B_x \cdot B_y$.

RIPFE for Linear Queries We are now ready to present the syntax of a RIPFE scheme in the private-key setting, adapted from [13, 73], for the family of functions $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$ defined above.

Definition 4.2.1 (Private-Key RIPFE)**.** *Let* $\lambda > 1$ *be a security parameter. A private-key randomized functional encryption scheme* RIPFE *for the family of functions* $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$ *consists of a tuple* (Setup, KeyGen, Enc, Dec) *defined as follows.*

- $\mathsf{Setup}(1^\lambda)$ *is a* PPT *algorithm which takes as input a security parameter* 1^λ*, and outputs a master secret key* msk *and possibly some public parameters* param *which are included in all the other algorithms.*
- $\mathsf{KeyGen}(\mathsf{msk}, f_{\mathbf{y}})$ *is a* PPT *algorithm which takes as input a master secret key* msk *and a description of the function* $f_{\mathbf{y}} \in \mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$ *and outputs a functional key* $\mathsf{sk}_{\mathbf{y}}$*.*
- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x})$ *is a* PPT *algorithm which takes as input the master secret key* msk *and a message* $\mathbf{x} \in \mathcal{X}$*, and returns a ciphertext* $c_{\mathbf{x}}$*.*
- $\mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, c_{\mathbf{x}})$ *is a deterministic algorithm which takes as input a functional key* $\mathsf{sk}_{\mathbf{y}}$ *and a ciphertext* $c_{\mathbf{x}}$ *and outputs a string* $z \in \mathbb{Z}_q$*.*

4.2.2 Correctness for RIPFE

As shown in Chap. 2, Sec 2.3, the classical definition for an IPFE informally ensures that during the decryption procedure for *one* ciphertext $c_{\mathbf{x}}$ and *one* functional key $\mathsf{sk}_{\mathbf{y}}$, it is possible to obtain the inner-product $\langle \mathbf{x}, \mathbf{y} \rangle$. In the more general case of RFE, as mentioned in [73, Remark 2.2], we need to define correctness for multiple ciphertexts and functional keys.

Our decryption procedure outputs an element of the form $\langle \mathbf{x}, \mathbf{y} \rangle + e$, where e is distributed according to $\mathsf{Geo}(\alpha)$ for some α. Following [13, 73], the definition considers that given a fresh encryption of a message $\mathbf{x}$ or a fresh key for a function $f_{\mathbf{y}}$, the output of the decryption algorithm is (computationally/statistically) indistinguishable from evaluating $f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle + e$ with some fresh randomness output. In our case, e is drawn from the two-sided geometric distribution.

This leads to the following definition, adapted from [13, 73] to the case of the inner-product and the geometric distribution.

Definition 4.2.2 (Statistical (resp. Computational) Correctness)**.** *We say that an* $\mathsf{RIPFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *supporting the family of functions* $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$ *is* correct *if for all* $\lambda \geq 0$*, for every polynomial* $Q = Q(\lambda)$*, and for every* $\mathbf{x}^1, \ldots, \mathbf{x}^Q$ *in* $\mathcal{X}^\ell$ *and* $\mathbf{y}^1, \ldots, \mathbf{y}^Q$ *in* $\mathcal{Y}^\ell$*, the following two distributions are statistically (resp. computationally) indistinguishable:*

- $\mathbf{Real}(\lambda) := \{\mathsf{Dec}(\mathsf{sk}_{\mathbf{y}^i}, c_{\mathbf{x}^j})\}_{i,j \in [Q]}$*, where*

  $(\mathsf{msk}, \mathsf{param}) \leftarrow \mathsf{Setup}(1^\lambda)$

  $c_{\mathbf{x}^j} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}^j)$ *for all* $j \in [Q]$*.*

  $\mathsf{sk}_{\mathbf{y}^i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_{\mathbf{y}^i})$ *for all* $i \in [Q]$*.*

- **Ideal**$(\lambda) := \{\langle \mathbf{x}^i, \mathbf{y}^j \rangle + e_{i,j})\}_{i,j \in [Q] \times [Q]}$ *for* $e_{i,j}$ *sampled according to the geometric distribution* $\mathsf{Geo}(\alpha)$.

The main difference between our definition and those of [13, 73] is that we directly consider the specific output distribution $\mathsf{Geo}(\alpha)$. In [13,73], the general case of a randomized algorithm f is treated and the formulation uses the *auxiliary input* configuration of randomized algorithms (see the discussion in Section 4.1.1). Our definition makes it possible to easily prove that our scheme fulfills the correctness condition for this special case of RIPFE. In particular, since our interest is for DP mechanisms, it is sufficient to prove that our decryption algorithm outputs elements that are distributed according to the specific geometric distribution.

4.2.3 Simulation-Security for RIPFE

In this section, we focus on the message privacy for an RIPFE supporting the family of functions $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$. Informally, the main general objective for any functional encryption scheme is to guarantee that an adversary who owns a functional secret key corresponding to a function f cannot learn more than $f(\mathbf{x})$ from the encryption of $\mathbf{x}$ (as shown in Chap 2, Sec 2.3.1). The case of randomized FE is intricate since there are multiple potential outputs for $f(\mathbf{x})$.

As mentioned in [13, 73], considering an indistinguishability-based definition is hard and could potentially provide circularity in the definition (see [73] for a more detailed discussion). So we consider, as in [13, 73], a simulation-based definition that we adapt to our specific case.

In [13, 73], the idea of the simulation-security for randomized FE is to simulate the functional key generation and the message encryption using the set of values $\{f_j(\mathbf{x}_i; r_{i,j})\}_{i,j \in Q}$ where (1) $\{f_j\}_{j \in Q}$ is the set of functions requested by the adversary, (2) $\{\mathbf{x}_i\}_{i \in Q}$ is the set of messages output by the adversary, and (3) each $r_{i,j}$ is a *uniformly* chosen random sample from $\mathcal{R}_\lambda$, where $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ is the randomness space.

We insist that all previous approaches for RFE consider this view of randomized functions as sampling uniformly from a random space[3]. As discussed in Section 2.1 from Chapter 2, we have taken another approach for the definition of randomized functions, which considers the specific output distribution. We then change the definition given in [13, 73] to fit with our discussion.

Also, we restrict our definition to the case of the *one* selective simulation security

[3] except for NoisyLinFE of [6, 7, 12], but this work is is not introduced or based as a special case of RFE.

(one-Sel-Sim), managing only one plaintext $\mathbf{x}^\star$ coming from the adversary, and not a set of messages as in [13, 73]. More formally, we have the following adapted definition for adaptive simulation security of [13, 73].

Definition 4.2.3 (Simulation-security for RIPFE)**.** *Let* RIPFE $=$ (Setup, Enc, KeyGen, Dec) *be a randomized* IPFE *according to the family of functions* $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$*. Let* Sim $= (\mathcal{S}^\star_{\mathsf{S}}, \mathcal{S}^\star_{\mathsf{E}}, \mathcal{S}^\star_{\mathsf{KG}})$ *be a* PPT *simulator and let* $\mathcal{A}$ *be a* PPT *adversary. Consider the following experiments.*

$\mathsf{Exp}^{real}_{\mathcal{A}}(1^\lambda)$

1 : $(\mathbf{x}^\star, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$
2 : $(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{st})$
3 : $(c_{\mathbf{x}^\star}, \mathsf{st}') \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}^\star)$
4 : $\gamma \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{param}, c_{\mathbf{x}^\star}, \mathsf{st}')$

$\mathsf{Exp}^{ideal}_{\mathcal{A},\mathsf{Sim}}(1^\lambda)$

1 : $(\mathbf{x}^\star, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$
2 : $(\mathsf{param}^\star, \mathsf{st}^\star) \leftarrow \mathcal{S}^\star_{\mathsf{S}}(1^\lambda, \mathsf{st})$
3 : $(c^\star, \mathsf{st}') \leftarrow \mathcal{S}^\star_{\mathsf{E}}(\mathsf{st}^\star, 1^{|\mathbf{x}^\star|})$
4 : $\gamma \leftarrow \mathcal{A}^{\mathcal{S}^\star_{\mathsf{KG}}(\mathsf{st}^\star, st', \cdot)}(\mathsf{param}^\star, c^\star, \mathsf{st}')$

In the real experiment, the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *oracle takes the master secret key* msk*, a query* $\mathbf{y}$ *and returns a (real) functional key* $sk_{\mathbf{y}}$ *corresponding to* $\mathbf{y}$*.*

In the simulated experiment, the $\mathcal{S}^\star_{\mathsf{KG}}(\mathsf{st}^\star, \cdot)$ *are the corresponding key generation oracle. The* $\mathcal{S}^\star_{\mathsf{KG}}$ *simulator has an oracle access to an ideal functional key generation oracle* $\mathsf{KeyIdeal}(\mathbf{x}^\star, \cdot)$ *which on input* $\mathbf{y}^\star$*, outputs an element* $\langle \mathbf{x}^\star, \mathbf{y}^\star \rangle + e$ *for* e *distributed according to* $\mathsf{Geo}(\alpha)$*. More formally, let* $\mathbf{y}^1, \cdots, \mathbf{y}^Q$ *be* $\mathcal{A}$*'s queries to* $\mathcal{S}^\star_{\mathsf{KG}}(\mathsf{st}^\star, \cdot)$*, then* KeyIdeal *pick (independent)* e_i *from* $\mathsf{Geo}(\alpha)$ *and define* $\mathsf{aux}_i := \langle \mathbf{x}^\star, \mathbf{y}^i \rangle + e_i$ *for all* $i \in [Q]$*. Finally, it returns* $\{\mathbf{y}^i, \mathsf{aux}_i, \mathsf{sk}_{\mathbf{y}^i}\}$ *to* $\mathcal{S}_{\mathsf{KG}}$*.*

An RIPFE *scheme satisfies the* ***One Selective Simulation-Based*** *(one-Sel-SIM) security if there exists a* PPT *simulator* Sim *such that for all efficient* PPT *adversary* $\mathcal{A}$*, the outputs of the* real *and* ideal *experiments above are computationally indistinguishable.*

In a FE-based indistinguishability game, there is an inherent condition imposed by the output of the security game that no adversary should be allowed to query vectors $\mathbf{x}_0, \mathbf{x}_1$ with $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$. This condition badly translates in the randomized FE setting, since we need to manage random values. It is unclear how it is possible to include this parameter.

On Indistinguishability type definition. In the case of RFE, having an indistinguishability based definition implies some circularity [73]. Another related notion in the literature is the one of noisy linear FE [6, 7, 12]. In this context, an indistinguishable-like definition is provided, in which it is requested to guarantee the indistinguishability between two ciphertexts encrypting respectively $\mathbf{x}_0, \mathbf{x}_1$, as long as they have approximately

the same inner-product. The given definition of [6, 7] also considers a one-message security. However, since indistinguishability type and simulation type definitions for RFE are incomparable in general, as shown in [13, 73], it is an interesting problem to find connections between those two notions.

4.3 Construction

In this section, we provide our construction of an RIPFE for the family of functions $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$ defined in the previous section. Our scheme is one-selective simulation secure under the DDH assumption.

4.3.1 High-Level Overview

Denote by $f_{\mathbf{y}}$ the corresponding randomized function: $f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle + e$, with e to be some DP-compliant noise.

Having IPFE in mind, we have to add in both the encryption and the functional key generation algorithms some partial noise that will imply, at the end of the decryption phase, a noise e which is compatible with the specifications of DP. The main issue is to ensure that this modification preserves the semantic security. We give in the following a *step-by-step* description of our construction with the aim to provide the subtleties that may arise when considering this primitive.

Fix a private-key FE scheme $\mathsf{IPFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ for the inner-product functions. Let $\mathbf{x}, \mathbf{y}$ be two vectors in some domain with say $\ell \geq 1$ coordinates. The transition from $\langle \mathbf{x}, \mathbf{y} \rangle$ (for a given $\mathbf{y}$) to the randomized function $f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle + e$ can be done by augmenting the vectors $\mathbf{x}$ and $\mathbf{y}$ with some randomness and considering vectors $\mathbf{x}^\star, \mathbf{y}^\star$ with $\ell + 1$ coordinates such that $\langle \mathbf{x}^\star, \mathbf{y}^\star \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + e$. The natural point is to *incorporate* e in one of the two vectors (augmenting the other element with 1 element). We provide two important remarks about this method.

1. Considering $\mathbf{x}^\star = (\mathbf{x}, e)$ and $\mathbf{y}^\star = (\mathbf{y}, 1)$ provides the desired answer but yield the same *noise* e for different functional decryption keys.

2. Considering $\mathbf{x}^\star = (\mathbf{x}, 1)$ and $\mathbf{y}^\star = (\mathbf{y}, e)$ is also a solution but notice that the noise remains the same for different encrypted messages.

These behaviours need to be avoided in practice, since in the DP literature, the noise must be freshly sampled for each output result.

Dividing DP noise. Starting from that, we give interest in the construction of [6] and adapt it to divide the noise into two parts: one which is generated during the encryption Enc, and another one that is chosen during the functional key generation KeyGen. Remark that since we are considerinh IPFE, if we augment the vectors $\mathbf{x}, \mathbf{y}$ as $\mathbf{x}^\star := (\mathbf{x}, \mathbf{x}')$ and $\mathbf{y}^\star := (\mathbf{y}, \mathbf{y}')$ for any vectors $\mathbf{x}', \mathbf{y}'$, we obtain by construction the inner-product $\langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}', \mathbf{y}' \rangle$. Having this view of what IPFE could provide makes a suggestion that if e is the targeted DP noise, then an appropriate choice of the elements $\mathbf{x}', \mathbf{y}'$ such that $\langle \mathbf{x}', \mathbf{y}' \rangle = e$ could return a valid answer to our problem.

Suppose that e is written as a difference between two elements $e_{\mathbf{x}} - e_{\mathbf{y}}$ (as suggested by lemma 4.1.2) following the *one-sided* geometric distribution, then the natural form is to view a plaintext having the following shape $\mathbf{x}^\star = (\mathbf{x}, e_{\mathbf{x}}, 1)$ where $\mathbf{x}$ is the vector to be encrypted and $e_{\mathbf{x}}$ the one-side geometric noise. Similarly, the functional key for function $f_{\mathbf{y}}$ is computed with the vector $\mathbf{y}^\star = (\mathbf{y}, 1, -e_{\mathbf{y}})$, where $e_{\mathbf{y}}$ is again a one-side geometric noise. After the decryption we obtain $\langle \mathbf{x}, \mathbf{y} \rangle + e_{\mathbf{x}} - e_{\mathbf{y}}$. Splitting e as $e_{\mathbf{x}} - e_{\mathbf{y}}$ gives us the targeted distribution.

It remains to fix some issues with this intuition.

1. Invoking the same KeyGen implies using the same e_y and the resulted output could be correlated. In particular, we still need to ensure correctness with fresh samples any time the decryption algorithm is invoked.

2. A more technical problem in known (non-function private) IPFE schemes is that the adversary has access (in addition to a ciphertext) to a functional key that generally contains the description of the function. In particular, the value $e_{\mathbf{y}}$ could be deduced from the description of the functional key $\mathsf{sk}_{\mathbf{y}}$, which obviously allows the adversary to learn the quantity $\langle \mathbf{x}, \mathbf{y} \rangle + e_{\mathbf{x}}$ from the output of the decryption. This is clearly more than what is permitted and then it follows that this first attempt could not be secure, and even not DP private.

3. It is still not clear how we *split* the noise e in two different separate noises.

Remark that we need somehow to hide the information related to the noise used during the functional key generation (and encryption). However, we do not exactly need a full function-private FE [28], in which the functional key should not reveal any information about the underlying vectors. In our case, the input $\mathbf{y}$ may be known since what we want to hide is the noise $e_{\mathbf{y}}$.

Using a two-input FE. Our approach is to view the function that we want to implement as a *two-input* function, and to consider a compatible functional encryption scheme, namely multi-input FE (MIFE). Informally speaking, a MIFE manages several encryption slots and given a functional key sk_f for a multi-input function f, it permits to decrypt a set $\{\mathsf{Enc}(\mathbf{d}_1), \ldots, \mathsf{Enc}(\mathbf{d}_n)\}$ of potentially n independent ciphertexts to obtain $f(\mathbf{d}_1, \ldots, \mathbf{d}_n)$ (and nothing more). Here, $\mathbf{d}_i$ is a certain vector of a certain size ℓ. We target a private-key two-input IPFE (denoted 2IPFE) build upon the construction given in [2] (which is itself based on any single-input IPFE scheme).

As in [2], we begin with their description of a *one-time* version of a private-key MIFE for n slots and the inner-product.

- The master key is a set $\mathbf{u}$ of n random vectors $\mathbf{u}_i$.
- The encryption of a vector message $\mathbf{d}_i$ in slot i ($i \in [n]$) is given by $\mathsf{c}_i := \mathbf{d}_i + \mathbf{u}_i$.
- The functional key for a vector $\mathbf{y}$, with n components denoted $\mathbf{y}_i$, is given by $z_{\mathbf{y}} := \sum_{i\in[n]} \langle \mathbf{u}_i, \mathbf{y}_i \rangle$.
- The decryption procedure, taking as input n ciphertexts c_i ($i \in [n]$) constructed as above, computes $\langle \mathbf{d}, \mathbf{y} \rangle := \sum_{i\in[n]} \langle \mathbf{d}_i, \mathbf{y}_i \rangle$ as $\sum_{i\in[n]} \langle \mathsf{c}_i, \mathbf{y}_i \rangle - z_{\mathbf{y}}$.

The scheme in [2] is proven to be *one ciphetext* selective simulation secure for multiple instances, one instance corresponding to one slot.

We use a 2-slot (i.e $n = 2$) variant for vectors of size $\ell + 1$: one for the vector $\mathbf{x}$ of size ℓ and its noise, that is $\mathbf{d} := (\mathbf{x}, e_{\mathbf{x}})$ of size $\ell + 1$, and the other one for the noise related to the query $\mathbf{y}$ of size ℓ, that is $\mathbf{d}' := (\mathbf{0}^\ell, -e_{\mathbf{y}})$.

Consider the master secret key as a two-component vector $(\mathbf{u}, \mathbf{u}')$, where $\mathbf{u}$ and $\mathbf{u}'$ are both vectors of size $\ell + 1$. The functional key is computed for a vector with two components of size $\ell+1$: $(\mathbf{y}, 1)$ and $(\mathbf{1}^{\ell+1})$. The decryption gives us what we are looking for: $\langle \mathbf{d}, (\mathbf{y}, 1) \rangle + \langle \mathbf{d}', (\mathbf{1}^{\ell+1}) \rangle$ which is by definition equal to $\langle \mathbf{x}, \mathbf{y} \rangle + e_{\mathbf{x}} - e_{\mathbf{y}}$.

Remark that in the MIFE setting, the encryption slots are generated independently from each other. This observation leads us to consider the encryption of second slot during the KeyGen execution phase. In more details, our scheme is then constructed as follows.

- During Setup, one generates the master secret key $\mathbf{u}$ for the first slot, together with some additional parameters.
- During encryption procedure Enc for a vector $\mathbf{x}$, we execute the encryption for the first slot, as $\mathbf{d} = (\mathbf{x}, e_{\mathbf{x}}) + \mathbf{u}$, where $e_{\mathbf{x}}$ is a fresh sample.

- During the functional key generation KeyGen, on input the vector $\mathbf{y}$, we first generate the master secret key $\mathbf{u}'$ for the second slot. We then perform the encryption for the second slot, as $\mathbf{d}' = (\mathbf{0}^\ell, -e_\mathbf{y}) + \mathbf{u}'$ for some fresh sample $e_\mathbf{y}$. After that, we compute the functional key as given above: $z_\mathbf{y} = \langle \mathbf{u}, (\mathbf{y}, 1) \rangle + \langle \mathbf{u}', \mathbf{1}^{\ell+1} \rangle$.

