Succinctly Verifiable Computation over Additively-Homomorphically Encrypted Data with Applications to Privacy-Preserving Blueprints

Scott Griffy[1], Markulf Kohlweiss[2], Anna Lysyanskaya[1], and Meghna Sengupta[3]

[1] Brown University, {anna_lysyanskaya,scott_griffy}(at)brown.edu
[2] University of Edinburgh and IOG, Edinburgh, markulf.kohlweiss(at)ed.ac.uk
[3] University of Edinburgh, M.Sengupta-1(at)ed.ac.uk

Abstract. With additively homomorphic encryption (AHE), one can compute, from input ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)$, and additional inputs $y_1, \ldots, y_k$, a ciphertext $c_f = \mathsf{Enc}(f(x_1, \ldots, x_n, y_1, \ldots, y_k))$ for any polynomial f in which each monomial has total degree at most 1 in the x-variables (but can be arbitrary in the y-variables). For AHE that satisfies a set of natural requirements, we give a non-interactive zero-knowledge proof system (in the random-oracle model) for showing that a ciphertext c_f is the result of homomorphically evaluating f on ciphertexts $c_1, \ldots, c_n$ and private inputs $y_1, \ldots, y_k$ that correspond to commitments $C_1, \ldots, C_k$. Our proofs are *succinct*, i.e., their size is independent of the number of ciphertexts n, and is instead $O(k \log d)$ where k is the number of private inputs, and d is the maximum degree of any variable in f.

We give two ways of instantiating this framework: with ElGamal-based encryption (under the DDH assumption) and with a variant of the Camenisch-Shoup cryptosystem (under the DCR assumption). Both yield proof systems where computing and verifying the proof takes a comparable amount of time to homomorphically evaluating f.

Next, we show that our framework yields a dramatically improved privacy-preserving blueprint (PPB) system. Introduced by Kohlweiss, Lysyanskaya, and Nguyen (Eurocrypt'23), an f-PPB system allows an auditor with secret input x to create a public encoding pk of the function $f(x, \cdot)$ that reveals nothing about x. Yet, it allows a user to compute an encoding, or escrow Z, of the value $f(x, y)$ on input the user's private data y corresponding to a commitment C_y; Z will verifiably correspond to the commitment C_y. The auditor will be able to recover $f(x, y)$ from Z, but will learn no other information about y. For example, if f is the watchlist function where $f(x, y)$ outputs y only in the event that y is on the list x, then an f-PPB allows the auditor to trace watchlisted users in an otherwise anonymous system.

Using our succinct zero-knowledge proof system for additively homomorphic computation we achieve the following results: (1) We provide efficient schemes for a bigger class of functions f; for example, we show how to realize f that would allow the auditor to trace e-cash transactions of a criminal suspect which was previously not efficient. (2) For the watchlist and related functions, we reduce the size of the escrow Z from linear in the size of the auditor's input x, to logarithmic. Additionally, we define and satisfy a stronger notion of security for f-PPBs, where a malicious auditor cannot frame a user in a transaction in which the user was not involved in.

Table of Contents

1 Introduction

Cryptography gives us powerful tools to trade off our fundamental need to protect our personal privacy with the legitimate needs of systems and governments to enforce rules and laws and to regulate finance. Among these, anonymous credentials [Cha90,LRSW99,CL01,Lys02,CL02,CV02,CL04,BCL04,BL13,HS21] [RWGM23,TZ23,HSS23] and related technologies such as e-cash [CFN90] are prominent examples: such systems allow a user with a cryptographic commitment C_y to his data y to prove that y is somehow certified by some authority or authorities; in the case of e-cash, they further allow to prove that an e-coin was computed correctly as a function of the user's data y.

In a recent paper, Kohlweiss, Lysyanskaya and Nguyen (KLN) [KLN23] added *privacy-preserving blueprints* (PPBs) to the repertoire of cryptographic algorithms for balancing privacy and accountability. In an f-PPB system, the goal is to allow an authorized auditor to learn $f(x, y)$ where x is the auditor's secret input that's fixed once and for all, and y is a user's secret input to a transaction; if a PPB system is used in tandem with an anonymous credential system, y can include meaningful information about the user's identity. Via an appropriate choice of f, an f-PPB system makes it possible to perform audits of the system while leaking no information other than what's leaked by f. For example, for x representing a watchlist of suspected criminals, let $f_{watchlist}$ be defined as follows: $f_{watchlist}(x, y) = y$ if y is on the list, and $\perp$ otherwise. An $f_{watchlist}$-PPB would allow the auditor to trace all of the suspects' transactions, but none of the transactions of other people. A PPB further requires that the secret x correspond to a publicly known commitment C_x that can be further certified by an external party, so that a malicious auditor cannot make up x at will.

In a PPB system, first, the auditor sets up his public key pk and secret key sk on input his secret x and a commitment C_x to x for which the auditor knows the opening (and which may be signed by an external validator who certifies that x is a correct input). A PPB includes a public verification procedure $\mathsf{VerPK}(\mathsf{pk}, C_x)$ for ensuring that pk corresponds to the commitment C_x. Now the system is ready for blueprinting transactions; there is no limit on the number of such transactions. In a transaction, a user with secret input y and a commitment C_y to y to which the user knows the opening r (and which meaningfully corresponds to some information about this user, for example validated via an anonymous credential system), computes the escrow $Z = \mathsf{Escrow}(\mathsf{pk}, y)$ of y under pk. A PPB includes a public verification procedure $\mathsf{VerEscrow}(\mathsf{pk}, C_y, Z)$ for ensuring that Z corresponds to pk and C_y. Finally, using sk, the auditor runs the decryption algorithm to recover $z = f(x, y)$ from Z. The reason that it is called a privacy-preserving *blueprint* is that we can think of pk as a "blueprint" of the function $f(x, \cdot)$ of the user's y.

An f-PPB is realizable for any efficiently computable function f from either fully homomorphic encryption (FHE) or non-interactive secure computation (NISC) [KLN23]; however, this general approach is not suitable for practical use. KLN additionally gave a much more practical construction of $f_{watchlist}$-PPB from the ElGamal cryptosystem and proof systems about discrete logarithm relations in the random-oracle model; the size of their escrow is linear in the size of the watchlist. They did not provide an efficient instantiation for any function other than $f_{watchlist}$; and even for $f_{watchlist}$ the size of the escrow was prohibitive.

As we argue below, this is not sufficient to be useful in practice. To bridge this gap, we develop a commit-and-proof framework for working with additively-homomorphically encrypted data. Additively homomorphic encryption (Definition 6) allows one to compute, on input ciphertexts $c_1, \ldots, c_n$ that encrypt $x_1, \ldots, x_n$, and additional inputs $y_1, \ldots, y_k$, the value $f(x_1, \ldots, x_n, y_1, \ldots, y_k)$

for any polynomial f in which each monomial has total degree at most 1 in the x-variables (but can be arbitrary in the y-variables).

Our Contribution 1: A modular framework for succinct verifiable secure computation on additively-homomorphically encrypted data. In this paper, we give a non-interactive zero-knowledge proof system (in the random-oracle model) for showing that a ciphertext c_{out} is the result of homomorphically evaluating f on $c_1, \ldots, c_n$ and private inputs $y_1, \ldots, y_k$ that correspond to commitments $C_1, \ldots, C_k$. Our proof system outputs *succinct* proofs, i.e. their size is $O(k \log d)$ where k is the number of private inputs, and d is an upper bound on the degree of any variable in f; note that the size of the proof is independent on the number n of the x-variables. Our construction diverges from those in the literature since in PPBs, the auditor (who must only learn $f(x, y)$) can decrypt manipulations of the ciphertexts, $c_1, \ldots, c_n$. For the proof to be efficient, we must include "intermediate" ciphertexts in the proof that allows the verifier to follow along to be convinced of the final evaluation. Thus, to protect these intermediate ciphertexts from being decrypted, we define and construct *commitments to ciphertexts* so that while we can prove relations between these ciphertexts, we can keep any intermediate ciphertexts hidden in commitments. We give two different practical instantiations of this framework: one under the DDH assumption (using the ElGamal cryptosystem) and the other under the Paillier assumption (using the Camenisch-Shoup cryptosystem).

PPBs for central bank digital currencies. Since the KLN paper first appeared, privacy-preserving blueprints received some attention in the civil liberties discourse [Sta23] because (among other things) of the following motivating application to central bank digital currencies (CBDCs): suppose that the auditor's input x is a list of suspected financial criminals' unique identifiers. Suppose a user's input y contains this user's unique identifier y_{id} as well as seed y_{seed} from which all of this user's e-coins' serial numbers are generated. This is consistent with, e.g., compact e-cash [CHL05] and related schemes [CHL06,CHK$^+$06,KKS22,TBA$^+$22], including those proposed specifically for the CBDC application [KKS22,TBA$^+$22]. The function f is as follows: $f(x, y) = y$ if $y_{id} \in x$, and $\perp$ otherwise. A PPB with these properties will allow the auditor to not only identify that a transaction was carried out by a suspect, but also to recover the seed y_{seed} and trace all of the user's e-coins, even as the rest of the users of the systems' privacy is protected[4].

[4] This application to cryptographic e-cash is attractive to those who advocate that a CBDC can be privacy-preserving even while enabling lawful investigations. Unfortunately, the alternative to yielding ground on this to law enforcement is that central banks throughout the world would adopt a CBDC that provides no privacy — even from third-party observers — to individuals, in the name of compliance with law enforcement. For example, the analysis of CBDC design choices provided by the White House [Gov22] is lukewarm on using ecash-like systems for that reason. See page 17 of [Gov22]. The existence of a practical cryptographic system that can provide a watchlist capability in a way that is transparent to citizens who, even if they shouldn't know who is on the watchlist, can still see the size of the watchlist and the fact that there was a lawfully obtained warrant for placing a person on it, would

For the CBDC application, $f_{watchlist}$ is not the right function. Instead, we need $f_{CBDC}(x, y) = y$ if $y = (y_{id}, y_{seed})$, and $y_{id} \in x$. KLN give a practical construction that works for $f_{watchlist}$ but not for f_{CBDC}, because of their use of ElGamal encryption. instead of recovering y, the auditor in their construction can only recover g^y where g is a generator of a group in which the discrete logarithm problem is hard. From g^y it is possible to recover y by brute-force search if only a small number of bits of y are still unknown; but it wouldn't be possible to recover y_{seed}, since the size of a pseudorandom seed must be too large to allow brute-force search. Here, we give a construction for the correct f.

Our second contribution: Realizing f_{CBDC}-PPBs. Let $f(x, y) = y$ if $y = (y_1, y_2)$, and $y_1 \in x$, and $\perp$ otherwise. We give a practical instantiation of a f-PPB construction. By "practical", we mean that it can be instantiated efficiently using proof systems for discrete logarithm relations in the random-oracle model.

The KLN approach is also not good enough for either f_{CBDC}-PPBs or even $f_{watchlist}$-PPBs because we expect the watchlist x to be quite large. In the KLN construction, the size of the escrow Z was linear in the size of the watchlist x. Using the fact that our framework produces succinct proofs, we give a substantial improvement:

Our third contribution: Exponential improvement in the size of escrow Z. We give practical constructions of a f_{CBDC}-PPB and a $f_{watchlist}$-PPB where the size of Z is logarithmic in the size of x.

Other improvements to PPBs. The KLN definition of security [KLN23] does not rule out that a malicious auditor would be able to produce pk, sk, C_y and Z such that the decryption algorithm will output $z \neq f(x, y)$. In Sect. 1.2, we discuss how the KLN construction of $f_{watchlist}$-PPB allowed for a "framing" attack: a malicious auditor causing an escrow to decrypt to the identity of an honest user y who is not a party to the transaction. Addressing these security issues using our new framework and the reworked functionality is our final contribution.

Our fourth contribution: Stronger security. We improve the definition of security of PPB to that of *non-frameable* PPB: we add the requirement that the decryption algorithm's output be publicly verifiable. Our constructions achieve non-frameability.

1.1 Our Framework for Verifiable Computation

Let us focus on a concrete example. At a high level, a f_{cbdc}-PPB scheme will work as follows: The auditor will first find the coefficients of the polynomial $P(\chi) = a_0 + a_1\chi + \ldots + a_n\chi^n$ of degree n whose roots are values on the list x, and it will output a public key pk of an encryption scheme, as well as the encryptions of the coefficients of P; i.e. $X = (\mathsf{pk}, \boxed{a_0}_{\mathsf{pk}}, \ldots \boxed{a_n}_{\mathsf{pk}})$, where $\boxed{m}_{\mathsf{pk}}$ denotes an encryption of a message m under the public key pk (and we drop the subscript

strike a reasonable balance, and, as a result, may sway the policy conversation (in which law enforcement voices are often louder than those of privacy advocates) in favor of using an ecash-like system for CBDCs.

when clear from the context). Let $f(a_0,\ldots,a_n,y_{id},y,s) = \big(s\sum_{i=0}^{n} a_i y_{id}^i\big) + y$. Note that if $f_{cbdc}(x,y_{id},y) \neq \perp$, then $f(\mathbf{a},y_{id},y,s) = y$; else, if the user picks s uniformly at random, then $f(\mathbf{a},y_{id},y,s)$ is also random. Thus, the goal is for the user to compute c_f, an encryption of $f(a_0,\ldots,a_n,y_{id},y,s)$, from X.

If the underlying encryption scheme is additively homomorphic, then $c_f = \boxed{f(a_0,\ldots,a_n,y_{id},y,s)}$ can be computed using homomorphic addition: Let the symbol '$\oplus$' denote the homomorphic operation on ciphertexts, and let $\odot$ denote multiplying a ciphertext by a scalar. Then $c_f = \big(\bigoplus_{i=0}^{n}(sy_{id}^i)\odot\boxed{a_i}\big)\oplus\boxed{y}$. We also need the user to compute a zero-knowledge proof that c_f was computed correctly from X and the user's secret inputs s, y_{id} and y that correspond to commitments C_s, C_y and $C_{y_{id}}$. While general-purpose ZK proof systems can be used here, a proof system designed hand-in-hand with the underlying encryption scheme can take advantage of efficient Σ-protocols and impose only a minimal overhead over encryption; the classical results on efficient multi-party computation of Cramer, Damgård and Nielsen [CDN01] serve as the inspiration for this approach.

We suggest a modular, commit-and-prove [BCF$^+$] approach for constructing a proof that a given ciphertext is the result of computing on additively-homomorphically encrypted data. For example, here the output ciphertext c_f is the result of applying a series of homomorphic operations, starting with the input ciphertexts $\{\boxed{a_i}\}$ and the user's inputs. In order to prove correctness of c_f in our framework, one forms commitments to the intermediate steps of this computation (for example, the intermediate ciphertexts $\boxed{a_i}^{y_{id}^i}$) and proves that each of these intermediate steps was carried out correctly.

Thus, our main new building block is an additively homomorphic encryption scheme equipped with (1) a cryptographic commitment scheme for committing to ciphertexts; and (2) proof systems for proving properties of committed ciphertexts, such as the property that a committed ciphertext c was obtained from committed ciphertexts c_1 and c_2, along with a committed scalar a, as follows: $c = c_1 \oplus (c_2 \odot a)$. (See Sect. 3.1 for the more formal treatment.)

Next, let us explain how to instantiate this framework with the ElGamal cryptosystem. Let G be a group of prime order q with generator g_1; an ElGamal public key is a group element g_2; an encryption of $M \in G$ is $(g_1^r, g_2^r M)$ where for random $r \in \mathbb{Z}_q$. ElGamal is not, strictly speaking, an additively homomorphic encryption scheme, but a multiplicatively homomorphic one: $(g_1^r, g_2^r M)\oplus(g_1^{r'}, g_2^{r'}M') = (g_1^{r+r'}, g_2^{r+r'}MM')$. However, we can define a "lifted" ElGamal cryptosystem: to encrypt the message m, use the ElGamal cryptosystem to encrypt g_1^m; i.e. $\boxed{m} = (g_1^r, g_2^r g_1^m)$. The problem is that, instead of outputting m, the decryption algorithm outputs g_1^m; converting it to m requires that m come from a small space, so that it can be found via brute-force search; we call this flavor of encryption "semi"-encryption. Still, for some applications (such as realizing $f_{watchlist}$-PPBs), this is good enough.

Our techniques for achieving succinct proofs. The naïve way for computing a proof π of correctness of c_f is to form a commitment to the ciphertext that is the result of each intermediate step in the computation (for example, the values $\boxed{a_i}^{y_{id}^i}$ in the example above), meaning that the size of the proof will need to be linear in the degree d of the polynomial f (and in the description of the polynomial altogether). To reduce the dependence on the degree from d to $O(\log d)$, we use a degree reduction technique inspired by the sum-check protocol of Lund, Fortnow, Karloff and Nisan [LFKN92]. The sum-check protocol was used more recently in cryptography by Goldwasser, Kalai and Rothblum [GKR08] and follow-up work on "proofs for Muggles" [XZZ$^+$19,ZLW$^+$21]. Pietrzak [Pie19,HHKP23] was the first to use it to halve the degree of a polynomial (as we do) rather than to eliminate a linear variable as in the other cited work. As far as we know, our paper is the first time that this technique is used in order to prove correctness of commit-and-prove-style computation on encrypted data.[5]

The overall idea, described in Sect. 5.3, is to recursively halve the degree of the polynomial. Suppose that we need to prove that a ciphertext $c_f = \boxed{f(x_1, \dots, x_n, y_1, \dots, y_k)}$; the prover and verifier both know $\boxed{x_i}$; further, the prover knows $y_1, \dots, y_k$ (and thus can compute c_f) while the verifier knows just the corresponding commitments $\{C_{y_i} = \mathsf{Com}(y_i; r_i)\}$. Suppose the degree of y_1 in f is d. The recursive step is to reduce the proof of this statement to the proof that another ciphertext $c_{f'}$ is an encryption of $f'(x_1, \dots, x_n, y_1, \dots, y_k)$, where in f' the degree of y_1 is $d/2$. This can be accomplished using the Schwartz-Zippel lemma: we obtain f' from f by replacing each occurrence of $y^{d/2}$ with a random scalar α; in the interactive version of the sum-check protocol α would be chosen by the verifier, but here it is chosen by the random oracle. It is important that the ciphertext $c_{f'}$ used in the recursive step not be given to the verifier in the clear; otherwise, it will leak information to the adversary who knows the decryption key. Instead, our proof system works for *committed* ciphertexts.

To obtain a commitment to an ElGamal ciphertext $\boxed{a} = (A, A')$, we first extend Pedersen commitments (with generators g and h) to commit to group elements. To commit to A, we sample $s_A, r_A \leftarrow \mathbb{Z}_q$ and the commitment is $C_A = (C_{A,1}, C_{A,2}) = (Ag^{s_A}, g^{s_A} h^{r_A})$; similarly, we can form a commitment $C_{A'} = (C_{A',1}, C_{A',2})$. Thus, a commitment to $\boxed{a}$ is $C_{\boxed{a}} = (C_A, C_{A'})$. It is easy to see that this commitment scheme has convenient homomorphic properties: if '$*$' denotes applying the group operation componentwise, then $C_{\boxed{a}} * C_{\boxed{b}} = C_{\boxed{a+b}}$. As shown in Sect. 4, this allows for efficient proof systems for properties of committed ciphertexts needed for our framework. Additionally, we show in Sect. 4 that our framework can also be instantiated, under the Paillier assumption, with a semantically secure variant of the Camenisch-Shoup cryptosystem [CS03].

Why f_{cbdc}-PPB was not achievable in KLN. KLN's limitation was that it used lifted ElGamal, and thus, in the event that the user was on the watchlist, the decryption algorithm was only able to recover g^y from the escrow, rather than y in the clear. As explained earlier, this is not good enough if y comes from a large enough domain (for example if it contains a seed for a PRF) and cannot be brute-force-searched. The Camenisch-Shoup based instantiation of the framework we just discussed allows the decryption algorithm to recover y, which yields f_{cbdc}-

[5] Previous work [BG13] used a completely different technique to give a succinct proof that a committed value corresponds to the evaluation of a polynomial, but with the important distinction that the polynomial was known to both Prover and Verifier.

blueprints. It turns out that the ElGamal-based instantiation can work as well (with some efficiency limitations), if we split the e-cash seed into sufficiently small chunks, see Section 5.4.

1.2 Non-Frameability and Why It Matters

Our additional contribution to privacy-preserving blueprints is an additional property — non-frameability, — and our constructions satisfy it. The concept of non-frameability was first introduced in the work of Camenisch [Cam97]. The paper introduced it for the group signature scheme setting as the property that the manager (even if they collude with a group member) cannot falsely accuse group members. Subsequently, Bellare, Shi and Zhang [BSZ05] formalized the property and called it Non-frameability - again for group signature schemes.

At a high-level there are similarities with the property of non-frameability as we define it and as defined by [BSZ05]. Both properties require that if some authority (the opener in the case of [BSZ05] and the auditor in our case) wants to prove that a user took some action (signing a message in the case of [BSZ05] and authenticating themselves in an anonymous credential scheme in the case of blueprints) they must provide verifiable proof. One difference between the schemes is that in [BSZ05] the opener traces any user indiscriminately. In our case, the auditor's functionality is not "trace" but the function f. (In the case of watchlists, that means the auditor can trace iff the user is on the watchlist.) Also, a group signature scheme provides tracing for group members who are signing messages, whereas in blueprints, the functionality is to trace users who are using an anonymous credential scheme, which does not imply that these traceable users sign any messages. Thus, it is not trivial to construct blueprints from the group signature scheme in [BSZ05].

The watchlist PPB scheme of [KLN23] is frameable, i.e., a malicious auditor can collude with a malicious user to produce Z that will decrypt to the identity of an honest user who was not a party to the transaction (and who may or may not be on the watchlist). The gist of their scheme is that pk includes encrypted coefficients of a polynomial P such that $P(y) = 0$ if and only if y is on the watchlist x. The escrow $Z = (\hat{Z}, \pi)$ produced by the user whose identity is y consists of the encryption $\hat{Z}$ of $rP(y) + y$ for a random r chosen by the user, as well as a proof π that indeed $\hat{Z}$ was computed correctly. In order to frame the user with identity y^*, a malicious user whose identity is y and to whom the coefficients of the polynomial P are known (as would be the case if the auditor is malicious) needs to solve for r^* in the $r^*P(y) + y = y^*$, and will produce an escrow $Z = (\hat{Z}, \pi)$ by following the original algorithm, but just using $r = r^*$.

This attack is outside the KLN security model, and therefore does not contradict their security analysis (which is correct). One could also argue that frameability, also known as deniability, can be a feature and not a bug. We discuss this at greater length in Section A.

In Sect. 5, we improve the KLN definition of privacy-preserving blueprints by incorporating non-frameability. The decryption algorithm must now produce a proof π_z of correct decryption, and a new algorithm Judge verifies this proof.

The proof π_z is important when the auditor's output is used as evidence in legal proceedings[6] or as input in a smart contract, e.g., an Ethereum Eigenlayer slashing operation or crime restitution.

In order to obtain a practical non-frameable f-PPB for the watchlist function, we modify the KLN construction as follows: our Escrow algorithm will output $(\hat{Z}, \hat{Z}', \pi)$, where $\hat{Z}$ is an encryption of $rP(y) + y$ (just as before), and the additional value $\hat{Z}'$ is an encryption of $r'P(y)$, while, as before, the proof π is to ensure that $\hat{Z}$ and $\hat{Z}'$ were computed correctly. If π verifies, the decryption algorithm will decrypt $\hat{Z}$ iff $\hat{Z}'$ decrypts to 0; it will output $\perp$ otherwise. Our succinct proofs are compatible with this non-framing construction.

1.3 Related work

Freedman, Nissim, and Pinkas (FNP) [FNP04] were the first to give a protocol for the evaluating an encrypted polynomial. Unlike here, the evaluator in their work was not committed to a particular input y on which to evaluate it; it only needed to ensure that some y exists that makes the evaluation correct. In our scheme, the user commits to a y before the protocol starts and must use this y throughout the protocol, making our proof system much more involved. FNP initiated the study on secure set intersection (PSI) which is by now an extremely well-studied [CMdG+21,CM20,RS21,GPR+21] [CRR21,RR22] special case of secure two-party computation. Our framework can be seen as a building block for verifiable PSI [KMRS14,ATD16,JWP22], since verifiable evaluation of encrypted polynomials is a subroutine in many of these protocols.

Recent years have seen an explosion of techniques for zero-knowledge proof systems [BMM+21,CBBZ23,GLS+23,WHV24,BFK+24]; many of these are for general circuits, but especially worthy of comparison to our work are those of them that, like us, take advantage of efficient Σ-protocols for algebraic relations over committed values and, like us, also achieve succinctness [BBB+18,ACC+22]. The main difference of our work from these is that our framework is suitable for verifiable computation on encrypted data, which is a scenario to which these cited works do not directly apply. Bhadauria, Hazay, Venkitasubramaniam, Wu, and Zhang [BHV+23] provide a way for a prover to compute and prove the encryption of an evaluation of a polynomial without knowing the polynomial. Where our work differs is that their proof system achieves zero-knowledge only in the event that the secret key of the encryption scheme is unknown to the adversary. Bartusek, Garg, Jain and Policharla's work [BGJP23] is related in spirit to privacy-preserving blueprints: they show a scheme that makes it possible to identify an originator of harmful content (relative to a database of harmful content) while protecting privacy in all other circumstances.

[6] Interestingly, this is currently rarely the case for existing investigations employing mass or targeted surveillance. Instead, law enforcement follow a complicated process of parallel construction where not always lawfully attained evidence is used to inform a lawful investigation [Boy].

2 Preliminaries

Black-box partially straight-line (BB-PSL) Non-Interactive Zero Knowledge (NIZK). Non-interactive zero-knowledge (NIZK) proofs are an important building block for us. We follow the KLN notation and definitions (Sec. 2.1 of [KLN23]) of the completeness and ZK properties of NIZK proof system, provided in abbreviated form in Def. 1 below.