- The decryption takes as input $(\mathbf{d}, \mathbf{d}', z_\mathbf{y})$ and finally outputs $\langle \mathbf{x}, \mathbf{y} \rangle + e_\mathbf{x} - e_\mathbf{y}$ as explained above.

Using the security of the one-time MIFE [2], the resulting scheme is secure for one ciphertext (using one instance/slot), and one functional key query (using the other instance/slot). But what we want in our scenario is that several entities could request a functional key for their own query, which means that we need one slot for the ciphertext (during database encryption) and the possibility to use it several times, with an unlimited number of functional key queries (using as many instances as needed). More precisely, we need to transform this 2IPFE so that it can manage several instances.

Adding a layer of IPFE. Another interesting novelty given in [2] is their idea of adding a layer of single-input IPFE on the top of this *one-time encryption* to obtain the security for polynomially many challenges (achieving the many-adaptive indistinguishability property). Adapted to our scheme above and given any One-Selective simulation secure IPFE scheme (for multiple instances), the construction now works as follows.

- The master key consists of the master key of the one-time MIFE, denoted $\mathbf{u}$, and the master secret key of the IPFE, denoted $\mathsf{IPFE.msk}$.

- The encryption of a vector $\mathbf{x}$ consists in computing the one-time ciphertext $\mathbf{d}$ as above, and then in encrypting the results using IPFE.Enc, as $c_\mathbf{x} = \mathsf{IPFE.Enc}(\mathsf{IPFE.msk}, \mathbf{d})$.

- The functional key for a vector $\mathbf{y}$ is given by (i) the IPFE ciphertext $c_{\mathbf{d}'}$ corresponding to the encryption of $\mathbf{d}' = (\mathbf{0}^\ell, -e_\mathbf{y}) + \mathbf{u}'$ for a new master secret key $\mathbf{u}'$, given by $c_{\mathbf{d}'} = \mathsf{IPFE.Enc}(\mathsf{IPFE.msk}, \mathbf{d}')$, (ii) the functional key $\mathsf{sk}_{(\mathbf{y},1)} = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.msk}, (\mathbf{y}, 1))$ of the used IPFE for the vector $(\mathbf{y}, 1)$, (iii) another functional key $\mathsf{sk}_{(\mathbf{1}^\ell,1)} = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.msk}, (\mathbf{1}^\ell, 1))$ of the used IPFE for the vector $(\mathbf{1}^\ell, 1)$ and (iv) the functional key $z_\mathbf{y}$ of the one-time MIFE, computed as above using $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{y}$.

- The decryption finally proceeds in two main steps. It first executes twice the decryption procedure of the IPFE to obtain $\mathsf{IPFE.Dec}(\mathsf{sk}_{(\mathbf{y},1)}, c_\mathbf{x}) = \langle \mathbf{d}, (\mathbf{y}, 1) \rangle$ and

$\mathsf{IPFE.Dec}(\mathsf{sk}_{(\mathbf{1}^\ell,1)}, c_{\mathbf{d}'}) = \langle \mathbf{d}', (\mathbf{1}^\ell, 1)\rangle$ (by construction). It finally extracts the result $\langle \mathbf{x}, \mathbf{y}\rangle + e_{\mathbf{x}} - e_{\mathbf{y}}$ as for the one-time decryption algorithm above, using $z_{\mathbf{y}}$.

Using this trick, we achieve our goal of being able to manage several key generation executions (and thus several queries to the same encrypted database). We provide in the following an instantiation from the DDH assumption using the IPFE given in [1] and an attempt that our construction could be generalized by considerinng two-input IPFE.

4.3.2 Construction from DDH

In the following we consider an instantiation of the above high-level description from the DDH assumption. We use the IPFE of from [1], in which the ciphertext is as following $\mathbf{c}_{\mathsf{d}} := (C, D, \mathbf{E})$ such that $C = g^r$, $D = h^r$, and $\mathbf{E} = \mathbf{g}^{\mathbf{d}}\mathbf{h}^r$ for a random r and vector d.

Next, given the related master secret key $(\mathbf{s}, \mathbf{t}, \mathbf{h})$ of the IPFE, our functional key generation produces one functional secret $\mathsf{sk}_{(\mathbf{y},1)}$ for the vector $(\mathbf{y}, 1)$, another one $\mathsf{sk}_{(\mathbf{1}^\ell,1)}$ for the vector $(\mathbf{1}^\ell, 1)$, and a final functional key for a one-time MIFE, as $z := \langle \mathbf{u}, (\mathbf{y}, 1)\rangle + \langle \mathbf{u}', (\mathbf{1}^\ell, 1)\rangle \pmod q$.

Finally, using the IPFE decryption and the property of the two-input MIFE, we can easily recover $\langle \mathbf{x}, \mathbf{y}\rangle + e_{\mathbf{x}} - e_{\mathbf{y}}$.

For the description, we consider a subroutine called MSKGen which corresponds to a Setup for IPFE and works on input g, h and proceeds as follows: for $i \in [\ell+1]$, it samples uniformly at random $s_i, t_i \leftarrow\!\$\, \mathbb{Z}_q^\star$ and computes $h_i := g^{s_i}h^{t_i}$. Let $\mathbf{u} \leftarrow\!\$\, \mathbb{Z}_q^{\ell+1}$ be a uniform element. Denote by

$$(\mathbf{s}, \mathbf{t}, \mathbf{h}) := ((s_i)_{i\in[\ell+1]}, (t_i)_{i\in[\ell+1]}, (h_i)_{i\in[\ell+1]}),$$

then MSKGen finally returns the master secret key $\mathsf{msk} := (\mathbf{u}, \mathbf{s}, \mathbf{t}, \mathbf{h})$.

We provide next the description of our scheme.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: given the security parameter λ, this algorithm outputs the public parameters $\mathsf{param} := (\mathbb{G}, q, g, \mathbf{g}, h, \mathcal{X}, \mathcal{Y}, \alpha)$ defined as follows. Let q be a λ-bit prime and let $\mathbb{G}$ be a cyclic group of order q where the Decisional Diffie-Hellman assumption holds. Let g, h be generators of $\mathbb{G}$ and let $\mathbf{g} = (g, \cdots, g)$ be a vector of size $\ell + 1$. Let $B_x, B_y \in \mathbb{Z}$ be some integer bounds and let $\mathcal{X} = \{0, \ldots, B_x\}$ and $\mathcal{Y} = \{0, \ldots, B_y\}$. Let α be the parameter associated to the DP-privacy. We now implicitly suppose that param is included in each of the following algorithms. Next, we execute MSKGen described above to produce $\mathsf{msk} := (\mathbf{u}, \mathbf{s}, \mathbf{t}, \mathbf{h})$.

- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x})$: using the master secret key msk, this algorithm returns an encryption of a vector $\mathbf{x} \in \mathcal{X}^\ell$. It first computes an element $e_{\mathbf{x}}$ over $\mathbb{Z}$ distributed according to *one-sided* geometric distribution (see Def. 4.1.3) with parameter α and considers the vector $\mathbf{d} := (\mathbf{x}, e_{\mathbf{x}}) + \mathbf{u} \pmod q$. For a uniform element $r \in \mathbb{Z}_q^\star$, the algorithm returns a ciphertext $\mathbf{c}_{\mathbf{x}} := (C, D, \mathbf{E})$ defined as

$$C = g^r,\ D = h^r,\ \mathbf{E} = \mathbf{g}^{\mathbf{d}}\mathbf{h}^r.$$

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$: this algorithm returns a functional key for the vector $\mathbf{y} \in \mathcal{Y}^\ell$, given the secret key msk. It first executes the MSKGen procedure (given in the Setup above) to create an ephemeral master secret key $\mathsf{msk}' = \mathsf{MSKGen}(g, h) = (\mathbf{u}', \mathbf{s}', \mathbf{t}', \mathbf{h}')$. It next selects an element $e_{\mathbf{y}}$ over $\mathbb{Z}$ distributed according to *one-sided* geometric distribution with parameter α. Let $\mathbf{d}' := (\mathbf{0}^\ell, -e_{\mathbf{y}}) + \mathbf{u}' \mod q$ and consider a uniformly random $r' \in \mathbb{Z}_q^\star$. Next the algorithm proceeds as follows.
  - It first computes an encryption of $\mathbf{d}'$ under the fresh master key msk', i.e

$$\mathbf{c}_{\mathbf{d}'} := (C' = g^{r'}, D' = h^{r'}, \mathbf{E}' = \mathbf{g}^{\mathbf{d}'}\mathbf{h}'^{r'}).$$

  - It then considers the two vectors $(\mathbf{y}, 1)$ and $(\mathbf{1}^\ell, 1)$ and computes

$$\mathsf{sk}_{(\mathbf{y},1)} := (\langle \mathbf{s}, (\mathbf{y}, 1)\rangle, \langle \mathbf{t}, (\mathbf{y}, 1)\rangle),$$
$$\mathsf{sk}_{(\mathbf{1}^\ell,1)} := (\langle \mathbf{s}', (\mathbf{1}^\ell, 1)\rangle, \langle \mathbf{t}, (\mathbf{1}^\ell, 1)\rangle).$$

  - It finally computes the final partial key z as

$$z := \langle \mathbf{u}, (\mathbf{y}, 1)\rangle + \langle \mathbf{u}', (\mathbf{1}^\ell, 1)\rangle \mod q$$

  The output of the algorithm is $\mathsf{sk}_{\mathbf{y}} := (\mathbf{c}_{\mathbf{d}'}, sk_{(\mathbf{y},1)}, sk_{(\mathbf{1}^\ell,1)}, z)$.

- $\mathsf{Dec}(sk_{\mathbf{y}}, \mathbf{c}_{\mathbf{x}})$: this procedure parses $sk_{\mathbf{y}} := (\mathbf{c}_{\mathbf{d}'}, sk_{(\mathbf{y},1)}, sk_{(\mathbf{1}^\ell,1)}, z)$, $\mathbf{c}_{\mathbf{x}} = (C, D, \mathbf{E})$ and $\mathbf{c}_{\mathbf{d}'} = (C', D', \mathbf{E}')$, and proceeds as follows.
  - It first obtains

$$K_1 := \frac{\prod \mathbf{E}^{(\mathbf{y},1)}}{(C, D)^{sk_{(\mathbf{y},1)}}},\ K_2 := \frac{\prod \mathbf{E}'^{(\mathbf{1}^\ell,1)}}{(C', D')^{sk_{(\mathbf{1}^\ell,1)}}}$$

  - It then computes $\log_g((K_1 \cdot K_2)/g^z) = \langle \mathbf{x}, \mathbf{y}\rangle + e_{\mathbf{x}} - e_{\mathbf{y}}$.

4.3.3 Correctness of the scheme

A direct computation shows that $K_1 = g^{\langle \mathbf{d}, (\mathbf{y}, 1) \rangle}$, $K_2 = g^{\langle \mathbf{d}', (\mathbf{1}^\ell, 1) \rangle}$, and finally we obtain $K_1 \cdot K_2 \cdot g^{-z} = g^{\langle \mathbf{x}, \mathbf{y} \rangle + \langle e_{\mathbf{x}}, 1 \rangle + \langle -e_{\mathbf{y}}, 1 \rangle} = g^{\langle \mathbf{x}, \mathbf{y} \rangle + e_{\mathbf{x}} - e_{\mathbf{y}}}$.

Setting the parameters. We first give a precise analysis of the magnitude of the elements that are used in our scheme to properly define the parameters we need to set. There are two main issues that we have to manage at the same time: (i) to ensure that the final computation result lies in a small domain in order to be correctly decrypted (as we are using a discrete logarithm computation); and (ii) to ensure the differential-privacy of the output.

1. We restrict the input messages to some interval $\mathcal{X}$ in $\mathbb{Z}_q$ of width B_x, and the functions keys fall into some interval $\mathcal{Y}$ of width B_y. Then, in order to decrypt correctly, that is to be able to retrieve the discrete logarithm, we need to ensure that $|\langle \mathbf{x}, \mathbf{y} \rangle + e_{\mathbf{x}} - e_{\mathbf{y}}| \leq L$ for some polynomially bounded $L = \mathsf{poly}(\lambda)$. Using standard methods for computing such discrete logarithm, the exact value could be recovered in time $\tilde{O}(L^{1/2})$.

2. As explained in Section 4.1.1, we can ensure ϵ-differential privacy by choosing the parameter α of the geometric distribution as $\alpha = \exp(\frac{B_x \cdot B_y}{\epsilon})$ since, by a simple calculation, the resulting ℓ_1-sensitivity is $\Delta_{\langle , \rangle} = B_x \cdot B_y$.

3. Finally, as proved in [63], and similarly to [109], the geometric distribution has unbounded magnitude. However, since the variance of Geo is $\frac{2\alpha}{(\alpha-1)^2}$ and $\frac{\sqrt{\alpha}}{\alpha-1} \leq \frac{1}{\ln \alpha}$, the magnitude of the error added is $O(\frac{1}{\ln \alpha})$ which is $O(\frac{\epsilon}{B_x \cdot B_y})$ by our choice of α. The introduced geometric error $e_{\mathbf{x}} - e_{\mathbf{y}}$ is then bounded by $O(\frac{\epsilon}{B_x \cdot B_y})$ with high probability, and is independent of the size ℓ of the vectors. Since the input vectors are picked from the domain $\mathcal{X} = \{0, \ldots, B_x\}$ and functional vectors from the domain $\mathcal{Y} = \{0, \ldots, B_y\}$, by applying the Cauchy-Schwarz inequality, the inner-product of $\mathbf{x}$ and $\mathbf{y}$ is bounded by $\ell \cdot B_x \cdot B_y$.

To conclude on this part, if there had been no noise, the decryption would have been a successful if $\ell \cdot B_x \cdot B_y$ would have lied in some polynomial size range (which implies that $\ell \cdot B_x \cdot B_y < q$). In our case, with high probability adding our noise will imply that the discrete log computation will lie in the following

$$\left[-O\left(\frac{\epsilon}{B_x \cdot B_y} \right), O\left((\frac{\epsilon}{B_x \cdot B_y} \right) + \ell \cdot B_x \cdot B_y \right].$$

Then, to ensure correct decryption, this range size needs to be polynomial in the security parameter λ.

Proof of correctness. We now prove that our scheme verifies the statistical correctness of Definition 4.2.2 for a RFE scheme supporting the family of functions $\mathcal{F}_{\mathcal{X},\mathcal{Y},\ell,\alpha,q}$. Let $\lambda \geq 0$, let $Q = Q(\lambda)$ be a polynomial, and let $\mathbf{x}^1, \ldots, \mathbf{x}^Q$ in $\mathcal{X}^\ell$ and $\mathbf{y}^1, \ldots, \mathbf{y}^Q$ in $\mathcal{Y}^\ell$. Let $(\mathsf{msk}, \mathsf{param}) \leftarrow \mathsf{Setup}(1^\lambda)$ and let $c_{\mathbf{x}^j} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}^j)$ for all $j \in [Q]$. Finally, let $\mathsf{sk}_{\mathbf{y}^i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_{\mathbf{y}^i})$ for all $i \in [Q]$.

We focus on the real game, and thus consider the following distribution $\{\mathsf{Dec}(sk_{\mathbf{y}^j}, c_{\mathbf{x}^j})\}_{i,j\in[Q]}$ for every possible bounded input $\mathbf{x}^i$ and vector functions $\mathbf{y}^j$. Since the encryption Enc and the functional key generation KeyGen algorithms are independently executed, we first have that

$$\{\mathsf{Dec}(sk_{\mathbf{y}^j}, c_{\mathbf{x}^j})\}_{i,j\in[Q]} = \{\langle \mathbf{x}^i, \mathbf{y}^j \rangle + e_i - e_j\}_{i,j\in[Q]},$$

where e_i, e_j are distributed according to the one-sided Geometric distribution produced during Enc and KeyGen respectively. Using Lemma 4.1.1, we deduce that $\mathsf{Dec}(sk_{\mathbf{y}^j}, c_{\mathbf{x}^j})\}_{i,j\in[Q]}$ is exactly distributed as $\{\langle \mathbf{x}^i, \mathbf{y}^j \rangle + e_{i,j}\}_{i,j\in[Q]}$ for an element $e_{i,j}$ sampled according to the distribution referred as $\mathsf{Geo}(\alpha)$, which corresponds to the ideal game.

4.3.4 Simulation-Security of our Scheme

As discussed in the introduction, our proof is inspired by the techniques developed in the case of MIFE [2]. Informally, this is done by using the security of the IPFE. Since we use multiple invocations of the KeyGen algorithm, we exploit the one-Sel-SIM security of the underlying IPFE [1] in the multi-instance setting. Our security proof can be summarized by the following steps.

We first modify the encryption algorithm and encrypt a "dummy" message, i.e., the zero vector. We then inject the expected intermediate value during the generation of the first partial functional key. This technique is similar to the proof of the one-AD-SIM security of [10]. In particular, we here only use the one-SEL-SIM security for one instance.

The second step is to modify the MIFE ciphertexts generated during our functional key generation step. We here follow the same approach as before, while requiring the one-SEL-SIM security in the multi-instance setting, since there are potentially many possible functional keys in our case.

Next, we modify the generation of the partial key z to include all the above changes, and rely on some statistical arguments.

Finally, in order to give a correct simulator, we plug the excepted final result (seen as a true DP-output) into the element z.

Theorem 4.3.1. *Under the DDH assumption, the scheme described in Section 4.3.2 is one-Sel-SIM secure. More formally, there exists a simulator* Sim *such that for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT $\mathcal{B}$ *such that*

$$|\mathsf{Exp}^{real}_{\mathcal{A}}(1^\lambda) - \mathsf{Exp}^{ideal}_{\mathsf{Sim}}(1^\lambda)| \leq Q \cdot \mathsf{Adv}(\mathcal{B})^{\mathsf{DDH}}.$$

Proof. To prove our theorem, we proceed via a series of Games. Let $\mathcal{A}$ be a PPT adversary playing the One-Sel-SIM security game and let λ be the security parameter. We suppose that param is included in all below games.

Game 0. This is the real game, taken from Def. 4.2.3. For an adversary $\mathcal{A}$, since we are in the selective game, $\mathcal{A}$ starts with providing $(\mathbf{x}, st) \leftarrow \mathcal{A}(1^\lambda)$,then $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{st}) := (\mathbf{u}, \mathbf{s}, \mathbf{t}, \mathbf{h})$, $(\mathbf{c_x}, \mathsf{st}') \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}) := (C, D, \mathbf{E})$ and $\gamma \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathbf{c_x}, \mathsf{st}')$.

Game 1. In this game, we modify the generation of the ciphertext $\mathbf{c_x}$ during the encryption algorithm, using **(s,t)** in msk, as $C = g^{r_0}$, $D = h^{r_0}$, $\mathbf{E} = \mathbf{g^d C^s D^t}$, where $r_0 \in \mathbb{Z}_q^\star$. Remark that the ciphertext is identical to the one in Game 0.

We proceed similarly for the ciphertext generated during the functional key generation algorithm. In particular, we use $(\mathbf{s}', \mathbf{t}')$ in msk' to compute $\mathbf{c_{d'}}$ as $C' = g^{r'_0}$, $D' = h^{r'_0}$, $\mathbf{E}' = \mathbf{g^{d'} C'^{s'} D'^{t'}}$, for some $r'_0 \in \mathbb{Z}_q^\star$.

Game 2. We modify the generation of the two first components of the ciphertext $\mathbf{c_x}$ by generating uniformly random elements $r_0, r_1 \in \mathbb{Z}_q^\star$ and setting $C = g^{r_0}$ and $D = h^{r_0+r_1}$.

Similarly, we modify the generation of the ciphertext $\mathbf{c_{d'}}$ by generating uniformly random elements r'_0, r'_1 in $\mathbb{Z}_q^\star$ and setting $C' = g^{r'_0}$ and $D' = h^{r'_0+r'_1}$.

Game 3. In this game, we modify the encryption algorithm. First, we encrypt the element of the master secret key $\mathbf{u}$ and compute $\mathsf{sk}_{(\mathbf{y},\mathbf{1})}$ as $(\mathsf{sk}^{(1)}_{\mathbf{y}}, \mathsf{sk}^{(2)}_{\mathbf{y}})$ where $\mathsf{sk}^{(1)}_{\mathbf{y}} = \langle \mathbf{s}, (\mathbf{y}, \mathbf{1}) \rangle + \frac{1}{r_1} \cdot \langle \mathbf{u} - (\mathbf{x}, \mathbf{e_x}), (\mathbf{y}, \mathbf{1}) \rangle$ and $\mathsf{sk}^{(2)}_{\mathbf{y}} = \langle \mathbf{t}, (\mathbf{y}, \mathbf{1}) \rangle - \frac{1}{\omega \cdot r_1} \cdot \langle \mathbf{u} - (\mathbf{x}, \mathbf{e_x}), (\mathbf{y}, \mathbf{1}) \rangle$.

Observe that up to this point, if $\omega = \log_g(h)$, we have $(C, D) = (g^{r_0}, g^{\omega \cdot (r_0+r_1)})$ Moreover, we now compute $\mathbf{E} := (E_1, \ldots, E_{\ell+1})$ as, for all $i = 1, \ldots, \ell+1$, $E_i = g^{u_i + \omega \cdot r_1 \cdot t_i} \cdot h_i^{r_0}$.

Putting all together, we obtain

$$\mathbf{E}^{(\mathbf{y},1)} = g^{\langle \mathbf{u}, (\mathbf{y},\mathbf{1}) \rangle + \omega \cdot r_1 \cdot \langle \mathbf{t}, (\mathbf{y},\mathbf{1}) \rangle} \cdot (g^{\langle \mathbf{s}, (\mathbf{y},\mathbf{1}) \rangle} \cdot h^{\langle \mathbf{t}, (\mathbf{y},\mathbf{1}) \rangle})^{r_0},$$

and

$$C^{\mathsf{sk}_{\mathbf{y}}^{(1)}} \cdot D^{\mathsf{sk}_{\mathbf{y}}^{(2)}} = g^{r_0 \cdot \mathsf{sk}_{\mathbf{y}}^{(1)} + \omega \cdot r_1 \cdot \mathsf{sk}_{\mathbf{y}}^{(2)}} h^{r_0 \cdot \mathsf{sk}_{\mathbf{y}}^{(2)}}.$$

This leads to $K_1^\star = g^{\langle \mathbf{d}, \mathbf{y} || \mathbf{1} \rangle}$, where $\mathbf{d} = (\mathbf{x}, \mathbf{e_x}) + \mathbf{u}$ and the decryption still works with those modifications.

Game 4. We proceed as in the previous game for the functional key generation. We generate a uniformly random element $\mathbf{u}'$ over $\mathbb{Z}_q^\ell$, and then compute the functional key $\mathsf{sk}_{(\mathbf{1}^\ell, \mathbf{1})}$ as $(\mathsf{sk}_{\mathbf{y}}^{(1)'}, \mathsf{sk}_{\mathbf{y}}^{(2)'})$ where

$$\mathsf{sk}_{\mathbf{y}}^{(1)'} = \langle \mathbf{s}', (\mathbf{1}^\ell, 1) \rangle + \frac{1}{r_1'} \cdot \langle \mathbf{u}' - (\mathbf{0}^\ell, \mathbf{e_y}), (\mathbf{1}^\ell, \mathbf{1}) \rangle,$$
$$\mathsf{sk}_{\mathbf{y}}^{(2)'} = \langle \mathbf{t}', (\mathbf{1}^\ell, 1) \rangle) - \frac{1}{\omega \cdot r_1'} \cdot \langle \mathbf{u}' - (\mathbf{0}^\mathbf{m}, \mathbf{e_y}), (\mathbf{1}^\ell, \mathbf{1}) \rangle.$$

Again, we deduce that $K_2' = g^{\langle \mathbf{d}', (\mathbf{1}^\ell, \mathbf{1}) \rangle}$, where $\mathbf{d}' = (\mathbf{0}, \mathbf{e_y}) + \mathbf{u}'$ and again the decryption works.

Game 5. In this game, we modify the generation of z. We first generate two uniformly random elements $z_x, z_y \in \mathbb{Z}_q^\star$. We then modify the functional key generations as follows

$$\mathsf{sk}_{\mathbf{y}}^{(1)} = \langle \mathbf{s}, (\mathbf{y}, \mathbf{1}) \rangle + \frac{1}{r_1} \cdot (\langle \mathbf{u}, (\mathbf{y}, \mathbf{1}) \rangle - z_x)$$
$$\mathsf{sk}_{\mathbf{y}}^{(2)} = \langle \mathbf{t}, (\mathbf{y}, \mathbf{1}) \rangle) - \frac{1}{\omega \cdot r_1} \cdot (\langle \mathbf{u}, (\mathbf{y}, \mathbf{1}) \rangle - z_x)$$
$$\mathsf{sk}_{\mathbf{y}}^{(1)'} = \langle \mathbf{s}', (\mathbf{1}^\ell, \mathbf{1}) \rangle + \frac{1}{r_1'} \cdot (\langle \mathbf{u}', (\mathbf{1}^\ell, \mathbf{1}) \rangle - z_y)$$
$$\mathsf{sk}_{\mathbf{y}}^{(2)'} = \langle \mathbf{t}', (\mathbf{1}^\ell, \mathbf{1}) \rangle) - \frac{1}{\omega \cdot r_1'} \cdot (\langle \mathbf{u}', (\mathbf{1}^\ell, \mathbf{1}) \rangle - z_y).$$

Finally, we compute $z := z_x + z_y - \langle (\mathbf{x}, \mathbf{e_x}), (\mathbf{y}, \mathbf{1}) \rangle - \langle (\mathbf{0}^\ell, \mathbf{e_y}), (\mathbf{1}^\ell, \mathbf{1}) \rangle$.

Game 6. This is the ideal Game. Giving some auxiliary information aux which corresponds to the functionality (i.e a true DP output), the simulator Sim is given by $(\mathcal{S}_\mathsf{S}^\star, \mathcal{S}_\mathsf{E}^\star, \mathcal{S}_\mathsf{KG})$ and defined as follows.

- $\mathcal{S}_\mathsf{S}^\star(1^\lambda, 1^\ell)$: the algorithm is identical to Setup, except that we set $\omega := \log_g(h)$. It outputs $\mathsf{param}^\star = \mathsf{param}$ and $\mathsf{st}^\star := (\omega, \mathbf{u}^\star, \mathbf{s}^\star, \mathbf{t}^\star, \mathbf{h}^\star)$.

- $\mathcal{S}^{\star}_{\mathsf{E}}(\mathsf{msk}^{\star})$: it generates the ciphertext by sampling r_0, r_1 uniformly at random and computing $\mathbf{c}^{\star} := (C^{\star}, D^{\star}, \mathbf{E}^{\star})$ where $C^{\star} = g^{r_0}$, $D^{\star} = h^{r_0+r_1}$ and $\mathbf{E}^{\star} = g^{\mathbf{u}^{\star}} C^{\mathbf{s}^{\star}} D^{\mathbf{t}^{\star}}$.

  In addition, we add to the state information $\mathsf{st}^{\star}$ the element (r_0, r_1).

- $\mathcal{S}^{\star}_{\mathsf{KG}}(\mathsf{st}^{\star}, \mathsf{st}', \mathbf{y})$: on input $\mathsf{st}^{\star} = (\omega, \mathbf{u}^{\star}, \mathbf{s}^{\star}, \mathbf{t}^{\star}, \mathbf{h}^{\star}, r_0, r_1)$, this algorithm uses the $\mathsf{KeyIdeal}$ oracle that returns a vector $\mathbf{y}$ and the evaluation function $\mathsf{aux} := \langle \mathbf{x}^{\star}, \mathbf{y} \rangle + e$ for a noise e distributed according to $\mathsf{Geo}(\alpha)$. Then, $\mathcal{S}^{\star}_{\mathsf{KG}}$works as follows.

  It first runs $\mathsf{st}'' := (\omega, \mathbf{u}', \mathbf{s}', \mathbf{t}', \mathbf{h}') \leftarrow \mathcal{S}^{\star}_{\mathsf{S}}(1^{\lambda}, 1^{\ell})$.

  It computes the ciphertext by first sampling $r'_0, r'_1 \in \mathbb{Z}^{\star}_q$ uniformly at random and computing $\mathbf{c}_{\mathbf{u}'} := (C', D', \mathbf{E}^{\star})$ where

$$C' := g^{r'_0},\ D' := h^{r'_0+r'_1},\ \mathbf{E}^{\star} := g^{\mathbf{u}'} C'^{\mathbf{s}'} D'^{\mathbf{t}'}.$$

  It generates uniformly at random $z_x, z_y \in \mathbb{Z}_q$, and sets

$$z^{\star} := z_x + z_y - \mathsf{aux}.$$

  It respectively computes $sk^{\star}_{(y,1)} = ((\mathbf{y}, 1), \mathsf{sk}^{(1)}_{\mathbf{y}}, \mathsf{sk}^{(2)}_{\mathbf{y}})$ where

$$\mathsf{sk}^{(1)}_{\mathbf{y}} = \langle \mathbf{s}^{\star}, (\mathbf{y}, \mathbf{1}) \rangle + \frac{1}{r_1} \cdot [\langle \mathbf{u}^{\star}, (\mathbf{y}, \mathbf{1}) \rangle - z_x],$$
$$\mathsf{sk}^{(2)}_{\mathbf{y}} = \langle \mathbf{t}^{\star}, (\mathbf{y}, \mathbf{1}) \rangle - \frac{1}{\omega \cdot r_1} \cdot [\langle \mathbf{u}^{\star}, (\mathbf{y}, \mathbf{1}) \rangle - z_x],$$

  and $sk'_{(\mathbf{1}^{\ell},1)} = ((\mathbf{1}^{\ell}, 1), \mathsf{sk}^{(1)'}_{\mathbf{y}}, \mathsf{sk}^{(2)'}_{\mathbf{y}})$ where

$$\mathsf{sk}^{(1)'}_{\mathbf{y}} = \langle \mathbf{s}', (\mathbf{1}^{\ell}, \mathbf{1}) \rangle + \frac{1}{r'_1} \cdot [\langle \mathbf{u}', (\mathbf{y}, 1) - z_{\mathbf{y}} \rangle],$$
$$\mathsf{sk}^{(2)'}_{\mathbf{y}} = \langle \mathbf{t}', (\mathbf{1}^{\ell}, \mathbf{1}) \rangle) - \frac{1}{\omega \cdot r'_1} \cdot [\langle \mathbf{u}', (\mathbf{y}, 1) - z_{\mathbf{y}} \rangle].$$

- The algorithm finally returns the simulated functional keys as

$$(\mathbf{c}_{\mathbf{u}'}, sk^{\star}_{(\mathbf{y},1)}, sk'_{(\mathbf{1}^{\ell},1)}, z^{\star}).$$

Analysis. Consider a challenger $\mathcal{C}$ that flips a coin $b \in \{0, 1\}$. If $b = 0$, it interacts with the adversary $\mathcal{A}$ as in $Game_i$, else it interacts as in $Game_j$. At the end of the

interaction $\mathcal{A}$ will have to make its guess b'. We define $\mathsf{Adv}(\mathcal{A})_{ij} := |\mathsf{Pr}[\mathsf{b}' = \mathsf{b}] - 1/2|$, for $i = 0, \ldots, 5$ and $j = i + 1$.

From Game 0 to Game 1. As Game 0 and Game 1 are identically distributed, we have $\mathsf{Adv}(\mathcal{A})_{01} = 0$.

From Game 1 to Game 2. Regarding this transition, the ciphertext components in both Game 1 and Game 2 are computed as similarly. As in [11], under the DDH assumption, this modification does not significantly affect the adversary's view. More formally, we have the following lemma.

Lemma 4.3.2. *There exists an adversary $\mathcal{B}$ such that* $\mathsf{Adv}(\mathcal{A})_{12} \leq Q \cdot \mathsf{Adv}(\mathcal{B})^{\mathsf{DDH}} + \frac{1}{q-1}$.

Proof. At first, we do not take into account the changes that we have done in the functional key generation. This will be done in the second part of the proof.

Remark that $\mathbf{E}$ has exactly the same distribution in the two games. We then only focus on C and D. Consider an adversary $\mathcal{A}$ that distinguish between Game 1 and Game 2. We build an adversary $\mathcal{B}$ that, using $\mathcal{A}$, can break the DDH assumption. Let (g^x, g^y, Z) be a DDH challenge. $\mathcal{B}$ generates all the parameters as in the description of the scheme, except for the element h which is build as $h := g^x$. $\mathcal{B}$ flips a bit b. If $b = 0$, it forms $(C, D) := (g^{yr_0}, Z^{r_0})$ for some uniformly random $r_0 \in \mathbb{Z}_q^\star$. Else, it computes $(C, D) := (g^{yr_0}, Z^{r_0} \cdot h^{r_1})$ for uniformly random $r_0, r_1 \in \mathbb{Z}_q^\star$. The other elements remain unchanged. $\mathcal{B}$ finally provides to $\mathcal{A}$ all these elements and returns to its challenger whatever $\mathcal{A}$ returns. Let us analyse this output by considering two cases.

- Suppose that $Z = g^{xy}$. If the bit $b = 0$, then $(C, D) = (g^{yr_0}, g^{yxr_0}) = (g^{yr_0}, h^{yr_0})$ which is distributed as in Game 1. If the bit $b = 1$, then $(C, D) = (g^{yr_0}, h^{yr_0} \cdot h^{r_1})$ which is distributed as in Game 2.
- Suppose now that Z is random. If $b = 0$, D has the same distribution as in Game 1 since $r_1 \in \mathbb{Z}_q^\star$. Else, $D = h^{r_0 + r_1}$, which corresponds as the Game 2.

We conclude that the advantage of $\mathcal{B}$ is exactly the one of $\mathcal{A}$ in distinguishing between Game 1 and Game 2.

If we now consider the changes that we made on the functional key generation, the same argument remains valid. We however have to consider the multi-instance setting of the same problem since the adversary could ask for Q different functional keys. A standard hybrid argument over all these instances induces a security loss proportional to Q. Then, we have $\mathsf{Adv}(\mathcal{A})_{12} \leq Q \cdot \mathsf{Adv}(\mathcal{B})^{\mathsf{DDH}}$. □

From Game 2 to Game 3. This transition corresponds to a modification of the master secret key of the IPFE, from $(\mathbf{s}, \mathbf{t})$ to some $(\bar{\mathbf{s}}, \bar{\mathbf{t}})$ where $\bar{\mathbf{s}} = \mathbf{s} + \frac{1}{r_1} \cdot (\mathbf{u} - (\mathbf{x}, e_{\mathbf{x}}))$ (mod q), and $\bar{\mathbf{t}} = \mathbf{t} - \frac{1}{\omega \cdot r_1} \cdot (\mathbf{u} - (\mathbf{x}, e_{\mathbf{x}}))$ (mod q).

Both have obviously the same distribution. Indeed, the elements $\mathbf{u}$ and $e_{\mathbf{x}}$ are picked after the adversary sends its selective challenge $\mathbf{x}$, and are therefore independent of it. In particular, we deduce this simple transformation does not affect the adversary's view. Note that this is however crucial to consider a selective challenge in this transition. We then deduce that $\mathsf{Adv}(\mathcal{A})_{23} = 0$.

From Game 3 to Game 4 In this Game, we use the same strategy as in the previous transition. However, we have to manage the fact that the adversary could ask for functional keys for a vector $\mathbf{y}$ of its choice. In particular, we have to consider an hybrid argument across all the functional keys' requests. Indeed, we obtain the following lemma.

Lemma 4.3.3. *We have* $\mathsf{Adv}(\mathcal{A})_{34} = 0$.

We consider Q hybrids $\mathrm{Game}_{3,i}$, $i \in [Q]$, one for each call to the $\mathcal{S}^{\star}_{\mathsf{K}}$ oracle by the adversary. We then modify each answer, for each request $\mathbf{y}^i$, $i \in [Q]$, as shown in Game 4. For each of them, let $\mathbf{u}'^i$ be a random element in $\mathbb{Z}_q^{\ell}$ and let $\mathbf{d}'^i := (0, e^i_{\mathbf{y}}) + \mathbf{u}'i$ for a $e^i_{\mathbf{y}}$ which is *independently* chosen (unlike the previous game). As in the previous step, each new transition corresponds to a modification of the master secret key of the IPFE, from $(\mathbf{s}'^i, \mathbf{t}'^i)$ to some $(\bar{\mathbf{s}'^i}, \bar{\mathbf{t}'^i})$ where

$\bar{\mathbf{s}'^i} = \mathbf{s}'^i + \frac{1}{r'^i_1} \cdot (\mathbf{u}'^i - (0, e^i_{\mathbf{y}}))$ and $\bar{\mathbf{t}'^i} = \mathbf{t}'^i - \frac{1}{\omega \cdot r'^i_1} \cdot (\mathbf{u}'^i - (0, e^i_{\mathbf{y}}))$.

Again, those distributions are obviously equals and we deduce that the advantage for each transition is equal to 0. We can then conclude that $\mathsf{Adv}(\mathcal{A})_{34} = 0$.

From Game 4 to Game 5. Here, we use the fact that for all $\mathbf{y}$ in the function query space, the following distributions over $\mathbb{Z}_q$ are equals: $\{z_x - \langle (\mathbf{x}, e_x), (\mathbf{y}, 1) \rangle\}$ and $\{z_x\}$, and also $\{z_y - \langle (\mathbf{0}^{\ell}, e_y), (\mathbf{1}^{\ell}, 1) \rangle\}$ and $\{z_y\}$.

Indeed, for any function query $\mathbf{y}$, the geometric noises e_x, e_y are generated independently of the other quantities and the elements z_x, z_y are uniformly chosen at random, independently of the challenge $\mathbf{x}$ (we rely on the fact that the games are selective). We deduce that $\mathsf{Adv}(\mathcal{A})_{45} = 0$.

From Game 5 to Game 6. Game 6 is obviously a rephrasing of Game 5, using

$$\mathsf{aux} := \langle (\mathbf{x}, \mathbf{e_x}), (\mathbf{y}, \mathbf{1}) \rangle - \langle (\mathbf{0}^{\ell}, \mathbf{e_y}), (\mathbf{1}^{\ell}, \mathbf{1}) \rangle.$$

Thanks to the statistical correctness condition that we discussed in sec.4.1.1, we deduce that the Game 5 and Game 6 are identical. In particular, $\mathsf{Adv}(\mathcal{A})_{56} = 0$.

Since Game 6 is the Ideal experiment, by combining all the previous results we have $|\mathsf{Exp}^{\mathrm{real}}_{\mathcal{A}}(1^\lambda) - \mathsf{Exp}^{\mathrm{ideal}}_{\mathsf{Sim}}(1^\lambda)| \leq Q \cdot \mathsf{Adv}(\mathcal{B})^{\mathsf{DDH}}$, which concludes the security proof.