Definition 1 (Completeness and ZK of NIZK [KLN23]). *Let $\mathcal{R}$ be a relation. Let S be a setup model (e.g., the CRS model or the random oracle model). Let P^{S} and V^{S} be (non-interactive) algorithms for the prover and the verifier in the S-setup model. $(\mathsf{P}^{\mathsf{S}}, \mathsf{V}^{\mathsf{S}})$ constitute a complete proof system if for all $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $\Pr\left[\pi \leftarrow \mathsf{P}^{\mathsf{S}}(\mathbb{x}, \mathbb{w}) : \mathsf{V}^{\mathsf{S}}(\mathbb{x}, \pi) = 0\right] = 0$.*

They satisfy the zero-knowledge property if for any PPT adversary Adv in the experiment of Fig. 2.1, the advantage function $\nu(\lambda)$ defined below is negligible: $\mathsf{Adv}^{\mathsf{NIZK}}_{\mathsf{Adv}} = |\Pr[\mathsf{NIZK}^{\mathsf{Adv},0}(1^\lambda) = 0] - \Pr[\mathsf{NIZK}^{\mathsf{Adv},1}(1^\lambda) = 0]| = \nu(\lambda)$

```
$\mathsf{NIZK}^{\mathsf{Adv},0}(1^\lambda)$
  return $\mathsf{Adv}^{\mathsf{S}(\cdot),\mathsf{P}^{\mathsf{S}}(\cdot,\cdot)}(1^\lambda)$
$\mathsf{NIZK}^{\mathsf{Adv},1}(1^\lambda)$
  return $\mathsf{Adv}^{\mathcal{O}_{\mathsf{S}}(\cdot),\mathcal{O}_{\mathsf{P}}(\cdot,\cdot)}(1^\lambda)$
```

```
$\mathcal{O}_{\mathsf{S}}(m)$
  $\mathsf{st}, h, \tau_{\mathsf{Ext}} \leftarrow \mathsf{SimS}(\mathsf{st}, m)$
  return $h$
```

```
$\mathcal{O}_{\mathsf{P}}(\mathbb{x}, \mathbb{w})$
  if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$ : return $\bot$
  $\mathsf{st}, \pi \leftarrow \mathsf{Sim}(\mathsf{st}, \mathbb{x})$
  return $\pi$
```

Fig. 2.1: NIZK game

Let us review BB-PSL simulation extractable proof systems [KLN23] (Def. 2). The straight-line extractor here does not extract the entire witness, but just some function of it; simultaneously, a black-box extractor (that's allowed to rewind the adversary) can extract the entire witness. In Sec. B, we motivate this definition further.

Definition 2 (Black-box partial straight-line simulation extractability). *A proof system (as defined in Def. 1) is BB-PSL simulation extractable if the advantage (defined below) of any PPT adversary is negligible:* $\mathsf{Adv}^{\mathsf{NISimBBPSLExt}}_{\mathsf{Adv},f} = \Pr[f\text{-}\mathsf{NISimBBPSLExt}^{\mathsf{Adv}}(1^\lambda) = 1] = \nu(\lambda)$ *for some negligible function ν.*

Proofs of Equivalent Representations of Discrete Logarithms. Using known techniques, we can construct a Σ-protocol that proves the following relation in Def. 3 in prime order cyclic groups where the DDH and CDH problems are hard. We describe a Σ-protocol that satisfies Def. 3 in Sec. D.1.

Definition 3 (Relation for proof of equality of discrete logarithm representations in cyclic groups of prime order). *Let $R_{\mathsf{eqrep}\text{-}p}$ be the following relation: $R_{\mathsf{eqrep}\text{-}p}(\mathbb{x}, \mathbb{w})$ accepts if $\mathbb{x} = (\mathcal{G}, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{k})$ where $\mathcal{G}$ is the description of a group of order q, and all the x_is and $g_{i,j}$s are elements of $\mathcal{G}$, and witness $\mathbb{w} = \{w_j\}_{j=1}^{m}$ such that $x_i = \prod_{j=1}^{m} g_{i,j}^{w_j}$.*

```
$f\text{-}\mathsf{NISimBBPSLExt}^{\mathsf{Adv}}(1^\lambda)$
    $\mathcal{Q}, \mathcal{Q}_{\mathsf{S}} \leftarrow [\,]; (\mathbb{x}, \pi) \leftarrow \mathsf{Adv}^{\tilde{\mathcal{O}}_{\mathsf{S}}(\cdot), \mathcal{O}_{\mathsf{Sim}}(\cdot)}(1^\lambda)$
    $\mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{BB(Adv)}}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi); \mathbb{w}' \leftarrow \mathsf{ExtSL}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi)$
    return $\mathsf{V}^{\mathcal{O}_{\mathsf{S}}}(\mathbb{x}, \pi) \wedge (\mathbb{x}, \pi) \notin \mathcal{Q} \wedge \big((\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \vee \mathbb{w}' \neq f(\mathbb{w})\big)$

$\mathcal{O}_{\mathsf{S}}(m)$ $\underline{\tilde{\mathcal{O}}_{\mathsf{S}}(m)}$
    $\mathsf{st}, h, \tau_{\mathsf{Ext}} \leftarrow \mathsf{SimS}(\mathsf{st}, m)$
    $\mathcal{Q}_{\mathsf{S}}.\mathsf{add}((m, h, \tau_{\mathsf{Ext}}))$
    return $h\underline{, \tau_{\mathsf{Ext}}}$

$\mathcal{O}_{\mathsf{Sim}}(\mathbb{x})$
    $\mathsf{st}, \pi \leftarrow \mathsf{Sim}(\mathsf{st}, \mathbb{x})$
    $\mathcal{Q}.\mathsf{add}((\mathbb{x}, \pi))$
    return $\pi$
```

Fig. 2.2: f-$\mathsf{NISimBBPSLExt}$ game

We can enhance this protocol to multiply witnesses with the relation in the following definition (Def. 4). We give examples of how to construct and use these protocols in Appx. D.3. While using this protocol, we use Camenisch-Stadler notation to denote witnesses and relations.

Definition 4 (Relation for proof of multiplication of witnesses over bases in cyclic groups of prime order). *Let $R_{\mathsf{eqrep}\text{-}p^*}$ be the following relation: $R_{\mathsf{eqrep}\text{-}p^*}(\mathbb{x}, \mathbb{w})$ accepts if the following two conditions hold:*
(1) $\mathbb{x} = (\mathcal{G}, \mu, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{k})$ where $\mathcal{G}$ is the description of a group of order q, and all the x_is and $g_{i,j}$s are elements of $\mathcal{G}$, and witness $\mathbb{w} = \{w_j\}_{j=1}^{m}$ such that $x_i = \prod_{j=1}^{m} g_{i,j}^{w_j}$.
(2) If $\forall i \in [m], w_i = \prod_{j \in \mu(i)} w_j$ where μ is a map $\mu : [m] \to P([m])$ and $P([m])$ is the set of all subsets of $[m]$.

The multiplication protocol holds for $\mathbb{Z}_{n^2}$ as well with a caveats: we can only prove the relations for the absolute values of elements (e.g., for the example above, we could only prove that $C = \pm g^{ab}h^r$). This is a limitation of extraction of Σ-protocols in $\mathbb{Z}_{n^2}$. We explain this limitation and other details in Appx. D.3. This proof can be constructed from known techniques [BCM05,DF02].

Definition 5 (Relation for proof of multiplication of witnesses over bases in composite order groups). *Let $R_{\mathsf{eqrep}\text{-}n^*}$ be the following relation: $R_{\mathsf{eqrep}\text{-}n^*}(\mathbb{x}, \mathbb{w})$ accepts if the following two conditions hold:*
(1) $\mathbb{x} = (n, \mu, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{k})$ where $n = pq$ and p, q are safe primes, and all the x_is and $g_{i,j}$s are elements of $\mathbb{Z}_{n^2}$, and witness $\mathbb{w} = (\{b_i\}_{i=0}^{k}, \{w_j\}_{j=1}^{m})$ such that $x_i = b_i \prod_{j=1}^{m} g_{i,j}^{w_j}$ where $b_i \in \{-1, 1\}$.
(2) If $\forall i \in [m], w_i = \prod_{j \in \mu(i)} w_j$ where μ is a map $\mu : [m] \to P([m])$ and $P([m])$ is the set of all subsets of $[m]$.

2.1 Privacy Preserving f-Blueprint Schemes (PPBs)

[KLN23] defines a blueprint scheme as in Def. 2.3. We will be modifying this definition to serve our new use-case of non-frameable privacy preserving blueprints in Sect. 5. A blueprint scheme has three parties - an auditor, a set of users and a set of recipients.

$\mathsf{Setup}(1^\lambda, cpar) \to \Lambda$: Outputs public parameters Λ including 1^λ and commitment scheme, $cpar$.
$\mathsf{KeyGen}(\Lambda, x, r_x) \to (\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$: The key generation algorithm for auditor A.
$\mathsf{VerPK}(\Lambda, \mathsf{pk}_\mathsf{A}, C_x) \to 1$ or 0: Takes the auditor's public key pk_A and a commitment C_x as input, verifies that the auditor's public key was computed correctly for the commitment C_x.
$\mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, y, r_y) \to Z$: Takes Λ, pk_A, and commitment value and opening (y, r_y) as input and outputs an escrow Z for commitment $C = \mathsf{Com}(y; r_y)$.
$\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C_y, Z) \to 1$ or 0: Takes the auditor's public key pk_A, a commitment C_y, and an escrow Z as input and verifies that the escrow was computed correctly for the commitment C_y.
$\mathsf{Dec}(\Lambda, \mathsf{sk}_\mathsf{A}, C_y, Z) \to f(x, y)$ or $\perp$: Takes the auditor's secret key sk_A, a commitment C_y and an escrow Z as input. It decrypts the escrow and returns the output $f(x, y)$ if C_y is a commitment to y and $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C_y, Z) = 1$.

Fig. 2.3: An f-blueprint scheme

[KLN23] also defines a *secure* f-blueprint scheme as one that possesses the following properties -

Correctness of VerPK *and* VerEscrow : The algorithms VerEscrow and VerPK accept with probability 1 for honestly generated values $(cpar, \mathsf{pk}_\mathsf{A}, C_x, C_y, Z)$.

Correctness of Dec : $\mathsf{Dec}(\Lambda, \mathsf{sk}_\mathsf{A}, C_y, Z) = f(x, y)$ holds with overwhelming probability for honestly generated values $(cpar, \mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A}, C_y, Z)$.

Soundness ensures that if, for a commitment C_y, escrow Z is accepted, then it correctly decrypts to $f(x, y)$ where x is opening of C_x and y is opening of C_y.

Blueprint Hiding : The blueprint pk_A does not reveal anything about x other than what the adversary can learn by forming valid escrows and submitting them for decryption.

Privacy against Dishonest Auditor ensures that even if the auditor is malicious, an honest user's escrow contains does not have access to any information apart from $f(x, y)$, where x is opening of C_x and y is opening of C_y.

Privacy with Honest Auditor ensures that an adversary that does not control the auditor learns no information from the escrow Z.

2.2 Additively Homomorphic Encryption

Additively homomorphic $\mathfrak{g}$-semi-encryption scheme. We need an appropriate additively homomorphic (AH) semantically secure public-key encryption scheme. Our application can tolerate a relaxed version of encryption, in which the decryption algorithm need not recover the original plaintext m, but just some function $\mathfrak{g}(m)$, where g is a (not necessarily efficiently) invertible function. This relaxation allows us to view the ElGamal cryptosystem as additively homomorphic. Let us define it formally.

Definition 6 (Additively homomorphic $\mathfrak{g}$-semi-encryption scheme). *A set of three polynomial-time algorithms $AH = (\mathsf{KeyGen}_{AH}, \mathsf{Enc}_{AH}, \mathsf{Dec}_{AH})$ constitutes a semantically secure homomorphic $\mathfrak{g}$-semi-encryption scheme if it satisfies the following input-output specification as well as correctness, security, and homomorphic properties:*

Input-output specification KeyGen_{AH} *and* Enc_{AH} *have the same input-output specifications as those for key generation and encryption algorithms, respectively, for a public-key encryption scheme. The message space,* $\mathcal{M}_{\mathsf{pk}_{AH}}$*, may be parameterized by the public key* pk_{AH} *of the cryptosystem.* $\mathsf{Dec}_{AH}(\mathsf{sk}_{AH}, c)$ *takes as input a secret key* sk_{AH} *and a ciphertext, and outputs a value* $m' = \mathfrak{g}_{\mathsf{pk}_{AH}}(m)$ *for some* $m \in \mathcal{M}_{\mathsf{pk}_{AH}}$.

Correctness *For all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}_{AH}$*, for all* $m \in \mathcal{M}_{\mathsf{pk}_{AH}}$*, for all* $c \in \mathsf{Enc}_{AH}(\mathsf{pk}, m)$*,* $\mathsf{Dec}_{AH}(\mathsf{sk}, c) = \mathfrak{g}_{\mathsf{pk}_{AH}}(m)$*. I.e., the decryption algorithm correctly recovers* $\mathfrak{g}_{\mathsf{pk}_{AH}}(m)$ *from an encryption of* m.

Security *A semantically secure* $\mathfrak{g}$*-semi-encryption scheme must satisfy the same definition of semantic security as a regular semantically secure encryption scheme [GM82].*

Additively homomorphic properties *(1)* $\mathcal{M}_{\mathsf{pk}_{AH}}$ *is an algebraic ring (we will use* $\mathbb{Z}_\tau$ *as the ring) and (2) there is an efficient deterministic algorithm* Op_{AH} *that takes as input the public key* pk_{AH} *and two ciphertexts,* c_1 *and* c_2 *and outputs a ciphertext* c' *such that for all* $\mathsf{pk}_{AH} \in \mathsf{KeyGen}_{AH}$*, for all* $m_1, m_2 \in \mathcal{M}_{\mathsf{pk}_{AH}}$*, for all ciphertexts* $c_1 \in \mathsf{Enc}(\mathsf{pk}_{AH}, m_1)$ *and* $c_2 \in \mathsf{Enc}(\mathsf{pk}_{AH}, m_2)$*, if* $c' = \mathsf{Op}_{AH}(\mathsf{pk}_{AH}, c_1, c_2)$*, then* $c' \in \mathsf{Enc}(\mathsf{pk}_{AH}, m_1 + m_2)$.

For our constructions in Sec. 4.1 we define $\mathcal{M}_{\mathsf{pk}_{AH}}$ as $\mathbb{Z}_p$ for a prime p for ElGamal or $\mathbb{Z}_N$ for an RSA modulus N for Camenisch-Shoup.

Further (inspired by Cramer, Damgård and Nielsen's [CDN01] formalization of an additively homomorphic cryptosystem), we also need a way to sample new encryptions of messages, i.e., compute $c' \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, m)$ given any $c \in \mathsf{Enc}(\mathsf{pk}_{AH}, m)$. I.e. we require that this be achieved by forming a fresh encryption of 0, $c_0 \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, 0)$ and then adding to c, resulting in $c' = c \oplus c_0$ Further, we need AH to include efficient algorithms for obtaining $c' \in \mathsf{Enc}(\mathsf{pk}_{AH}, am)$ from $c \in \mathsf{Enc}(\mathsf{pk}_{AH}, m)$ and $a \in \mathbb{Z}_\tau$. [7]. Our application to privacy-preserving blueprints requires that the user's input y is in the message space $\mathcal{M}_{\mathsf{pk}_{AH}} = \mathbb{Z}_\tau$.

Note that the function $\mathfrak{g}_{\mathsf{pk}_{AH}}$ that determines the output of the decryption algorithm is parameterized by pk_{AH}; when clear from the context, we omit the parameterization. Also note that, when $\mathfrak{g}$ is the identity function, a semantically secure additively homomorphic $\mathfrak{g}$-semi-encryption scheme is just a regular additively homomorphic semantically secure encryption scheme.

Notation for additively-homomorphic encryption. We will generally use the lowercase c label to refer to ciphertexts (while uppercase C refers to commitments).

[7] In (1), we require randomization by adding an encryption of 0. This is needed for technical reasons that lead to a simpler construction; it may be possible to relax this requirement at the expense of a more complicated construction and proof. (2) follows generically from homomorphic properties, so explicitly requiring it is somewhat redundant, but we choose to do so for ease of presentation.

If c_1 and c_2 are ciphertexts, will use $c_1 \oplus c_2$ to denote the output of $\mathsf{Op}(\mathsf{pk}, c_1, c_2)$. We use $\boxed{a}_{\mathsf{pk}}$ to represent an encryption of a under the public key pk using the scheme AH; we will drop the subscript and denote it $\boxed{a}$ when pk is clear from the context. By $\boxed{a} = \boxed{c} \oplus \boxed{d}$ we denote that the ciphertext $\boxed{a}$ was generated by running the algorithm $\mathsf{Op}(\mathsf{pk}, \boxed{c}, \boxed{d})$; thus $\boxed{a} = \boxed{c+d}$. $y \odot \boxed{a}$ denotes applying this operation y times; in our instantiations this will yield $\boxed{ya}$ and is efficient for large y with repeated squaring; $\bigoplus_{i=0}^{n} \boxed{a_i}$ denotes applying Op n times on the set $\{\boxed{a_i} : i \in [0...n]\}$.

3 Our Succinct Proofs for Verifiable Secure Computation on Additively-Homomorphic Ciphertexts

Suppose that we have an additively homomorphic cryptosystem $\Gamma^{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \oplus, \odot)$. $\oplus$ denotes the algorithm for homomorphically adding two ciphertexts, and $\odot$ denotes the algorithm for multiplying a ciphertext with a known scalar. Recall (see Sect. 2.2) that, for any function $\mathfrak{g}$, by $\mathfrak{g}$-semi encryption we mean the following generalization of the notion of encryption: instead of outputting the plaintext m, the decryption algorithm outputs $\mathfrak{g}(m)$. Suppose that Γ^{Enc} is also a $\mathfrak{g}$-semi encryption scheme.

Let pk be a public key for this cryptosystem. Given a set of ciphertexts $c_1, \ldots, c_n$ whose plaintexts are $x_1, \ldots, x_n$, and a set of scalars $y_1, \ldots, y_k$, the additively homomorphic property of the cryptosystem allows anyone to compute a ciphertext c_f which is the encryption of $f(x_1, \ldots, x_n, y_1 \ldots, y_k)$, where f is a polynomial where each monomial is of the form $a_i x_i^{b_i} \prod_{j=1}^{k} y_j^{d_j}$, b_i is a bit ($\{0,1\}$), a_i is a coefficient of f, and d_j can be any integer. The time it takes to compute c_f is proportional to the time it would take to compute f in the clear.

Let Com be a non-interactive commitment scheme. In this section, we provide a framework for efficiently obtaining a proof system, in the random-oracle model, for the following relation, parameterized by public key pk and the function f:

$R_{params,\mathsf{pk},f}((r_1, \ldots, r_k, y_1, \ldots, y_k), (C_1, ..., C_k, c_1, \ldots, c_n, c_f)) = 1$ iff

$$\begin{aligned} &\exists x_1, \ldots, x_n \text{ such that} \\ &C_j = \mathsf{Com}(y_j, r_j)\ \forall 1 \le j \le k \\ &\wedge c_i \in \mathsf{Enc}(\mathsf{pk}, x_i)\ \forall 1 \le i \le n \\ &\wedge c_f \in \mathsf{Enc}(\mathsf{pk}, f(x_1, \ldots, x_n, y_1, \ldots, y_n)) \end{aligned}$$

Where *params* include the parameters for the homomorphic encryption scheme, a commitment scheme for scalars, and a commitment scheme for ciphertexts. In the remainder of the paper, we will omit the parameters and key $(params, \mathsf{pk})$ from this notation when it is clear, relabeling this relation as R_f.

The resulting proof system is complete, zero-knowledge and satisfies the definition of a (not straight-line extractable) proof of knowledge in the random-oracle model. To compile it into a partially straight-line extractable (g-BB-PSL) proof system, it will be sufficient to combine it with a g-BB-PSL proof of knowledge of the opening of the commitments $C_1, \ldots, C_k$ which we do in Sect. 5.3.

Construction of a proof system for R_f. Using a general NIZK proof system to prove R_f would yield a proof of size $\Omega(kd_{\mathsf{max}})$ where d_{max} is the largest degree among any $y_i, i \in [k]$. To make this more succinct, our proof system that halves the degree with each step. This reduces the size of the proof from linear in d_{max} to $O(k\log(d_{\mathsf{max}}))$, which is an exponential improvement. As we will see below, the proof size will be independent on the number of monomials in f and ciphertexts, and depends only on k (the number of variables $y_1, \ldots, y_k$) and the degree d_{max}.

Each step of this proof will reduce the task of proving the correct evaluation of a polynomial f to that of another polynomial, f'. To achieve succinctness, we will ensure that the degree of f' in one of the variables is at most half that of f. For example, proving that $c_f = \boxed{f(x_1, x_2, y_1, y_2) = x_1y_1^8y_2 + x_2y_1^7y_2}$ will be reduced to proving that $c_{f'} = \boxed{f'(x_1, x_2, y_1, y_2) = x_1'y_1^4y_2 + x_2'y_1^3y_2}$ where the ciphertexts $\boxed{x_1'}$ and $\boxed{x_2'}$ are derived from $\boxed{x_1}$ and $\boxed{x_2}$ in a way that is known to both prover and verifier. Because we want to achieve zero knowledge even when the adversary knows the secret key of the encryption scheme, a zero-knowledge simulator cannot simply make up an arbitrary value for $c_{f'}$: the adversary would be able to decrypt it and detect simulation. Thus, we need to instead commit to this value and perform the proof that the committed value was computed correctly. We call these *commitments to additively homomorphic ciphertexts* and we define them in Sect. 3.1 and construct them in Sect. 4.

3.1 Basic Building Blocks

Commitment to $\{y_1, \ldots, y_k\}$ Recall that our relation R_f is defined relative to a non-interactive commitment scheme $(\mathsf{CSetup}, \mathsf{Com})$. Com takes as input an element y from $\mathbb{Z}_\tau$, and a random value r sampled uniformly at random from $[R]$ for some integer R.

Proofs of correct modular addition and multiplication of committed values. In order to construct this proof system, we need to add and multiply the values in our scalar commitments together (modulo τ). Let us define the following relations:

- $R_{\mathsf{add}}((C_1, C_2, C_3), (x_1, r_1, x_2, r_2, x_3, r_3)) = 1$ iff $\forall i \in [3] : C_i = \mathsf{Com}(x_i; r_i)$, and $x_3 = x_1 + x_2 \bmod \tau$. Let $(\mathsf{Prove}^{\mathsf{add}}, \mathsf{Verify}^{\mathsf{add}})$ be a BB NIZK proof system for R_{add}.
- $R_{\mathsf{mult}}((C_1, C_2, C_3), (x_1, r_1, x_2, r_2, x_3, r_3)) = 1$ iff $\forall i \in [3] : C_i = \mathsf{Com}(x_i; r_i)$, and $x_3 = x_1x_2 \bmod \tau$. Let $(\mathsf{Prove}^{\mathsf{mult}}, \mathsf{Verify}^{\mathsf{mult}})$ be a BB NIZK proof system for R_{mult}.

We also need this commitment scheme to have a zero-knowledge proof of knowledge $(\mathsf{Prove}^{\mathsf{Com}}, \mathsf{Verify}^{\mathsf{Com}})$ of opening, i.e. a BB NIZK for the relation $R_{\mathsf{Com}} = ((C), (m, r))$ iff $\mathsf{Com}(m; r) = C$.

Commitment to ciphertexts. In order to prove correctness of an intermediate step in a longer computation over (semi-)encrypted data without revealing the ciphertext obtained in that step itself (which would leak data), we need to be able to commit to ciphertexts and prove properties of committed ciphertexts. Thus, we need a non-interactive statistically hiding, computationally binding commitment scheme Com_{AH} (parameterized by public parameters *params* generated by Setup_{AH}) for committing to ciphertexts $c \in \mathsf{Enc}_{AH}(\mathsf{pk}, \cdot)$ and we need protocols for proving statements about committed ciphertexts, as described below. We use a subscript notation (i.e. Com_{AH}) to distinguish this scheme from our commitments to scalars which do not have a subscript (the commitment function for scalars is Com). If randomness is not supplied to Com_{AH}, it will sample randomness and output it, e.g.: $(C, r) = \mathsf{Com}_{AH}(\boxed{a})$ implies that $C = \mathsf{Com}_{AH}(\boxed{a}; r)$.

Proofs of relations between committed ciphertexts. We need BB NIZK proof systems for (1) proving knowledge of a committed ciphertext; (2) proving that a committed ciphertext is the result of applying Op_{AH} to other committed ciphertexts; (3) proving that a committed ciphertext is the result of applying Op_{AH} to another committed ciphertext α times, where α is the opening of a commitment (under the commitment scheme Com) to an element of $\mathbb{Z}_\tau$; and (4) proving that a committed ciphertext is an encryption of a committed scalar. (4) is often called "verifiable encryption" (VE). More precisely, let us define the following relations:

- $R_{\mathsf{Com}_{AH}}(C, (c, r)) = 1$ iff $C = \mathsf{Com}_{AH}(c; r)$;
- $R_\oplus((C_1, C_2, C_3), (c_1, r_1, c_2, r_2, c_3, r_3)) = 1$ iff $\forall i \in [3] : C_i = \mathsf{Com}_{AH}(c_i; r_i)$ and $c_3 = \mathsf{Op}_{AH}(c_1, c_2)$;
- $R_\odot((C_1, C_2, C_3), (c_1, r_1, c_2, r_2, x, r_3)) = 1$ iff $\forall i \in [2] : C_i = \mathsf{Com}_{AH}(c_i; r_i)$, $C_3 = \mathsf{Com}(x; r_3)$ and $c_2 = c_1 \odot x$.
- $R_{VE}((C_1, C_2), (c_1, r_1, r_{c_1}, y, r_2)) = 1$ iff $C_1 = \mathsf{Com}_{AH}(c_1; r_1)$, $C_2 = \mathsf{Com}(y; r_2)$ and $c_1 = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y; r_{c_1})$.

Our construction will use as building blocks BB NIZK proof systems $(\mathsf{Prove}^{\mathsf{Com}_{AH}}, \mathsf{Verify}^{\mathsf{Com}_{AH}})$ for the relation $R_{\mathsf{Com}_{AH}}$, $(\mathsf{Prove}^\oplus, \mathsf{Verify}^\oplus)$ for the relation $R_\oplus$, $(\mathsf{Prove}^\odot, \mathsf{Verify}^\odot)$ for the relation $R_\odot$, and $(\mathsf{Prove}^{\mathsf{enc}}, \mathsf{Verify}^{\mathsf{enc}})$ for the relation R_{VE}. As before, we omit the parameters and public keys from these relations when it is clear. These proof systems exist generically for any cryptosystem and any set of commitment schemes; however, for the specific instantiations of semi-encryption and commitment schemes we consider, we also show how to construct them efficiently in Sec. 4.