□

We finish this section by the following possible improvement for the bound in the conclusion of Thm. 4.3.1. Indeed, we have a security loss of Q where considering an hybrid approach. In fact, it is also possible to build an adversary $\mathcal{B}$ such that $\mathsf{Adv}(\mathcal{A})_{12} \leq \mathsf{Adv}(\mathcal{B})^{\mathsf{DDH}} + \frac{1}{q-1}$. For this, consider the transition between **Game 2** and **Game 3**. We can use the random self reducibility of DDH by considering a variant which is the *Q-Fold DDH assumption* as studied in [55]. In such Q-fold DDH, the *challenge* consists in Q independent ones from the DDH assumption, which is precisely our case. We then obtain in this situation a more tight construction and can argue that $\mathsf{Adv}(\mathcal{A})_{12} \leq \mathsf{Adv}(\mathcal{B})^{\mathsf{DDH}} + \frac{1}{q-1}$.

4.3.5 Towards a construction from any IPFE.

We provide some intuitions on how our construction could be generalized with generic IPFE. In addition, we give evidence on why our construction could be proven secure but we leave a formal proof in a future work.

Security for private-key 2IPFE. For the two-input IPFE setting (denoted hereafter 2IPFE), the situation is different. An adversary could *mix-and-match* different messages and in the public key setting, it could learn much more information than the basic evaluation of the function (for example by encrypting any $\mathbf{x}^\star$ and learning $\langle(\mathbf{x}_1, \mathbf{x}^\star), (\mathbf{y}_1, \mathbf{y}_2)\rangle$). These capabilities must enforce additional restrictions that could make in some situation the scheme useless (we refer to [68] for a discussion).

As in the context of classical IPFE, one can consider two different flavours of security: simulation or indistinguishability . The simulation security for the general two-input setting is more difficult to deal with, especially in the public-key setting. In 2IPFE, the ideal functionality consists of the following $(\mathbf{y}_1, \mathbf{y}_2, \langle(\mathbf{x}_1^j, \mathbf{x}_2^j), (\mathbf{y}_1, \mathbf{y}_2\rangle))$ for any possible messages per slot j and any functional keys. The simulation security paradigm is quite strong and yet no known general simulation secure constructions of 2IPFE (in the multi-instance setting) are known.

There exists 2IPFE secure [2, 3] in an indistinguishable type security. However, we used simulation type arguments in our DDH instantiation and we only need a one-time

simulation secure 2IPFE which exists as builded in [2]. As a side, it is a interesting challenge to build a simulation secure 2IPFE (or multi-input IPFE).

Fortunately, the needed 2IPFE for our construction (i.e. one selective simulation secure) exists and is given in the high-level description (for $n = 2$) of Sec. 4.3.1.

Description of our solution from 2IPFE. As for the DDH construction, the idea is to encapsulate the one-time construction into an IPFE. In particular, consider $\mathsf{IPFE} := (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ which is a secret key single-input IPFE and a one-time scheme $\mathsf{2IPFE} = (\mathsf{Setup}^{\mathsf{2IPFE}}, \mathsf{Enc}^{\mathsf{2IPFE}}, \mathsf{KeyGen}^{\mathsf{2IPFE}}, \mathsf{Dec}^{\mathsf{2IPFE}})$. In the following, we assume that IPFE and 2IPFE handle vectors of length $\ell + 1$.

Our RIPFE follows the same structure as for the DDH construction. We maintain the notation for our previous construction.

- $\mathsf{Setup}^{\mathsf{RIPFE}}(1^\lambda)$: this algorithm generates the master secret key msk. First, compute $\mathbf{u}_1 \leftarrow\$ \mathsf{Setup}^{\mathsf{2IPFE}}(1^\lambda, \ell + 1, 1)$ and $(\mathsf{param}, \mathsf{msk}_1^{\mathsf{IPFE}}) \leftarrow \mathsf{Setup}^{\mathsf{IPFE}}(1^\lambda, 1^{\ell+1})$. This algorithm returns
$$\mathsf{msk} := (\mathsf{param}, \mathbf{u}_1, \mathsf{msk}_1^{\mathsf{IPFE}}).$$

- $\mathsf{Enc}^{\mathsf{RIPFE}}(\mathsf{msk}, \mathbf{x})$: select a random $e_{\mathbf{x}}$ according to the distribution $\mathsf{Geo}^+(\alpha)$, then returns the encryption of $\mathbf{x}$ as
$$\mathsf{c}_1 := \mathsf{Enc}^{\mathsf{IPFE}}(\mathsf{msk}_1^{\mathsf{IPFE}}, \mathsf{Enc}^{\mathsf{2IPFE}}(\mathbf{u}_1, (\mathbf{x}, e_{\mathbf{x}}))).$$

- $\mathsf{KeyGen}^{\mathsf{RIPFE}}(\mathsf{msk}_1, y)$: we first generate a fresh parameter $\mathbf{u}_2 \leftarrow\$ \mathsf{Setup}^{\mathsf{2IPFE}}(1^\lambda, \ell + 1, 2)$ and $(\mathsf{param}, \mathsf{msk}_2^{\mathsf{IPFE}}) \leftarrow \mathsf{Setup}^{\mathsf{IPFE}}(1^\lambda, 1^{\ell+1})$. Select a random $e_{\mathbf{y}}$ according to distribution $\mathsf{Geo}^+(\alpha)$. Then, compute the following quantities.
$$\mathsf{c}_2 \leftarrow\$ \mathsf{Enc}^{\mathsf{IPFE}}(\mathsf{msk}_2^{\mathsf{IPFE}}, \mathsf{Enc}^{\mathsf{2IPFE}}(\mathbf{u}_2, (\mathbf{0}^\ell, -e_{\mathbf{y}})))$$
$$\mathsf{sk}_1 \leftarrow\$ \mathsf{KeyGen}^{\mathsf{IPFE}}(\mathsf{msk}_1^{\mathsf{IPFE}}, (\mathbf{y}, 1))$$
$$\mathsf{sk}_2 \leftarrow\$ \mathsf{KeyGen}^{\mathsf{IPFE}}(\mathsf{msk}_2^{\mathsf{IPFE}}, (\mathbf{1}^\ell, 1))$$
$$z \leftarrow\$ \mathsf{KeyGen}^{\mathsf{2IPFE}}(\mathbf{u}_1, \mathbf{u}_2, ((\mathbf{y}, 1), (\mathbf{1}^\ell, 1)))$$
Finally, return $sk_y := (\mathsf{c}_2, \mathsf{sk}_1, \mathsf{sk}_2, z)$.

- $\mathsf{Dec}^{\mathsf{RIPFE}}(sk_{\mathbf{y}}, \mathsf{c}_1)$: Parse $sk_{\mathbf{y}} := (c_2, sk_1, sk_2, z)$. First run, $\mathbf{d}_i = \mathsf{Dec}^{\mathsf{IPFE}}(\mathsf{c}_i, sk_i)$ for $i = 1, 2$. Then compute $\mathsf{Dec}^{\mathsf{2IPFE}}(z, \mathbf{d}_1, \mathbf{d}_2)$

Correctness. For the correctness, remark informally that c_1 encrypts the vector $(\mathbf{x}, \mathbf{e_x})$ and c_2 encrypts $(\mathbf{0}^\ell, -e_\mathbf{y})$. In particular, the decryption recovers the quantity

$$\langle (\mathbf{x}, e_\mathbf{x}), (\mathbf{y}, \mathbf{1}^\ell) \rangle + \langle (\mathbf{0}^\ell, -e_\mathbf{y}), (\mathbf{1}^\ell, \mathbf{1}) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + e_\mathbf{x} - e_\mathbf{y} \mod L,$$

for some module L. Similarly to the previous DDH case, the correctness condition of our scheme is similar and we do not provide the full details. We notice however that we need a 2IPFE and IPFE scheme that ensure perfect correctness in order to provide the statistical correctness.

Additional parameter selection. Observe that similarly to the DDH construction, we could provide a precise analysis of the magnitude for the used elements in our scheme. For this purpose, remark that unlike DDH, the decryption depends on the underlying IPFE. For instance, it is not necessarily essential to ensure that the final computation lies in a small domain. For example, the Paillier-based IPFE scheme of [11] implies an efficient decryption process. Similarly, the CL-based IPFE (built upon CL homomorphic scheme that we used in chapter 3) also enjoys this property. More importantly, to define the parameters for DP, one has to ensure that the output is large enough to be compatible with the requirement for DP.

On the security of the scheme. Intuitively, the proof of security follows the same proof path as the previous DDH construction. We do not provide in this thesis a full proof of this generalization and leave it for a future work. We conjecture in the sequel the required security properties in order to obtain a one selective simulation secure scheme.

- Several queries could be requested for different functional keys, which means that we need a one slot security for the ciphertext (during database encryption) consisting of one instance of the One-Sel-SIM security of the 2IPFE scheme.
- In addition, we have the possibility to use this encrypted element several times, with an unlimited number of functional key queries (using as many instances as needed). It follows that we have to use the One-Sel-Sim security of IPFE and the $\mathsf{2IPFE}^{\mathsf{ot}}$ in the multi-instance setting. This property is verified by classical IPFE schemes.
- Finally, as in the DDH case, embedding the functionality (which is a noisy DP inner-product) into the functional keys in the proof requires some structural homomorphism which are also verified by classical instances of popular IPFE schemes.

4.4 Conclusion

This chapter proposes a construction that brings together techniques from functional encryption and differential privacy literatures. Our result can be thought of as a design of a randomized functional encryption scheme where instead of using generic results, it proposes a new randomized inner product functional encryption scheme which is specifically tuned to work with the geometric distribution and is thus DP-compliant. That is, an analyst does not exactly learn the output of the inner-product but only a differentially private version of it.

We present several new techniques in order to obtain these results. Special cares were taken in order to integrate the desired noises where thinking about the security definitions. In addition, while simple solutions would be to consider the noise directly as a component in the functional key of a standard IPFE scheme (implying function hiding schemes), we showed that this is not fully satisfactory. Our proposed solution relies on a noise splitting technique particularly suitable with the use of a multi-input IPFE.

We believe that our solution provides an interesting new path to obtain non-interactive differential encrypted database and we show that IPFE is a powerful primitive that could be extended to handle more advanced primitives.

CHAPTER 5

COMPUTING FAULT-TOLERANT PRIVATE STATISTICS FOR MOBILE USAGE

Overview of the problematic

In this section, we propose a solution for solving the problem of aggregating data from different sources while maintaining the fault-tolerance of the scheme. We start from prior works on this subject but relies on functional encryption. In particular, we use one variant called *dynamic multi-client* FE which allows to aggregate a set of data issued from a group of predetermined users. We show how to modify and implement this primitive in order to fulfil the requirements of a practical scenario. Namely, we consider the case where there is possibly some faulty users. We demonstrate in this chapter how it is possible to collect some mobile data users and aggregate them, while reducing to a minimum the information leakage to the involved parties.

We first start by describing our architecture, giving the different actors that participate to this general protocol and highlighting the power of any potential malicious behaviour from each entity. Moreover, we will discuss some possible security limitations.

Then, we exhibit our secure proposition to solve this problematic. At a high-level, we embed the construction of Chotard et al. [42] (for the sum function) with a modification using some ideas from the fault-tolerant scheme of Chan et al. [39].

Finally, we show an instantiation of the underlying primitives using classical symmetric in a bilinear environment.

5.1 Our case study

Consider how to provide an additional tool to a team of social scientists in order to improve their statistical analysis by having a fine-grained access to their mobile data usage. One important requirement is to provide a privacy-preserving solution to control how much information is leaked during a particular study.

The privacy concerns that we discussed are not limited to this specific example. This work starts and was part of one of the problems addressed by the European research project **PAPAYA** [1]. It aims to question security and privacy concerns on situations where some are delegated to an untrusted party.

There is of course other possible situations when our preoccupation could be relevant. A particular example is given by *Telemetry* which is an old concept and a central tool in many industrial or public areas. Transportation, logistics, energy monitoring...etc. Several sensible systems have to collect a huge amount of information in order to exploit them and enhance the underlying usage. With the rise and evolution of connected devices, it makes no doubt that future practical companies must take into account that their working environment will include multiple devices. Each device will report their individual data usage and transmit over some untrusted networks. Similarly to the team of social scientists, this comes with a challenging task of being (legally) compliant with the privacy expectations (or regulations) about the data treatment.

In fact, in the PAPAYA project, we also consider the situation where we separate an aggregator entity (that collects data) from the party who wishes to obtain the study result. This could correspond to a third party platform initiated by a company or public service where each time a study is required, the platform permits to collect the data with for the production of the desire study.

We consider a system, that we called **WeStat**, where three types of *actors* are possible:

- a set of *individuals* or *users* that will obtain a notification for each new study. After having the precise information about the general context of it, they eventually provide their participation consent leading to an aggregation step and their data without knowing in advance which other users are going to participate;

- an *Aggregator* that can obtain individuals' data so as to perform the statistics in a privacy-preserving manner. It makes the interface with *Requestors* to prepare the study and to permit them to obtain the results;

[1] https://www.papaya-project.eu/about

- *Requestors*, that can contact the Aggregator for a new study, filling some on-line form (for eg. the type of statistics) with the parameters for a study.

Moreover, users must send the minimum number of messages. Another important feature for practical situations, that will be crucial for our WeStat architecture, is *fault-tolerance*. If one user fails, the final result should still be possible to compute over the remaining users without compromising their privacy. The security treatment here is regarding to these two above entities (Aggregator and Requestor).

As an additional requirements, we only consider statistics that can be managed using an *inner prodcut* between a vector where each component is the entry of one individual and another vector related to the study. For example, the sum (resp. mean) of all the entries can be computed with vector $(1, \cdots, 1)$ (resp. $(1, \cdots, 1)$ divided by the number of entries). More complex statistics can also be computed using inner-product, such as e.g. linear regression or data classification.

Using FE and limitations. By its definition, FE is a choice for delegating an evaluation to an external entity while ensuring that only the result of the computation is released. However, our context is a little different and fits into answering the above privacy concerns about data aggregation for *multiple sources* that communicate with *untrusted* entities.

Consider, as a potential implementation for WeStat, to set up a public key FE such that the master secret key owner is the aggregator (which is responsible of the data aggregation platform). We can then make users sending their encrypted data using FE. However, basic FE schemes consider *single input*[2] functions. Hence, this implies to decrypt every single ciphertext for a certain function and eventually combine them for the aggregation step. Moreover, it is not fully satisfactory to use a *centralized* solution. Recall that individuals do not trust the aggregator which by definition can generate any functional keys. In particular, they are not automatically convinced that the function for which they give their consent is indeed the one used for the study.

Using known primitives. Of course, other solutions are possible. The literature dealing with the desired functionality, i.e. privately computing over the encrypted data, is huge and differs for one reference to the other depending on the architecture one has to consider. As a high-level, they are all special purposes of a multi-party computation protocol. While there are some generic constructions for general functionalities, previous

[2]Here single mean that f took one input, which could be a bit, a vector of bits, or more general object.

works (for example [21] and included references) try to obtain more efficient solutions by exploiting the structure of some specific functions, such as summing over the inputs or considering inner-products of data with some vectors. We provide in the following some solutions for WeStat.

- (Threshold[3]) fully homomorphic encryption FHE provides a way of computing (theoretically) any function over encrypted data. A simple solution consists of the Requestor generating a public key for this primitive and consider each user sending an encryption of their inputs using (Threshold) FHE with requestor's public key. Then, the Aggregator can compute a *study function* over the received ciphertexts and can reveal the resulted homomorphic computation to the Requestor. Finally, Requestor (or threshold requestors) decrypt the resulted ciphertext to obtain the desired statistics.

- (Threshold, function) secret-sharing schemes permits to split a secret input into different shares. Then, it is possible from a (threshold) number of shares to reconstruct function over the shares [107] without revealing any additional information. With this primitive, a user could split its input to the Aggregator and the Requestor Then, it is possible to compute the statistics by reconstructing the received shares.

Even if these two sketched solutions seems to fit with our problematic, we however remark that there is an inherent limitation of learning more functions that it is permitted. Hence, they do not provide evidence that indeed the *right* computation is carried out. Moreover, we need to impose that at least one party is trusted during the execution of protocol.

A way to circumvent this issue is to use ZKPoK to convince other parties that computations were done correctly, and reject the trust if it is not verified. It would represent a cost (especially if the users has some performance limitation) and we choose to focus on solutions without generic ZKPoK.

From variants of FE. Since we are considering multiple source of data (users) for WeStat, there is a generalization of FE that could be used for our case.

Decentralized Multi-Client FE (here after DMCFE) [41] is a symmetric-variant (each user encrypts its data under some label[4]) of FE with the particularity of generating functional keys in a completely decentralized way by the users (interactively or not). A

[3] A variant where it is possible to decrypt a ciphertext where at least a subset some decryptors provide their secret keys.

[4] Which corresponds to time, a study...etc.

decryptor can recover a function over the data only if she has access to all the ciphertexts and the functional key generated by *all* the users (under the same label).

At first glance, it is suitable to use DMCFE for our problematic: *the users dictate the nature of the computation*. For WeStat, all users interact during an initial Setup phase[5] and then use an encryption algorithm to send their message. Whenever a function needs to be evaluated by a Requestor, a functional key is generated. Having so, any decryptor cannot obtain other than the predetermined function.

Viewing the specifications of DMCFE, we remark however that it does not give a complete answer to our problematic and we highlight the main reasons in the following.

1. Each user has to know in advance the other participants in order to generate its parameters. The set of users is then fixed and if one user does not contribute with its *part* of a functional key (or ciphertext), it is not possible (by definition) to obtain the evaluated function over the remaining users. In our case, we tolerate the idea that some users could be dropped and will not participate during the computation. The DMCFE based solution is not *fault-tolerant*.

2. The roles of the Aggregator and the Requestor are not clear. In fact, examinating the definition of DMCFE allows us to conclude that this primitive is more suitable for a situation where there is a set of users and *one decryptor* (for example the Requestor). Providing an aggregation step needs to examinate in a *non black-box* manner the details of DMCFE, which is not clear from known instantiations.

3. In a recent work, Chotard et al. introduce the notion of dynamic DMCFE which provide more flexibility to users for their group belonging. Roughly speaking each user can decide to contribute in the scheme for any subset of users of its choice. The evaluation of the data is revealed only when all parties in the *same* group have sent their contributions.

In other context, the *Ad-hoc multi-input* FE [9] is an FE where users can join a system (by encrypting input) on-the-fly and functional keys can be generated in a decentralized way, by each client, without any interaction. The main difference with DMCFE of [42] is that each user encrypts its input once and for all using some public parameters (independent from other users) and the work is then transferred into the functional key generation procedure. When asked, the user needs to know the set of users for which the

[5] In fact, there is a minimum of interaction in order to obtain the public and global parameters (handled by some trusted party for example). After this initial Setup, if there is no *direct* interaction between users, then the scheme is considered as totally decentralized.

functional key is issued. To recover the result, it is possible to accumulate any choice of ciphertexts, then asking the different involved parties to send the partial decryption keys corresponding to their ciphertext.

Notice that in particular, after a functional key generation, if one user fails to send one partial decryption key, then we have the same problem of *users vs. one decryptor* discussed above.

From DMCFE to our solution. In fact, we exploit ideas from variants of FE in the multi-client setting. More precisely, we puts some high-level ideas of our construction.

- To handle faulty users, we use the *binary tree* idea of the *Private Stream Aggregation*[6] scheme of Chan et al. [39] that we modify according to our WeStat specification. Informally, we add redundancy by regrouping users into different groups. Then, we consider a Dynamic DMCFE over each group. If some user fails, than there is a way to recover remaining users.

- MCFE based constructions do not consider two entities (Aggregator and Requestor) by essence as we do. In addition, all discussed multi party solutions suppose at least a level of trust in one of the two entities. For our solution, we suppose there is no collusion between the Requestor and the Aggregator.

The basic composition of the schemes of Chotard et al. [42] with Chan et al. [39] does not answer to our model because the Aggregator can learn the sum of each of the involved group. However, we modify the construction and leverage the Requestor by carefully adding some *masks* that will hide the intermediate values.

In fact, regarding WeStat, we only need a DSum (for Distributed Sum) [42] which is a particular class of Dynamic DMCFE [7], where there is no functional key generation but it is possible to obtain the sum of inputs of a group of users only when all their ciphertexts are available.

Summing up our contribution, the main challenge of our WeStat solution is to build a protocol implementing a *decentralized sum protocol robust to fault-tolerance*. In particular, our solution combines the DSum from Chotard et al. [42] for different subgroups and the fault-tolerant idea of Chan et al. [39].

[6] PSA is a centralized solution (with a trusted aggregator) of the problem of aggregating stream data.
[7] Authors provide a construction for the more general case of inner-product.

Organization of this chapter

We start by providing a description of our WeStat architecture in Sec. 5.2. Then, we develop in Sec. 5.3 the security model concerning the Aggregator and the Requestor, discussing at the same time the limitation of the model. In Sec. 5.4, we propose a construction of WeStat from any DSum scheme. Finally, we discuss one instantiation of the scheme from bilinear maps in Sec.5.5.

5.2 General definition of WeStat

For the ease of the description, our model considers three phases and for each of them, we describe a specific computation done by each entity. At high-level, there is an initial setup for registration and study creation, an accumulation process for collecting data from users and a final computation to recover the results.

Involved parties and general terminology. For the rest of this chapter, we fix a *set of users* $\{\mathcal{U}_i\}_{i\in\mathcal{I}}$ ($\mathcal{I}$ represents the set of all users using the WeStat service), an *Aggregator* Agg with the role of collecting all the informations from the users and a *Requestor* R asking for a new study. These entities are represented as PPT algorithms. A *label* represents in the following parameters that are related to a particular study. Generally speaking, a label could correspond to the time where the protocol is executed or some public information that are used in order to authenticate all involved parties.

A *message* designates the information sent between each entity, while the *data* corresponds to the user's input to be evaluated.

WeStat architecture. We provide in the following the general description of the WeStat protocol. For each of the following phase, we describe algorithms or protocols that are *sequentially* executed, while specifying at the same time the underlying concerned entities. We suppose also that each communication (i.e. exchanged messages) between the involved parties are executed through a secure and authenticated transmission channel. Let λ be a security parameter.

Phase 1: study creation. This phase generates the starting parameters for a new requested study. In particular, R is interested in the evaluation of a function f over inputs.

Setup(λ): a PPT algorithm, executed by the aggregator Agg, with the help of R. It takes as input a security parameter λ and outputs a global public parameters

param, which is included in all of the involved algorithms executed by all the entities.

$\mathsf{StudyRq}(\mathsf{Agg}(\mathsf{param}), \mathsf{R}(\mathsf{param}, \ell))$: this is a two-party protocol executed between the requestor R, interested in a new study, and the aggregator Agg that agrees on starting it. During this preliminary protocol, they generate a label $\ell$to perform the study. In addition, during this step, R and Agg generate then output a pair of keys. If so, for the label ℓ, we specify $(\mathsf{sk}_{\mathsf{R},\ell}, \mathsf{pk}_{\mathsf{R},\ell})$ as the private and public keys for R and $(\mathsf{sk}_{\mathsf{Agg},\ell}, \mathsf{pk}_{\mathsf{Agg},\ell})$ for Agg. The function f is specified during this interaction.

Finally, param is updated to include new public informations as $\mathsf{param} \leftarrow \mathsf{param} \cup \{f, \ell, \mathsf{pk}_{\mathsf{Agg},\ell}, \mathsf{pk}_{\mathsf{R},\ell}\}$.

$\mathsf{KeyGen}(\lambda, \mathsf{param})$: a PPT algorithm that returns a pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ which represents a private key and public key for each user $\mathcal{U}_i$. These elements are not necessarily dependent on the label ℓ nor the function f. Each pk_i is included in param.

$\mathsf{ParticipatingUsers}(P, \mathsf{pk}_j)$: this deterministic algorithm is executed by the aggregator Agg and maintains a list P_ℓ of participating users (the starting list P_ℓ contains $\mathsf{pk}_{\mathsf{Agg},\ell}$ and $\mathsf{pk}_{\mathsf{R},\ell}$). It takes as input a public registration list (represented as a set) and updates it if user $\mathcal{U}_j$ participates $P_\ell \leftarrow P_\ell \cup \{\mathsf{pk}_j\}$.

This list can be provided on-demand to any other party.

Phase 2: collecting Data. During this phase, Agg requests the data from users.

$\mathsf{DataSend}(x_i, \mathsf{sk}_i, \mathsf{param})$: This is a PPT algorithm that is executed by user $\mathcal{U}_i$ on input some data x_i, his personal secret key sk_i. The algorithm returns a message denoted $m_{i,\ell}$ that is transmitted to Agg.

$\mathsf{ConnectedUsers}(C_\ell)$: this deterministic algorithm is executed by the aggregator. Agg receives from some user $\mathcal{U}_i$ a message $m_{i,\ell}$, then the algorithm updates $C_\ell \leftarrow C_\ell \cup \{\mathsf{pk}_i\}$.

Phase 3: recovering the result. In this phase, the final result of the study is computed.

$\mathsf{Aggregate}(\{m_{i,\ell}\}_{i \in C}, P_\ell, \mathsf{param}, \mathsf{sk}_{\mathsf{Agg},\ell})$: this is a PPT algorithm executed by the Aggregator Agg. Having received messages from the previous connected users C_ℓ and using its secret key $\mathsf{sk}_{\mathsf{Agg},\ell}$, it returns an aggregated value $M_{\ell,C}$. This information is transmitted to R.

$\mathsf{Recover}(M_{\ell,C}, \mathsf{param}, \mathsf{sk}_{\mathsf{R},\ell})$: this is a PPT algorithm executed by R on input the aggregated value $M_{\ell,C}$ which extracts the result of the study using its secret key $\mathsf{sk}_{\mathsf{R},\ell}$ and output an element $result$.

Correctness. The correctness condition follows the idea that if some users are in a list C_ℓ of connected users (and of course in P_ℓ), then $result$ will correspond to $f(\{x_j\}_{j\in C_\ell})$.

More formally, for λ a security parameter, for any label ℓ, let param generated during $\mathsf{Setup}(\lambda)$ and updated after $\mathsf{StudyRq}(\mathsf{Agg}(\mathsf{param}), \mathsf{R}(\mathsf{param}, \ell))$. Consider P_ℓ generated during $\mathsf{ParticipatingUsers}(\{\mathsf{pk}_{\mathsf{Agg},\ell}, \mathsf{pk}_{\mathsf{R},\ell}, \cdot\})$ and C_ℓ as $\mathsf{ConnectedUsers}(\emptyset)$ after respectively the end of phase 1 and phase 2. Finally, consider pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for all $i \in P_\ell$ generated using $\mathsf{KeyGen}(\lambda, \mathsf{param})$. The correctness condition states that for each $i \in C_\ell$, if $m_{i,\ell} \leftarrow \mathsf{DataSend}(x_i, \mathsf{sk}_i, \mathsf{param})$ and

$$M_{\ell,C,} \leftarrow \mathsf{Aggregate}(\{m_{i,\ell}\}_{i\in C}, P_\ell, \mathsf{param}, \mathsf{sk}_{\mathsf{Agg},\ell}),$$

then $\mathsf{Recover}(M_{\ell,C,}, \mathsf{param}, \mathsf{sk}_{\mathsf{R},\ell}) = f(\{x_j\}_{j\in C_\ell})$ with overwhelming probability.

On the arity of the function f. In the correctness condition, the function f has to satisfy the arity constraint of accepting $|C_\ell|$ inputs. Even if it can be seen as a strong restriction, since we do not know in advance the position of the real connected users that will effectively send a message, we argue in fact that for our consideration the inner-product function $\langle \mathbf{x}, \mathbf{y} \rangle$ has the particular property that the arity needs not to be specified when $\mathbf{y}$ is public. Indeed, by simply replacing some coordinates equal to 0 in the underlying vector function (i.e. in $\mathbf{y}$) for some specific places (i.e. in the indices provided by the connected users), we can manage the above constraint.

5.3 Security definitions

In this section, we present the main security properties needed for WeStat.

Overview. The main guideline for any security property that we are considering is to ensure that a minimum amount of information to each party is leaked during the computation over the previous phases. Informally, there are three main coexisting entities (Users, Agg,$\mathcal{R}$) that interact and each of them one could involve a potential different attacks depending on the elements it has access to. In more details, we develop in the following the main issues.

1. The Requestor R asking for a study has the ability to learn the final result. Thanks to our architecture, it does not have a direct access to the messages sent by the users. However, any malicious R could corrupt, or simply add some fully-controlled corrupted users, then produce some messages that could perturb the final result computation. R could *pollute* the computation that leads to a corrupted result, a secure scheme should ensure that a malicious R can not learn any additional useful information about the remaining honest users, other than what it could learn with the corrupted messages. In other words, corrupting some subset of users can not help him to learn new information about the non-corrupted users.

2. The Aggregator Agg sees the message sent by the users, then a notion of *obliviousness* for the aggregator is needed. Indeed, we want to verify that the aggregator cannot have an access to any particular user's data. However, cares should be taken when formalizing such security property. As for the previous case, Agg could potentially corrupt some users and proceed to an aggregation step over a mixed set of corrupted and honest users. There is however two cases depending on whether Agg has or has not access to the final result. In the former case, even if it looks similar, this is not the exactly the same as in the previous consideration for R, since Agg has more power than R. Agg could aggregate any set of values, learning at the same time all possible partial results. The latter case however better restricts the attack ability of a malicious Agg and security should ensure that Agg does not learn anything about the received (honest) messages.

Recall that we are not considering a collusion between Agg and R. Otherwise, this means that Agg and R has the same information as an aggregator with a direct access to the result. Having fully collusion resistance is an ideal property that one wants to ensure. As we mentioned in the beginning of the chapter, previous works (for e.g. [21] and included references) provided a non-collusion condition between at least two parties. We do not have a solution to this problem, and we leave it as an interesting question to consider the general case. In particular, we suppose in the sequel that Agg and R do not collude.

In the following section, we consider a PPT adversary with the aim to obtain additional information depending on the leaked one it gets during the interactions. We consider in the following two useful sets for our descripition: $\mathcal{HU}$ (for honest users) and $\mathcal{CU}$ (for corrupted users).

5.3.1 Requestor security

In this section, we formalize the security notion needed for the requestor. Notice that unlike the aggregator, the requestor has no access to the elements $m_{i,\ell}$ from any non-corrupted users. More formally, we have the following definition.

Definition 5.3.1 ($\mathsf{Ind}-\mathsf{Req}$)**.** *For a* PPT *adversary* $\mathcal{A}$*, consider the following experiment.*

- *Initialization: the experiment starts by generating* $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$*. A random bit* $b \in \{0,1\}$ *is chosen and* param *is given to* $\mathcal{A}$*.*
- *Study request:* $\mathcal{A}$ *has access to a* StudyRq *oracle which on input* param *and a label* ℓ*, permits to interact with an honest* Agg *and obtains the new updated parameters* $\mathsf{param} \leftarrow \mathsf{param} \cup \{f, \ell, \mathsf{pk}_{\mathsf{Agg},\ell}, \mathsf{pk}_{\mathcal{A},\ell}\}$ *with the corresponding secret keys* $\mathsf{sk}_{\mathsf{Agg}}$*.*
- *Corrupt User creation:* $\mathcal{A}$ *has access to a* QCKeygen *oracle, which on input an index* i*, runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{param})$ *and returns* sk_i *to* $\mathcal{A}$ *and adds* i *to* $\mathcal{CU}$*.*
- *Honest User creation:* $\mathcal{A}$ *has access to a* QHKeygen *oracle, which on input an index* i*, runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{param})$*, returns* pk_i *to* $\mathcal{A}$ *and adds* i *to* $\mathcal{HU}$*..*
- *Aggregation challenge:* $\mathcal{A}$ *has an adaptive access to a* QAggregate *oracle, which on inputs the index* i*, updated* param*, a set of connected users (with possibly corrupted users) and data* $\{(x^0_{i,\ell}, x^1_{i,\ell})\}_{i\in C}$*, first consider* $m^b_{i,\ell} \leftarrow \mathsf{DataSend}(x^b_i, \mathsf{sk}_i, \mathsf{param})$*, then returns an element*

$$M^b_{C,\ell} \leftarrow \mathsf{Aggregate}(\{m^b_{i,\ell}\}_{i\in C}, P_\ell, \mathsf{param}, \mathsf{sk}_{\mathsf{Agg},\ell}).$$

- *Guessing challenge:* $\mathcal{A}$ *makes a guess* b'*.*

The output b' *of the game depends on some trivial condition. Consider* $\mathcal{CU}$ *the set of corrupted users and* $\mathcal{HU}$ *the set of the remaining honest users. Denote the requested connected list by* C_ℓ *deduced from the in the aggregating challenge. If for all* $i \in \mathcal{HU} \cap C_\ell$*, there exist some* (i, x^0_i, x^1_i, ℓ) *such that* $f(\{x^0_i\}_{i\in C_\ell}) \neq f(\{x^1_i\}_{i\in C_\ell})$ *or* $x^0_i = x^1_i$ *for all* $i \in \mathcal{CU}$*, then sets* b' *to be a random bit. Otherwise, define the advantage of* $\mathcal{A}$ *is then defined as the quantity*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{IND}-\mathsf{Req}}(1^\lambda) = |\Pr[b' = b] - 1/2|,$$

and we say that the protocol ***is oblivious*** *to the requestor if the advantage is negligible.*

5.3.2 Aggregator security

We discuss in this section the security model for the aggregator.

No result access. Intuitively, the idea is that a malicious aggregator cannot distinguish between two incoming messages even if it could corrupt some users for each study request initiated by an honest requestor R. This should hold without having direct access to the final result. More formally, we have the following definition.

Definition 5.3.2 ($\mathsf{Ind}-\mathsf{Agg}$)**.** *For a* PPT *adversary* $\mathcal{A}$*, consider the following experiment.*

- *Initialization:* $\mathcal{A}$ *initiates the experiment by generating* $\mathsf{param} \leftarrow \mathsf{Setup}(\lambda)$*. A random bit* $b \in \{0,1\}$ *is picked.*
- *Study request:* $\mathcal{A}$ *has access to a* StudyRq *oracle which permits to interact with an honest* R *and recover in a the new updated parameters* $\mathsf{param} \leftarrow \mathsf{param} \cup \{f, \ell, \mathsf{pk}_{\mathcal{A},\ell}, \mathsf{pk}_{\mathsf{R},\ell}\}$*.*
- *Corrupt User creation:* $\mathcal{A}$ *has access to a* QCKeygen *oracle, which on input an index* i*, runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{param})$ *and returns* sk_i *to* $\mathcal{A}$ *and adds* i *to* $\mathcal{CU}$*.*
- *Honest User creation:* $\mathcal{A}$ *has access to a* QHKeygen *oracle, which on input an index* i*, runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{param})$ *and returns* pk_i *to* $\mathcal{A}$ *and adds* i *to* $\mathcal{HU}$*.*
- *Data challenge:* $\mathcal{A}$ *has an adaptive access to an oracle* QData*, which on input* $(i, x_i^0, x_i^1, \ell, \mathsf{param})$ *returns* $m_{i,\ell}^b \leftarrow \mathsf{DataSend}(x_i^b, \mathsf{sk}_i, \mathsf{param})$*.*
- *Guessing challenge:* $\mathcal{A}$ *makes a guess* b'*.*

The output b' *of the game depends on some trivial condition. Consider* $\mathcal{CU}$ *the set of corrupted users and* $\mathcal{HU}$ *the set of the remaining honest users. Denote by* C_ℓ *all the connected list of users resulting from* ConnectedUsers *deduced from* $\mathcal{A}$*'s request. If during the data challenge for all* $i \in \mathcal{HU} \cap C_\ell$*, there exists some* (i, x_i^0, x_i^1, ℓ) *such that* $f(\{x_i^0\}_{i \in C_\ell}) \neq f(\{x_i^1\}_{i \in C_\ell})$*, or* $x_i^0 = x_i^1$ *for all* $i \in \mathcal{CU}$ *then sets* b' *to be a random bit. Otherwise, define the advantage of* $\mathcal{A}$ *as the quantity*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{IND}-\mathsf{Agg}}(1^\lambda) = |\Pr[b' = b] - 1/2|,$$

and we say that the protocol ***is oblivious*** *to the aggregator if the advantage is negligible.*

With result access. A natural extension of our security definition is to consider that Agg has access to the final result (using some specific oracle) of the computation provided by a honest R. Notice the similarities with the classical CCA security definition of FE. We mention however that giving access an oracle that provides results on any request represents a considerable leakage of information. Indeed, since Agg could aggregate any sum of its choice, it could obtain polynomially many relations between the inputs.

A possible way to integrate this leakage would be to restrict the queries issued by Agg in order to capture how much partial sums it could learn. We do not studied this case in full detail and we leave it as an interesting open question to see how to measure the potential leakage and how to adapt the definitions.

5.4 Our proposed solution for WeStat

In this section, we present our cryptographic construction to build WeStat. The starting point is the *binary tree* idea of Chan et al. [39] that we modify according to our needs. First, we split users in different groups. Then, each user is associated to one leaf in a binary tree, and is related to all the nodes from her leaf to the root. We then run a decentralized sum FE for each node, using the DSum construction given in [42] and defined in Sec.2.3.2 If some users fail, we are then able to find a set of subgroups that can cover the connected users. This gives us the fault-tolerant property. The problem with this solution is that Agg can potentially obtain partial sums (one for each subgroup). More precisely,

- each user $\mathcal{U}_i$ select a set $\mathcal{N}$ of nodes where it appears and generates a secret key $\mathsf{sk}_{i,\mathcal{N}}$ for each node $\mathcal{N}$, using the KeyGen algorithm of the DSum scheme of Sec2.3.2;
- using the encryption algorithm of the DSum on input her contribution x_i and the secret key $\mathsf{sk}_{i,\mathcal{N}}$, user $\mathcal{U}_i$ generates as many ciphertexts as the number of nodes in which she appears (in at most the depth of the binary tree);
- after having received the contribution of all participating users, the Aggregator has to find a set of blocks that contains all of them. Then, these blocks permit to obtain at first all partial sums, and then the whole result, which is finally sent to the requestor R;
- to avoid learning partial informations, our solution is to introduce intermediate *masks* to the partial sums by treating R as an extra user in the binary tree but with the particularity that R is belonging to all leaves in the tree. The resulting

sum is noised with some global randomness that it can be removed after having received the result from the Aggregator.

Remark that the scheme of Chan et al. [39] argue their security, in particular when considering partial sums, by relying on differential-privacy. Indeed, Chan et al. provide several techniques that introduce noise to perturb (in a DP manner) the partial sums. *Security* of all the aggregated value is provided following some composition lemma inherited from the differential-privacy literature. Our attempt is to propose a solution solely based on cryptographic primitives, as the one proposed in the previous sections. An interesting path is to integrate differential-private techniques within our consideration.