Notation. We will use the following notation when invoking a proof system (inspired by the Camenisch-Stadler notation): $\pi = \mathsf{NIZK}[X, W : R(X, W)]$ denotes that the proof π is computed using the proof system for R on input a statement X and a witness W. When X is clear from the description of the relation R, we may omit it. For example, if we have $A = \mathsf{Com}_{AH}(\boxed{a}; r_a)$, $B = \mathsf{Com}_{AH}(\boxed{b}; r_b)$, and $C = \mathsf{Com}(c; r_c)$ and want to prove that $a = bc$, we'll denote the output of the prover's computation as $\pi = \mathsf{NIZK}[\boxed{a}, \boxed{b}, c, r_a, r_b, r_c : A = \mathsf{Com}_{AH}(\boxed{a}, r_a) \wedge B = \mathsf{Com}_{AH}(\boxed{b}, r_b) \wedge C = \mathsf{Com}(c; r_C, a_C) \wedge \boxed{a} = \boxed{b} \odot c]$. This π is computed by calling $\mathsf{Prove}^\odot(A, B, C, \boxed{a}, r_a, \boxed{b}, r_b, c, r_c)$. If π is accepted by the verification algorithm (i.e. $\mathsf{Verify}^\odot(A, B, C, \pi) = 1$) we can extract openings for A, B and C to ciphertexts $\boxed{a}$, $\boxed{b}$ and scalar c respectively, such that $\boxed{a} = \boxed{b} \odot c$.

3.2 Efficient Instantiation of Proof of R_f for $k = 1$

In this section we show how to efficiently instantiate a NIZK proof for the relation R_f when $k = 1$, i.e. there is a single variable y. Our main result in Appx. 3.3 subsumes the result in this section; however, this section makes it easier for the reader to understand the results in Appx. 3.3.

Observe that it is sufficient to provide a proof system for the polynomial $P = \sum_{i=0}^{n-1} x_i y^i$[8]. Thus we give a proof system for the relation R_P. Further, it is sufficient to give a proof system for a slightly more general relation, R_P^* in which the statement contains not the ciphertext c_P but a commitment $C_P = \mathsf{Com}_{AH}(c_P, r_P)$. To get a proof system for R_P, prover and verifier set $C_P = \mathsf{Com}_{AH}(c_P, 0)$ and invoke the proof system for R_P^*. Assume WLOG[9] that n (the number of ciphertexts) is a power of two. More formally,

$$\begin{aligned} R_P^*((r, y, c_P, r_P)\ ,\ (C_y, c_0, \ldots, c_{n-1}, C_P)) = 1 \text{ iff} \\ R_P(r, y, C_y, c_0, \ldots, c_{n-1}, c_P) = 1 \land C_P = \mathsf{Com}_{AH}(c_P, r_P). \end{aligned}$$

Input to the recursive step. Our PoK_P^* algorithm in Algorithm 2 recursively computes a proof until R_P^* is satisfied, i.e., C_P is a commitment to $c_P = \boxed{e} = \boxed{P(x_0, \ldots, x_{n-1}, y)}$. The input to PoK_P^* includes an auxiliary input aux, in addition to the statement and witness for the relation R_P^*. aux consists of (1) the part of the proof computed so far; (2) commitments to a logarithmic number of powers of y, i.e. commitments $\{C_{y^{2^i}}\}$ to $\{y^{2^i}\} = \{y^2, y^4, y^8, ..., y^{n/2}\}$ and (3) NIZK proofs that for $i > 2$, each $C_{y^{2^i}}$ is computed correctly from $C_{y^{2^{i-1}}}$ (using the proof system $(\mathsf{Prove}^{\mathsf{mult}}, \mathsf{Verify}^{\mathsf{mult}})$ described above). aux is of size that is logarithmic in n and the verifier need not verify any proofs in it more than once. We assume that the prover remembers how it computed aux (so we won't explicitly pass the openings of the commitments in aux to the recursive step). Alg. 1 is a "wrapper" algorithm that, on input the statement-witness pair for relation R_P transforms it into the statement-witness pair for relation R_P^*, initializes aux with $\{C_{y^{2^i}}\}$ and their proofs of correctness, and calls PoK_P^* .

Ensuring soundness for the recursive proof. The prover and verifier can both compute encrypted evaluations of the polynomial $P(x_0, \ldots, x_{n-1}, \gamma)$ on any input γ using the ciphertexts $\{c_i\}$. They can further break P into two parts such that $P(x_0, \ldots, x_{n-1}, \gamma) = P_1(x_0, \ldots, x_{n/2-1}, \gamma) + P_2(x_{n/2}, \ldots, x_{n-1}, \gamma)$ where P_1 contains the monomials $x_i \gamma^i$ for $i < n/2$, and P_2 contains monomials of higher

[8] From here, to obtain the proof system for any $f = a_{00} + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{i,j} x_j y^i$, we use the homomorphic properties of the cryptosystem to compute $c'_i = \boxed{\sum_{j=0}^{n-1} a_{i,j} x_j}$ for $0 \leq i < n$, (deterministically, using the all-0 string for encryption) incorporate the term a_{00} by letting $c''_0 = c'_0 \oplus \boxed{a_{00}}$ and then invoke the proof system for P on input ciphertexts $c''_0, c'_1, \ldots, c'_{n-1}$.

[9] This is without loss of generality: to reduce to this case, prover and verifier can both compute the extra ciphertexts $c_n, \ldots, c_{2^a-1}$ (so that the total number is a power of two) by encrypting 0 with fixed randomness.

degree in γ. We can represent P as $P(x_0, \ldots, x_{n-1}, \gamma) = P_1(x_0, \ldots, x_{n/2-1}, \gamma) + \gamma^{n/2} P_3(x_{n/2}, \ldots, x_{n-1}, \gamma)$ where $P_3(\gamma) = P_2(\gamma)/\gamma^{n/2}$.

To recurse, the prover commits to ciphertexts $\boxed{e_1} = \boxed{P_1(x_0, \ldots, x_{n/2-1}, y)}$, $\boxed{e_2} = \boxed{P_2(x_{n/2-1}, \ldots, x_{n-1}, y)}$, $\boxed{e_3} = \boxed{P_3(x_{n/2}, \ldots, x_{n-1}, y)}$, and then proves (using the proof systems for proving properties of committed ciphertexts) that $\boxed{e} = \boxed{e_1} \oplus \boxed{e_2}$ and $\boxed{e_2} = y^{n/2} \odot \boxed{e_3}$ using the commitment $C_{y^{n/2}}$ found in aux. Thus, the prover has reduced the task of proving that C_P is a commitment to $\boxed{e} = \boxed{P(x_0, \ldots, x_{n-1}, y)}$ for a polynomial P of degree $n-1$ to the task of proving that C_{P_1} is a commitment to $\boxed{e_1} = \boxed{P_1(x_0, \ldots, x_{n/2-1}, y)}$ and C_{P_3} is a commitment to $\boxed{e_3} = \boxed{P_3(x_0, \ldots, x_{n/2-1}, y)}$, where P_1 and P_3 are both polynomials of degree $n/2-1$.

To take advantage of recursion, we need to use just one recursive call in order to prove that the openings of C_{P_1} and C_{P_3} (i.e., $\boxed{e_1}$ and $\boxed{e_3}$ respectively) are encrypted evaluations of P_1 and P_3. To do so, prover and verifier define a new polynomial P' of degree $(n-1)/2$ by taking a random linear combination of P_1 and P_3: let α be the output of the random oracle on input the elements of the proof that have been computed so far. Let $P'(x_0, \ldots, x_{n-1}, y) = P_1(x_0, \ldots, x_{n-1}, y) + \alpha P_3(x_0, \ldots, x_{n-1}, y)$. By the Schwartz-Zippel Lemma (Lemma 1), if committed $\boxed{e_1} \neq \boxed{P_1(y)}$ or committed $\boxed{e_3} \neq \boxed{P_3(y)}$, then with overwhelming probability over the choice of α, $\boxed{e_1} \oplus (\alpha \odot \boxed{e_3}) \neq \boxed{P'(x_0, \ldots, x_{n-1}, y)}$. Let $C_{P'}$ be a commitment to the ciphertext $\boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_3})$; the prover can provide a proof that indeed $C_{P'}$ is a commitment to $\boxed{e'}$ computed this way based on C_{P_1} and C_{P_3} and α using the proof systems for committed ciphertexts.

Next, we use recursion in order to prove that $C_{P'}$ corresponds to correctly evaluating the polynomial P', i.e. it is a commitment to $\boxed{P'(x_0, \ldots, x_{n-1}, y)}$. To do so, we call PoK^*_P on input ciphertexts $(c'_0, \ldots, c'_{n/2-1})$ where $c'_i = \boxed{x_i} \oplus (\alpha \odot \boxed{x_{n/2} + i})$.

Notational remarks. For compactness, here we only present the prover's algorithms; the verifier's algorithms (provided in the appendix) should follow from the prover's algorithms. For readability, in the list of inputs to the prover, we underline those inputs that are also given to the verifier.

Algorithm 1 $\mathsf{PoK}_P(r, y, \underline{C_y, c_0, \ldots, c_{n-1}, c_P}) \to \pi$

```
    Let $c_i = \{\boxed{x_i}\}_{i \in [0 \ldots n-1]})$;
    Prover needs to prove that $c_P = \boxed{e} = \bigoplus_{i=0}^{n}(\boxed{x_i} \odot y^i)$
    To format $c_P$ for the recursion, we commit to it with known randomness e.g. 0
$C_P \leftarrow \mathsf{Com}_{AH}(c_P; 0)$
For $i = 1$ to $\log n$, let $(C_{y^{2^i}}, r_i) = \mathsf{Com}(y^{2^i})$
    and let $\pi_{y^{2^i}} \leftarrow \mathsf{NIZK}[(z, r_{i-1}, r_i) : C_{y^{2^{i-1}}} = \mathsf{Com}(z; r_{i-1}) \wedge C_{y^{2^i}} = \mathsf{Com}(z^2; r_i)]$.
Initialize $\mathsf{aux} = (\{C_{y^{2^i}}\}, \{\pi_{y^{2^i}}\})$.
**return** $\mathsf{PoK}^*_P(r_y, y, c_P, r_P, C_y, c_0, \ldots, c_{n-1}, C_P, \mathsf{aux})$
```

Algorithm 2 $\mathsf{PoK}^*_P(r_y, y, c_P, r_P, \underline{C_y, c_0, \dots, c_{n-1}, C_P, \mathsf{aux}}) \to \pi$

```
Let $c_i = \{\boxed{x_i}\}_{i \in [0 \dots n-1]}; c_P = \boxed{e}$
    Prover needs to prove that $C_P = \mathsf{Com}_{AH}(\boxed{e}; r_P)$ where $\boxed{e} = \bigoplus_{i=0}^{n-1} \boxed{x_i} \odot y^i = \boxed{\sum_{i=0}^{n-1} y^i x_i}$ and $C_y = \mathsf{Com}(y; r_y)$

If the degree of the polynomial is low enough, prove its computation directly:
if $n = 1$, return $(\mathsf{aux}, \pi_1)$ where $\pi_1 \leftarrow \mathsf{NIZK}[r : \mathsf{Com}_{AH}(\boxed{x_0}, r) = C_P]$
If not, we will need to reduce the degree needed to prove $C$ and recurse.
To do so, first, commit to the lower half of the polynomial:
$(C_1, \rho_1) = \mathsf{Com}_{AH}(\boxed{e_1})$ where $\boxed{e_1} = \bigoplus_{i=0}^{n/2-1} \boxed{x_i} \cdot y^i = \boxed{\sum_{i=0}^{n/2-1} y^i x_i}$
Next, commit to the upper half of the polynomial
$(C_2, \rho_2) = \mathsf{Com}_{AH}(\boxed{e_2})$
    where $\boxed{e_2} = \bigoplus_{i=0}^{n/2-1} \boxed{x_{i+n/2}} \odot y^{i+n/2} = \boxed{\sum_{i=0}^{n/2-1} y^{i+n/2} x_{i+n/2}}$
Lastly, commit to the upper half of the polynomial with the degree lowered by half
$(C_3, \rho_3) = \mathsf{Com}_{AH}(\boxed{e_3})$ where $\boxed{e_3} = \bigoplus_{i=0}^{n/2-1} \boxed{x_{i+n/2}} \odot y^i = \boxed{\sum_{i=0}^{n/2-1} y^i x_{i+n/2}}$
Query the random oracle on the current transcript of the proof so far,
  i.e. on $\tau = (\mathsf{aux}, C_1, C_2, C_3)$ to get a random value, $\alpha$.
$\alpha \leftarrow H(\tau)$
Compute the encryptions of the new coefficients for a reduced degree polynomial
$\forall i \in [n/2 - 1], c'_i = \boxed{x'_i} = \boxed{x_i} \oplus (\boxed{x_{i+n/2}} \odot \alpha)$
Compute a new evaluation over this reduced degree polynomial:
$(C', r') = \mathsf{Com}_{AH}(\boxed{e'})$ where $\boxed{e'} = \bigoplus_{i=0}^{n/2-1} \boxed{x'_i} \odot y^i$
Prove that this new commitment $C'$ is consistent with $C_P$,$C_1$,$C_2$, and $C_3$.
$\pi_\alpha \leftarrow \mathsf{NIZK}[r, \rho_1, \rho_2, \rho_3, r', r_y, y, \boxed{e}, \boxed{e_1}, \boxed{e_2}, \boxed{e_3}, \boxed{e'}$ :
    $\mathsf{Com}_{AH}(\boxed{e}, r) = C_P \wedge \mathsf{Com}_{AH}(\boxed{e'}, r') = C' \wedge \forall 1 \leq i \leq 3 : \mathsf{Com}_{AH}(\boxed{e_i}, \rho_i) = C_i$
    $\wedge \boxed{e} = \boxed{e_1} \oplus \boxed{e_2}$
    $\wedge \boxed{e_2} = y^{n/2} \odot \boxed{e_3}$                    ▷ proven relative to $C_{y^{n/2}}$ in aux
    $\wedge \boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_3})]$
Append $(C_1, C_2, C_3, C', \pi_\alpha)$ to aux
return $(\mathsf{PoK}^*_P(r_y, y, \boxed{e'}, r', C_y, c'_0, \dots, c'_{n/2-1}, C', \mathsf{aux}))$
```

Theorem 1. *Our scheme in Algs. 1 and 2 are complete and ZK (Def. 1).*

Theorem 2. *The* PoK^*_P *function in Alg. 2 is black-box (BB) simulation extractable with respect to Def. 2 for the relation* R^*_f.

We provide the verification function for PoK^*_P (V^*_P) in Alg. 3.

We prove Thms. 1 and 2 next.

Proof of Thm. 1 (Completeness and ZK). Completeness is clear by inspection.

The zero knowledge property of Alg. 2 relies on the hiding and zero knowledge property of our underlying ciphertext and scalar commitment scheme and associated protocols described in Sec. 3.1 and constructed in Sec. 4. Since we have committed to all values and do all proofs with a NIZK scheme with a trapdoor that allows our simulator to produce proofs for relations not in the language, we can simply choose random elements as our commitments and simulate all proofs.

Algorithm 3 $\mathsf{V}^*_P(\underline{C_y, c_0, \ldots, c_{n-1}, C_P, \mathsf{aux}}, \pi) \to \{0,1\}$

```
**parse** $\pi = \big(\pi', C_y, c_0, ..., c_n, \mathsf{aux}_i = (C_1, C_2, C_3, C', \pi_\alpha, \pi_1)\big)$
**if** $n = 1$,
    Verify $\pi_1$
    **return** 0 if $\pi_1$ didn't verify, otherwise, **return** 1
Random oracle hash current transcript ($\tau$) of the proof (including all inputs)
$\alpha \leftarrow H(\tau)$
Verify $\pi_\alpha$
Verify $\pi'$ by recursing into $\mathsf{V}^*_P$.
If any proof failed to verify, **return** 0, otherwise **return** 1
```

We show the simulator for PoK_f and $R^*_{f_y}$ in Algs. 5 and 4 for completeness in Sec. 3.2. We can see that if we replace the real commitments and proofs one-by-one with hybrids, an adversary that can distinguish these hybrids can defeat either the hiding of the commitment or the zero knowledge of the proof systems.

We quickly review the Schwartz-Zippel lemma [Sch80,Sho97] in Lemma 1. We will use this in our proof of black-box simulation extractability proof for Alg. 2 in Thm. 2

Lemma 1 (Schwartz-Zippel [Sch80,Sho97]). *For two distinct polynomials, $r(\chi)$, $r'(\chi)$, over a field, $\mathbb{F}$ of size p, the probability that $r(\alpha) = r'(\alpha)$ when α is sampled randomly from $\mathbb{F}$ is d/p where d is the larger degree out of either polynomial, $d = \max\{\deg r, \deg r'\}$. Where "distinct polynomials" means there exists some power where the coefficients for r and r' differ.*

We need one more form of the Schwartz-Zippel lemma in order to prove our construction sound for Camenisch-Shoup encryptions which we show in Lemma 2

Lemma 2 (Schwartz-Zippel for $\mathbb{Z}_n$). *For two distinct polynomials, $r(\chi)$, $r'(\chi)$, over a ring, $\mathbb{Z}_n$ where $n = pq$ for p, q prime, the probability that $r(\alpha) = r'(\alpha)$ when α is sampled randomly from $\mathbb{Z}_n$ is d/p where d is the larger degree out of either polynomial, $d = \max\{\deg r, \deg r'\}$ and WLOG $q \geq p$. Where "distinct polynomials" means there exists some power where the coefficients for r and r' differ.*

Proof of Lemma 2. Let us label the polynomial, $r(\chi) - r'(\chi)$, as $t(\chi)$. We can see that because $t(\alpha) = 0 \mod n$, we have that $t(\alpha) = 0 \mod p$ and $t(\alpha) = 0 \mod q$ since $p|n$ and $q|n$. Let us define a map from $\mathbb{Z}_n[x]$ to $\mathbb{Z}_p[x]$, ϕ_p where for $t(\chi) = t_0 + t_1\chi + ... + t_d\chi^d$ we have that $\phi_p(t(\chi)) = \sum s_i\chi^i$ where $s_i = t_i \mod p$. Thus, if $t(\alpha) = u \mod n$, then $s(\alpha) = u \mod p$. We also know that the polynomial, $t(\chi)$ in $\mathbb{Z}_n[\chi]$ is not identically zero for one of the two polynomial $\phi_p(t)$ or $\phi_q(t)$. If this were not true, then the coefficients of $t(\chi)$ in $\mathbb{Z}_n$ would be multiples of both p and q (since p, q prime and $pq = n$) and thus the coefficients would be multiples of n. This would mean the coefficients would be zero in $\mathbb{Z}_n$ but we've assumed that $t(\chi) \in \mathbb{Z}_n[x]$ is not identically zero. WLOG we'll assume $\phi_p(t)$ is a non-zero polynomial in $\mathbb{Z}_p[\chi]$. We thus know that we can map this polynomial onto a non-zero polynomial in $\mathbb{Z}_p[\chi]$. We'll call this polynomial $s(\chi) \in \mathbb{Z}_p[\chi]$. Thus, we know that $s(\alpha) = 0 \mod p$ since $t(\alpha) \in \mathbb{Z}[x]$ is some multiple of n and $p|n$. Because $s(\alpha) = 0 \mod p$ and $s(\chi) \mod p$ is not identically zero, we can use Lemma 1 for the field $\mathbb{Z}_p$ to determine the probability of this evaluating to 0 (for a random evaluation point) is d/p. Because this must be true if $t(\alpha) = 0 \mod n$, this must only occur with at most d/p probability. By choosing p to be the smaller prime factor of n, we've proven our bound in Lemma 2. □

Proof of Thm. 2 (Simulation extractability of PoK^*_P*).* This property of Alg. 2 relies on the BB-extraction and binding of our underlying ciphertext and scalar commitment scheme and associated protocols described in Sec. 3.1 and constructed in Sec. 4. We can use the simulator ($\mathsf{SimPoK_P}$) in Alg. 5 for this reduction. Because our simulator is zero knowledge, the BB-simulation-extractability adversary gets no advantage when given these proofs.

To do this, we'll prove that C is correctly computed and that we can extract the witnesses for the relation. We can prove that we can extract recursively. As a base case, we see that when ProveRecursive is called with $n = 1$. We can see on line 1 that in this case, the correct computation of P is directly computed.

Thus, if we can prove that C is correctly computed, assuming that C' is correctly computed, we can use induction to conclude that the original commitment given to the recursion from $\Psi_2.P$ (on line 4 of Alg. 7) was correctly computed. From the proof, π_{rec}, we know that $P'(y) = e_1 + \alpha e_3$. We see that α is computed from a hash of the transcript, including C_1 and C_3. Thus, the adversary cannot make e_1 or e_3 depend on α, since this would reduce to either distinguishing a random oracle or double opening C_1 or C_3. We now rewrite these polynomials and fix y to reform these as: $q(\chi) = e_1 + \chi e_3$ and $q'(\chi) = \sum_{i=0}^{n/2-1} y^i a_i + \sum_{i=0}^{n/2} \chi y^i a_{i+n/2}$. For the proof to succeed, $q(\chi)$ must equal $q'(\chi)$ when evaluated at the random value, α. We know from the Schwartz-Zippel lemma (Lemma 1) that the probability of this occurring when $q(\chi)$ is distinct from $q'(\chi)$ is negligible in the size of the ring, $\mathbb{Z}_\tau$. Thus, with overwhelming probability, these must be equivalent polynomials. Because α is multiplied by the right term and not the left, and (with overwhelming probability) the polynomials are equivalent, this further proves that $e_1 = \sum_{i=0}^{n/2-1} y^i a_i$ and $e_3 = \sum_{i=0}^{n/2} y^i a_{i+n/2}$. This is because e_1 is the 0-degree coefficient in $q(\chi)$ and $\sum_{i=0}^{n/2-1} y^i a_i$ is the 0-degree coefficient in $q'(\chi)$ (with similar reasoning for e_3 and $\sum_{i=0}^{n/2} y^i a_{i+n/2}$ for being the 1-st degree coefficient of $q(\chi)$ and $q'(\chi)$). We then see that π_C proves that $e_2 = e_3 \odot y^{n/2}$. Thus, $e_2 = e_3 \odot y^{n/2}$ and since we proved e_3 correctly with π_C, we now know that $e_2 = \sum_{i=0}^{n/2} \chi y^{i+n/2} a_{i+n/2}$. We then see that π_{rec} proves that $e = e_1 + e_2$, which proves that $e = \sum_{i=0}^{n} \chi y^i a_i$, thus, proving C to be correctly formed. Thus, after extracting all witnesses from the underlying NIZKs, we know that these are correct witnesses for the relation.

Algorithm 4 $\mathsf{SimPoK_P}(\underline{C_y, c_0, \ldots, c_{n-1}, c_P}) \to \pi$

```
$C_P \leftarrow \mathsf{Com}_{AH}(*; 0)$ where $*$ is a random value
For $i = 1$ to $\log n$, let $(C_{y^{2^i}}, r_i) = \mathsf{Com}(*)$
        and let $\pi_{y^{2^i}} \leftarrow \mathsf{Sim}[(z, r_{i-1}, r_i) : C_{y^{2^{i-1}}} = \mathsf{Com}(z; r_{i-1}) \wedge C_{y^{2^i}} = \mathsf{Com}(z^2; r_i)]$.
Initialize $\mathsf{aux} = (\{C_{y^{2^i}}\}, \{\pi_{y^{2^i}}\})$.
**return** $\mathsf{SimPoK}^*_P(C_y, c_0, \ldots, c_{n-1}, C_P, \mathsf{aux})$
```

Algorithm 5 $\mathsf{SimPoK}^*_P(\underline{C_y, c_0, \ldots, c_{n-1}, C_P, \mathsf{aux}}) \to \pi$

```
**if** $n = 1$, **return** $(\mathsf{aux}, \pi_1)$ where $\pi_1 \leftarrow \mathsf{Sim}[r : \mathsf{Com}_{AH}(\boxed{x_0}, r) = C_P]$
$(C_1, \rho_1) = \mathsf{Com}_{AH}(*)$
$(C_2, \rho_2) = \mathsf{Com}_{AH}(*)$
$(C_3, \rho_3) = \mathsf{Com}_{AH}(*)$
$\alpha \leftarrow H(\tau)$
$\forall i \in [n/2 - 1], c'_i = \boxed{x'_i} = \boxed{x_i} \oplus (\boxed{x_{i+n/2}} \odot \alpha)$
$(C', r') = \mathsf{Com}_{AH}(*)$
$\pi_\alpha \leftarrow \mathsf{Sim}[r, \rho_1, \rho_2, \rho_3, r', r_y, y, \boxed{e}, \boxed{e_1}, \boxed{e_2}, \boxed{e_3}, \boxed{e'}$ :
    $\mathsf{Com}_{AH}(\boxed{e}, r) = C_P \wedge \mathsf{Com}_{AH}(\boxed{e'}, r') = C' \wedge \forall 1 \leq i \leq 3 : \mathsf{Com}_{AH}(\boxed{e_i}, \rho_i) = C_i$
    $\wedge \boxed{e} = \boxed{e_1} \oplus \boxed{e_2}$
    $\wedge \boxed{e_2} = y^{n/2} \odot \boxed{e_3}$                    ▷ proven relative to $C_{y^{n/2}}$ in $\mathsf{aux}$
    $\wedge \boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_3})]$
Append $(C_1, C_2, C_3, C', \pi_\alpha)$ to $\mathsf{aux}$
**return** $\big(\mathsf{SimPoK}^*_P(C_y, c'_0, \ldots, c'_{n/2-1}, C', \mathsf{aux})\big)$
```

3.3 Proof System for Multivariate Polynomials

We present our algorithm for polynomials with multiple y_i values in Alg. 6. This algorithm proves the relation R_y described at the start of this section. In essence, the algorithm will perform the same recursive step as Alg. 2 until it has reduce the degree of a y_i variable to 0. The algorithm then recurses on the remaining $k - 1$ variables until none are left. At this point, the evaluation has been fully proven.