5.4.1 The proposed system

We are now ready to describe our main construction. We fix a DSum denoted by $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$. In the sequel, we sometimes omit the label ℓ for the ease of exposition. We define similarly to Chan et al. the notion of *block* (or segments) of users: having some integers, $r \geq 0$ and $m \geq 1$, we define the following sets of integers

$$\begin{aligned} B_m^r &:= \{2^r(m-1)+s : 1 \leq s \leq 2^r\}, \\ T(N) &:= \{B_m^r | n \geq 0, m \geq 1, B_m^r \subseteq [1, N]\} \\ B(i) &:= \{B | B \in T(N) \text{ and } i \in B\}. \end{aligned}$$

Suppose that N is a power of 2, i.e $N = 2^n$ for a certain n. Then, set of integers B_m^r corresponds to nodes in the binary tree construction. First, notice that these sets correspond to integers between $2^r(m-1)+1$ and $2^r m$ by definition. Moreover, the integer r, which is the size of the block B_m^r, corresponds to the *level* of the nodes in the binary tree. The integer m corresponds to the index position of the nodes (from left to right) for each level. In particular, we have $B_m^0 = \{m\}$ for all integer $m \geq 1$ which corresponds to a leaf and $B_1^r = \{1, \cdots, 2^r\}$ which corresponds to the root.

The above definition of blocks considers arbitrary r and m as well as $T(N)$. However, by our choice of N as a power of 2, $T(N)$ represents the relevant blocks of a certain size and can be seen as the set of all nodes in the binary tree with N leaf nodes. Finally, the number of blocks is bounded by $2n$.

Remark. *An important observation that will be useful when recovering the set of connected users (fault-tolerance) is the following mathematical facts. Notice that if i is in a set $[1, 2^n]$, then i is contained in at most $n+1$ blocks and, given $0 \leq r \leq n$, in at most one block of the form B_m^r for a certain $m \geq 1$. The first part is easy to check. For the*

second part, observe that ***for any*** r*, since* i *is a positive integer, by Euclidean division, we know that there exists (unique)* $a, b \in \mathbb{N}$ *such that* $i = 2^r \cdot a + b$*, where* $0 \leq b < 2^r$*. Then,* $i = 2^r \cdot (m-1) + b$ *for* $m := a + 1 \geq 1$*, implies that* $i \in B_m^r$ *but with the condition that* $1 \leq b < 2^r$*. If* $b = 0$*, it follows that* $i = 2^r \cdot (a - 1 + 1) = 2^r \cdot (a-1) + 2^r$*. Because* $i \geq 1$ *and* $b = 0$ *implies that* $a \geq 1$*, we deduce that* $i \in B_m^r$ *with* $m := a$*.*

We will use this basic facts when providing the fault-tolerance of the scheme.

We are now ready to provide a description of our system using this binary tree idea. Consider that there are at most $N = 2^n$ users in the system. The binary tree is defined as follows

- each leaf is represented by a unique number from 1 (extreme left) to N (extreme right);
- each internal node is represented by "$[i, j]$" where i is the extreme left number of node's left son, and j is the extreme right number of node's right son.

Our proposed system is as follows (see an example in Figure 5.1 for an illustration).

Phase 1. We describe the main components of the first phase.

$\mathsf{Setup}(\lambda)$: Agg generates the binary tree of height $n+1$ in such a way that each node is associated with a set of cryptographic keys.

Each leaf of the tree represents a unique user and each user is related to the nodes from the root to its leaf.

This tree is managed and maintained by the Aggregator and is common to all studies. The Aggregator also executes the $\mathsf{DSum.Setup}$ procedure (see Section 2.3.4), which outputs param. All the details of the tree are also put on this set of parameters.

$\mathsf{StudyRq}(\mathsf{Agg}(\mathsf{param}), \mathsf{R}(\mathsf{param}, \ell))$ We describe in the following the protocol between Agg and R. We consider a requestor who wants to create a new study, labelled as ℓ. The Requestor is associated to every node in the tree. It computes its public/private keys as $\{\mathsf{pk}_{\mathsf{R},B}, B \in T(N)\}$ and $\{\mathsf{sk}_{\mathsf{R},B}, B \in T(N)\})$, where

$$(\mathsf{pk}_{\mathsf{R},B}, \mathsf{sk}_{\mathsf{R},B}) \leftarrow \mathsf{DSum.KeyGen}(1^\lambda).$$

The Requestor then proceeds as follows, again for each $B \in T(N)$:

- chooses uniformly random r_B;

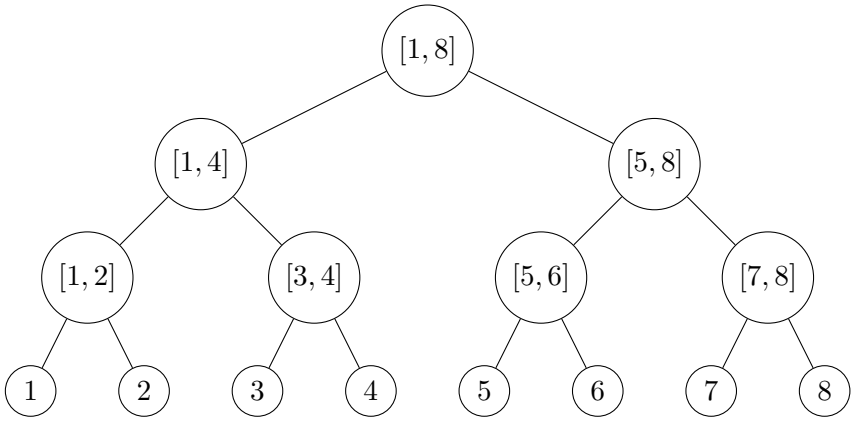

Figure 5.1: Tree structure. Here, each leaf of the tree represents a unique user and each user is related to the nodes from the root to its leaf.

- computes $\mathsf{ct}_{\mathsf{R},B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_{\mathsf{R},B}, \mathsf{sk}_{\mathsf{R},B}), r_B, \mathsf{pk}_{\mathsf{R},B}, \ell)$.

This makes available to the Aggregator the following set of ciphertexts $\mathsf{ct}_{\mathsf{R},B} := \{\mathsf{ct}_{\mathsf{R},B}, B \in T(N)\}$. Together with the above set of public keys, the public parameters $\mathsf{pk}_{\mathsf{R},\ell} := \{\mathsf{pk}_{\mathsf{R},B}, B \in T(N)\}$ consists of each generated public key. The secret output of the Requestor, $\mathsf{sk}_{\mathsf{R},\ell} := \{\mathsf{sk}_{\mathsf{R},B}, r_B, B \in T(N)\}$ is given by the set of random number $\{r_B, B \in T(N)\}$ and the underlying DSum secret keys.

$\mathsf{KeyGen}(\lambda, \mathsf{param})$: We now consider that user $\mathcal{U}_i$ wants to register to the service. The Aggregator sends him the parameter param and associates this new user to a particular leaf on the tree (leaf number i). User $\mathcal{U}_i$ executes the $\mathsf{DSum.KeyGen}(1^\lambda)$ algorithm from the DSum for each node in the tree in which it is involved. In particular, following our introduced notation for each block $B \in B(i)$, user $\mathcal{U}_i$ executes and returns $(\mathsf{pk}_{i,B(i)} := \{\mathsf{pk}_{i,B}, B \in B(i)\}, \mathsf{sk}_{i,B(i)} := \{\mathsf{pk}_{i,B}, B \in B(i)\})$, where

$$(\mathsf{pk}_{i,B}, \mathsf{sk}_{i,B}) \leftarrow \mathsf{DSum.KeyGen}(1^\lambda).$$

$\mathsf{ParticipatingUsers}(P, \mathsf{pk}_j)$: Maintained by the Agg, it starts with an empty list and add each pk_i whenever a user is registered.

Phase 2: Sending Data for a Study We then consider the participation phase by user $\mathcal{U}_i$. First, $\mathcal{U}_i$ gets back all the public key sets as defined in the previous phase and more specifically, obtain the ones related to his own nodes.

$\mathsf{DataSend}(x_i, \mathsf{sk}_i, \mathsf{param})$: The user $\mathcal{U}_i$ takes then as input his entry x_i and the label ℓ corresponding to the study and computes the following ciphertexts using the encryption scheme given by DSum and each block B in $B(i)$:

- computes $\mathsf{ct}_{i,B} = \mathsf{DSum.Enc}((\mathsf{pk}_{i,B}, \mathsf{sk}_{i,B}), x_i, \mathsf{pk}_{i,B(i)} \cup \mathsf{pk}_{\mathsf{R},B}, \ell)$.

Notice the inclusion of the requestor's public key during the encryption ($\mathsf{pk}_{i,B(i)} \cup \mathsf{pk}_{\mathsf{R},B}$).

Finally, $\mathcal{U}_i$ computes $m_{i,\ell} := \{\mathsf{ct}_{i,B}, B \in B(i)\}$. All the ciphertexts are then sent by $\mathcal{U}_i$ to the Aggregator as a participation to the study.

$\mathsf{ConnectedUsers}(C_\ell)$: the Aggregator maintains the list of participating users whenever there exists a message $m_{i,\ell}$.

Phase 3: Obtaining the Result Having access to all ciphertexts of all participating users (plus the ones of the Requestor), the Aggregator and the Requestor can start to proceed to the computation of the result.

$\mathsf{Aggregate}()$: this algorithm proceeds as follows.

- *Find "target nodes" uniquely covering all the leaves of participant*: This algorithm consists of finding the blocks $\mathcal{B}_{C_\ell} \subseteq T(N)$ such that it covers the connected users C_ℓ. We explain after the description how do we compute these nodes (See the paragraph in Sec.5.4.1)
- *Execute the decryption procedure for each given target node and aggregate the whole blinded sum*: Having the block set of the remaining users, consider for the received messages $m_{i,\ell}$ ($:= \{\mathsf{ct}_{i,B}, B \in B(i)\}$), select in particular the concerned blocks $B \in \mathcal{B}$, then compute

$$M_{\ell,\mathcal{B}} := \sum_{B \in \mathcal{B}} \mathsf{DSum.Dec}(\mathsf{param}, \{ct_{i,B}\}_{i \in B})$$

Finally, it sends this partial result to the Requestor, together with the used target nodes.

$\mathsf{Recover}(M_{\ell,\mathcal{B}}, \mathsf{param}, \mathsf{sk}_{\mathsf{R},\ell})$: from $\mathcal{B}$ the set of blocks computed in the $\mathsf{Aggregate}$ algorithm, R recovers the randomness r_B for all block $B \in \mathcal{B}$ used during the initial protocol with Agg, then it computes the following result

$$result := M_{\ell,\mathcal{B}} - \sum_{B \in \mathcal{B}} r_B.$$

The final steps are always possible since we have obtained all the relevant ciphertexts, which permits the Aggregator to obtain as many partial blinded sums[8] as target nodes. For example, in the tree given in Figure 5.1, if user 5 haven't participated, one can obtain the final result by using nodes $[1, 4]$, 6 and $[7, 8]$, which permit to scan all true participants.

The way we can treat a more general inner-product with this scheme is quite easy: each individual i can be associated to a scalar α_i and can easily replace, in the above computations, its input value x_i by $\alpha_i \cdot x_i$.

Finding target nodes.

Suppose that there are some users $0 \leq k < N$ that do not participate. By the specification of the DSum scheme, this leads to a situation where it is not possible to recover the exact sum for some blocks. In more details, similarly to Chan et al. [39], this implies that the interval $[1, N]$ is divided into $k + 1$ intervals. The main task is to efficiently find a way to circumvent the lost of information and cover the users that ineed send a message. We explain the procedure for one interval.

The key observation is to notice that every interval $[s, t]$ within $[1, N]$ can be uniquely covered using this trick: for each integer in $[s, t]$ it is possible to have exactly one block of that contains it. One possible solution to build $\mathcal{B}$ is to observe that iteratively, for an integer r decreasing from $\lceil \log(t) \rceil$ to $\lceil \log(s) \rceil$, we can build a block of size 2^r (by using our method described in remark 5.4.1) of the desired form B_m^r that is contained in the interval $[s, t]$ and which is not contained in a previously constructed block. This leads to a disjoint set of blocks such that their union cover the interval $[s, t]$.

Notice however that even if the time for searching this covering blocks is reasonable, we stress that there are some situations where there is an exponential blow-up in the number of possible blocks. For example, the worst case scenario is when we have to consider each individual block ($[i, i]$) in order to cover all the remaining users (this corresponds to the unlucky situation where one out of two predetermined ordered users fails).

5.4.2 Security proof

Our solution is obtained using the same arguments of [39] (without the DP part) and our adaptation of the DSum functionality in [42].

[8] We consider the sum as blinded since, at this step, it still includes the secret randomness coming from the Requestor.

First, the security of the DSum dynamic DMFE building block given in [42] provides the guarantee, using the fact that all the parties additionally make use of the same label (which imposes a constraint on which values can be added together), that,

- for each block/node in the tree, the sum of the contributions is automatically revealed when all the parties belonging to this block/node have sent their contributions;

- the individual contributions of non-participating users, together with the sum related to block/nodes where there are non-participating users, remain hidden for any actor, including the Aggregator that makes the computations.

As an example, if users 1, 3 and 4 have sent their contributions, but not user 2, the partial sums of nodes 1, 3, 4 and $[3, 4]$ can be retrieved, while the ones of nodes $[1, 2]$, $[1, 4]$ and $[1, 8]$ cannot, since the DSum is secure [42].

Then, from that result, the security of the fault-tolerant tree-based system given in [39] permits us to argue that

- the sum of the contributions of all the participating users (including the one by the Requestor, see below) is automatically revealed when all the parties have sent their contribution;

- any other partial sum (except the above intermediate ones related to full groups/nodes) cannot be obtained, since the users contribution are provided for the whole set of "target nodes" of the tree that uniquely cover the participating users.

Recall that we do not consider the case of a coalition between the Aggregator and the Requestor. Then, as the non-corrupted Requestor is seen as an extra user that contributes to hide the intermediate values by adding/removing randomness to all nodes, the Aggregator only obtains noised sum, so that even if it has compromised some users, what it can get is either the sum of compromised users, or a noised sum of compromised and non-compromised users.

Regarding the indistinguishability against the Requestor, we observe informally that the only information that it could obtain is the final sum.

Proof of Aggregator security.

We have the following theorem.

Theorem 5.4.1. *Suppose that* DSum *multi-client* FE *is Ind secure as per Definition 2.3.4. The WeStat construction of Sec 5.4.1 is* $\mathsf{Ind}-\mathsf{Agg}$ *Secure as per Definition 5.3.2. More formally, for any* PPT *adversary* $\mathcal{A}$ *attacking the aggregator security game, there exists a* PPT *adversary* $\mathcal{D}$ *such that*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Ind}-\mathsf{Agg}}(1^\lambda) \leq 3n \cdot (N-c) \cdot \mathsf{Adv}_{\mathcal{D},\mathsf{Ind}-\mathsf{DSum}}(1^\lambda),$$

where c *is the number of calls to the* QCKeyGen *oracle.*

Informally, we obtain the security of the WeStat construction by reducing the task for any adversary to the task of breaking the underlying DSum scheme. However, applying a direct reduction does not work, since we consider several instantiations of DSum. In fact, one for each block. The difficulty is to argue that any advantage over a block does not provide any useful advantage against another block. To provide such a result, we will rely on a hybrid argument across all the used blocks as well as honest users.

We describe in the following the proof.

Proof. Let $\mathcal{A}$ be a PPT adversary playing the security game as in Def. 5.3.2. Let λ be a security parameter and b a random bit. We prove the theorem via a series of intermediate games.

For $k \in [0, N]$, consider Game_k to be the following hybrid. During any QData request and for any honest user $i \in \mathcal{HU}$, let x_i^0, x_i^1 be the i-th client input vector submitted by $\mathcal{A}$. For all users $i \leq k$, compute i's ciphertext using x_i^0, and for all other i, computes i's ciphertext using x_i^b. Remark that Game_0 corresponds to the initial game, and Game_N is independent of the bit b since the oracle QData returns the encryption of x_i^0 for all (honest) users i.

We have now to show that Game_k and Game_{k+1} are computationally indistinguishable for all $k \in [0, N]$. The main argument is to reduce each transition to the security of the DSum primitive, thus we can build an adversary attacking the ind security game of DSum as per definition 2.3.4. The main argument is to think *over the underlying blocks corresponding to user* $\mathcal{U}_k$ (i.e. $B(k)$).

To argue about our proof, we need the following intermediate Games $\mathsf{Game}_{k,t}$ for $t = 0, 1, 2, 3$.

- $\mathsf{Game}_{k,0}$ is Game_k.
- $\mathsf{Game}_{k,1}$ is similar to $\mathsf{Game}_{k,0}$ with the difference in the generation of $\mathsf{ct}_{\mathsf{R},B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_{\mathsf{R},B}, \mathsf{sk}_{\mathsf{R},B}), r_B, \mathsf{pk}_{\mathsf{R},B}, \ell)$. A honest requestor generates all the uniformly random r_B as in the description of the protocol. All random r_B are

used for $\mathsf{ct}_{\mathsf{R},B}$ except for all blocks $B \in B(k)$, where we consider instead an encryption of the input 0. In addition, we modify the generation of $ct^b_{k,B}$ as $\mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x^b_k + r'_{k,B}, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$ such that $r'_{k,B}$ is defined as follows: having previously constructed $r_{j,B}$ for all $j < k$ (recall that we are on a hybrid argument for k), $r'_{k,B}$ is generated uniformly generated with the condition $r'_{k,B} + \sum_{j \in B, j<k} r_{j,B} := r_B$ for all $B \in B(k)$.

- $\mathsf{Game}_{k,2}$ is similar to $\mathsf{Game}_{k,1}$ except that $\forall B \in B(k)$, we modify the encryption of the challenge data by setting the following ciphertext $\mathsf{ct}^b_{k,B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x^0_k + r^\star_{k,B}, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$ where $r^\star_{k,B} = (x^b_k - x^0_k) - r'_{k,B}$. Observe that $x^0_k + r^\star_{k,B} = x^b_k - r'_{k,B}$.
- $\mathsf{Game}_{k,3}$ is Game_{k+1}.

As we will show in the following lemmas, $\mathsf{Game}_{k,t}$ is computationally indistinguishable from $\mathsf{Game}_{k,t+1}$ for $t = 0, 1, 2$. From the result, we deduce that Game_k is computationally indistinguishable from Game_{k+1} and by a standard hybrid argument we have the claim of the theorem. Let us now come back to the different transitions. We give and prove them by the following lemmas.

Lemma 5.4.2. *For any adversary $\mathcal{A}$ that distinguish between $\mathsf{Game}_{k,0}$ and $\mathsf{Game}_{k,1}$, there exists a PPT algorithm $\mathcal{A}'$ breaking the ind security of DSum such that*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Game}_{k,0},\mathsf{Game}_{k,1}}(1^\lambda) \leq n \cdot \mathsf{Adv}_{\mathcal{A}',\mathsf{Ind-DSum}}(1^\lambda).$$

Proof. Consider the following PPT algorithm $\mathcal{A}'$ attacking the ind security of DSum and that works as follows.

- Initialization: $\mathcal{A}'$ receives $\mathsf{param}_{\mathsf{DSum}}$ issued from the $\mathsf{DSum.Setup}(\lambda)$ executed by the DSum challenger. In addition, $\mathcal{A}'$ generates the corresponding parameters for the WeStat construction, i.e. a tree with the parameters in the description. During this procedure, $\mathcal{A}'$ simulates an interaction between an honest requestor R and $\mathcal{A}$ by following the description of the protocol (recall that there is no collusion between an aggregator and the requestor). In particular, $\mathcal{A}'$ output the following set of ciphertexts $\mathsf{ct}_{\mathsf{R},T(N)} := \{\mathsf{ct}_{\mathsf{R},B}, B \in T(N)\}$ together with an update of the public parameters $\mathsf{pk}_{\mathsf{R},\ell}$ of the study. Up to this point the label ℓ is also provided and $\mathcal{A}$ is given these elements. Recall that the ciphertexts $\mathsf{ct}_{\mathsf{R},B}$ for each $B \in T(N)$ are as follows $\mathsf{ct}_{\mathsf{R},B} = \mathsf{DSum.Enc}((\mathsf{pk}_{\mathsf{R},B}, \mathsf{sk}_{\mathsf{R},B}), r_B, \mathsf{pk}_{\mathsf{R},B}, \ell)$ for a uniformly random r_B except for all blocks $B \in B(k)$. In this situation, $\mathcal{A}'$ considers its QEncrypt

oracle on inputs $(r_{k,B}, 0)$ for a uniformly random $r_{k,B}$ and obtains depending on its DSum experiment an encryption of $r_{k,B}$ or 0.

- Honest User creation: $\mathcal{A}'$ runs $\mathcal{A}$ to obtain the index i that it requests for honest users and makes the same requests to its own $\mathsf{QHKeygen}$ oracle obtaining revealing pk_i to $\mathcal{A}'$.

- Corrupt User creation: $\mathcal{A}'$ runs $\mathcal{A}$ to obtain the index i that it requests for corruption and makes the same requests to its own $\mathsf{QCKeygen}$ oracle obtaining the corresponded sk_i.

- Data challenge: for all $i \in [1, N]$ and for each block $B \in B(i)$, each time $\mathcal{A}$ which has an adaptive access to his oracle QData makes a request (i, x_i^0, x_i^1, ℓ), $\mathcal{A}'$ first run $\mathcal{A}$ to obtain these inputs and proceeds as follows:

  For $i \neq k$ and $i \in \mathcal{HU}$ (all honest users different from user k), $\mathcal{A}'$ computes the ciphertexts $\{\mathsf{ct}_{i,B}\}$ and forms the elements

  $$ct_{i,B}^b \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_i, \mathsf{sk}_i), x_i^0, \mathsf{pk}_{i,B(i)} \cup \mathsf{pk}_{\mathsf{R},B}, \ell),\ i < k$$

  $$ct_{i,B}^b \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_i, \mathsf{sk}_i), x_i^b, \mathsf{pk}_{i,B(i)} \cup \mathsf{pk}_{\mathsf{R},B}, \ell),\ i > k$$

  using its own $\mathsf{QEncrypt}$ oracle (notice that we used $\mathsf{pk}_{\mathsf{R},B}$ generated during the initialization phase). Then, it forms $m_{i,\ell} := \{\mathsf{ct}_{i,B}, B \in B(i)\}$.

  Next, for user k that is a honest user, then $\mathcal{A}'$ uses the uniformly random $r_{k,B}$ for all $B \in B(k)$, considers $r'_{k,B}$ as described in $\mathsf{Game}_{k,1}$ with the condition $r'_{k,B} + \sum_{j \in B, j \neq k} r_{j,B} := r_{k,B}$ and asks its own $\mathsf{QEncrypt}$ oracle (for DSum) on input $(x_k^b, x_k^b + r'_{k,B})$ in order to obtain

  $$ct_{k,B}^b \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x_k^b, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$$

  or

  $$ct_{k,B}^b \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x_k^b + r'_{k,B}, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$$

  depending on its DSum experiment. $\mathcal{A}'$ then computes

  $$m_{k,\ell}^b := \{ct_{k,B}^b, B \in B(k)\}.$$

  Finally, $m_{k,\ell}^b$ is given in addition to previous generated $m_{i,\ell}$ to $\mathcal{A}$ by $\mathcal{A}'$.

- Guessing challenge: when $\mathcal{A}$ makes a guess b' and $\mathcal{A}'$ returns the same bit b' to its experiment.

When the $\mathsf{QEncrypt}$ oracle returns the encryption of the left input, then $\mathcal{A}'$ simulates the view of $\mathcal{A}$ playing $\mathsf{Game}_{k,0}$, and where it returns the encryption on the right input it simulates the view for $\mathsf{Game}_{k,1}$. Indeed, these oracle queries (which have the same distribution as before) do not change the result of the output and are legitimate since

$$\begin{aligned} x^b_{k,B} + r'_{k,B} + \sum_{j\in B, j<k} \left(x^0_{j,B} + r_{k,j}\right) + \sum_{j\in B, j>k} \left(x^1_{j,B} + r_{k,j}\right) \\ = \left(x^b_{k,B} + \sum_{j\in B, j<k} x^0_{j,B} + \sum_{j\in B, j>k} x^1_{j,B}\right) + \left(r'_{k,B} + \sum_{j\in B, j\neq k} r_{k,j}\right) \\ = x^b_{k,B} + \sum_{j\in B, j<k} x^0_{j,B} + \sum_{j\in B, j>k} x^1_{j,B} + r_{k,B} \end{aligned}$$

Notice however that our claim is valid for a block $B \in B(k)$. If we start with considering another hybrid game where the modifications described above are done for every block in $B(k)$. Since the number of blocks containing user k is at most $\log N := n$. We deduce that we obtain for each block $\mathsf{Adv}_{\mathcal{A}',\mathsf{Ind-DSum}}(1^\lambda)$. Hence, the result of the lemma follows. □

Lemma 5.4.3. *For any adversary $\mathcal{A}$ that distinguish between $\mathsf{Game}_{k,1}$ and $\mathsf{Game}_{k,2}$, there exists a PPT algorithm $\mathcal{A}^\star$ such that*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Game_{k,0}},\mathsf{Game_{k,1}}}(1^\lambda) \leq n \cdot \mathsf{Adv}_{\mathcal{A}^\star,\mathsf{Ind-DSum}}(1^\lambda).$$

Proof. We proceed as in the previous lemma by providing an adversary $\mathcal{A}^\star$ that will attack the security game of the underlying DSum. Indeed, $\mathcal{A}^\star$ acts exactly as the previously described adversary $\mathcal{A}'$ except that this time, $\mathcal{A}^\star$ asks its own $\mathsf{QEncrypt}$ oracle (for DSum) on input $(x^b_k + r'_{k,B}, x^0_k + r^\star_{k,B})$ in order to obtain

$$ct^b_{k,B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x^b_k + r'_{k,B}, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$$

or

$$ct^b_{k,B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x^0_k + r^\star_{k,B}, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$$

depending on its DSum experiment.

Here again, we clearly do not modify the view of the adversary and the queries are legitimate since it is a simple change of variables. The argument is valid for each block

$B \in B(k)$ so we deduce the result of the lemma. □

Lemma 5.4.4. *For any adversary $\mathcal{A}$ that distinguish between $\mathsf{Game}_{k,2}$ and $\mathsf{Game}_{k,3}$, there exists a* PPT *algorithm $\mathcal{A}^\dagger$ such that*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Game}_{k,0},\mathsf{Game}_{k,1}}(1^\lambda) \leq n \cdot \mathsf{Adv}_{\mathcal{A}^\dagger,\mathsf{Ind-DSum}}(1^\lambda).$$

Proof. We use again the same approach as in the transition between $\mathsf{Game}_{k,0}$ and $\mathsf{Game}_{k,1}$. The idea is to take a step back to the generation of the (honest) requestor ciphertext by encrypting, instead of 0, the sum of $r^\star_{k,B}$ as the new *mask*, i.e $\mathcal{A}^\dagger$ computes $r^\dagger_{k,B} := \sum_{B \in B(k)} r^\star_{k,B}$ with the previously chosen $r^\star_{k,B}$ (which is in particular independent of the bit b by construction). In addition, $\mathcal{A}^\dagger$ asks its $\mathsf{QEncrypt}$ oracle on input $(0, r\dagger_{k,B})$ (instead of $(0, r_{k,B})$) to obtain a ciphertext corresponding to 0 or $r\dagger_{k,B}$. Then, $\mathcal{A}^\dagger$ asks its own $\mathsf{QEncrypt}$ oracle (for DSum) on input $(x^0_k + r^\star_{k,B}, x^0_k)$ with the same description of $r^\star_{k,B}$ to obtain

$$ct^b_{k,B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x^0_k + r^\star_{k,B}, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell)$$

or

$$ct^b_{k,B} \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_k, \mathsf{sk}_k), x^0_k, \mathsf{pk}_{k,B(k)} \cup \mathsf{pk}_{\mathcal{A}',B}, \ell),$$

depending on its DSum experiment. The proof of the lemma follows. □

This concludes the lemma proofs. The conclusion of the theorem follows as we should notice that we have used an hybrid argument over all honest users k. In particular, summing up all the hybrids, we obtain the claim of the theorem, i.e. by letting $\mathcal{D} := (\mathcal{A}', \mathcal{A}^\star, \mathcal{A}^\dagger)$, we have

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Ind-Agg}}(1^\lambda) \leq 3n \cdot (N - c) \cdot \mathsf{Adv}_{\mathcal{D},\mathsf{Ind-DSum}}(1^\lambda).$$

□

Proof of Requestor security

We have the following theorem concerning the requestor security result.

Theorem 5.4.5. *Suppose that* DSum *mutli-client* FE *is ind secure as per Definition 2.3.4. The WeStat construction of Sec 5.4.1 is* $\mathsf{Ind-Req}$ *to any requestor and is secure as per Definition 5.3.1. More formally, for any* PPT *adversary $\mathcal{A}$ attacking the requestor*

security game, there exists some PPT *adversary* $\mathcal{D}$ *such that*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Ind-Req}}(1^\lambda) \leq n \cdot (N-c) \cdot \mathsf{Adv}_{\mathcal{D},\mathsf{Ind-DSum}}(1^\lambda),$$

where c *is the number of calls to the* QCKeyGen *oracle.*

The proof is quite similar to the previous aggregator security one, except that a malicious requestor does not see the resulted individual ciphertext coming from honest users. In fact, as explained in the scheme description, the requestor is an extra user and can thus act maliciously. The requestor could then influence the final result bit not the honest ciphertexts. Since decryption of the DSum is performed by a honest Aggregator, we are in the same situation as having an adversary attacking the ind security of the IND security of DSum.

We use again an hybrid argument over all challenge messages x_i^b for a bit b. In addition, we also need an hybrid argument over the blocks to argue security. This time, the proof is more direct than the previous one since we do not have to rely on secret sharing argument.

Proof. Let $\mathcal{A}$ be a PPT adversary playing the security game as in Def. 5.3.1. Let λ to be the security parameter and b to be a random bit. We will prove the theorem via a series of intermediate games

For $k \in [N]$, consider Game_k to be the following hybrid. During any QAggregate request and for any honest user $i \in \mathcal{HU}$, let x_i^0, x_i^1 to be the i-th client input vector submitted by $\mathcal{A}$. For all users $i \leq k$, compute i's ciphertext using x_i^0, and for all other i, computes i's ciphertext using x_i^b. Remark that Game_0 corresponds to the initial game, and Game_N is independent of the bit b since the oracle QData always returns the encryption of x_i^0 for all (honest) users i.

We have now to show that Game_k and Game_{k+1} are computationally indistinguishable for all $k \in [0, N]$. The main argument is to reduce each transition to the security of the DSum primitive, thus we can build an adversary that will attack the ind security game of DSum as per definition 2.3.4. The main guideline is to reason *over the underlying blocks corresponding to user* k (i.e $B(k)$).

We describe in the following the adversary $\mathcal{A}'$ that attacks n copies of the underlying DSum. Recall that $n := \log N$, where N is the number of users.

- Initialization: $\mathcal{A}'$ receives $\mathsf{param}_{\mathsf{DSum}}$ issued by the $\mathsf{DSum.Setup}(\lambda)$ execution from the DSum challenger. In addition, $\mathcal{A}'$ generates the corresponding parameters for the WeStat construction, i.e a tree with the above parameters in the description.

In addition, $\mathcal{A}'$ simulate an interaction between an honest aggregator R and $\mathcal{A}$ by following the description of the protocol (recall that there is no collusion between an aggregator and the requestor). Eventually, $\mathcal{A}$ returns the following set of ciphertexts $\mathsf{ct}_{\mathsf{A},T(N)} := \{\mathsf{ct}_{\mathsf{A},B}, B \in T(N)\}$ together with an update of the public parameters $\mathsf{pk}_{\mathsf{A},\ell}$ of the study. Up to this point a label ℓ is provided.

- Honest User creation: $\mathcal{A}'$ runs $\mathcal{A}$ to obtain the index i that it requests for honest users and makes the same requests to its own $\mathsf{QHKeygen}$ oracle obtaining revealing pk_i to $\mathcal{A}'$.
- Corrupt User creation: $\mathcal{A}'$ runs $\mathcal{A}$ to obtain the index i that it requests for corruption and makes the same requests to its own $\mathsf{QCKeygen}$ oracle obtaining the corresponded sk_i.
- Data challenge: for each block $B \in B(i)$ for user i, $\mathcal{A}'$ runs $\mathcal{A}$ which has an adaptive access to his oracle $\mathsf{QAggregate}$. This oracle takes inputs elements of the form (i, x_i^0, x_i^1, ℓ). $\mathcal{A}'$ first run $\mathcal{A}$ whenever it requests these inputs and forwards them to its own $\mathsf{QEncrypt}$ oracle (for DSum) in order to obtain $ct_{i,B}^b \leftarrow \mathsf{DSum.Enc}((\mathsf{pk}_i, \mathsf{sk}_i), x_i^b, \{\mathsf{pk}_B\}_{B\in B(i)}, \ell)$, for each block $B \in B(i)$. $\mathcal{A}'$ computes

$$\begin{aligned} M_{C,\ell}^b &\leftarrow \mathsf{Aggregate}(\{m_{i,\ell}^b\}_{i\in C}, P_\ell, \mathsf{param}, \ell) \\ &= \sum_{B\in\mathcal{B}_C} \mathsf{DSum.Dec}(\mathsf{param}, \{ct_{i,B}^b\}_{i\in B}) \end{aligned}$$

  where $m_{i,\ell}^b \leftarrow \mathsf{DataSend}(x_i^b, \mathsf{sk}_i, \mathsf{param})$ and $\mathcal{B}_C$ is the recovering connected users. Finally, the aggregated value $M_{C,\ell}^b$ is given to $\mathcal{A}$ by $\mathcal{A}'$.
- Guessing challenge: $\mathcal{A}$ makes a guess b' and $\mathcal{A}'$ returns the same bit b'.

We analyse the security of this reduction. First, notice that $\mathcal{A}'$ correctly simulates the view of $\mathcal{A}$ since the ciphertexts are well generated for all corrupted users.

The arguments are the same in the security of the Aggregator. Since $\mathcal{A}'$ uses these same requests as $\mathcal{A}$, we deduce that Game_k is indistinguishable from Game_{k+1}. The hybrid argument is over all honest users k and affected blocks that contain user $\mathcal{U}_k$(at most $\log N = n$), we deduce that for $\mathcal{D} := \mathcal{A}'$, we have the claim of the theorem. In particular, we get

$$\mathsf{Adv}_{\mathcal{A},\mathsf{Ind}-\mathsf{Req}}(1^\lambda) \leq n \cdot (N - c) \cdot \mathsf{Adv}_{\mathcal{D},\mathsf{Ind}-\mathsf{DSum}}(1^\lambda),$$

where c is the number of calls to the $\mathsf{QCKeyGen}$ oracle by $\mathcal{A}$. □

5.5 Instantiation from bilinear maps

In this section, we discuss a possible implementation of our instantiation of WeStat from bilinear maps. We do not give in details the full description of the protocol but only the DSum part and how the needed cryptographic primitives are used in the application.

In addition, the DSum construction of Chotard et al. [42] uses NIKE [57] together with the concept of All-or-Nothing Encapsulation [42] that is also derived from chameleon hash as described in Sec. 2.2.2 as well as some classical symmetric primitives. The full description of the DSum is given in [42, Sec. 6.2].

Our next paragraph provides a direct adaptation. In particular, we provide instantiations of all the underlying used primitives of the presented construction in [42, Sec. 6.2]. Hence, for NIKE we consider the DL-based chameleon hash scheme given in Sec. 2.2.2. Moreover, we use this instantiation to discuss our implementation issues. We denote fo by $\mathbf{x}_i[k]$ the k-th component of vector $\mathbf{x}_i$ (similar notation is used for other vectors).

Description of DSum. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ to be a bilinear environment where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order prime p and $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Let q a prime number and k an integer such that $p = kq + 1$ for some k. Let g of order q in $\mathbb{Z}_p^*$. Finally, let $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ and $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ be two hash functions. In addition, we consider classical and known symmetric primitives such as AES algorithm and a hash-based pseudo-random function (such as HMAC) that we denote by PRF-SHA256.

The instantiated DSum multi-client functional encryption consists of the following (Setup, Enc, KeyGen, Dec).

- DSum.Setup(1^λ). During the setup, one[9] has to generate $u_0, u_1, u_2, S \in \mathbb{G}_1$, generate $\mathsf{ck} \in \mathbb{Z}_q^*$ and compute $\mathsf{hk} = g^{\mathsf{ck}} \pmod p$. The parameters param is then defined as $(u_0, u_1, u_2, S, \mathsf{hk})$, together with the bilinear environment $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ and the two hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$.

- DSum.KeyGen(i, param). The following steps for each user i, on input param are executed:

  - choose at random $z_i \in \mathbb{Z}_p^*$ and $r_i \in \mathbb{Z}_q^*$;
  - compute $Z_i = g_2^{z_i}$ and $t_i = g^{H_2(Z_i)}\mathsf{hk}^{r_i} \pmod p$;
  - compute $Y_i = u_0 u_1^{t_i} u_2^{t_i^2}$ and $X_i = Y_i^{z_i}$;

[9] for example a Agg

– choose at random $v_i \in \mathbb{Z}_p^*$ and compute $T_i = g_2^{v_i}$.

The i-th public key is $\mathsf{pk}_i = (X_i, Z_i, r_i, T_i)$ and the corresponding private key is $\mathsf{sk}_i = (z_i, v_i)$.

- $\mathsf{DSum.Enc}(\mathsf{sk}_i, \mathbf{x}_i, \mathcal{I}, \ell)$. We encrypt a data $\mathbf{x}_i$ defined as a vector of length L. The encryption procedure is executed as follows:

  – $\forall j \in \mathcal{I}, j \neq i$, compute

    * $t_j = g^{\mathcal{H}_2(Z_j)}\mathsf{hk}^{r_j}$;
    * $K_{i,j} = e(S^{z_i}, Z_j)$ iff $e(X_j, g_2) = e(u_0 u_1^{t_j} u_2^{t_j^2}, Z_j)$;
    * $\mathbf{r}_{i,j}[k] = \mathsf{PRF\text{-}SHA256}(K_{i,j}, \mathcal{I}||\ell||k)$ for all $k \in [0, L[$;

  – compute $\mathbf{c}_i = \mathbf{x}_i + \sum_{j<i} \mathbf{r}_{i,j} - \sum_{j>i} \mathbf{r}_{i,j}$ (in the L-length vector space with $\mathbf{c}_i$ being then a vector of length L);

  – choose $w_i \in \mathbb{Z}_p^*$ and compute $W_i = g_2^{w_i}$;

  – compute $K_i = e(\mathcal{H}_1(\mathsf{pk}_i||\ell), (\prod_j T_j)^{w_i})$, the ciphertext $C_i = \mathsf{AES}(K_i, \mathsf{c}_i)$;

  – compute $S_i = \mathcal{H}_1(\mathsf{pk}_i||\ell)^{w_i}$.

  The ciphertext is finally $\mathsf{ct}_i = (C_i, W_i, S_i, \mathsf{pk}_i, \ell)$.