For intuition, we provide an example polynomial: $f(x_1, x_2, y_1, y_2) = a_1x_1y_1y_2 + a_2x_2y_1^2y_2$. We can see that our proof will first focus on y_1, finding that the maximum degree of this variable, $d_{\mathsf{max}} = 2$. It will then compute $f_1(x_1, x_2, y_1, y_2) = a_1x_1y_1y_2$ and $f_2(x_1, x_2, y_1, y_2) = a_2x_2y_1^2y_2$. It will then compute $f_3(\ldots) = (a_2x_2y_1^2y_2)/y_1 = a_2x_2y_1y_2$, commit to encryptions of these polynomials, and hash the transcript to receive the challenge, α. It will then prove the relation $f(\ldots) = f_1(\ldots) + f_2(\ldots)$ and $f_2(\ldots) = y * f_3(\ldots)$. It will then compute $f_4(\ldots) = f_1(\ldots) + \alpha f_3(\ldots) = a_1 x_1 y_1 y_2 + a'_2 x_2 y_1 y_2$ where $a'_2 = a_2 * \alpha$. This process will repeat for f_4, and this time we'll see that $f_1(\ldots) = 0$ (since no monomial has degree of y_1 less than $d_{\mathsf{max}}/2 = 1/2$) and $f_3(\ldots) = (a_1 x_1 y_1 y_2 + a_2 x_2 y_1 y_2)/y_1 = a_1 x_1 y_2 + a_2 x_2 y_2$. Thus, $f_4(\ldots) = f_1(\ldots) + \alpha f_3(\ldots) = 0 + a_1 x_1 y_2 + a_2 x_2 y_2$ and thus, we've removed y_1 from the polynomial to be proven. Once this repeats to remove y_2, we're left with $f(\ldots) = a_1 x_1 + a_2 x_2$ where a_1 and a_2 are some combination of the coefficients of f and the challenges (α's) from the previous recursive steps. This is a linear function in the x_i's where the α's are known by the verifier so the verifier can simply compute the encryption of $f(\ldots)$ at this point and the prover can prove that they've committed to this encryption.

In this proof function, we prove a special class of polynomials, which is simpler to present, though just as powerful. In this class of polynomials, we break the polynomial down in terms of monomials (polynomials with a single term) of powers of the different y_i variables. Specifically, each polynomial is defined by a vector of coefficients, $(a_1, ..., a_n)$, and a vector of powers of y_i's, for each a_i, $((d_{1,1}, \ldots, d_{1,k}), \ldots, (d_{n,1}, \ldots, d_{n,k}))$ such that $d_{i,j}$ is the power of y_j in the monomial with coefficient a_i. The resulting form of the polynomial looks as: $f = \sum_{i=1}^{n} a_i x_i \prod_{j=1}^{k} y_j^{d_{i,j}}$. We then show that any polynomial (which is linear in the x_i's) can be proven correct using this proof by possibly duplicating x_i's and adding an extra encryption of 1 to the x_i's to ensure the polynomial can have a degree-0 term in any x_i. As in Alg. 2, we assume that the prover also has already created a commitment to each $\{y_i, y_i^2, y_i^4, y_i^8, ..., y_i^{d_i}\}$ where d_i is the largest power of y_i in the polynomial and proved that it was correct, and these commitments and proofs are included in the aux variable passed to the proof and they are implicit and used in line 17 in Alg. 6. We also prove the relation such that the verifier only has a commitment to c_f instead of the actual ciphertext, similar to PoK_P^* in Sect. 3.2. This allows us to recursively call PoK_f^* without revealing intermediate ciphertexts.

In this proof of knowledge, we reduce the degree of y_1 by half at each step. We assume that the maximum degree of each variable, y_i, is a power of 2 [10]. After a logarithmic number of recursions, we'll have that y_1 only has degree 1 when calling the proof. This will be divided out in line 10 of the proof (in Alg. 6) and thus, we'll be left with f_4 (the polynomial we recurse on) being a degree 0 polynomial in y_1. Thus, on the next recursive step, we'll trigger the conditional on line 4 and will remove y_1 from the witnesses (and polynomial). Thus, our proof will remove variables, y_i, one-by-one, until we have 0 left, in which we'll trigger the conditional on line 1, in which we're almost finished since at this point, f is a function of linear operations on the x_i values which the verifier can compute. The prover simply needs to prove that the C_f is a commitment to the

[10] If not, we can add a "dummy" monomial with the smallest power of 2 in each variable such that this degree is larger than any degree of that variable in the original polynomial. This dummy monomial can simply have a coefficient of 0 to ensure it doesn't affect the outcome.

c_f computed by the verifier. We note the steps that a verifier can also compute with a star (*). We give our results for this relation in Theorems 4 and 3.

Theorem 3. *Our scheme in Alg. 6 is complete and ZK (Def. 1).*

Theorem 4. *The function in Alg. 6 is black-box (BB) simulation extractable with respect to Def. 2 for the relation R_f defined in Sect. 3.*

Complexity analysis. We can see that at each step, we reduce the degree of one of the y_i variables by half. By the end, all of the y_i variables have been removed from the polynomial and thus because our polynomial is linear in the x_i's, the verifier can compute the encryption themselves, meaning our proof is independent of n. Thus, our complexity will be $O(k \log(d_{\mathsf{max}}))$ where d_{max} is the maximum degree among all y_i variables in the polynomial.

Proof of theorems 3 and 4. For zero knowledge, it's easy to see that because we're committing to every encryption and variable, and using ZKPs to manipulate them, our proof is also ZK. On the last recursion, the verifier does see an encryption in the clear, which seems to contradict zero-knowledge, but we can see that this is simply a combination of the original coefficients (x_i) and random outputs from the random oracle. For BB extraction, we can prove this by induction. If $f_4(\ldots)$ is correctly computed, and C_4^* is truely a commitment to $c_1^* \oplus \alpha c_3^*$. Then, we know that $f_3(\ldots)$ and $f_1(\ldots)$ must be correctly computed (due to similar logic as the proof for PoK_{f_y}). Thus, because we've also proven that $f_2(\ldots) = y^{d_{\mathsf{max}}/2}$ and $f(\ldots) = f_1(\ldots) + f_2(\ldots)$, we've proven correctness of $f(\ldots)$. When $d_{\mathsf{max}} = 0$, we simply relabel our witnesses, removing one which isn't necessary to prove $f(\ldots)$ anymore. As our base case, we have that if there are no y_i variables left, we can prove correctness of the encryption of c_f.

4 Constructions of Commitments to Additively-Homomorphic Ciphertexts

We first define variants of ElGamal and Camenisch-Shoup encryption, in Sec. 4.1. Specifically, we define "lifted" ElGamal and Camenisch-Shoup in a "commitment-friendly" group. We then construct commitments to ciphertexts and associated proof systems for adding and multiplying ElGamal ciphertexts and Camenisch-Shoup ciphertexts. We use (Lifted) ElGamal which is a $\mathfrak{g}$-semi-encryption as defined in Sec. 3 with message space $\mathcal{M}_{\mathsf{pk}} = \mathbb{Z}_p$ and $\mathfrak{g}(x) = h^x \mod p$. Camenisch-Shoup encryption has the advantage that it allows for the efficient computation of discrete logarithms in a subgroup of size n where n is an RSA modulus. Thus, with Camenisch-Shoup encryption, we can efficiently decrypt ciphertexts when the message space has exponential size. Thus, our Camenisch-Shoup construction is a $\mathfrak{g}$-semi-encryption where $\mathfrak{g}$ is the identity function (i.e. a standard encryption scheme). In our Camenisch-Shoup construction, the message space is $\mathcal{M}_{\mathsf{pk}} = \mathbb{Z}_n$. In Sec. 4.2 we construct commitments to ElGamal ciphertexts. In Sec. 4.3 we construct commitments to Camenisch-Shoup ciphertexts.

Algorithm 6 $\mathsf{PoK}^*_f(params, f, X, W)$

```
parse $f = \sum_{i=1}^{n} a_i x_i \prod_{j=1}^{k} y_j^{d_{i,j}}$; in other words, $f$ consists of $n$ monomials $(m_1, ..., m_n)$ and for $1 \leq i \leq n$, the $i^{th}$ monomial involves is linear in $x_i$; it is a product of $x_i$ and the monomials of $y$-variables, $m_i(y_1, \dots, y_k) = \prod_{j=1}^{k} y_j^{d_{i,j}}$ where $d_{i,j}$ is the degree of variable $y_j$ in the $i^{th}$ monomial.
parse $X = (\mathsf{pk}_{AH}, \boxed{x_1}, \dots, \boxed{x_n}, C_1, \dots, C_k, C_f)$
    and $W = (y_1, ..., y_k, c_f, r_1, ..., r_y, r_f)$.
$W = (y_1, \dots, y_k, r_1, \dots, r_k, c_f = \boxed{f(x_1, \dots, x_n, y_1, \dots, y_k)}, r_f)$
if $k = 0$,
    return Prove that $C_f$ is the commitment to $c_f = \boxed{\sum_{i=1}^{n} a_i x_{j_i}}$ (the verifier can compute $c_f$ autonomously).
Let $d_{\mathsf{max}}$ be the maximum degree of $y_1$ in any monomial.
if $d_{\mathsf{max}} = 0$ (i.e. $y_1$ does not appear in $f$),
    return $\mathsf{PoK}_f(params, f', X', W')$ where $f' = f$, $X' = (\mathsf{pk}_{AH}, \boxed{x_1}, \dots, \boxed{x_n}, C_2, \dots, C_k, C_f)$, $W' = (y_2, \dots, y_k, r_1, \dots, r_k, c_f = \boxed{f(x_1, \dots, x_n, y_1, \dots, y_k)}, r_f)$.
Recursive step:
* Let $(e'_1, ..., e'_t)$ be the indices such that $y_1$ in the monomials $(m_{e'_1}, ..., m_{e'_t})$ has degree $\geq d_{\mathsf{max}}/2$. Let $(e^*_1, ..., e^*_s)$ be the indices of the remaining monomials $(m_{e^*_1}, ..., m_{e^*_s})$ with degree $< d_{\mathsf{max}}/2$ over $y_1$. Note that $s + t = n$.
* Let $f_1(x_1, ..., x_n, y_1, ..., y_k) = \sum_{i=1}^{t} a_{e^*_i} x_{e^*_i} \prod_{j=1}^{k} y_j^{d_{e^*_i,j}}$
* Let $f_2(x_1, ..., x_n, y_1, ..., y_k) = \sum_{i=1}^{s} a_{e'_i} x_{e'_i} \prod_{j=1}^{k} y_j^{d_{e'_i,j}}$
* Let $f_3(x_1, ..., x_n, y_1, ..., y_k) = \sum_{i=1}^{s} a_{e'_i} x_{e'_i} (\prod_{j=1}^{k} y_j^{d_{e'_i,j}})/y_1^{d_{\mathsf{max},1}/2}$
Compute $\forall i \in [3], c^*_i = \boxed{(f_i(x_1, ..., x_n, y_1, ..., y_k))}$ computed homomorphically from the input to the prover, and let $\forall i \in [3], (C^*_i, \kappa_i) = \mathsf{Com}(c^*_i)$.
Let $\alpha = H(\tau)$ where $\tau$ is a trascript of the proof so far (along with the statement and parameters) that includes $C^*_1$,, $C^*_2$ and $C^*_3$.
* Let $x'_1, \dots, x'_n$ be a reordering of $x_1, \dots, x_n$ such that $x'_1, \dots x'_t$ correspond to the monomials in which $y_1$ was of degree $< d_{\mathsf{max}}/2$, and $x'_{t+1}, \dots, x_n$ correspond to those where the degree was $\geq d_{\mathsf{max}}/2$.
* Let $(x^*_1, \dots, x^*_n) = (x'_1, \dots, x'_t, \alpha x'_{t+1}, \dots, \alpha x'_n)$. Compute $\boxed{x^*_1}, \dots, \boxed{x^*_n}$, and let $X^*$ be the same as $X$ except that $\boxed{x_1}, \dots, \boxed{x_n}$ are replaced by $\boxed{x^*_1}, \dots, \boxed{x^*_n}$, so the order in which the encrypted $x$ variables appear in $X^*$ corresponds to the order in which they appear in the monomials of $f_4$.
* Let $f_4(x_1, ..., x_n, y_1, ..., y_k) = f_1(x_1, ..., x_n, y_1, ..., y_k) + \alpha f_3(x_1, ..., x_n, y_1, ..., y_k)$.
Compute $c^*_4 = \mathsf{Enc}(f_4(x_1, ..., x_n, y_1, ..., y_k))$ homomorphically using $X^*$, and $(C^*_4, r^*_4) = \mathsf{Com}(c^*_4))$.
Prove that $c^*_2 = c^*_3 \odot y_1^{d_{\mathsf{max}}/2}$ using the commitments, $C^*_i$ and openings, $\kappa_i$, using $\mathsf{Prove}^{mult}_{AH}$, yielding $\pi_\alpha$.
Prove that $c_f = c^*_1 \oplus c^*_2$ using the commitments, $C_f$, $C^*_i$ and openings, $r_f, \kappa_i$, using $\mathsf{Prove}^{add}_{AH}$, yielding $\pi_f$.
Prove that $c^*_4 = c^*_1 \oplus \alpha c^*_3$ using the commitments, $C_4$, $C^*_i$ and openings, $r_4, \kappa_i$, using $\mathsf{Prove}^{add}_{AH}$, yielding $\pi_4$.
return $(\pi_f, \pi_\alpha, \pi_4, \mathsf{PoK}_f(params, f_4, X^*, W))$
```

4.1 Encryption Schemes

We review (Lifted) ElGamal encryption in Fig. 4.1a. We include an extra generator (h) for lifting to exponents in ElGamal so that we can draw parallels between ElGamal and Camenisch-Shoup (ElGamal encryption generally uses the default generator, $h = g$). We also slightly modify Camenisch-Shoup encryption in Fig. 4.1b, replacing some values (parameter $g \in \mathbb{Z}_n$ and ciphertext c) with their absolute values.

Modifying Camenisch-Shoup ensures that the elements of honest Camenisch-Shoup ciphertexts lie in a "commitment-friendly" sub-group $|QR_{n^2}|$ that shares more properties with $\mathbb{G}_p$ than $\mathbb{Z}_{n^2}$. This is done by computing the absolute values of elements (i.e. the elements of the public key and ciphertexts). The two commitment schemes are very similar at a high-level and only differ due to limitations with the *eqrep*-$\mathbb{Z}_{n^2}$ protocol (Def. 5 from Sect. 2) which is the protocol we use to prove relations between the ciphertexts in Camenisch-Shoup commitments. Namely, the limitation is that the *eqrep*-$\mathbb{Z}_{n^2}$ protocol only guarantees the absolute values of group elements. The *eqrep*-$\mathbb{G}_p$ protocol which we use for the relations between ciphertexts in ElGamal commitments does not have this limitation and thus is much simpler.

Another modification we've made to the Camenisch-Shoup cryptosystem is that we remove the third element from ciphertexts. Camenisch and Shoup [CS03] construct their scheme with a third element to prove CCA security. We've removed the third element from these ciphertexts as we do not need CCA security for our scheme. Since we don't need the third element to correctly decrypt *honest* ciphertexts, we can simply drop the element and attain CPA security.

The description of $|QR_{n^2}|$. The group $|QR_{n^2}|$ uses an absolute value function shown in Equation 1:

$$|x| = \begin{cases} n^2 - x & x > \lfloor n^2/2 \rfloor \\ x & \text{otherwise} \end{cases} \tag{1}$$

We define $|QR_{n^2}|$ as the group of absolute value of elements in QR_{n^2}, i.e.: $|QR_{n^2}| = \{|x| : x \in QR_{n^2}\}$. A fact that will prove useful is that g and h (in the public parameters in 4.1b) are both in the group $|QR_{n^2}| = \{|x| : x \in QR_{n^2}\}$. We see that g is in $|QR_{n^2}|$ because it is equal to $|(g')^{2n}|$. Squaring $g' \in \mathbb{Z}_{n^2}$ ensures that the result is in QR_{n^2} and taking the absolute value of an element in QR_{n^2} ensures the result is in $|QR_{n^2}|$. We prove that h is $\in QR_{n^2}$ in the proof of Lemma 6 and $h \in |QR_{n^2}|$ follows from the fact that $|1 + n| = 1 + n$. We also see that $|QR_{n^2}|$ comprises $1/4$ of $\mathbb{Z}^*_{n^2}$ in Lemma 12 and the fact that $|QR_{n^2}|$ is isomorphic to QR_{n^2} (proved in Appendix 4.3). From Lemma 6 in Appx. D.2 and the fact that g is in QR_{n^2}, we can see that both elements of our modified Camenisch-Shoup ciphertexts are in $|QR_{n^2}|$. Additionally, $|QR_{n^2}|$ is efficiently sampleable by sampling a random element of $\mathbb{Z}_{n^2}$, squaring it, and taking its absolute value. Unfortunately, $|QR_{n^2}|$ is not efficienctly recognizable. Thus, we need to ensure that honest users in our scheme only commit to encryptions that have an associated proof of correct encryption. Verifying this proof ensures that the encryption algorithm was run correctly and thus the resulting ciphertext lives in $|QR_{n^2}|$. Ensuring that Camenisch-Shoup ciphertexts are in $|QR_{n^2}|$ is useful because $|QR_{n^2}|$ is cyclic (which helps with our hiding and ZK proofs) and also $(-1)x = x$ for elements in $|QR_{n^2}|$. This is important because it means that using *eqrep*-$\mathbb{Z}_{n^2}$ (as defined in Sect. 2) to prove relations between $|QR_{n^2}|$ elements works perfectly, where-as for $\mathbb{Z}_{n^2}$ it only holds for the absolute values of these elements. As an example, if we wanted to prove that we know a such that $c = g^a$ in $\mathbb{Z}^*_{n^2}$, we could only prove that $c = bg^a$ where $b \in \{-1, 1\}$. Intuitively, what we really want is to ensure that after performing exponentiation and multiplication proofs over commitments to ciphertexts, the ciphertext decrypts to the correct value. We can see in Fig. 4.1b that the encryption scheme decrypts the absolute value of a ciphertext exactly the same as the original ciphertext. This is clear from rewriting the decryption process as $m = (((c_1^2/(c_0^2)^x)^t \mod n^2) - 1)/n$. The first operation the decryptor does is square both elements of the ciphertext, and our claim follows from the fact that $|x|^2 = x^2 \in \mathbb{Z}_{n^2}$. Thus, if we can create commitments to elements of $|QR_{n^2}|$, we can use them to commit to our modified Camenisch-Shoup ciphertexts and construct the associated protocols for multiplication and exponentiation.

Drawing more parallels, we see that both ElGamal and Camenisch-Shoup have similar homomorphic properties. Specifically for two encryptions, $(g^r, k^r h^m)$ and $(g^{r'}, k^{r'} h^{m'})$, $(g^r \cdot g^{r'}, k^r h^m \cdot k^{r'} h^{m'})$ is a valid encryption of $\mathfrak{g}(m + m')$ in both encryption schemes. Also, exponentiation is similar, i.e. $((g^r)^y, (k^r h^m)^y)$ is a valid encryption of $\mathfrak{g}(ym)$ in both encryption schemes. Thus, if we can commit to elements of $\mathbb{G}_p$ and $|QR_{n^2}|$ and provide generic protocols for proving the multiplication and exponentiation of committed group elements, we can easily construct commitments to ciphertexts for ElGamal and Camenisch-Shoup along with associated protocols. We use this insight to construct commitments to ElGamal ciphertexts in Sec. 4.2 and commitments to ciphertexts in Camenisch-Shoup ciphertexts in Sec. 4.3.

We quickly prove useful properties about our modified Camenisch-Shoup encryption scheme below:

Correctness of simplified Camenisch-Shoup in Fig. 4.1a. Since the third element is only used in [CS03] for CCA security, our decryption algorithm works for honest encryptions. This is because $h^m = (1+n)^m = \sum_{i=0}^{m} \binom{m}{i} 1^{m-i} n^i = 1 + mn + (m-1)n^2 + ... = 1 + mn \mod n^2$ and y^r can be cancelled out with u^x. We can see that taking the absolute value of ciphertexts does not affect this correctness because part of the decryption squares the ciphertexts. Because $c^2 = (|c|)^2$, after squaring the ciphertexts our decryption algorithm works correctly.

CPA security of simplified Camenisch-Shoup in Fig. 4.1a. Assume we have an adversary that can defeat the CPA security of this scheme. We can then construct a reduction to CCA security of [CS03] by having the reduction simply pass through encryption queries to the CCA challenger and strip the third element from encryptions when returning them to the adversary. Our reduction also takes the absolute value of ciphertexts when passing them to the assumed adversary. These modified encryptions look exactly like encryptions for our modified

Fig. 4.1: Encryption schemes

```
$\mathsf{Setup}(1^\lambda) \to \mathit{params}$
    Generate cyclic group of
      prime order $p$, $\mathbb{G}_p$
    $g, h \leftarrow\$ \mathbb{G}_p$
    $\textbf{return } g, h, \mathbb{G}_p$
$\mathsf{KeyGen}(\mathit{params}) \to (\mathsf{pk}, \mathsf{sk})$
    $x \leftarrow\$ \mathbb{Z}_p;$
    $\textbf{return } \mathsf{pk} \leftarrow g^x, \mathsf{sk} \leftarrow x;$
$\mathsf{Enc}(\mathsf{pk} = k, m) \to c$
    $r \leftarrow\$ \mathbb{Z}_p;$
    $\textbf{return } c = (g^r, k^r h^m)$
$\mathsf{Dec}(\mathsf{sk}, c = (c_0, c_1)) \to M$
    $z = c_0^{\mathsf{sk}} = k^r$
    $\textbf{return } c_1/z = M = h^m$
```

(a) Lifted ElGamal

```
$\mathsf{Setup}(1^\lambda) \to \mathit{params}$
    Sample a safe RSA modulus,
      $n = pq = (2p' + 1)(2q' + 1)$
    $g' \leftarrow\$ |QR_{n^2}|, g = |(g')^n|, h = (1 + n),$
    $\textbf{return } \mathit{params} = (n, g, h)$
$\mathsf{KeyGen}(\mathit{params}) \to (\mathsf{pk}, \mathsf{sk})$
    $\mathsf{sk} = x \leftarrow\$ [n^2/4], \mathsf{pk} = k = |g^x|$    // in $|QR_{n^2}|$
    $\textbf{return } \mathsf{pk}, \mathsf{sk}$
$\mathsf{Enc}(\mathsf{pk}, m \in [n]) \to c$
    $r \leftarrow\$ [n/4],$
    $\textbf{return } c = (|g^r|, |k^r h^m|)$    // in $|QR_{n^2}|$
$\mathsf{Dec}(\mathsf{sk}, c = (c_0, c_1)) \to m$
    $t = 2^{-1} \mod n$
    $M = c_1/c_0^x$    // in $\mathbb{Z}_{n^2}$
    $\textbf{return } m = ((M^{2t} \mod n^2) - 1)/n$
```

(b) Simplified Camenisch-Shoup

scheme. Since the CPA adversary never issues decryption requests, our reduction does not need to decrypt any ciphertexts for the original scheme. Thus, our reduction's probability of success is the same as this adversary's.

4.2 Commitments to $\mathbb{G}_p$ Elements and ElGamal Ciphertexts

In this section, we introduce commitments to group elements (in $\mathbb{G}_p$) and then construct a commitment scheme to ElGamal ciphertext in Fig. 4.3 which relies on those commitments to group elements. Note that the generators g and h used in this section are distinct from those used in the encryption schemes in Sec. 4.1. In this section, g and h refer to commitment bases for a Pedersen commitment.

Commitments to $\mathbb{G}_p$ group elements. In Alg. 4.2 we present a commitment scheme for committing to group elements. Our parameters for the scheme are the same as a Pedersen commitment, yielding g and h. We then commit to a group element by computing $C_1 = Mg^s$ and $C_2 = g^s h^r$. We can see that C_2 is a Pedersen commitment and that s is hidden by C_2. Thus, for any $M, C_1, C_2 \in \mathbb{G}_p$, there exists an s, r that forms a valid opening. We can see that using the opening information, the group element can be retrieved by computing $M = C_1/g^s$.

Proof of opening of an committed group element. We can create a ZK proof of knowledge of an opening of the commitment $C = (C_1, C_2) = \mathsf{Com}_{\mathbb{G}_p}(M)$ by proving knowledge of an opening for C_2 as a Pedersen commitment, i.e. it is the proof of knowledge of representation of C_2 in bases g and h.

Proof of equality of committed group elements. Proving that two group commitments $C = (C_1, C_2) = (Mg^s, g^s h^r)$ and $C' = (C'_1, C'_2) = (M'g^{s'}, g^{s'}h^{r'})$ are committed to the same value ($M = M'$) reduces to a proof of knowledge of equality of representations: $\mathsf{NIZK}[M, M', s, r, s', r' : C_1/C'_1 = g^{s-s'} \wedge C_2/C'_2 = g^{s-s'}h^{r-r'}]$. We can see that this proof works because $C_1/C'_1 = M'g^s/(M'g^{s'}) = g^{s-s'}$ and $C_2/C'_2 = g^s h^r/(g^{s'}h^{r'}) = g^{s-s'}h^{r-r'}$. If the second commitment were committed to a distinct value, then C_1/C'_1 would equal $Mg^s/(Mg^{s'}) = (M/M')g^{s-s'}$ which the adversary could not prove was equivalent to $g^{s-s'}$.

Proof of multiplication of committed group elements. We can also prove that a commitment $C_c = (C_{c,1}, C_{c,2}) = (cg^{s_c}, g^{s_c}h^{r_c})$ opens to the product c of two group elements a, b committed to by two other group element commitments, $C_a = (C_{a,1}, C_{a,2}) = (ag^{s_a}, g^{s_a}h^{r_a})$ and $C_b = (C_{b,1}, C_{b,2}) = (bg^{s_b}, g^{s_b}h^{r_b})$ using *eqrep*-$\mathbb{G}_p$. This can be done by having the verifier and prover compute $D_1 = C_{c,1}/(C_{a,1}C_{b,1}) = cg^{s_c}/(bg^{s_b}ag^{s_a})$ and $D_2 = C_{c,2}/(C_{a,2}C_{b,2}) = g^{s_c}h^{r_a c}/(g^{s_a}h^{r_a}g^{s_b}h^{r_b})$. We can see that if the relation is true, c will be cancelled out by ab in D_1, leading to D_1 being simply the result of an exponentiation of g (we'll label this exponent $\beta_1 = s_c - s_a - s_b$). Further, we see that if the relation is true, D_2 is a Pedersen commitment to β_1. The prover then proves the relation: $\mathsf{PoK}_{\textit{eqrep-}\mathbb{G}_p}[s_a, s_b, s_c, r_a, r_b, r_c, \beta_1, \beta_2 : D_1 = g^{\beta_1} \wedge D_2 = g^{\beta_1}h^{\beta_2}]$ where $\beta_1 = s_c - s_a - s_b$ and $\beta_2 = r_c - r_a - r_b$. We can see that if D_1 can be represented as g^{β_1} and D_2 can be represented as a Pedersen commitment to β_1, we know that C_c is a commitment to ab.

Proof of exponentiation of committed group elements. We can also prove the exponentiation of a $\mathbb{G}_p$ commitment using a scalar in a Pedersen commitment. This can be done by using the *eqrep*-$\mathbb{G}_p$ relation described in Sec. 2. An exponentiation proof takes group element commitments C_a to $\mathbb{G}_p$ element, a, and C_b to element b. It also takes in a Pedersen commitment C_y to y. The goal of this proof is to prove that $a = b^y$. To do this, we prove that $\mathsf{PoK}_{\textit{eqrep-}\mathbb{G}_p}[y, r_y, \beta_1, \beta_2 : C_y = g^y h^{r_y} \wedge C_{a,1} = C_{b,1}^y g^{\beta_1} \wedge C_{a,2} = C_{b,2}^y g^{\beta_1}h^{\beta_2}]$ where $\beta_1 = s_a - ys_b$ and $\beta_2 = r_a - yr_b$ and where $C_y = g^y h^{r_y}$, $C_{a,1} = ag^{s_a}$, $C_{b,1} = bg^{s_b}$, $C_{b,2} = g^{s_b}h^{r_b}$, and $C_{a,2} = g^{s_a}h^{r_a}$.

Another notable feature of this commitment scheme is that the commitments are homomorphic, i.e. if $C = \mathsf{Com}_{\mathbb{G}_p}(M; (s, r))$ and $C' = \mathsf{Com}_{\mathbb{G}_p}(M'; (s', r'))$, then $C \cdot C' = \mathsf{Com}_{\mathbb{G}_p}(MM'; (s + s', r + r'))$.

Theorem 5. *Our construction in Fig. 4.2 is binding.*

Proof of Thm. 5 If a PPT adversary can produce (C, M, M', s, s', r, r') such that $C_1 = Mg^s = M'g^{s'}$ and $C_2 = g^s h^r = g^{s'}h^{r'}$ where $M \neq M'$, we can double open C_2 as a Pedersen commitment. We see that if $M \neq M'$, then $s \neq s'$ because otherwise $M = C_1/g^s = C_1/g^{s'} = M'$. Thus, $s \neq s'$ and s, r, s', r' is a valid double opening for C_2 as a Pedersen commitment. The binding property of

Fig. 4.2: Commitments to $\mathbb{G}_p$ elements

```
$\mathsf{Setup}_{\mathbb{G}_p}(1^\lambda) \rightarrow \mathit{params}$
    Generate a group of prime order $p, \mathbb{G}_p = \langle g \rangle$.
        (or using an existing group e.g. from a bilinear pairing)
    Generate a random element $h \in \mathbb{G}_p$ as the base for opening.
    return $\mathit{params} = (\mathbb{G}_p, g, h)$

$\mathsf{Commit}_{\mathbb{G}_p}(\mathit{params}, M \in \mathbb{G}_p) \rightarrow C, O$
    $s \leftarrow\!\!\$\ \mathbb{Z}_p; r \leftarrow\!\!\$\ \mathbb{Z}_p$
    $C \leftarrow (C_1, C_2) = (Mg^s, g^s h^r)$
    return $C, O = (s, r)$
```

Pedersen commitments relies on the computational Diffie-Hellman assumption and so our $\mathbb{G}_p$ commitments are computationally binding.

Theorem 6. *Our construction in Fig. 4.2 is hiding.*

Proof of Thm. 6 For any $M, C_1, C_2 \in \mathbb{G}_p$, we see that $\exists s, r$ such that $C_1 = Mg^s, C_2 = g^s h^r$. This is because g is a generator for $\mathbb{G}_p$ and thus $\exists$ s such that $g^s = C_1/M$. Because C_2 is a Pedersen commitment which is perfectly hiding, there exists an r such that $C_2 = g^s h^r$ for our picked s. Finally, because s is chosen randomly from $\mathbb{Z}_p$, we see that any M is equally likely given C and thus this commitment scheme is perfectly hiding.

So far, we've constructed commitments to elements of $\mathbb{G}_p$ and discussed their associated proof protocols for opening and multiplication. Next we'll use these commitments and the intuition about their protocols to build commitments to ElGamal ciphertexts. We build these commitments to ElGamal ciphertexts in Fig. 4.3. Verifying these proofs is a direct application of the *eqrep*-$\mathbb{G}_p$ verification protocol. We put square brackets $[\cdot]$ around secret values for proof functions. We can see in this ElGamal commitment scheme that we set it up by generating Pedersen commitment bases, g, h, while labeling the parameters for the ElGamal encryption scheme as g' and h'. To commit, we form a $\mathbb{G}_p$ commitment to each the two elements of an ElGamal ciphertext, $c = (c_1, c_2)$, yielding C_1, C_2 as a commitment to c_1 and C_3, C_2 as a commitment to c_2. Because our $\mathbb{G}_p$ commitments are perfect hiding and computationally binding to elements of $\mathbb{G}_p$, our ElGamal commitments are perfectly hiding and computationally binding as well.

Proofs over commitments to ciphertexts. Inspecting our construction, we see that many of our proofs ($\mathsf{Prove}^{\mathsf{Com}}_{ElG}, \mathsf{Prove}^{\mathsf{add}}_{ElG}, \mathsf{Prove}^{\mathsf{mult}}_{ElG}$) consists of simply performing the proof on both group elements. For example, to prove knowledge of an opening of an ElGamal commitment, we open the Pedersen commitments of each $\mathbb{G}_p$

commitment, C_2 and C_4. This allows an extractor to recover s_1, s_2, r_1, r_2 allowing the extractor to compute $c_1 = C_1/g^{s_1}$ and $c_2 = C_3/g^{s_2}$. This is how we described opening those $\mathbb{G}_p$ commitments earlier in this section. As another example, we see in $\mathsf{Prove}^{\mathsf{add}}_{ElG}$ that we want to prove that C_c is committed to ciphertext c where $c = ab$ and C_b is committed to ciphertext b and C_a is committed to ciphertext a. We label this add "addition" because multiplying two ciphertexts results in the addition of their encrypted messages. Intuitively, $\mathsf{Prove}^{\mathsf{mult}}_{ElG}$ requires the verifier to use the homomorphic properties of the commitment scheme to multiply two group elements and then requires the prover to prove that the resulting commitment is equivalent to C_a. We can see in this algorithm that $D_1 = C_{c,1}/(C_{a,1}C_{b,1})$ will be a power of g if (and only if) $c = ab$ because $D_1 = cg^{s_c}/(ag^{s_a}bg^{s_b}) = cg^{s_c - s_a - s_b}/(ab)$. The same is true for D_3 and D_4.

Proving a ciphertext is an encryption of a Pedersen committed message. Proving that a committed ciphertext is an encryption of a Pedersen committed message somewhat breaks our ciphertext commitment scheme's paradigm of simply performing proofs on either element in the ciphertext. In this proof, $\mathsf{Prove}^{\mathsf{enc}}_{ElG}$, the prover must prove that the commitment is correctly formed for the message y (whereas in the other proofs, we assume the ciphertexts are correctly formed and proofs can be created without knowledge of the randomness of ciphertexts). Thus, we prove that $c_1 = (g')^{\rho_c}$ and $c_2 = k^{\rho_c}(h')^y$ where g' and h' are the generators for the encryption scheme (in the case of ElGamal, $g' = h'$ but in Sec. 4.3 we'll see that these may differ). We can see that verifying π ensures that the prover knows c (along with its randomness and message) such that is correct ElGamal encryption of y with randomness ρ_c and C_y is a scalar commitment to y.

Theorem 7 (Hiding of the commitments in Fig. 4.3). *Our commitments to ElGamal ciphertexts in Fig. 4.3 are statistically hiding.*

Proof (Proof of Thm. 7). We can see that (C_1, C_2) is identical to a $\mathbb{G}_p$ commitment to c_1 and (C_3, C_4) is identical to a $\mathbb{G}_p$ commitment to c_2, we can see that they statistically hide c_1 and c_2.

Theorem 8 (Binding of the commitments in Fig. 4.3). *Our commitments to ElGamal ciphertexts in Fig. 4.3 are computationally binding.*

Proof (Proof of Thm. 8). We can see that (C_1, C_2) is identical to a $\mathbb{G}_p$ commitment to c_1 and (C_3, C_4) is identical to a $\mathbb{G}_p$ commitment to c_2, thus, if a PPT adversary can produce a double opening such that one of these commitments opens to some c_1' or c_2' in $\mathbb{G}_p$, we obtain a double opening for our $\mathbb{G}_p$ commitments.

Theorem 9 (Zero-knowledge of Fig. 4.3). *Our protocols in Fig. 4.3 ($\mathsf{Prove}^{\mathsf{Com}}_{ElG}$, $\mathsf{Prove}^{\mathsf{enc}}_{ElG}$, $\mathsf{Prove}^{\mathsf{mult}}_{ElG}$, and $\mathsf{Prove}^{\mathsf{add}}_{ElG}$) are zero-knowledge against any PPT adversary.*

Proof (Proof of Thm. 9). We can see that in each of these NIZKs, we simply return a proof computed from the *eqrep*$-p^*$ protocol. Thus, we can use the simulator for this protocol to produce proofs in the zero knowledge games. Thus, if a PPT adversary can distinguish these simulated proofs from real proofs, we can break the zero knowledge of the *eqrep*$-p^*$ protocol.

Fig. 4.3: Commitments to ElGamal ciphertexts

```
$\mathsf{Setup}_{ElG}(1^\lambda, \mathit{params}_{ElG}) \to \mathit{params}$

    parse $\mathit{params}_{ElG} = (\mathbb{G}_p, g', h')$
    $(g, h) \leftarrow\$ \mathbb{G}_p$
    $\mathit{params} = (g, h, \mathit{params}_{ElG})$
    return $\mathit{params}$
```

```
$\mathsf{Commit}_{ElG}(\mathit{params}, c = (c_1, c_2)) \to C, O$

    $s_1, s_2 \leftarrow\$ \mathbb{Z}_p; r_1, r_2 \leftarrow\$ \mathbb{Z}_p$
    $C \leftarrow (C_1, C_2, C_3, C_4)$
        $= (c_1 g^{s_1}, g^{s_1} h^{r_1}, c_2 g^{s_2}, g^{s_2} h^{r_2})$
    return $(C, O = (s_1, s_2, r_1, r_2))$
```

```
$\mathsf{Prove}^{\mathsf{Com}}_{ElG}(\mathit{params}, C, M, O) \to \pi$

    parse $C = (C_1, C_2, C_3, C_4)$,
        $O = (s_1, s_2, r_1, r_2)$
    $\pi = \mathsf{NIZK}_{eqrep}[s_1, s_2, r_1, r_2 :$
        $C_2 = g^{s_1} h^{r_1}, C_4 = g^{s_2} h^{r_2}]$
    return $\pi$
```

```
$\mathsf{Prove}^{\mathsf{enc}}_{ElG}(\mathit{params}, \mathsf{pk} = k, C_c, C_y,$
        $[c, \rho_c, y, O_c, O_y]) \to \pi$

    parse $\mathit{params} = (g, h, \mathit{params}_{ElG})$
        $\mathit{params}_{ElG} = (\mathbb{G}_p, g', h')$
        $O_c = (s_{c,1}, s_{c,2}, r_{c,1}, r_{c,2})$
        $c = ((g')^{\rho_c}, k^{\rho_c}(h')^y)$,
        $O_y = (r_y)$
    $\pi = \mathsf{NIZK}[$
        $s_{c,1}, s_{c,2}, s_y, \rho_c, r_{c,1}, r_{c,2}, r_y, y :$
        $C_y = g^y h^{r_y}$
        $\wedge C_{c,1} = (g')^{\rho_c} g^{s_{c,1}}$
        $\wedge C_{c,2} = g^{s_{c,1}} h^{r_{c,1}}$
        $\wedge C_{c,3} = k^{\rho_c}(h')^y g^{s_{c,2}}$
        $\wedge C_{c,4} = g^{s_{c,2}} h^{r_{c,2}}]$
    return $\pi$
```

```
$\mathsf{Prove}^{\mathsf{mult}}_{ElG}(\mathit{params}, C_a, C_b, C_y,$
        $[c, a, b, y, O_a, O_b, O_y]) \to \pi$

    parse $O_a = (s_{a,1}, s_{a,2}, r_{a,1}, r_{a,2})$
        $O_b = (s_{b,1}, s_{b,2}, r_{b,1}, r_{b,2})$
        $O_y = (r_y)$
    $\beta_1 = s_{a,1} - y s_{b,1}$
    $\beta_2 = r_{a,1} - y r_{b,1}$
    $\beta_3 = s_{a,2} - y s_{b,2}$
    $\beta_4 = r_{a,2} - y r_{b,2}$
    $\pi = \mathsf{NIZK}[y, r_y, \beta_1, \beta_2, \beta_3, \beta_4 :$
        $C_y = g^y g^{r_y}$
        $\wedge C_{a,1} = (C_{b,1})^y g^{\beta_1}$
        $\wedge C_{a,2} = (C_{b,2})^y g^{\beta_1} h^{\beta_2}$
        $\wedge C_{a,3} = (C_{b,3})^y g^{\beta_3}$
        $\wedge C_{a,4} = (C_{b,4})^y g^{\beta_3} h^{\beta_4}]$
    return $\pi$
```

```
$\mathsf{Prove}^{\mathsf{add}}_{ElG}(\mathit{params}, C_a, C_b, C_c,$
        $[a, b, c, O_a, O_b, O_c]) \to \pi$

    parse $O_a = (s_{a,1}, s_{a,2}, r_{a,1}, r_{a,2})$
        $O_b = (s_{b,1}, s_{b,2}, r_{b,1}, r_{b,2})$
        $O_c = (s_{c,1}, s_{c,2}, r_{c,1}, r_{c,2})$
    $D_1 \leftarrow C_{c,1}/(C_{a,1} * C_{b,1})$
    $D_2 \leftarrow C_{c,2}/(C_{a,2} * C_{b,2})$
    $D_3 \leftarrow C_{c,3}/(C_{a,3} * C_{b,3})$
    $D_4 \leftarrow C_{c,4}/(C_{a,4} * C_{b,4})$
    $\beta_1 = s_{c,1} - s_{a,1} - s_{b,1}$
    $\beta_2 = r_{c,1} - r_{a,1} - r_{b,1}$
    $\beta_3 = s_{c,2} - s_{a,2} - s_{b,2}$
    $\beta_4 = r_{c,2} - r_{a,2} - r_{b,2}$
    $\pi = \mathsf{NIZK}[\beta_1, \beta_2, \beta_3, \beta_4 :$
        $D_1 = g^{\beta_1}$
        $\wedge D_2 = g^{\beta_1} h^{\beta_2}$
        $\wedge D_3 = g^{\beta_3}$
        $\wedge D_4 = g^{\beta_3} h^{\beta_4}]$
    return $\pi$
```

Theorem 10 (Black box knowledge extraction of Fig. 4.3). *Given a PPT adversary that can produce a proof that verifies for our protocols in Fig. 4.3 (*$\mathsf{Prove}^{\mathsf{Com}}_{ElG}$*,* $\mathsf{Prove}^{\mathsf{enc}}_{ElG}$*,* $\mathsf{Prove}^{\mathsf{mult}}_{ElG}$*, and* $\mathsf{Prove}^{\mathsf{add}}_{ElG}$*) there exists an extractor with* *black-box access to the adversary that can extract a witness that proves the relations true.*

Proof (Proof of Thm. 10). Similar to our proof of zero-knowledge for these protocols, because these protocols simply return *eqrep*-$\mathbb{G}_p$ proofs, we can use the black-box extractor for these proofs to extract the witnesses. This extractor is described in Sec. 2.

4.3 Commitments to $|QR_{n^2}|$ and Camenisch-Shoup Ciphertexts

To construct commitments to Camenisch-Shoup ciphertexts, we need to construct commitments to the elements of the group in which components of a Camenisch-Shoup ciphertexts lie. To construct efficient commitments, we need to use a group that retains similar algebraic structure to Camenisch-Shoup ciphertexts. We accomplish this by using Damgård-Fujisaki integer commitments [DF02] that are similar to Pedersen commitments for ElGamal. First, we adapt Damgård-Fujisaki commitments to "live" in $\mathbb{Z}_{n^2}$. We then construct commitments to $|QR_{n^2}|$ elements in a similar way to how we constructed commitments to $\mathcal{G}_p$ (i.e. by creating commitments of the form: $C = (Mg^s, g^s h^r)$). Because both elements in our $|QR_{n^2}|$ commitments will belong to $\mathbb{Z}_{n^2}$, this will allow us to use the *eqrep*-$\mathbb{Z}_{n^2}$ protocol defined in Def. 5 to complete proofs of multiplication and exponentiation of our $|QR_{n^2}|$ commitments.[11]

Modifications to Damgård-Fujisaki Damgård and Fujisaki [DF02] construct a commitment scheme to integers which works over any group $\mathcal{G}$ as long as $\mathcal{G}$ is efficiently recognizable and sampleable and has certain properties – mainly, having hidden order. They then prove the group $\mathbb{Z}_n$ satisfies these properties.

We present our modified version of Damgård-Fujisaki commitments which lie in $\mathbb{Z}_{n^2}$ in Fig. 4.4. In this construction, 2^B is roughly the order of $\phi(n^2)$ (where ϕ is Euler's totient function) though 2^B is computable without knowing $\phi(n^2)$ (as defined in [DF02]).

Damgård and Fujisaki [DF02] list four properties sufficient for an Abelian group to create an integer commitment scheme. They then prove that the group $\mathbb{Z}_n$ satisfies these properties. We will prove these properties for the group $\mathbb{Z}_{n^2}$.

The assumptions Damgård and Fujisaki required to prove their integer commitment scheme secure are shown below. They [DF02] provide a construction and prove that if a group meets all four requirements, their construction is secure. We will modify these requirements slightly and prove that $\mathbb{Z}_{n^2}$ satisfies them. In these assumptions, C is some number which is super polynomial in the security parameter, but smaller than the primes, p, q, p', q'.

[11] We could use Damgård-Fujisaki commitments as-is (such that they live in $\mathbb{Z}_n$), but our $|QR_{n^2}|$ commitments would then consist of elements in $\mathbb{Z}_{n^2}$ and $\mathbb{Z}_n$, requiring a new *eqrep* protocol that spans both groups. It is not clear if this alternative approach would be more efficient or simpler.

Fig. 4.4: Simplified Damgård-Fujisaki commitments in $\mathbb{Z}_{n^2}$

```
$\mathsf{Setup}(1^\lambda) \to params$ :
Sample $O(\lambda)$-bit SG primes $p', q'$ and compute $p = 2p'+1, q = 2q'+1, n = pq$.
Sample random $g, h \in \mathbb{Z}_{n^2}$.
return $params = (g, h)$

$\mathsf{Commit}(params, m) \to (C, O)$ :
To commit to integer, $m$, compute: $C = g^m h^r$
    where $r \leftarrow\!\!\$ \, [2^{B+\lambda}]$
Let the opening be $O = r$
return $(C, O)$
```

Damgård-Fujisaki commitment properties:

1. **Strong root property** - Let Adv be any PPT algorithm. After generating the group with security parameter, λ, then, with a description of the group, $\mathcal{G}$, (without the trapdoor) and a random $h \in \mathcal{G}$, Adv is tasked with outputting $y \in \mathcal{G}$ and a number, $t > 1$, such that $y^t = h$. The probability of this occurring is negligible.
2. **Small order property** - Let Adv be an PPT algorithm. With a description of the group, $\mathcal{G}$, Adv is tasked with outputting $b \in \mathcal{G}, \sigma \in \mathbb{Z}$ such that $b \neq 1$, $b^2 \neq 1$, $0 < \sigma < C$, and $b^\sigma = 1$. The probability of this occurring is negligible.
3. **No large even powers in orders** - Any element in $\mathcal{G}$ of the form a^{2t} has odd order.
4. **Many elements with only large prime factors in orders** - If h is chosen randomly in $\mathcal{G}$, then theres is an overwhelming $(1 - O(2^{-\lambda}))$ probability that the order of h has no prime factors less than C.

Damgård and Fujisaki [DF02] prove that $\mathbb{Z}_n$ satisfies these properties where $n = pq$ and $p \equiv q \equiv 3(mod\ 4)$ and p, q are safe primes. The primes, p and q, are not given to the adversary in these assumptions.

We now prove that these properties hold for $\mathbb{Z}_{n^2}$ with n formed the same way as in Damgård-Fujisaki [DF02]. We review the strong RSA assumption (Assumption 1 of [DF02]), and prove a useful lemma (Lemma 3).

Assumption 1 (Strong RSA assumption[DF02]) *Given $n = pq$ (where $|n| = O(2^\lambda)$), and a number, $t \in \mathbb{Z}_n$, no PPT algorithm can find a pair, v, e such that $v^e = t$ and $e > 1$ with non-negligible probability in λ.*

Lemma 3. *If $a = b \mod n^2$, then $a = b \mod n$.*

Proof of Lemma 3 Take values $a, b \neq 0 \in \mathbb{Z}_{n^2}$ such that $a = b \mod n^2$. This implies that $a = mn^2 + d, b = on^2 + d$ for some $m, o \in \mathbb{Z}$ where $0 < d < n^2$. This implies that $a = m'n + d, o'n + d$ where $m' = mn, o' = on$. If we take the remainder of d mod n, as $d = ln + \rho$ for some $l \in \mathbb{Z}$ where $0 < \rho < n$, we find that the following equation holds: $a = (m' + l)n + \rho, (o' + l)n + \rho$. Since division with remainder is unique for $0 \leq \rho < n$, we've shown that a and b are equal mod n.

Proof of DF Property 1 for $\mathbb{Z}_{n^2}$. Assume we have a PPT algorithm that given $t \in \mathbb{Z}_{n^2}$ can produce a $g \in \mathbb{Z}_{n^2}, y$ such that $g^y = t \mod \mathbb{Z}_{n^2}$. We are then tasked with creating a reduction to strong RSA in $\mathbb{Z}_n$. Let our reduction take t in $\mathbb{Z}_n$ and give $t + bn \mod n^2$ to this adversary where b is a random number drawn from 0 to $n-1$. The adversary then provides g, y such that $g^y = (t+bn) \mod n^2$. Since this equality holds in $\mathbb{Z}_{n^2}$, it holds in $\mathbb{Z}_n$ as well due to Lemma 3. We can see that $t + bn = t \mod n$. Thus $g^y = t \mod n$. Lastly, we have to prove that $(t + bn)$ is distributed indistinguishably from a uniform drawing from $\mathbb{Z}_{n^2}$. We can see that $t + bn$ can "reach" almost every element of $\mathbb{Z}_{n^2}$ since if $t = n - 1$ and $b = n-1$, then $t+bn = n-1+(n-1)n = n-1+n^2-n = n^2-1$ and if $t = 1, b = 0$, we get 1. Then, we see that there are no duplicates of $t + bn$ across this range since no $t, b, t', b' \in \{0, ..., m - 1\}$ exist such that $t + bn = t' + b'n$. There are $(n - 1)n$ possible possible combinations of t and b from our ranges. Thus, each value mapped to by $t + bn$ uniformly maps to a random element of $\mathbb{Z}_{n^2}$ except for values of $\mathbb{Z}_{n^2}$ where n is a factor. There are only n samples of $\mathbb{Z}_{n^2}$ that are divisible by n out of a total of n^2 instances and thus the probability of drawing one of these samples is negligible and our assumed strong RSA adversary in $\mathbb{Z}_{n^2}$ must be able to solve problems when the challenge is not a multiple of n with non-negligible probability.

Proof of DF Property 2 for $\mathbb{Z}_{n^2}$. The only possible orders of elements in $\mathbb{Z}_{n^2}$ are $2, 4, p, q, p', q'$ or some product of these. If the adversary outputs a b with $\sigma = 2$, we see that is must be that $b^2 = 1$ and thus this is not a valid solution. If σ is a multiple of p, q, p', or q', then $\sigma > C$ and thus this solution doesn't work for this property. Thus, the only possible values for σ is 4. We can see that, in this case, if b^2 is a non-trivial root of 1 (i.e. $b^2 \neq -1$) we can factor by rewriting $(b - 1)(b + 1) = 0 \mod n^2$ thus ensuring that taking the gcd of $b - 1$ or $b + 1$ with p, q, p', or q' yields a factorization. We see that if $b^4 = 1$ and $b^2 = -1$, this must be true in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ due to the Chinese remainder theorem. We can see that because $p \equiv 3 \mod 4$, it must be that $p = 4k + 3$ and thus $(p - 1)/2$ is odd and so $(-1)^{(p-1)/2} = -1$ implying that (-1) is not a quadratic residue mod p. Thus, if $b^4 = 1$ but $b^2 = -1$, this would be a contradiction and thus b^2 must be a non-trivial square root allowing us to factor.

Proof of DF Property 3 for $\mathbb{Z}_{n^2}$. We see that the order of $\phi(n^2)$ is $2pqp'q'$ and thus, if a^{2t} has even order, then a has order $4k$ but $4 \nmid 2pqp'q'$ and thus does not divide the order of the group and thus we have a contradiction and a^{2t} cannot have even order.

Fig. 4.5: $|QR_{n^2}|$-Commitments

```
$\mathsf{Setup}_{\mathsf{QR}}(1^\lambda) \rightarrow (params)$
Sample a safe RSA modulus,
    $n = pq = (2p' + 1)(2q' + 1)$
Sample random $g \leftarrow |QR_{n^2}|$
Sample random $(g', h') \leftarrow \mathbb{Z}_{n^2}$
return $params = (n, g, g', h')$
```

```
$\mathsf{Com}_{\mathsf{QR}}(params, M) \rightarrow (C, O)$
$(s, r) \leftarrow\!\$\; [2^{B+\lambda}]$
$C = (|Mg^s|, (g')^s(h')^r)$
return $(C, O)$ where $O = (s, r)$
```

Proof of DF Property 4 for $\mathbb{Z}_{n^2}$. If we find a non-trivial square root of 1, we factor and we showed in the proof of DF Property 2 that if we find a 4-th root of 1, it must be that when we square the value, we can factor. Thus, these must be hard to sample, otherwise, it would be trivial to factor. Thus, the only orders of sampleable elements (by a PPT algorithm) must be some product of p, q, p' and q'. We can simply set $C < p, q, p', q'$ and $p, q, p', q' \approx O(2^\lambda)$ to satisfy this.