- $\mathsf{DSum.Dec}(\mathcal{C} := \{\mathsf{ct}_i\}_{i\in\mathcal{I}}, \mathsf{param})$. It takes as input a set of ciphertexts $\mathcal{C} = \{\mathsf{ct}_i\}_{i\in\mathcal{I}}$ and works as follows:

  – $\forall i \in \mathcal{I}$, compute $K_i = e(\prod_j S_j, W_i)$;

  – $\forall i \in \mathcal{I}$, compute $\mathbf{c}_i = \mathsf{AES}^{-1}(K_i, C_i)$;

  – compute the result $R = \sum_{i\in\ \mathcal{I}} \mathbf{c}_i$ (which computes $\sum_{i\in\ \mathcal{I}} \mathbf{x}_i$).

  As the initial encrypted message is given by a vector, this last step is performed component-wise.

Intuitively, in the encryption algorithm, $\mathbf{r}$ serves as a *user-dependent* mask (for a set of users $\mathcal{I}$) and is derived using NIKE. The sum is obtained in a telescopic manner. Moreover, the All-or-Nothing encapsulation step consists of bringing the elements K_i, which are used as a symmetric keys (for AES), in order to encapsulate informations that permits to recover the ciphertexts $\mathbf{c}_i$ only when all the W_i's and S_i's are all present.

Practical implementation. This work has been implemented as a prototype in the scope of the PAPAYA project [10]. We present in the following some details about this construction.

The studied use-case is to count the carbon emission of a smart phone when its owner uses social networks apps. We would like to correlate such carbon emission w.r.t. the age range and the residence area. One complex point we had to treat is the fact that the group for the chameleon hash signature should be $\mathbb{Z}_p^*$ of prime order q, and where p should itself be the prime order of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of the bilinear map e. As this last step is the less flexible one (finding a pairing-friendly group is not as easy [19, 43]), we need first to fix p according to the used pairing environment, and then test whether it is compatible with the chameleon hash. In fact, we are using BN-256 pairing-friendly elliptic curves, see [19] which provides 128-bits security. The resulted scheme provides however 50-bits security (and not 128-bits as claimed in our published paper [32]). This is mainly due to our choice of the chameleon hashing since we work in $\mathbb{Z}_p^\star$ which is a *small* subgroup and only provides 50-bits security. This is of course a limitation.

A way to enhance our proposition is to consider another group of order p it is is an interesting task to investigate a friendly chameleon hash that suits with this bilinear environment. Moreover, a possible path is to increase all the parameters, with of course an important impact over the global performance.

We refer to the published work [32] for a more detailed description about the implementation (how data is parsed, cryptographic libraries usage, online/offline optimization...etc) as well as the encouraging resulted benchmarks that ensure that such kind of schemes are possible to imagine for real-world applications.

5.6 Conclusion

The main purpose of the WeStat architecture is to make an in-depth privacy-by-design study of a real-life use case, namely mobile data usage statistics.

Motivated by the possible limitations for practical deployment, we have presented a service platform called WeStat that could tolerate fault-tolerance aggregation of mobile data users in a privacy-preserving manner. In this scenario, a special care will be given to the information leakage delegated to an external entity.

Our work identifies the possible limitation of existing solutions and propose a general formal cryptographic protocol that could provide useful statistics to third parties, in a

[10] https://www.papaya-project.eu/

multi-device environment with users failure.

We propose a security model with the aim of capturing the possible areas of attacks. Our choice makes a focus on the sum function and shows how it is possible to exploit and modify some advanced cryptographic mechanisms, such as variants of FE that have recently been published, in order to control the leakage of delegated information as well as to dictate the nature of authorized statistics.

CHAPTER 6

CONCLUSION AND OPEN PROBLEMS

In the umbrella of our discussed use cases, functional encryption arises in this thesis as a base primitive for solving different problems in security that have in common one objective: *the control of information leakage to a designed external entity.* We have combined several techniques from the cryptographic literature and have seen how brining functional encryption benefits to solve multiple practical scenarios.

In chapter 3, we have considered a new notion of function's protection for an interactive FE, where it is possible to obtain a functional key for a function without revealing the exact specification of it. Concretely, we have defined a general framework that considers FE in an interactive setting. For this, we have introduced IFE, with an adapted extension of the classical security properties of FE. Moreover, we have defined the new security notion of blindness and our study has shown the possible interplay between all these considerations. The formal model being on track, we have been able to provide a generic secure construction of IFE from any FE by relying on a two-move private function evaluation and zero-knowledge proofs of knowledge. Finally, we have proposed an efficient blind interactive IPFE. For this, we have exploited the power of a linearly homomorphic encryption, using the Castagnos – Laguillaumie homomorphic scheme, with some adapted zero-knowledge proofs.

For the chapter 4, we have highlighted the links between differential privacy and functional encryption. In particular, we have suggested with our work a new view and construction for a differential private mechanism based on FE. This leads us to consider a randomized version of IPFE that we introduced in this thesis. As usual with a new primitive, we have exhibited the desired security model with a focus on the special case

of linear queries. Using ideas from multi-input IPFE, we have designed in a non-black box way a mechanism that permits to release an encrypted dataset, so that it becomes possible to recover a DP-compatible statistics over it. Our work has shown that it is possible to provide both confidentiality and differential privacy.

Finally, in the last chapter 5, we have tackled the problem of privacy-preserving aggregation of mobile data usage, in a fault-tolerant manner. Motivated by the practical objectives of the PAPAYA project, we have proposed an architecture, called WeStat, that permits to describe a general protocol for computing specific analytics. Concretely, our security model has permitted to guarantee that the only allowed leakage in the WeStat architecture is the sum of all participating user inputs during one specific study. The proposed secure solution is obtained by a modification of a variant of FE, in the multi-user setting for the sum function, in combination with known adapted fault-tolerance techniques.

Perspectives

Finding a balance between security and the growing need for functionality is a constant challenge for modern cryptography. To address positively more complex demands, tools such as homomorphic encryption and multiparty computation already exist, and will surely play a determining role in the further evolution of the digital world.

As a part of these tools, functional encryption is a versatile primitive for delegating computation. This thesis seeks to highlight the diversity in its usage by presenting several scenarios where it brings new solutions. Of course, there remain many open questions and directions that could be explored.

Blind IFE. On the theoretical side, the result of [75] shows that blind IBE implies efficient oblivious transfer. Does any similar result or implications exist for blind IFE? A future improvement is to consider our security properties on a more general setting such as the Universally Composable (UC) framework [35]. While adapting blind signature definitions is a challenging task in UC model [4], our blind IFE naturally inherits these difficulties. We leave as an interesting path to explore a satisfying UC definition that meets our requirements.

Another natural extension is to consider the multi-input or the multi-client settings. The main challenge then is to handle several entities that derive a functional key. Since our proposed constructions are a special two-party protocols, there is no doubt that generic techniques from the multi-party computation literature could be adapted for

these generalizations.

Finally, as we did for the inner product IFE, is it possible to consider other specific functionalities? There exists construction of quadratic FE [16]. Hence, designing a blind protocol that computes quadratic functions is an intriguing open question.

Randomized FE and DP. Regarding security of our RIPFE, a first improvement consists in providing a full proof for our generic construction from any IPFE. For the model, studying the situation where an adversary could request multiple challenge ciphertexts in an adaptive manner, should provide stronger guarantees.

Multi-input or multi-client RIPFEs (or more generally RFE) is a possible generalization which is interesting on its own. Returning to DP, *Local differential privacy* [48] is an important variant that provides DP mechanisms in a multi-user setting. Consequently, providing a clear relationship between a DP-compliant multi-user RFE and local differential privacy is an exciting future work.

Similarly to our previous work, we can also mention that having a DP compatible RFE for quadratic functions is an interesting problem. For applications, the DP literature is in fact diverse in terms of DP functionalities and of course, an obvious perspective is to find a particular RFE that is suitable for specific DP randomized functions. In fact, we believe that building general DP-compliant RFE for these described situations would probably and remarkably require new techniques.

Improving WeStat. Further refinement for the WeStat design might improve its security, its efficiency, or it supported functionality. Considering collusion between an aggregator and requestor is one of the main challenges. Finding more suitable DSum schemes could moreover leverage the efficiency of the global scheme. Furthermore, as in the previous contributions, handling more functionalities fits in our planned work.

Another development consists of adding differential privacy into the WeStat architecture. In fact, a possible start could be to consider a multi-user DP-compliant *randomized* DSum as in Chapter 4. This merits to be inspected.

BIBLIOGRAPHY

[1] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, Mar. / Apr. 2015.

[2] 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 H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Heidelberg, Aug. 2018.

[3] M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, Apr. / May 2017.

[4] M. Abe and M. Ohkubo. A framework for universally composable non-committing blind signatures. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 435–450. Springer, Heidelberg, Dec. 2009.

[5] A. Agarwal, M. Herlihy, S. Kamara, and T. Moataz. Encrypted databases for differential privacy. *PoPETs*, 2019(3):170–190, July 2019.

[6] S. Agrawal. New methods for indistinguishability obfuscation: Bootstrapping and instantiation. Cryptology ePrint Archive, Report 2018/633, 2018. https://eprint.iacr.org/2018/633.

[7] S. Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Y. Ishai and V. Rijmen, editors, *EURO-*

CRYPT 2019, Part I, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019.

[8] S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai. On the practical security of inner product functional encryption. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 777–798. Springer, Heidelberg, Mar. / Apr. 2015.

[9] S. Agrawal, M. Clear, O. Frieder, S. Garg, A. O'Neill, and J. Thaler. Ad hoc multi-input functional encryption. Cryptology ePrint Archive, Report 2019/356, 2019. https://eprint.iacr.org/2019/356.

[10] S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *PKC 2020, Part I*, LNCS, pages 34–64. Springer, Heidelberg, 2020.

[11] S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, Aug. 2016.

[12] S. Agrawal and A. Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In V. Rijmen and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 110–140. Springer, Heidelberg, May 2020.

[13] S. Agrawal and D. J. Wu. Functional encryption: Deterministic to randomized functions from simple assumptions. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 30–61. Springer, Heidelberg, Apr. / May 2017.

[14] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[15] M. Azraoui, S. Brunet, S. Canard, A. Diop, L. Eveillard, A. Filipiak, A. Hamdi, F. Misarsky, D. N. Kuate, M. Paindavoine, Q. Santos, and B. Vialla. Cybercrypt: Learn basic cryptographic concepts while playing. Cryptology ePrint Archive, Report 2021/063, 2021. https://eprint.iacr.org/2021/063.

[16] C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In J. Katz and

H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, Aug. 2017.

[17] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001.

[18] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, Heidelberg, May / June 2010.

[19] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, Oct. 2019.

[20] A. Beimel, K. Nissim, and E. Omri. Distributed private data analysis: Simultaneously solving how and what. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 451–468. Springer, Heidelberg, Aug. 2008.

[21] J. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure single-server aggregation with (poly)logarithmic overhead. Cryptology ePrint Archive, Report 2020/704, 2020. https://eprint.iacr.org/2020/704.

[22] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997.

[23] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, Aug. 2001.

[24] D. Boneh, A. Raghunathan, and G. Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 461–478. Springer, Heidelberg, Aug. 2013.

[25] D. Boneh, A. Raghunathan, and G. Segev. Function-private subspace-membership encryption and its applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 255–275. Springer, Heidelberg, Dec. 2013.

[26] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, Mar. 2011.

[27] F. Boudot. Efficient proofs that a committed number lies in an interval. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444. Springer, Heidelberg, May 2000.

[28] Z. Brakerski and G. Segev. Function-private functional encryption in the private-key setting. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 306–324. Springer, Heidelberg, Mar. 2015.

[29] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, Oct. 2011.

[30] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 196–214. Springer, Heidelberg, Mar. 2009.

[31] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 107–122. Springer, Heidelberg, May 1999.

[32] S. Canard, N. Desmoulins, S. Hallay, A. Hamdi, and D. Le Hello. *WeStat: A Privacy-Preserving Mobile Data Usage Statistics System*, page 5–14. Association for Computing Machinery, New York, NY, USA, 2021.

[33] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt. BlindIDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic. In R. Karri, O. Sinanoglu, A.-R. Sadeghi, and X. Yi, editors, *ASIACCS 17*, pages 561–574. ACM Press, Apr. 2017.

[34] S. Canard, A. Hamdi, and F. Laguillaumie. Blind functional encryption. In *ICICS 20*, LNCS, pages 183–201. Springer, Heidelberg, 2020.

[35] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

[36] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ecdsa from hash proof systems and efficient instantiations. To appear in Proc. of CRYPTO 2019.

[37] G. Castagnos and F. Laguillaumie. Linearly homomorphic encryption from DDH. In K. Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 487–505. Springer, Heidelberg, Apr. 2015.

[38] G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 733–764. Springer, Heidelberg, Dec. 2018.

[39] T.-H. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In A. D. Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 200–214. Springer, Heidelberg, Feb. / Mar. 2012.

[40] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, Dec. 2016.

[41] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, Dec. 2018.

[42] J. Chotard, E. Dufour-Sans, R. Gay, D. H. Phan, and D. Pointcheval. Dynamic decentralized functional encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2020, Part I*, LNCS, pages 747–775. Springer, Heidelberg, Aug. 2020.

[43] R. Clarisse, S. Duquesne, and O. Sanders. Curves with fast computations in the first pairing group. Cryptology ePrint Archive, Report 2020/760, 2020. https://eprint.iacr.org/2020/760.

[44] E. Cuvelier, O. Pereira, and T. Peters. Election verifiability or ballot privacy: Do we need to choose? In J. Crampton, S. Jajodia, and K. Mayes, editors, *ESORICS 2013*, volume 8134 of *LNCS*, pages 481–498. Springer, Heidelberg, Sept. 2013.

[45] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Heidelberg, Feb. 2001.

[46] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994.

[47] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[48] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *54th FOCS*, pages 429–438. IEEE Computer Society Press, Oct. 2013.

[49] C. Dwork. Differential privacy (invited paper). In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 1–12. Springer, Heidelberg, July 2006.

[50] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, Heidelberg, Mar. 2006.

[51] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, Aug. 2014.

[52] C. Dwork, A. Smith, T. Steinke, and J. Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4(1):61–84, 2017.

[53] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, Aug. 1984.

[54] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[55] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.

[56] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[57] E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, Feb. / Mar. 2013.

[58] H. F. Gaines. *Cryptanalysis a Study of Ciphers and Their Solutions*. Dover Publications, Inc., USA, 1989.

[59] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. Applications of Algebra to Cryptography.

[60] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, Oct. 2013.

[61] S. Garg, V. Rao, A. Sahai, D. Schröder, and D. Unruh. Round optimal blind signatures. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Heidelberg, Aug. 2011.

[62] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[63] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 351–360. ACM Press, May / June 2009.

[64] 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, Oct. 2006.

[65] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 104–120. Springer, 2004.

[66] O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[67] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[68] 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 P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.

[69] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[70] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, Aug. 2012.

[71] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, Aug. 2015.

[72] V. Goyal. Reducing trust in the PKG in identity based cryptosystems. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 430–447. Springer, Heidelberg, Aug. 2007.

[73] V. Goyal, A. Jain, V. Koppula, and A. Sahai. Functional encryption for randomized functionalities. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 325–351. Springer, Heidelberg, Mar. 2015.

[74] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309.

[75] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer, Heidelberg, Dec. 2007.

[76] A. Groce, J. Katz, and A. Yerukhimovich. Limits of computational differential privacy in the client/server setting. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 417–431. Springer, Heidelberg, Mar. 2011.

[77] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. Au. PPDCP-ABE: Privacy-preserving decentralized ciphertext-policy attribute-based encryption. In M. Kutylowski and J. Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 73–90. Springer, Heidelberg, Sept. 2014.

[78] R. V. L. Hartley. Transmission of information. *The Bell System Technical Journal*, 7(3):535–563, 1928.

[79] S. Inusah and T. J. Kozubowski. A discrete analogue of the laplace distribution. *Journal of Statistical Planning and Inference*, 136(3):1090 – 1102, 2006.

[80] V. Iovino and K. Zebrowski. On the power of public-key functional encryption with function privacy. Cryptology ePrint Archive, Report 2015/470, 2015. https://eprint.iacr.org/2015/470.

[81] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164. Springer, Heidelberg, Aug. 1997.

[82] J. Katz and L. Malka. Constant-round private function evaluation with linear complexity. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 556–571. Springer, Heidelberg, Dec. 2011.

[83] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, Apr. 2008.

[84] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu. Function-hiding inner product encryption is practical. In D. Catalano and R. De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 544–562. Springer, Heidelberg, Sept. 2018.

[85] I. Komargodski, G. Segev, and E. Yogev. Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 352–377. Springer, Heidelberg, Mar. 2015.

[86] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS 2000*. The Internet Society, Feb. 2000.

[87] A. K. Lenstra. Key length. contribution to the handbook of information security, 2004.

[88] I. Mironov, O. Pandey, O. Reingold, and S. P. Vadhan. Computational differential privacy. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 126–142. Springer, Heidelberg, Aug. 2009.

[89] P. Mohassel and S. S. Sadeghian. How to hide circuits in MPC an efficient framework for private function evaluation. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 557–574. Springer, Heidelberg, May 2013.

[90] P. Mohassel, S. S. Sadeghian, and N. P. Smart. Actively secure private function evaluation. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 486–505. Springer, Heidelberg, Dec. 2014.

[91] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[92] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In S. R. Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, Jan. 2001.

[93] M. Naveed, S. Agrawal, M. Prabhakaran, X. Wang, E. Ayday, J.-P. Hubaux, and C. A. Gunter. Controlled functional encryption. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 2014*, pages 1280–1291. ACM Press, Nov. 2014.

[94] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In D. S. Johnson and U. Feige, editors, *39th ACM STOC*, pages 75–84. ACM Press, June 2007.

[95] H. Nyquist. Certain factors affecting telegraph speed. *Transactions of the American Institute of Electrical Engineers*, XLIII:412–422, 1924.

[96] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. https://eprint.iacr.org/2010/556.

[97] R. Ostrovsky, A. Paskin-Cherniavsky, and B. Paskin-Cherniavsky. Maliciously circuit-private FHE. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 536–553. Springer, Heidelberg, Aug. 2014.

[98] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[99] T. B. Pedersen and E. Savas. Impossibility of unconditionally secure scalar products. *Data Knowl. Eng.*, 68(10):1059–1070, 2009.

[100] M. O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Jan. 1979.

[101] A. Rial. Blind attribute-based encryption and oblivious transfer with fine-grained access control. *Des. Codes Cryptography*, 81(2):179–223, Nov. 2016.

[102] R. L. Rivest and M. Dertouzos. On data banks and privacy homomorphisms. 1978.

[103] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[104] K. Schmeh and E. Antal, editors. *Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt 2019, Mons, Belgium, June 23-26, 2019*, volume 158 of *Linköping Electronic Conference Proceedings*. Linköping University Electronic Press, 2019.

[105] T. Schneider. Practical secure function evaluation. In *Fachwissenschaftlicher Informatik-Kongress (Informatiktage 2008)*, volume S-6 of *LNI*, pages 37–40, Bonn, Germany, March 14, 2008. GI.

[106] C.-P. Schnorr. Efficient identification and signatures for smart cards (abstract) (rump session). In J.-J. Quisquater and J. Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 688–689. Springer, Heidelberg, Apr. 1990.

[107] A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, Nov. 1979.

[108] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[109] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS 2011*. The Internet Society, Feb. 2011.

[110] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332.

[111] J. Tomida and K. Takashima. Unbounded inner product functional encryption from bilinear maps. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 609–639. Springer, Heidelberg, Dec. 2018.

[112] I. Tucker. *Functional encryption and distributed signaturesbased on projective hash functions,the benefit of class groups*. PhD thesis, 2020.

[113] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, Heidelberg, May / June 2010.

[114] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, Nov. 1982.

[115] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, Oct. 1986.