Commitments to $|QR_{n^2}|$ elements Next, by employing Damgård-Fujisaki commitments, we can construct a scheme for committing to elements of $|QR_{n^2}|$ (and then we can use $|QR_{n^2}|$ commitments to construct commits to Camenisch-Shoup ciphertexts). We show this scheme in Fig. 4.5. In this scheme, B is such that 2^B is larger than the order of $|QR_{n^2}|$ (i.e. $2^B = n^2/4$). We show that such commitments are hiding and binding in Appx. 4.3. We can see that these $|QR_{n^2}|$ commitments are multiplicatively homomorphic, i.e. if you take two $|QR_{n^2}|$ commitments $c = (c_1, c_2)$ committing to element M and $d = (d_1, d_2)$ committing to element N, then if you compute their pair-wise multiplication: $e = (c_1 * d_1, c_2 * d_2)$, this results in a commitment to $M * N$ with opening information $s_c + s_d, r_c + r_d$, computed pair-wise, where s_c, r_c is the opening information for c and s_d, r_d is the opening information for d.

Proofs of hiding and binding for $|QR_{n^2}|$-commitments in Fig. 4.5 We provide number theory background in Appx. D.2.

Hiding proof for Fig. 4.5. To prove that our commitments are hiding, we show that, for any group element M, the commmitment algorithm (which samples a commitment $C = (|Mg^s|, g^s h^r)$) provides a distribution that is statistically close to the distribution $(R_1, R_2) \in (|QR_{n^2}| \times \mathbb{Z}_{n^2})$ drawn uniformly at random.

We can see that since $n = pq$ where p, q are safe primes, then g with overwhelming probability generates QR_{n^2} due to Lemma 8. If s is large, g^s is indistinguishable from a random element of QR_{n^2} since s is much larger than $ord(g)$ (Lemma 9). Let $\star$ be the "multiply-and-absolute-value" operation in that it takes two elements, multiplies them and then takes the absolute value. We see that $|QR_{n^2}|$ is a group under this operator as $x \star y = |x * y| = |x| * |y|$, $1 = |1|$, $(|QR_{n^2}|, \star)$ is closed since QR_{n^2} is closed and $|\cdot|$ maps QR_{n^2} to $|QR_{n^2}|$, and the inverse of any $x \in (|QR_{n^2}|, \star)$ is $|x^{-1}|$ where $x^{-1} \in QR_{n^2}$ ($|x * x^{-1}| = |1|$). Let $|\cdot| : QR_{n^2} \to |QR_{n^2}|$ be the map defined by the absolute value function. We see that $|\cdot|$ is a homomorphism as $|x * y| = |x| * |y|$ and $|1| = |1|$. We can see that $|x|$ is bijective as the only values of $\mathbb{Z}_{n^2}$ that map to the same value have the form $-x$ and x, but if $x \in QR_{n^2}$, then $-x \notin QR_{n^2}$ since (-1) is not a quadratic residue (Lemma 7). We also defined $|QR_{n^2}|$ as the image of this function and thus because it is also injective, it is bijective. Thus, $(QR_{n^2}, *) \cong (|QR_{n^2}|, \star)$. This means that $|QR_{n^2}|$ is cyclic and any randomly sampled element of $|QR_{n^2}|$ is likely a generator due to Lemma 8. Thus, $M \star g^s$ is indistinguishable from a random element of $|QR_{n^2}|$.

We note that C_2 is simply a Damgård-Fujisaki integer commitment and thus is indistinguishable from a random element in $\mathbb{Z}_{n^2}$.

Binding proof for $|QR_{n^2}|$-commitments in Fig. 4.5. If a PPT adversary can open a commitment $C = (C_1, C_2)$ to two values $M, M' \in \{|x| : x \in |QR_{n^2}|\}$ (providing openings, s, s', r, r') such that $M \neq M'$, we see that it must be that $C_1/g^s \neq C_1/g^{s'}$. If $s \neq s'$, we see that $C_2, (s, r), (s', r')$ is a double opening for the Damgård-Fujisaki integer commitment scheme. Because we proved that these Damgård-Fujisaki commitments are binding for $\mathbb{Z}_{n^2}$ (In Appendix 4.3), this double opening violation still holds even if C_1 and C_2 are created maliciously (i.e., they are not in QR_{n^2}, but instead some arbitrary element of $\mathbb{Z}_{n^2}$). Thus, $s = s'$ and it must be that $|C_1/g^s| = |C_1/g^{s'}|$. This tells us that $|M| = |M'| \in \mathbb{Z}_{n^2}$ and since $|M| = M \forall M \in |QR_{n^2}|$, we see that $M = M' \in |QR_{n^2}|$. Thus it is impossible (based on the strong RSA assumption) for a PPT adversary to double open our $|QR_{n^2}|$ commitments without double opening a Damgård-Fujisaki commitment.

Auxiliary proofs for commitments to $|QR_{n^2}|$ We now describe protocols that we can use to create proofs of opening, multiplication, and exponentiation of elements in $|QR_{n^2}|$ which can be verified using only their commitments.

Proof of knowledge of opening for $|QR_{n^2}|$-commitments. We can see that the second part of a $|QR_{n^2}|$ commitment is simply an integer commitment from Damgård-Fujisaki [DF02] which we described previously in this section. Using their opening protocol to create a proof of opening of the second part of the commitment suffices as a proof of opening for a $|QR_{n^2}|$ commitment as we can extract s, r from C_2 and compute: $M = |C_1/(g^s)|$.

Proof of multiplication of $|QR_{n^2}|$-commitments. We show how to prove knowledge of multiplication of committed $|QR_{n^2}|$ elements by utilizing the homomorphic property of the commitments. Given three commitments, C_1, C_2, C_3, committing to $|QR_{n^2}|$ elements E_1, E_2, E_3 (where each commitment consists of two elements of $|QR_{n^2}|$, $C_i = (C_{i,1}, C_{i,2})$), we prove that a forth commitment $C_4 = (C_{4,1}, C_{4,2})$ is a commitment to 1, where $C_{4,1} = C_{1,1}/(C_{2,1}C_{3,1})$ and $C_{4,2} = C_{1,2}/(C_{2,2}C_{3,2})$ – the verifier can compute C_4 using $(C_i)_{i\in[3]}$. This is equivalent to proving multiplication because of the homomorphic properties of the relation and can be proven using *eqrep*-$\mathbb{Z}_{n^2}$ from Sect. 2 using relation $R((\gamma_1, \gamma_2, \beta_1, \beta_2), (C_{4,1}, C_{4,2})) = 1$ iff $C_{4,1} = \beta_1 g^{\gamma_1} \wedge C_{4,2} = \beta_2 (g')^{\gamma_1} (h')^{\gamma_2} \wedge \beta_1 \in \{-1, 1\} \wedge \beta_2 \in \{-1, 1\}$. This proves that $|C_4| = (|g^{\gamma_1}|, (g')^{\gamma_1} (h')^{\gamma_2})$ which is a commitment to 1. The prover uses $\gamma_1 = s_1 - s_2 - s_3$ and $\gamma_2 = r_1 - r_2 - r_2$ to satisfy this relation, where (s_i, r_i) is the opening of C_i.

Proof of exponentiation of $|QR_{n^2}|$-commitments with Damgård-Fujisaki commitments. We prove this with *eqrep*-$\mathbb{Z}_{n^2}$ from Sec. 2. This proof operates over two commitments C_1, C_2 to $|QR_{n^2}|$ elements E_1, E_2 and one commitment C_y to scalar y and proves that $E_1 = E_2^y$. First let $C_1 = (C_{1,1}, C_{1,2})$, $C_2 = (C_{2,1}, C_{2,2})$ and $C_y = (g')^y (h')^{r_y}$. This can be proven with relation $R((\gamma_1, \gamma_2, \beta_1, \beta_2), (C_1, C_2, C_y)) = 1$ iff $C_{1,1} = \beta_1 C_{2,1}^y g^{\gamma_1} \wedge C_{1,2} = \beta_2 C_{2,2}^y g^{\gamma_1} h^{\gamma_2} \wedge \beta_1 \in \{-1, 1\} \wedge \beta_2 \in \{-1, 1\}$. The Prover uses $\gamma_1 = s_1 - y s_2$ and $\gamma_2 = r_1 - y r_2$ to satisfy this relation. If the prover can open C_2 then, $C_{1,1} = \beta_1 E_2^y g^{y s_2 + \gamma_1}$ and $C_{1,2} = \beta_2 (g')^{y s_2 + \gamma_1} (h')^{y r_2 + \gamma_2}$ which is exactly a commitment to $|E_2^y|$.

Remark 1 (Reducing the size of scalars.). Our protocols for commitments must have a maximum size of the witnesses (the committed values). We label this as T. This bound ensures that our protocols remain zero knowledge. For our Camenisch-Shoup scheme, this will need to be $T = \mathbb{Z}_n$ since $\mathbb{Z}_n$ is our message space for these ciphertexts. We run into a problem with $|QR_{n^2}|$ commitments that we didn't have with $\mathbb{G}_p$ commitments here because the scalar commitments we use (Damgård-Fujisaki commitments) do not directly commit to the message space of Camenisch-Shoup commitments. Thus, in order to keep exponents small after an exponentiation proof, we'll also include a proof of modular arithmetic over n in our exponentiation proof. This ensures that the values needed in the proofs never grow large enough to violate our zero knowledge property. This proof of modular arithmetic works by computing a commitment to n and then proving that a remainder of n in a commitment is equal to the original commitment summed with a multiple of n. This ensures that honest provers can reduce the size of the commitments while still proving equivalence modulo n. As an example, let a prover have two $|QR_{n^2}|$ commitments and one scalar commitment, $C_M = (|M g^{s_M} a_M|, (g')^{s_M} (h')^{r_M})$, $C_N = (|N g^{s_N} a_N|, (g')^{s_N} (h')^{r_N})$, $C_y = (g')^y (h')^r$. To prove that $|N| = |M^{y \mod n}|$, the prover will construct $|QR_{n^2}|$ commitment $C_P = (|P g^{s_P} a_P|, (g')^{s_P} (h')^{r_P})$ where $|P| = |M^y|$ and $C_Q = (|Q g^{s_Q} a_Q|, (g')^{s_Q} (h')^{r_Q})$ where $|Q| = |M^n|$. They will then prove that $|N| = |M^{y \mod n} * (M^n)^k|$ where $k = y - (y \mod n)$. This can be done generically using *eqrep*$-n^*$ described in Sec. 2. Notice that a prover could select an incorrect k value in this proof. This is not a problem because larger scalars only affects zero knowledge and not soundness. Thus any honestly created commitments and proofs will remain zero knowledge and any malicious proofs will remain sound.

Commitments to Camenisch-Shoup encryptions Since we constructed commitments to elements of $|QR_{n^2}|$ along with their associated proof protocols, we can use these commitments with Camenisch-Shoup ciphertexts. We present the full construction in Fig. 4.6

Fig. 4.6: Commitments to Camenisch-Shoup ciphertexts

```
$\mathsf{Setup}_{CS}(1^\lambda, params_{CS}, params_{DF}) \rightarrow params$

$\textbf{parse } params_{CS} = (\mathbb{Z}_{n^2}, g^*, h^*)$
$\textbf{parse } params_{DF} = (\mathbb{Z}_{n^2}, g', h')$
$g \leftarrow\!\!\$ \; |QR_{n^2}|$
$params = (\mathcal{G}, g, g^*, h^*, g', h')$
$\textbf{return } params$

$\mathsf{Commit}_{CS}(params, c) \rightarrow C, O$

$\textbf{parse } c = (c_1, c_2)$
$s_1, s_2 \leftarrow\!\!\$ \; [2^{B+\lambda}]; r_1, r_2 \leftarrow\!\!\$ \; [2^{B+\lambda}]$
$a_1, a_2, b_1, b_2 \leftarrow\!\!\$ \; \{-1, 1\}$
$C_1 \leftarrow a_1 c_1 g^{s_1}; C_2 \leftarrow b_1 (g')^{s_1} (h')^{r_1}$
$C_3 \leftarrow a_2 c_2 g^{s_2}; C_4 \leftarrow b_2 (g')^{s_2} (h')^{r_2}$
$C \leftarrow (C_1, C_2, C_3, C_4)$
$O \leftarrow (a_1, a_2, s_1, s_2, r_1, r_2, b_1, b_2)$
$\textbf{return } (C, O)$

$\mathsf{Prove}^{\mathsf{add}}_{CS}(params, C_a, C_b, C_c, [a, b, c, O_a, O_b, O_c]) \rightarrow \pi$

$\textbf{parse } C_a = (C_{a,i})_{i \in [4]}$
  $C_b = (C_{b,i})_{i \in [4]}$
  $C_c = (C_{c,i})_{i \in [4]}$
  $O_a = (a_{a,i}, s_{a,i}, r_{a,i}, b_{a,i})_{i \in [2]}$
  $O_b = (b_{b,i}, s_{b,i}, r_{b,i}, b_{b,i})_{i \in [2]}$
  $O_c = (b_{c,i}, s_{c,i}, r_{c,i}, b_{c,i})_{i \in [2]}$
$\forall i \in [4], D_i \leftarrow C_{c,i} / (C_{a,i} * C_{b,i})$
$\gamma_1 \leftarrow s_{c,1} - s_{a,1} - s_{b,1}$
$\gamma_2 \leftarrow r_{c,1} - r_{a,1} - r_{b,1}$
$\gamma_3 \leftarrow s_{c,2} - s_{a,2} - s_{b,2}$
$\gamma_4 \leftarrow r_{c,2} - r_{a,2} - r_{b,2}$
$\beta_1 \leftarrow a_{c,1} / (a_{a,1} * a_{b,1})$
$\beta_2 \leftarrow b_{c,1} / (b_{a,1} * b_{b,1})$
$\beta_3 \leftarrow a_{c,2} / (a_{a,2} * a_{b,2})$
$\beta_4 \leftarrow b_{c,2} / (b_{a,2} * b_{b,2})$
$\pi = \mathsf{NIZK}[\{\gamma_i, \beta_i\}_{i \in [4]} :$
  $D_1 = \beta_1 g^{\gamma_1}$
  $\wedge D_2 = \beta_2 (g')^{\gamma_1} (h')^{\gamma_2}$
  $\wedge D_3 = \beta_3 g^{\gamma_3}$
  $\wedge D_4 = \beta_4 (g')^{\gamma_3} (h')^{\gamma_4}$
  $\wedge \{\beta_i\}_{i \in [4]} \in \{-1, 1\}]$
$\textbf{return } \pi$
```

```
$\mathsf{Prove}^{\mathsf{Com}}_{CS}(params, C, [M, O]) \rightarrow \pi$

$\textbf{parse } C = (C_1, C_2, C_3, C_4),$
  $O = (a_1, a_2, s_1, s_2, r_1, r_2, b_1, b_2)$
$\pi = \mathsf{NIZK}[O :$
    $C_2 = b_1 (g')^{s_1} (h')^{r_1} \wedge C_4 = b_2 (g')^{s_2} (h')^{r_2}$
    $\wedge b_1 \in \{-1, 1\} \wedge b_2 \in \{-1, 1\}]$
$\textbf{return } \pi$

$\mathsf{Prove}^{\mathsf{mult}}_{CS}(params, C_a, C_b, C_y, [a, b, y, O_a, O_b, O_y, b_y, \{b_i\}_{i \in [4]}]) \rightarrow \pi$

$\textbf{parse } C_a = (C_{a,i})_{i \in [4]}$
  $C_b = (C_{b,i})_{i \in [4]}$
  $O_a = (a_{a,i}, s_{a,i}, r_{a,i}, b_{a,i})_{i \in [2]}$
  $O_b = (b_{b,i}, s_{b,i}, r_{b,i}, b_{b,i})_{i \in [2]}$
$\gamma_1 \leftarrow s_{a,1} - y s_{b,1}; \gamma_2 \leftarrow r_{a,1} - y r_{b,1}$
$\gamma_3 \leftarrow s_{a,2} - y s_{b,2}; \gamma_4 \leftarrow r_{a,2} - y r_{b,2}$
$\beta_1 \leftarrow a_{a,1} / a_{b,1}; \beta_2 \leftarrow b_{a,1} / b_{b,1}$
$\beta_3 \leftarrow a_{a,2} / a_{b,2}; \beta_4 \leftarrow b_{a,2} / b_{b,2}$
$\pi = \mathsf{NIZK}[\{\gamma_i, \beta_i\}_{i \in [4]} :$
  $C_y = b_y (g')^y (g')^{r_y}$
  $\wedge C_{a,1} = b_1 (C_{b,1})^y (g')^{\gamma_1}$
  $\wedge C_{a,2} = b_2 (C_{b,2})^y (g')^{\gamma_1} (h')^{\gamma_2}$
  $\wedge C_{a,3} = b_3 (C_{b,3})^y (g')^{\gamma_3}$
  $\wedge C_{a,4} = b_4 (C_{b,4})^y (g')^{\gamma_3} (h')^{\gamma_4}$
  $\wedge \beta_y, \beta_1, \beta_2, \beta_3, \beta_4 \in \{-1, 1\}]$
$\textbf{return } \pi$

$\mathsf{Prove}^{\mathsf{enc}}_{CS}(params, \mathsf{pk}_{AH} = k, C_a, C_y, [a, r_a, y, O_a, O_y, b_y, \{b_i\}_{i \in [4]}]) \rightarrow \pi$

$\textbf{parse } C_a = (C_{a,i})_{i \in [4]}$
  $O_a = (a_{a,i}, s_{a,i}, r_{a,i}, b_{a,i})_{i \in [2]}$
$\pi = \mathsf{NIZK}[O_a, s_y, r_a, r_y, y :$
    $C_y = b_y (g')^y (h')^{r_y}$
    $\wedge C_{a,1} = b_1 (g^*)^{r_a} (g')^{s_{a,1}}$
    $\wedge C_{a,2} = b_2 (g')^{s_{a,1}} (h')^{r_{a,1}}$
    $\wedge C_{a,3} = b_3 k^{r_a} (g^*)^y (g')^{s_{a,2}}$
    $\wedge C_{a,4} = b_4 (g')^{s_{a,2}} (h')^{r_{a,2}}$
    $\wedge b_y, b_1, b_2, b_3, b_4 \in \{-1, 1\}]$
$\textbf{return } \pi$
```

Theorem 11 (Security of Camenisch-Shoup commitments). *The construction in Fig. 4.6 satisfies four properties: (1) statistically hiding; (2) computationally binding; (3) our protocols in Fig. 4.6 (*$\mathsf{Prove}_{CS}^{\mathsf{Com}_{AH}}$*,* $\mathsf{Prove}_{CS}^{\mathsf{enc}}$*,* $\mathsf{Prove}_{CS}^{\odot}$*, and* $\mathsf{Prove}_{CS}^{\oplus}$*) are computationally zero-knowledge; and (4) computationally black-box knowledge extractable.*

Proofs for commitments to Camenisch-Shoup ciphertexts We split Thm. 11 into the following theorems:

Theorem 12 (Zero-knowledge of proofs in Fig. 4.6). *Our protocols in Fig. 4.6 (*$\mathsf{Prove}_{CS}^{\mathsf{Com}}$*,* $\mathsf{Prove}_{CS}^{\mathsf{enc}}$*,* $\mathsf{Prove}_{CS}^{\mathsf{mult}}$*, and* $\mathsf{Prove}_{CS}^{\mathsf{add}}$*) are zero-knowledge against any PPT adversary.*

Proof (Proof of Thm. 12). We can see that in each of these NIZKs, we simply return a proof computed from the *eqrep* protocol. Thus, we can use the simulator for this protocol to produce proofs in the zero knowledge games. Thus, if a PPT adversary can distinguish these simulated proofs from real proofs, we can break the zero knowledge of the *eqrep* protocol.

Theorem 13 (Black box knowledge extraction of proofs in Fig. 4.6). *Given a PPT adversary that can produce a proof that verifies for our protocols in Fig. 4.6 (*$\mathsf{Prove}_{CS}^{\mathsf{Com}}$*,* $\mathsf{Prove}_{CS}^{\mathsf{enc}}$*,* $\mathsf{Prove}_{CS}^{\mathsf{mult}}$*, and* $\mathsf{Prove}_{CS}^{\mathsf{add}}$*) there exists an extractor with black-box access to the adversary that can extract a witness that proves the relations true.*

Proof (Proof of Thm. 13). Similar to our proof of zero-knowledge for these protocols, because these protocols simply return *eqrep* proofs, we can use the black-box extractor for these proofs to extract the witnesses. This extractor is described in Sec. 2.

Theorem 14 (Hiding of the commitments in Fig. 4.6). *Our commitments to Camenisch-Shoup ciphertexts in Fig. 4.6 are statistically hiding.*

Proof (Proof of Thm. 14). We can see that (C_1, C_2) is identical to a $|QR_{n^2}|$ commitment to c_1 and (C_3, C_4) is identical to a $|QR_{n^2}|$ commitment to c_2, we can see that they statistically hide c_1 and c_2.

Theorem 15 (Binding of the commitments in Fig. 4.6). *Our commitments to Camenisch-Shoup ciphertexts in Fig. 4.6 are computationally binding.*

Proof (Proof of Thm. 15). We can see that (C_1, C_2) is identical to a $|QR_{n^2}|$ commitment to c_1 and (C_3, C_4) is identical to a $|QR_{n^2}|$ commitment to c_2, thus, if a PPT adversary can produce a double opening such that one of these commitments opens to some c_1' or c_2' in $|QR_{n^2}|$, we obtain a double opening for our $|QR_{n^2}|$ commitments.

5 Non-Frameable Privacy-Preserving Blueprints

Given this new efficient framework for verifiable computation on ciphertexts, we are now equipped to build a PPB scheme with stronger security which can withstand the framing attack in Sect. 1.2. We first define the property of non-frameability for PPBs, and then focus our attention on proving it. We extend the formal definition of a blueprint scheme as introduced in [KLN23], see Sect. 2.1.

In order to systematically prevent framing attacks and formally define the notion of non-frameability, we change the Dec algorithm to additionally outputs a proof. We introduce an additional Judge algorithm to be included in a (non-frameable) blueprint scheme for verifying this proof.

Definition 7 (A non-frameable f-blueprint scheme). *For a non-interactive commitment scheme* $(\mathsf{CSetup}, \mathsf{Com})$*, a non-frameable f-blueprint scheme consists of all the algorithms of a basic f-blueprint scheme with an adapted* $\mathsf{Decrypt}$ *algorithm and an additional* Judge *algorithm:*

$\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C_y, Z) \rightarrow (f(x,y), \pi_z)$ or $\perp$: Takes the auditor's secret key $\mathsf{sk_A}$, commitment C_y and escrow Z such that $\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C_y, Z) = 1$ as input. Decrypts the escrow and returns the output $f(x,y)$ if C_y is a commitment to y. Additionally it returns a proof, π_z, that proves to the Judge algorithm that $f(x,y)$ was decrypted correctly from Z.

$\mathsf{Judge}(\Lambda, \mathsf{pk_A}, C_x, C_y, Z, z, \pi_z) \rightarrow 0$ or 1: Takes as input all the inputs of VerPK, $\mathsf{VerEscrow}$, z, π and verifies that z was obtained correctly from escrow Z.

Correctness of Judge: Assume values $(\Lambda, \mathsf{pk_A}, C_x, C_y, Z, z, \pi)$ are generated honestly that is: (1) $cpar \in \mathsf{CSetup}(1^\lambda)$; (2) $\Lambda \in \mathsf{Setup}(1^\lambda, cpar)$; (3) $(\mathsf{pk_A}, \mathsf{sk_A}) \in \mathsf{KeyGen}(\Lambda, x, r_x)$; (4) $C_x = \mathsf{Com}_{cpar}(x; r_x)$; (5) $C_y = \mathsf{Com}_{cpar}(y; r_y)$; (6) $Z \in \mathsf{Escrow}(\Lambda, \mathsf{pk_A}, y, r_y)$; (7) $(z, \pi_z) \in \mathsf{Dec}(\Lambda, \mathsf{sk_A}, C_y, Z)$. We require that algorithm Judge accept with probability 1 i.e. $\mathsf{Judge}(\Lambda, \mathsf{pk_A}, C_x, C_y, Z, z, \pi_z) = 1$.

We want to make sure that even if the auditor colludes with dishonest users, it is not possible for a dishonest auditor to frame an honest user.

Definition 8 (Non-Frameability). *Let C_x and C_y be commitments computed from (x, r_x) and (y, r_y) respectively. Non-frameability guarantees that any pk_A, Z, z, π_z that passes $\mathsf{Judge}(\Lambda, \mathsf{pk}_A, C_x, C_y, Z, z, \pi_z)$ will imply that $f(x,y) = z$ with overwhelming probability. More formally, for all PPT adversaries $\mathcal{A}$, there exists a negligible function ν such that: $Pr\big[\mathsf{NonFraming}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda) = 1\big] < \nu(\lambda)$*

[KLN23] uses a "homomorphic-enough" encryption (HEC) scheme to construct their PPB scheme. The existing HEC schemes that are only correct and sound as defined in [KLN23] will not be sufficient to construct Non-Frameable Blueprint schemes. We define a stronger HEC scheme in the following subsection.

5.1 Consistent Homomorphic-Enough Encryption

The [KLN23] HEC scheme is parameterized by a function family and is correct if it is possible to compute any function from that family using only the ciphertexts.

```
$\mathsf{NonFraming}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$
$cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
$\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
$(\mathsf{pk_A}, x, r_x, y, r_y, Z, z, \pi_z) \leftarrow \mathcal{A}(1^\lambda, \Lambda)$
$C_x = \mathsf{Com}_{cpar}(x, r_x); C_y = \mathsf{Com}_{cpar}(y, r_y)$
$\textbf{return } [(\mathsf{Judge}(\Lambda, \mathsf{pk_A}, C_x, C_y, Z, z, \pi_z) = 1) \wedge (f(x, y) \neq z)]$
```

Fig. 5.1: Experiments $\mathsf{NonFraming}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$

Definition 9 (Homomorphic-enough cryptosystem (HEC) for a function family). *Let $F = \{f \mid f : domain_{f,x} \times domain_{f,y} \mapsto range_f\}$ be a set of polynomial-time computable functions. We say that algorithms* HEC = (HECsetup, HECenc, HECeval, HECdec, HECdirect) *constitute a HEC for F if they satisfy the input-output, correctness, and security requirements below:*

- HECsetup(1^λ) → *hecpar* takes the security parameter as input, outputs the parameters *hecpar*.
- HECenc(*hecpar*, f, x) → (X, d) takes parameters *hecpar*, a function $f \in F$, and a value $x \in domain_{f,x}$ as input, outputs an encrypted representation X of the function $f(x, \cdot)$, and a decryption key d.
- HECeval(*hecpar*, f, X, y) → Z takes as input the parameters *hecpar*, a function $f \in F$, an encrypted representation of $f(x, \cdot)$, and a value $y \in domain_{f,y}$ and outputs a ciphertext Z, an encryption of $f(x, y)$.
- HECdec(*hecpar*, d, Z) → z takes as input the parameters *hecpar*, the decryption key d, and a ciphertext Z, decrypts Z to obtain a value z.
- HECdirect(*hecpar*, X, z) → Z on input *hecpar*, an encrypted representation X of some function, and a value z, outputs a ciphertext Z.

Fig. 5.2: Algorithms of HEC scheme for F

HEC correctness. For a given adversary Adv and HEC, let $\mathsf{Adv}_{\mathrm{HEC},\mathsf{Adv}}$ be the probability that the experiment HECcorrect in Fig. 5.3 accepts. HEC is *correct* if $\mathsf{Adv}_{\mathrm{HEC},\mathsf{Adv}}$ is negligible for all PPT algorithms Adv.
HEC security. We provide the formal definitions for the Security of x, security of x and y from third parties, and security of DirectZ in Appx. E.1.

Our main insight for adapting the generic construction of blueprints from a HEC scheme is that the adversary now controls the randomness r to the HEC encryption algorithm, in addition to the randomness r_Z, and can thus exercise additional control over the output of HECenc. We refer to this strengthening of the correctness property w.r.t. adversarial inputs as *HEC consistency.*

Definition of Consistent HEC. In the HEC consistency game, the adversary outputs x, y, and the randomness for the HEC scheme (r, r_Z), and the encryp-

```
$\mathrm{HEC}\textsc{correct}^{\mathsf{Adv}}(\lambda)$
$hecpar \leftarrow \mathrm{HEC}\textsc{Setup}(\lambda)$
$(f, x, \mathsf{st}) \leftarrow \mathsf{Adv}(1^\lambda, hecpar)$
if $f \in F, x \in domain_{f,x}$
    $(X, d) \leftarrow \mathrm{HEC}\textsc{Enc}(hecpar, f, x)$
    $(y, r_Z) \leftarrow \mathsf{Adv}(\mathsf{st}, X)$
    if $y \in domain_{f,y}$
        $Z \leftarrow \mathrm{HEC}\textsc{Eval}(hecpar, f, X, y; r_Z)$
        if $\mathrm{HEC}\textsc{Dec}(hecpar, d, Z) \neq f(x, y)$
            return 1
return 0
```

```
$\mathrm{HEC}\textsc{consistent}^{\mathsf{Adv}}(\lambda)$
$hecpar \leftarrow \mathrm{HEC}\textsc{Setup}(\lambda)$
$(f, x, \mathsf{st}, r, y, r_Z) \leftarrow \mathsf{Adv}(1^\lambda, hecpar)$
if $f \notin F \vee x \notin domain_{f,x} \vee y \notin domain_{f,y}$
    return 0
$(X, d) \leftarrow \mathrm{HEC}\textsc{Enc}(hecpar, f, x; r)$
$Z \leftarrow \mathrm{HEC}\textsc{Eval}(hecpar, f, X, y; r_Z)$
if $\mathrm{HEC}\textsc{Dec}(hecpar, d, Z) \neq f(x, y)$
    return 1
return 0
```

Fig. 5.3: HEC correctness, consistency and security games

tion and evaluation algorithms cannot produce a ciphertext that decrypts to a plaintext other than $f(x, y)$. We formalize this in Fig. 5.3.

Modifying the Generic Blueprint Scheme from HEC to Obtain Non-Frameability. As described previously (Def. 7), to obtain non-frameability, the Dec algorithm now returns a proof of knowledge of correct decryption and a new algorithm Judge is introduced.

Incorporating the property of non-frameability in the definition of blueprint schemes gives us the following theorem which is virtually identical to the result obtained in [KLN23] (Theorem 2) apart from adding the condition on the new NIZK PoK, Ψ_3, and the properties of consistency and non-frameability.

Theorem 16. *If* HEC *is a consistent and secure homomorphic-enough cryptosystem, the commitment scheme is binding, and the NIZK PoKs Ψ_1, Ψ_2 and Ψ_3 are zero-knowledge and BB-PSL simulation extractable then our generic blueprint scheme is a secure, non-frameable f-blueprint scheme.*

Proof. Since the property of HEC consistency implies HEC correctness, the proofs of correctness of VerEscrow, VerPK and Dec from the original PPB proof of [KLN23], goes through unchanged. Similarly, the soundness of the generic f-blueprint scheme is also proven using the BB-Extractability of the NIZK Ψ_2 in the same reduction as in [KLN23].

Using these properties and the correctness of the Judge which we prove in Lemma 4, we prove the non-frameability of the HEC scheme in 5.1.

Lemma 4. *If the NIZK PoKs Ψ_1, Ψ_2 and Ψ_3 are complete, then the generic blueprint scheme satisfies correctness of* Judge*.*

Proof. Consider Judge as defined in Fig. 5.4. Suppose this algorithm Judge returns 0 in the above mentioned experiment. This can happen if either VerEscrow returns a reject or if $\mathsf{V}_3^{\mathsf{S}_3}(Z, f_{xy}, hecpar, cpar) = 0$. From correctness of VerEscrow, we know that VerEscrow returns 1, so Judge only returns 0 if $\mathsf{V}_3^{\mathsf{S}_3}(Z, f_{xy}, hecpar, cpar) = 0$. However, this contradicts completeness of the NIZK scheme because the proof π_Z in Z is generated by Dec on a valid statement and witness pair.

```
$\mathsf{Setup}(\lambda, cpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $hecpar \leftarrow \text{HEC}\textsc{setup}(1^\lambda)$
  $\textbf{return } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$

$\mathsf{KeyGen}(\Lambda, x, r_x)$
  $\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $(X, d) \xleftarrow{r} \text{HEC}\textsc{enc}(hecpar, f, x)$
  $C_x = \mathsf{Com}_{cpar}(x; r_x)$
  $\pi_{\mathsf{A}} \leftarrow \mathsf{PoK}^{\mathsf{S}_1}_{\Psi_1}\Big\{(x, d, r, r_x) :$
      $(X, d) = \text{HEC}\textsc{enc}(hecpar, f, x; r)$
      $\wedge\, C_x = \mathsf{Com}_{cpar}(x; r_x))\Big\}$
  $\mathsf{pk}_{\mathsf{A}} \leftarrow (X, C_x, \pi_{\mathsf{A}}); \mathsf{sk}_{\mathsf{A}} \leftarrow (\mathsf{pk}_{\mathsf{A}}, d)$
  $\textbf{return } (\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}})$

$\mathsf{VerPK}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_x)$
  $\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $\textbf{parse } \mathsf{pk}_{\mathsf{A}} = (X, C'_x, \pi_{\mathsf{A}})$
  $\textbf{return } \mathsf{V}^{\mathsf{S}_1}_1((X, hecpar, f, C_x, cpar), \pi_{\mathsf{A}})$
      $\wedge\, (C'_x = C_x)$

$\mathsf{Judge}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_x, C_y, Z = (\hat{Z}, \pi_{\mathsf{U}}), z, \pi_Z)$
  $\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $\textbf{return } \mathsf{V}^{\mathsf{S}_3}_3((z, hecpar, \hat{Z}), \pi_Z)$
      $\wedge\, \mathsf{VerPK}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_x)$
      $\wedge\, \mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_y, Z)$

$\mathsf{Escrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, y, r_y)$
  $\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $\textbf{parse } \mathsf{pk}_{\mathsf{A}} = (X, C_x, \_)$
  $\textbf{if } \mathsf{VerPK}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_x) = 0$
    $\textbf{return } 0$
  $\hat{Z} \xleftarrow{r_{\hat{Z}}} \text{HEC}\textsc{eval}(hecpar, f, X, y)$
  $C_y = \mathsf{Com}_{cpar}(y; r_y)$
  $\pi_{\mathsf{U}} \leftarrow \mathsf{PoK}^{\mathsf{S}_2}_{\Psi_2}\Big\{(y, r_y, r_{\hat{Z}}) :$
    $\hat{Z} = \text{HEC}\textsc{eval}(hecpar, f, X, y; r_{\hat{Z}})$
    $\wedge\, C_y = \mathsf{Com}_{cpar}(y; r_y)\Big\}$
  $\textbf{return } (\hat{Z}, \pi_{\mathsf{U}})$

$\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_y, Z = (\hat{Z}, \pi_{\mathsf{U}}))$
  $\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $\textbf{parse } \mathsf{pk}_{\mathsf{A}} = (\_, C_x, \_)$
  $\textbf{return } \mathsf{VerPK}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_x)$
    $\wedge\, \mathsf{V}^{\mathsf{S}_2}_2((\hat{Z}, hecpar, f, X, C_y, cpar), \pi_{\mathsf{U}})$

$\mathsf{Dec}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C_y, Z = (\hat{Z}, \pi_{\mathsf{U}}))$
  $\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$
  $\textbf{parse } \mathsf{sk}_{\mathsf{A}} = (\mathsf{pk}_{\mathsf{A}}, d)$
  $\textbf{if } \mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C_y, Z) = 0$
    $\textbf{return } \bot$
  $z \leftarrow \text{HEC}\textsc{dec}(hecpar, d, \hat{Z})$
  $\pi_Z \leftarrow \mathsf{PoK}^{\mathsf{S}_3}_{\Psi_3}\Big\{d : z = \text{HEC}\textsc{dec}(hecpar, d, \hat{Z})\Big\}$
  $\textbf{return } (z, \pi_Z)$
```

Fig. 5.4: Construction of generic f-blueprint scheme from HEC and NIZK PoKs Ψ_1, Ψ_2 and Ψ_3 with setup $\mathsf{S}_1, \mathsf{S}_2$, and S_3 respectively.

Therefore, if the NIZK PoKs Ψ_1, Ψ_2 and Ψ_3 are complete, then the generic blueprint scheme satisfies correctness of Judge.

Lemma 5. *Let Ψ_3 be a BB extractable NIZK scheme, let* $(\mathsf{CSetup}, \mathsf{Com})$ *be a computationally binding commitment scheme, and* HEC *be consistent with adversarial evaluation randomness, then our proposed scheme achieves Non-frameability.*

Proof. Consider Fig. 5.1. Suppose, for the sake of contradiction, that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{NonFraming}}_{\mathcal{A},\mathsf{Blu}} = \nu(\lambda)$ is non negligible. Let Z, one of the adversary's output in the experiment, be divided into $\hat{Z}$ and a proof π_{U} to validate $\hat{Z}$. The events where $\mathcal{A}$ outputs 1 can be divided into four cases: (i) when $C = \mathsf{Com}(y; r)$, $C = \mathsf{Com}(y'; r')$ and $\hat{Z} = \text{HEC}\textsc{eval}(hecpar, f, X, y'; r_{\hat{Z}})$

for $y \neq y'$, (ii) when $C = \mathsf{Com}(y; r)$ and $\hat{Z} = \text{HEC}\textsc{Eval}(hecpar, f, X, y; r_{\hat{Z}})$ for some $r_{\hat{Z}}$ where in both (i) and (ii) X is a part of $\mathsf{pk_A}$, (iii) the case where neither of these equalities holds and (iv) when $C = \mathsf{Com}(y; r)$ and $(X, d) = \text{HEC}\textsc{Enc}(hecpar, f, x; r)$.

We express the probabilities of these events with the functions $\nu_0(\lambda)$, $\nu_1(\lambda)$, $\nu_2(\lambda)$, and $\nu_3(\lambda)$ respectively. Since $\nu(\lambda)$ is non negligible and these three events covers all cases where Adv would output 1, at least one of $\nu_0(\lambda)$, $\nu_1(\lambda)$, $\nu_2(\lambda)$ or $\nu_3(\lambda)$ must be non negligible.

Suppose $\nu_2(\lambda)$ is non negligible. The adversary produced a proof of a false statement and we can construct a reduction $\mathcal{B}$ to the BB extractable NIZK system. $\mathcal{B}$ runs $\mathcal{A}$ the same way as Sound, see Fig. C.1, but outputs $(\hat{Z}, hecpar, f, X, C_y, cpar), \pi_{\mathsf{U}})$ instead. By BB extractability of the NIZK, $\Pr[\mathcal{B} \text{ wins}]$ of extraction failure is negligible, which contradicts our assumption $\nu_2(\lambda)$ is non negligible.

Similarly, consider $\nu_3(\lambda)$ to be non-negligible. As proved above, we can reduce this case to a contradiction of the BB-extractability of the NIZK Ψ_3.

We now assume that the BB extractor extracts a witness $(y', r'_y, r_{\hat{Z}})$, such that $\hat{Z} = \text{HEC}\textsc{Eval}(hecpar, f, X, y'; r_{\hat{Z}})$ and $C_y = \mathsf{Com}_{cpar}(y'; r'_y)$. Suppose $\nu_0(\lambda)$ is non negligible. In this event, we break the computational binding property using a reduction that outputs (y, r, y', r'). Suppose $\nu_1(\lambda)$ is non negligible. In this event, we get a situation where both $\mathsf{pk_A}$ and Z were generated correctly with adversarial randomness $r_{\hat{Z}}$, but the output of decrypt is incorrect. We can construct a reduction $\mathcal{B}$ using $\mathcal{A}$ to HEC consistency with adversarial evaluation randomness. $\mathcal{B}$ runs $\mathcal{A}$, in the same way as Sound, see Fig. C.1, but instead of returning a bit at the end, it outputs the tuple $(y, r_{\hat{Z}})$.

The f-blueprint scheme having the properties of *Blueprint Hiding*, *Privacy against dishonest auditor* and *Privacy against honest auditor* can be shown using the same proofs as in [KLN23].

Consistent HEC from fully homomorphic encryption (FHE) We provide an efficient construction for a secure, consistent HEC scheme for the watchlist function in Sec. 5.2. We show that the existing construction of a HEC scheme for any function f from FHE, as provided in [KLN23], is also secure and consistent. The full details of the construction and the proof of Thm. 17 is in Appx. E.2.

Theorem 17. *For a FHE scheme,* $(\mathsf{FHEKeyGen}, \mathsf{FHEEnc}, \mathsf{FHEDec}, \mathsf{FHEEval})$ *with the Correctness property, for a circuit family* $\{\mathcal{C}_j^f : f \in F\}$ *(as defined in [KLN23]), the construction in [KLN23] is a consistent HEC for the family* F*.*

5.2 Instantiation of Consistent HEC Scheme

In this section, we provide a HEC scheme that satisfies our definition of consistent HEC from Sec. 5.1. In Sec. 5.3 we show a succinct proof system Ψ_2 which ensures escrows are created honestly.

To obtain a non-frameable watchlist scheme, we construct the algorithms $\text{HEC}\textsc{Eval}$ and $\text{HEC}\textsc{Dec}$ in Fig. 5.5 for the function family $\{f_{n,k}\}_{n,k\in\mathbb{Z}}$, where n is the length of the auditor's list $x = \{x_1, \ldots, x_n\}$ and k is the bit length of the user's attribute y_{at}, where the user's input consists of the user's identifier

y_{id} and an attribute: $y = (y_{id}, y_{at})$. $f_{n,k}$ is defined as $f_{n,k}(x, y) = y$ if $y_{id} \in x$ and $f_{n,k} = \emptyset$ otherwise. We discuss why this watchlist function is useful for the watchlist/CBDC application in Sec. 1. y_{id} uniquely identifies a user and y_{at} could be any useful data about the user such as a seed for the user's e-cash. We construct a HECEval algorithm for multiple attributes in Sec. 5.4.

Overview of the construction. The HECEnc algorithm (Fig. 5.5) takes as input the list x of n watchlisted identities, and computes a polynomial $P(\chi) = \sum_{i\in[n]} a_i \chi^i$ such that $P(y_{id}) = 0$ if and only if $y_{id} \in x$. Then, it samples a key pair $(\mathsf{pk}_{AH}, \mathsf{sk}_{AH})$ for a semantically secure $\mathfrak{g}$-semi-encryption scheme (Def. 6), and outputs the public key $X = (\mathsf{pk}_{AH}, \{A_i = \mathsf{Enc}(\mathsf{pk}_{AH}, a_i)\}_{i\in[0...n]})$ where the a_i's are coefficients of P, and the decryption key $d = (\mathsf{sk}_{AH}, x)$.

On input the public key X and the value $y = (y_{id}, y_{at})$, HECEval will output the escrow $Z = (Z_{id}, Z_{at}, Z_{nf})$ which consists of three ciphertexts under the key pk_{AH}; these will decrypt to the values $(y_{id}, y_{at}, 0)$ if and only if $y_{id} \in x$; otherwise they will decrypt to uniformly random elements of the message space, independent of y. As we show in more detail in Fig. 5.5, additively homomorphic properties of the underlying (semi-)encryption scheme allow the evaluator to form the ciphertext E so that it will be an encryption of $P(y_{id})$. The evaluator also encrypts the identity y_{id} and attribute y_{at}, yielding ciphertexts Y_{id} and Y_{at}. The escrow of y_{id} is then formed as $Z_{id} = (r_1 \odot E) \oplus Y_{id} = ((r_1 \odot \boxed{P(y_{id})}) \oplus \boxed{y_{id}} = \boxed{r_1 P(y_{id}) + y_{id}}$, which is an encryption of y_{id} if E is an encryption of 0 (i.e. whenever $y_{id} \in x$), and an encryption of a random value otherwise, thanks to the randomizer r_1. Similarly, the escrow of y_{at} is $Z_{at} = (r_2 \odot E) \oplus Y_{at} = \boxed{r_2 P(y_{id}) + y_{at}}$. To make the HEC consistent, we include $Z_{nf} = r_3 \odot E = \boxed{r_3 P(y_{id})}$, which will decrypt to 0 if and only if $y_{id} \in x$.

HECDec takes as input the HEC decryption key $d = (\mathsf{sk}_{AH}, x)$ and the escrow Z. It recovers y'_{id}, y'_{at}, and y' by decrypting the escrows (Z_{id}, Z_{at}, Z_{nf}) using the secret key, sk_{AH}. By the correctness property the decryption algorithm for $\mathfrak{g}$-semi-encryption, we know that for $Z \in$ HECEnc(X, y), $y' = \mathfrak{g}(r_3 P(y_{id})) = \mathfrak{g}(0)$ if and only if $y_{id} \in x$; so if $y' \neq \mathfrak{g}(0)$, HECDec outputs $\perp$. Else, we know that $y_{id} \in x$, so HECDec must somehow determine (1) y_{id} from $y'_{id} = \mathfrak{g}(y_{id})$, and (2) y_{at} from $y'_{attr} = \mathfrak{g}(y_{at})$. Let us explain how HECDec can do so.

If $\mathfrak{g}$ is the identity function then this step is trivial; we will show in Sec. 4 that we can achieve an additively homomorphic $\mathfrak{g}$-semi-encryption scheme where $\mathfrak{g}$ is the identity function under the decisional composite residuosity assumption using the Camenisch-Shoup cryptosystem.

If, however, $\mathfrak{g}$ is a one-way injective function, then (1) can be done by looking for $\mathfrak{g}(y_{id})$ on the list $\mathfrak{g}(x_1), \ldots, \mathfrak{g}(x_n)$ where $x_i \in x$ and (2) can only be done by exhaustive search, which is only possible if y_{at} comes from a small space. This is the approach that was (implicitly) taken by the original PPB paper of Kohlweiss et al.: since the ElGamal cryptosystem is only additively homomorphic when viewed as a $\mathfrak{g}$-semi-encryption scheme, and $\mathfrak{g}$ is a one-way function, they could only achieve attributes from a small space.

Theorem 18 (Security of the construction in Fig. 5.5). *Our construction in Fig. 5.5 achieves* HEC *consistency in Def. 5.3, security of* DirectZ,

$\text{HEC}\textsc{dec}(hecpar, d, Z)$

```
parse $d = (\mathsf{sk}_E, f_{n,k,\ell}, x)$,
    $Z = (Z_{id}, Z_{at}, Z_{nf})$
$y'_{id} \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{id})$
$y'_{at} \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{at})$
$y' \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{nf})$
if $y' \neq \mathfrak{g}(0)$
  return $\emptyset$
for $y_{id} \in x$
  if $\mathfrak{g}(y_{id}) = y'_{id}$
    return $(y_{id}, y_{at})$
        where $y_{at} \in domain_{f,y,at}$
          $\wedge\ \mathfrak{g}(y_{at}) = y'_{at}$
return $\emptyset$
```

$\text{HEC}\textsc{enc}(hecpar, f_{n,k}, x)$

```
$(\mathsf{pk}_{AH}, \mathsf{sk}_E) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$s \leftarrow\$ \mathcal{M}_{\mathsf{pk}_{AH}}$
$P \leftarrow s \prod_{i=1}^{n} (\chi - x_i)$
for $i$ in$\{1, \ldots, n+1\}$
    $A_i \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, P_i)$
return $(X = (\mathsf{pk}_{AH}, A_1, \ldots, A_{n+1})$,
   $d = (\mathsf{sk}_E, f_k, x)))$
```

$\text{HEC}\textsc{eval}(hecpar, f_{n,k,\ell}, X, y; r_{\hat{Z}})$

```
parse $X = (\mathsf{pk}_{AH}, A_1, ..., A_{n+1})$,
    $y = (y_{id}, y_{at})$,
    $r_{\hat{Z}} = (r_{id}, r_{at}, r_1, r_2, r_3)$
if $r_3 = 0$, return $\perp$
$E \leftarrow \bigoplus_{i=1}^{n+1} (A_i \odot y_{id}^i)$
$Y_{id} \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, y_{id}; r_{id})$
$Y_{at} \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, y_{at}; r_{at})$
$Z_{id} \leftarrow (r_1 \odot E) \oplus Y_{id}$
$Z_{at} \leftarrow (r_2 \odot E) \oplus Y_{at}$
$Z_{nf} = r_3 \odot E$
return $Z = (Z_{id}, Z_{at}, Z_{nf})$
```

$\text{HEC}\textsc{direct}(hecpar, X, z)$

```
parse $X = (\mathsf{pk}_{AH}, A_1, \ldots, A_{n+1})$
      $z = (z_1, z_2, z_3)$
if $z = \emptyset$
  $\beta_1 \leftarrow\$ \mathcal{M}_{\mathsf{pk}_{AH}}$
  $\beta_2 \leftarrow\$ \mathcal{M}_{\mathsf{pk}_{AH}}$
  $\beta_3 \leftarrow\$ \mathcal{M}_{\mathsf{pk}_{AH}}$
  return $(\mathsf{Enc}(\mathsf{pk}_{AH}, \beta_1)$,
      $\mathsf{Enc}(\mathsf{pk}_{AH}, \beta_2), \mathsf{Enc}(\mathsf{pk}_{AH}, \beta_3))$
return $(\mathsf{Enc}(\mathsf{pk}_{AH}, \mathfrak{g}(z_1))$,
    $\mathsf{Enc}(\mathsf{pk}_{AH}, \mathfrak{g}(z_2)), \mathsf{Enc}(\mathsf{pk}_{AH}, \mathfrak{g}(z_3)))$
```

Fig. 5.5: HEC algorithms

***Security of** y, and **security of** x **and** y **from third parties**, defined in Def. 15.*

Proof of Thm. 18 Because we include $Z_{nf} = E \odot r_3$ in the escrow, an auditor can prove that this is an encryption of 0. This ensures that the y_{id} is actually on the watchlist as the polynomial has roots at each entry of the watchlist. Formally, if an adversary were to be able to produce a $(f, x, \mathsf{st}, r, y, r_Z)$ such that $Z \leftarrow \text{HEC}\textsc{eval}(hecpar, f, X, y; r_Z)$ but $\text{HEC}\textsc{dec}(hecpar, d, Z) \neq f(x, y)$, we see that $E \odot r_3 = 0$ in this case, which implies that $r_3 P(y) = 0$. This is only true if $y \in x$ since $r_3 > 0$. In this case, because $\text{HEC}\textsc{eval}$ is proven to be correctly computed, $E \odot r_1$ decrypts to 0. Thus, $y' = \mathsf{Dec}(Y)$. Thus, this decrypts to the correct value.

We split Theorem 18 into Theorems 19, 20, and 21.

Theorem 19 (Security of DirectZ for Fig. 5.5). *Our construction in Fig. 5.5 achieves security of* DirectZ *defined in Def. 15.*

Proof of Thm. 19 We prove the theorem for the two separate cases of when the user is in the watchlist and when they are not.

For the former, since the user is on the watchlist, $f(x, y) \neq 0$. In HECEval, (Z_{id}, Z_{at}) is an encryption of $f(x, y)$ and in HECDirect, Z_{nf} is an encryption of 0. Considering the experiments $\text{DirectZ}_0^{\mathsf{Adv}}$ and $\text{DirectZ}_1^{\mathsf{Adv}}$, since the ciphertext of $f(x, y)$ is output in both cases, the indistinguishability of the experiments can be reduced to the IND-CPA security of the underlying encryption scheme.

In the case where the user is not on the watchlist, $f(x, y) = \emptyset$. Since separate randomness is used for each of r_1, r_2, and r_3 in HECEval, therefore each ciphertext is the encryption of a random value, $Z_{id} = r_1 P(y_{id}) + y_{id}$, $Z_{at} = r_2 P(y_{id}) + at$ and $Z_{nf} = r_3 P(y_{id})$ because $P(y_{id}) \neq 0$. This makes the three ciphertext values indistinguishable from random in $\text{DirectZ}_0^{\mathsf{Adv}}$. In experiment $\text{DirectZ}_1^{\mathsf{Adv}}$, the HECDirect function simply encrypts random values when $f(x, y) = 0$. Therefore, the two experiments are indistinguishable and we achieve security of DirectZ.

Theorem 20 (Security of x and y for Fig. 5.5). *Our construction in Fig. 5.5 achieves* ***security of x and y from third parties****.*

Proof of Thm. 20. Let us assume there exists an adversary for whom $|p^{\text{SecXY}}_{\mathsf{Adv},0}(\lambda) - p^{\text{SecXY}}_{\mathsf{Adv},1}(\lambda)|$ is non-negligible. This implies that either (i) the adversary can distinguish an encryption of x_0 from x_1 or (ii) the adversary can distinguish an encryption of y_0 from y_1. From Thm. 21, the adversary distinguishing an encryption of x_0 from an encryption of x_1 can be reduced to the IND-CPA game of the underlying scheme. This holds similarly for y_0 and y_1.

Theorem 21 (Security of x for Fig. 5.5). *Our construction in Fig. 5.5 achieves* ***Security of*** *y*

Proof of Thm. 21. Let us assume there exists an adversary Adv for whom $|p^{\text{SecX}}_{\mathsf{Adv},0}(\lambda) - p^{\text{SecX}}_{\mathsf{Adv},1}(\lambda)|$ is non-negligible. Let x_0 and x_1 be the input for which Adv wins the SecX game by correctly distinguishing the ciphertext of x_0 from the ciphertext of x_1. In that case, we can construct an IND-CPA adversary Adv' that wins the IND-CPA game by using the same input x_0 and x_1. This is possible since Adv does not possess the secret key for the HEC scheme. Thus, IND-CPA security of the underlying encryption scheme implies the SecX security of the HEC scheme.

5.3 Efficient Instantiation of HEC Evaluation Proof Ψ_2

In this section we show how to use the techniques introduced in 3.2 to efficiently instantiate a NIZK proof used in the Escrow algorithm in Fig. 5.4 to compute π_{U}. This proof is for the following relation: $R_{\Psi_2}((y, r_y, r_{\hat{Z}}), (\hat{Z}, X, f_{n,k}, C_y)) = 1$

iff $\hat{Z} = \text{HEC}\textsc{eval}(hecpar, f_{n,k}, X, y; r_{\hat{Z}}) \wedge C_y = \mathsf{Com}(y; r_y)$ where $f_{n,k}$ is the watchlist blueprinting function described at the start of this section.

In Alg. 7, we give the construction of Ψ_2 for HECEVAL. This function calls the proof function for R_f from Sec. 3 on lines 11, 12, and 13 in order to prove correct computation of Z_{id}, Z_{at}, and Z_{nf}. Because of the succinctness of our proof for R_f, the complexity of our proof will be $O(\log(n))$ since we evaluate with a constant number of variables.

Our proof system must have the zero-knowledge and extractability properties needed for the proofs of both blueprint hiding (Def. 11) and user privacy (Def. 12 and 13) for our construction in Fig. 5.4. The zero-knowledge property is standard; for extractability recall that we require both the usual black-box proof of knowledge property, as well as partial straight-line extraction of $\mathfrak{g}(y)$; $\mathfrak{g}$ is some function such that $\mathfrak{g}(y)$, jointly with x is sufficient to compute $f(x, y)$ because there is some efficiently computable function f^* such that $f^*(x, \mathfrak{g}(y)) = f(x, y)$. In order to achieve straight-line extractability of $\mathfrak{g}(y)$, our proof system requires that the prover $\mathfrak{g}$-semi-encrypt y under a public key "in the sky", i.e. a public key that's part of the parameters generated during setup; the knowledge extractor's trapdoor will be the decryption key. To that end, we need a semantically secure public-key $\mathfrak{g}$-semi-encryption scheme ($\Gamma_{sky} = \{\mathsf{KeyGen}_{sky}, \mathsf{Enc}_{sky}, \mathsf{Dec}_{sky}\}$). (Using our notation from Def. 2, the prover retrieves the public key in the sky by querying the setup S_2.)

Algorithm 7 $\mathsf{PoK}_{\Psi_2}^{S_2}(hecpar, f, X, y, r_y, r_{\hat{Z}}) \rightarrow \pi$

```
    parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in[0\ldots n]}); r_{\hat{Z}} = (r_1, r_2, r_3, r_{\mathsf{id}}, r_{\mathsf{attr}})$
 1: $(y_{id}, y_{at}) \leftarrow y; (C_{id}, r_{id}) = \mathsf{Com}(y_{id}); (C_{at}, r_{at}) = \mathsf{Com}(y_{at}); C_y \leftarrow \mathsf{Com}(y; r_y)$
 2: $Z_{id} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{id}; r_{\mathsf{id}}) \oplus (r_1 \odot \boxed{e}); Z_{at} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{at}; r_{\mathsf{attr}}) \oplus (r_2 \odot \boxed{e})$
 3: $Z_{nf} = r_3 \odot \boxed{e}; Z = (Z_{id}, Z_{at}, Z_{nf})$
 4: $\mathsf{pk}_{sky} \leftarrow S_2(1^\lambda); C_{sky} = \mathsf{Enc}_{sky}(\mathsf{pk}_{sky}, y; r_{sky});$
 5: $\pi_{sky} = \mathsf{NIZK}[y, r_y, r_{sky} : C_{sky} = \mathsf{Enc}(\mathsf{pk}_{sky}, y; r_{sky}) \wedge C_y = \mathsf{Com}(y; r_y)]$
 6: $\forall i \in [3], (C_{r_i}, \rho_i) = \mathsf{Com}(r_i)$
 7: $f_{id}(a_0, \ldots, a_n, y_{id}, r_1) = y_{id} + a_0 r_1 y_{id}^0 + a_1 r_1 y_{id}^1 + \ldots + a_n r_1 y_{id}^n$
 8: $f_{at}(a_0, \ldots, a_n, y_{id}, y_{at}, r_2) = y_{at} + a_0 r_2 y_{id}^0 + a_1 r_2 y_{id}^1 + \ldots + a_n r_2 y_{id}^n$
 9: $f_{nf}(a_0, \ldots, a_n, y_{id}, r_3) = a_0 r_3 y_{id}^0 + a_1 r_3 y_{id}^1 + \ldots + a_n r_3 y_{id}^n$
10: $\pi_y = \mathsf{NIZK}[y_{id}, r_{id}, y_{at}, r_{at} : C_{id} = \mathsf{Com}(y_{id}; r_{id}) \wedge C_{at} = \mathsf{Com}(y_{at}; r_{at}) \wedge (y_{id}, y_{at}) = y \wedge C_y = \mathsf{Com}(y; r_y)]$
11: $\pi_{id} = \mathsf{NIZK}[y_{id}, r_{id}, r_1, \rho_{id} : Z_{id} = \mathsf{Enc}_{AH}(f_{id}(a_0, \ldots, a_n, y_{id}, r_1)) \wedge C_{id} = \mathsf{Com}(y_{id}; r_{id}) \wedge C_{r_1} = \mathsf{Com}(r_1; \rho_1)]$
12: $\pi_{at} = \mathsf{NIZK}[y_{id}, r_{id}, y_{at}, r_{at}, r_2, \rho_2 : Z_{at} = \mathsf{Enc}_{AH}(f_{at}(a_0, \ldots, a_n, y_{id}, y_{at}, r_2)) \wedge C_{id} = \mathsf{Com}(y_{id}; r_{id}) \wedge C_{at} = \mathsf{Com}(y_{at}; r_{at}) \wedge C_{r_2} = \mathsf{Com}(r_2; \rho_2)]$
13: $\pi_{nf} = \mathsf{NIZK}[y_{id}, r_3 : Z_{nf} = \mathsf{Enc}_{AH}(f_{nf}(a_0, \ldots, a_n, y_{id}, r_3)) \wedge C_{id} = \mathsf{Com}(y_{id}; r_{id}) \wedge C_{r_1} = \mathsf{Com}(r_1; \rho_1)]$
14: return $(\pi_{id}, \pi_{at}, \pi_{nf}, \pi_{sky}, C_{sky}, \{C_{r_i}\}_{i\in[3]}, \pi_y)$
```

We present the corresponding verification functions for PoK_{Ψ_2} ($V_{\Psi_2}^{S_2}$) in Alg. 8

Theorem 22. *Our scheme in Alg. 7 is complete and ZK (Def. 1).*

Algorithm 8 $\mathsf{V}_{\Psi_2}^{S_2}(hecpar, f, X, C_y, Z, \pi) \to \{0,1\}$

```
parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in[0\ldots n]})$;
parse $\pi = (C, C_{id}, \pi_{rec}, \pi_{\hat{Z}}, \pi_{sky}, C_{sky}, C_{Y_{\mathsf{id}}}, C_{Y_{\mathsf{attr}}})$
$\mathsf{pk}_{sky} \leftarrow S_2(1^\lambda)$;
Verify $\pi_{sky}$
Verify $\pi_{rec}$ using $\mathsf{V}_P^*$
Verify $\pi_{\hat{Z}}$
If any proof failed to verify, return 0, otherwise return 1
```

Theorem 23 (g^*-BB-PSL for Ψ_2). *If PoK_P^* is a BB NIZK for the relation R_P (where R_P is defined as $R_P((C, C_{y_{id}}, X, n), (O, O_{y_{id}}, y_{id})) = 1$ iff $C = \mathsf{Com}_{AH}(\mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, \bigoplus_{i=0}^{n}(\boxed{a_i} \odot y_{id}^i); O)) \wedge C_{y_{id}} = \mathsf{Com}(y, O_{y_{id}}))$ and if $\Gamma_{sky} = \{\mathsf{KeyGen}_{sky}, \mathsf{Enc}_{sky}, \mathsf{Dec}_{sky}\}$ is a semantically secure $\mathfrak{g}$-semi-encryption scheme, our Ψ_2 proof is a g^*-BB-PSL protocol, where $g^*(y, r_{\hat{Z}}) = \mathfrak{g}(y)$.*

We prove Thms. 22 and 23 next.

Proof of Thm. 22 (Completeness and ZK). Our scheme is correct by inspection. We see that because the proof only consists of commitments and zero-knowledge proofs, it is zero-knowledge as well.

Proof of Thm. 23 (BB-PSL). We assume in this theorem that we can extract a witness for the relation R_f in a black-box way (Thm. 2) by instantiating the NIZKs in Alg. 7 with the proof function for R_f in Alg. 6 (Appendix 3.3). Thus, we know that the ciphertext (C_{sky}) containing $g(y)$ is correct, and thus, our straight line extractor (defined in 2) can extract $g(y) = g^*(y, r_{\hat{Z}})$ by decrypting this ciphertext. We can also use our homomorphic additive encryption along with our ciphertext commitment schemes to construct proofs for Ψ_1 and Ψ_3 using similar techniques to that of Ψ_2.

5.4 Multi-attribute HEC Scheme

In this section, we provide a HEC scheme that satisfies Def. 8 and supports multiple attributes. Including multiple attributes increases the size of values that can be escrowed. In the case of ElGamal, this becomes $\mathsf{poly}(\lambda)^\ell$ and in the case of Camenisch-Shoup, this becomes $(\mathbb{Z}_n)^\ell$. Notice in the case of ElGamal, this allows us to efficiently encrypt and decrypt public keys. This is still not as efficient as in the case of Camenisch-Shoup as the key has to be broken up into logarithmically sized chunks in the case of ElGamal. This makes proving properties of keys escrowed with the ElGamal scheme inefficient while with Camenisch-Shoup, the key can be encrypted while retaining more algebraic structure.

This allows for our Camenisch-Shoup scheme to potentially achieve more efficient proofs for extended properties such as retrospective blueprints.

Our function family for multi-attributes is $\{f_{n,k,\ell}\}_{n,k,\ell\in\mathbb{Z}}$, where n is the length of the auditor's list $\mathbf{x} = \{x_1, \ldots, x_n\}$ and k is the bit length of each user attribute y_i^{attr}, where the user's input consists of the user's identifier y_{id} and ℓ attributes: $y = (y_{id}, y_1^{attr}, \ldots, y_\ell^{attr})$. $f_{n,k,\ell}$ is defined as follows:

$$f_{n,k,\ell}(\mathbf{x}, y) = \begin{cases} y & y_{id} \in \mathbf{x} \\ \emptyset & \text{otherwise} \end{cases} \tag{2}$$

We construct a HEC scheme for this function in Fig. 5.6. In our previous construction in Alg. 7 we have a commitment to E which is the commitment C. Remember, C is a commitment to the auditor's polynomial $p(\chi)$ evaluated at the users identity y_{id}. Thus, E will be an encryption of zero if the user is on the watchlist ($y_{id} \in x$). In Fig. 5.6, we then scale C with the different randomization factors ($\{r_{E,i}\}_{i\in[\ell]}$) yielding the new commitments: $\{C_i\}_{i\in[\ell]}$ to these scaled encryptions. If the user is not on the watchlist, these ℓ commitments now encrypt random values. We then homomorphically add each scaled encryption C_i with the encryptions of attributes $\{Y_i\}_{i\in[\ell]}$ to ensure that they can only be decrypted if the user is on the watchlist. We need to use separate randomization scalars for each attribute because we will reveal each encryption. If the encryptions used the same random scalar, the adversary could homomorphically remove them by dividing one encryption by the other. Using independent randomness ensures that each of these commitments are scaled by a random factor and are independent of one another. We still need to include an encryption of E scaled by a random factor $Z_{nf} = r_{nf} \odot E$ to ensure non-framing. Because we only compute one commitment to E, when modifying the ψ_2 proof from Sec. 3 to work for multiple attributes, we only need to perform the proof of correct encryption of E once. Then, we simply use our auxiliary proofs of commitments to ciphertexts to prove that the rest of the encryptions of attributes are correct, without needing to reprove the commitment to E. This makes our ψ_2 scheme's communication size equal to $O(\log(x) + \ell)$ for multiple attributes.

6 Acknowledgements

Anna Lysyanskaya and Scott Griffy were supported by NSF Grants 2312241, 2154170, and 2247305 as well as the Peter G. Peterson Foundation and the Ethereum Foundation. Markulf Kohlweiss and Meghna Sengupta were supported by Input Output (iohk.io) through their funding of the University of Edinburgh ZK Lab. We'd also like to acknowledge Victor Youdom Kemmoe for his helpful discussions.

References

ACC$^+$22. Thomas Attema, Ignacio Cascudo, Ronald Cramer, Ivan Damgård, and Daniel Escudero. Vector commitments over rings and compressed Σ-protocols. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 173–202. Springer, Cham, November 2022.

ATD16. Aydin Abadi, Sotirios Terzis, and Changyu Dong. VD-PSI: Verifiable delegated private set intersection on outsourced private datasets. In Jens

Fig. 5.6: Multi-attribute HEC functions

```
$\text{HECEVAL}(hecpar, f_{n,k,\ell}, X, y; r_Z)$
parse $X = (\mathsf{pk}_{AH}, A_1, ..., A_{n+1})$,
    $y = (y_{id}, y_1^{at}, ..., y_\ell^{at})$,
    $r_Z = (\{r_{E,1}, ..., r_{E,\ell}\}, \{r_1, ..., r_\ell\}, r_{nf})$
$E \leftarrow \bigoplus_{i=1}^{n+1} (A_i \odot y_{id}^i)$
$\forall i \in [\ell] : Y_i \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, y_i^{at}; r_i)$
$\forall i \in [\ell] : Z_i \leftarrow ((r_{E,i} \odot E) \oplus Y_i$
$Z_{nf} = r_{nf} \odot E$
return $Z = (\{Z_1, ..., Z_\ell\}, Z_{nf})$
```

```
$\text{HECDEC}(hecpar, d, Z)$
parse $d = (\mathsf{sk}_E, f_{n,k,\ell}, x)$,
    $Z = (\{Z_1, ..., Z_\ell\}, Z_{nf})$
$\forall i \in [\ell] : y_{\mathfrak{g},i}^{at} \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_i)$
$y_g \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{nf})$
if $y_g \neq \mathfrak{g}(0)$
    return $\emptyset$
for $y_{id} \in x$
    if $\mathfrak{g}(y_{id}) = y_g$
        return $(y_{id}, y_1^{at}, ..., y_\ell^{at})$
            where $\forall i \in [\ell] : y_i^{at} \in domain_{f,y}$
            $\wedge\ \mathfrak{g}(y_i^{at}) = y_{\mathfrak{g},i}^{at}$
return $\emptyset$
```

Grossklags and Bart Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 149–168. Springer, Berlin, Heidelberg, February 2016.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BCF$^+$. Daniel Benarroch, Matteo Campanell, Dario Fiore, Jihye Kim, Jiwon Lee, Hyunok Oh, and Anaïs Querol. Proposal: Commit-and-prove zero-knowledge proof systems and extensions. `https://docs.zkproof.org/pages/standards/accepted-workshop4/proposal-commit.pdf`.

BCL04. Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols Workshop*, volume 3957 of *Lecture Notes in Computer Science*, pages 20–42. Springer, 2004.

BCM05. Endre Bangerter, Jan Camenisch, and Ueli Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 154–171. Springer, Berlin, Heidelberg, January 2005.

BdMW16. Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 62–89. Springer, Berlin, Heidelberg, August 2016.

BFK$^+$24. Alexander R. Block, Zhiyong Fang, Jonathan Katz, Justin Thaler, Hendrik Waldner, and Yupeng Zhang. Field-agnostic SNARKs from expand-accumulate codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 276–307. Springer, Cham, August 2024.

BG13. Stephanie Bayer and Jens Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In Thomas Johansson and

Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 646–663. Springer, Berlin, Heidelberg, May 2013.

BGJP23. James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 35–66. Springer, Cham, April 2023.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

BHV$^+$23. Rishabh Bhadauria, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Wenxuan Wu, and Yupeng Zhang. Private polynomial commitments and applications to MPC. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 127–158. Springer, Cham, May 2023.

BL13. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013.

BMM$^+$21. Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97. Springer, Cham, December 2021.

Boy. Dennis Boyle. The problem of "parallel construction" in criminal investigations. `https://www.boylejasari.com/the-problem-of-parallel-construction-in-criminal-investigations/`. Accessed: 2024-02-13.

BSZ05. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Berlin, Heidelberg, February 2005.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

Cam97. Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 465–479. Springer, Berlin, Heidelberg, May 1997.

CBBZ23. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023.

CDN01. Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Berlin, Heidelberg, May 2001.

CFN90. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, New York, August 1990.

Cha90. David Chaum. Showing credentials without identification transferring signatures between unconditionally unlinkable pseudonyms. In Jennifer Seberry and Josef Pieprzyk, editors, *AUSCRYPT'90*, volume 453 of *LNCS*, pages 246–264. Springer, Berlin, Heidelberg, January 1990.

CHK$^+$06. Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 201–210. ACM Press, October / November 2006.

CHL05. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Berlin, Heidelberg, May 2005.

CHL06. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In Roberto De Prisco and Moti Yung, editors, *Proceedings of the 5th International Conference on Security and Cryptography for Networks (SCN)*, volume 4116 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2006.

CL01. Jan Camenisch and Anna Lysyanskaya. An identity escrow scheme with appointed verifiers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 388–407. Springer, Berlin, Heidelberg, August 2001.

CL02. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Berlin, Heidelberg, August 2002.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Berlin, Heidelberg, August 2004.

CM20. Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Cham, August 2020.

CMdG$^+$21. Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1135–1150. ACM Press, November 2021.

CRR21. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Cham.

CS03. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Berlin, Heidelberg, August 2003.

CV02. Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 21–30. ACM Press, November 2002.

DD22. Nico Döttling and Jesko Dujmovic. Maliciously circuit-private FHE from information-theoretic principles. Cryptology ePrint Archive, Report 2022/495, 2022.

DF02. Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, 2002.

FNP04. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Berlin, Heidelberg, May 2004.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of STOC 2009*, pages 169–178, 2009.

GKL21. Matthew Green, Gabriel Kaptchuk, and Gijs Van Laer. Abuse resistant law enforcement access systems. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 553–583. Springer, Cham, October 2021.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

GLS$^+$23. Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226. Springer, Cham, August 2023.

GM82. Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

Gov22. United States Government. Technical design choices for a U.S. central bank digital currency system. `https://www.whitehouse.gov/wp-content/uploads/2022/09/09-2022-Technical-Design-Choices-US-CBDC-System.pdf`, 2022.

GPR$^+$21. Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 395–425, Virtual Event, August 2021. Springer, Cham.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Berlin, Heidelberg, August 2013.

HHKP23. Charlotte Hoffmann, Pavel Hubácek, Chethan Kamath, and Krzysztof Pietrzak. Certifying giant nonprimes. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 530–553. Springer, Cham, May 2023.

HS21. Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2004–2023. ACM Press, November 2021.

HSS23. Julia Hesse, Nitin Singh, and Alessandro Sorniotti. How to bind anonymous credentials to humans. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 3047–3064. USENIX Association, 2023.

IR90. K. Ireland and M.I. Rosen. *A Classical Introduction to Modern Number Theory*. Graduate Texts in Mathematics. Springer, 1990.

JWP22. Yuting Jiang, Jianghong Wei, and Jing Pan. Publicly verifiable private set intersection from homomorphic encryption. In *Security and Privacy in Social Networks and Big Data - 8th International Symposium, SocialSec 2022, Xi'an, China, October 16-18, 2022, Proceedings*, volume 1663 of *Communications in Computer and Information Science*, pages 117–137. Springer, 2022.

KKS22. Aggelos Kiayias, Markulf Kohlweiss, and Amirreza Sarencheh. PEReDi: Privacy-enhanced, regulated and distributed central bank digital currencies. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1739–1752. ACM Press, November 2022.

KL20. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press, 2020.

KLN23. Markulf Kohlweiss, Anna Lysyanskaya, and An Nguyen. Privacy-preserving blueprints. In Carmit Hazay and Martijn Stam, editors, *EURO-*

CRYPT 2023, Part II, volume 14005 of *LNCS*, pages 594–625. Springer, Cham, April 2023.

KM15. Markulf Kohlweiss and Ian Miers. Accountable metadata-hiding escrow: A group signature case study. *PoPETs*, 2015(2):206–221, April 2015.

KMRS14. Seny Kamara, Payman Mohassel, Mariana Raykova, and Seyed Saeed Sadeghian. Scaling private set intersection to billion-element sets. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 195–215. Springer, Berlin, Heidelberg, March 2014.

LFKN92. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

LRSW99. Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *LNCS*, 1999.

Lys02. Anna Lysyanskaya. *Signature schemes and applications to cryptographic protocol design*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.

OPP14. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 536–553. Springer, Berlin, Heidelberg, August 2014.

PEB21. Charlotte Peale, Saba Eskandarian, and Dan Boneh. Secure complaint-enabled source-tracking for encrypted messaging. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1484–1506. ACM Press, November 2021.

Pie19. Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, January 2019.

RR22. Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2505–2517. ACM Press, November 2022.

RS21. Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Cham, October 2021.

RWGM23. Michael Rosenberg, Jacob D. White, Christina Garman, and Ian Miers. zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. In *2023 IEEE Symposium on Security and Privacy*, pages 790–808. IEEE Computer Society Press, May 2023.

Sch80. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, oct 1980.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

Sta23. Jay Stanley. Paths toward an acceptable public digital currency. ACLU White Paper, 2023. `https://www.aclu.org/wp-content/uploads/legal-documents/cbdc_white_paper_-_0882_0.pdf`.

TBA$^+$22. Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. UTT: Decentralized ecash with accountable privacy. Cryptology ePrint Archive, Report 2022/452, 2022.

TZ23. Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721. Springer, Cham, April 2023.

WHV24. Ruihan Wang, Carmit Hazay, and Muthuramakrishnan Venkitasubramaniam. Ligetron: Lightweight scalable end-to-end zero-knowledge proofs post-quantum zk-snarks on a browser. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 1760–1776. IEEE, 2024.

XZZ$^+$19. Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Cham, August 2019.

ZLW$^+$21. Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 159–177. ACM Press, November 2021.

A Discussion on Non-frameability vs. Deniability

Non-frameability is a desirable feature, but it is fundamentally at odds with deniability. In a deniable system, data may be authenticated at the moment when it is received, but this authentication information quickly becomes useless. This way, Alice cannot use her authenticated transcript from a conversation with Bob to prove to a third party what Bob did or did not say. Typically, to define deniability, one would explicitly give Alice an algorithm to “frame” Bob, i.e., to authenticate any transcript on his behalf. That way, a real transcript will not be any more believable than a bogus one, and Bob may convincingly deny ever talking to Alice. Deniability of a ciphertext’s origin, for example, is valuable for encrypted messaging systems, especially when users might face coercion, and in other contexts [PEB21,GKL21]. Kohlweiss and Miers [KM15] attempted to address the question whether the properties of non-frameability and deniability can both be achieved together and reached disappointing conclusions, as did Bartusek et al. [BGJP23].

In a system like PPBs, deniability would allow for an efficient algorithm for creating a convincing-looking escrow that would decrypt to any value the algorithm takes as input. A deniable PPB would give an auditor a meaningful ability to monitor the system only so long as it trusts the escrow recipients that they did not make up the escrows but in fact collected them as part of a legitimate transaction. It may be an interesting direction to pursue in future work if well-motivated in practice.

In this work, however, similarly to Bartusek et al. [BGJP23], we prioritized non-frameability and thus abandoned deniability, because, in our view, systems like ours that are designed to detect illegal activity require not only the ability to identify a watchlisted user’s actions but also the means to only convince a judge of these actions if they have in fact taken place. It is more important to us that innocent users cannot be credibly accused of wrongdoing than that perpetrators be able to deny theirs activities.

B Motivation for BB-PSL

For concreteness, let us imagine that π is the NIZK we get by running a Σ-protocol for a proof of knowledge, and making it non-interactive by replacing the message from the verifier with the output of the random oracle. The prover's side of the Σ-protocol consist's two algorithms, P_1 and P_2. $P_1(\mathsf{pk}, m, r; R)$ generates the first message, a, of the proof of knowledge of how $c = \mathsf{Enc}(\mathsf{pk}, m, r)$ was computed using random coins R; $P_2(\mathsf{pk}, m, r, e; R)$ generates the prover's response, z, to the challenge e using the same randomness. The verifier's part of the Σ protocol is just the algorithm $V(\mathsf{pk}, c, a, e, z)$. It is well-known that, in the random-oracle model, the following proof system is black-box simulation-extractable: the prover computes $a = P_1(\mathsf{pk}, m, r; R)$, $e = H(\mathsf{pk}, c, a)$, and $z = P_2(\mathsf{pk}, m, r, e; R)$ and outputs the proof $\pi = (a, z)$. To verify π, the verifier computes $e = H(\mathsf{pk}, c, a)$ and runs $V(\mathsf{pk}, c, a, e, z)$.

However, when we plug this proof system into the attempted construction above of a CCA-secure cryptosystem from a semantically secure one, we don't (easily) get a proof of CCA security. This is because the adversary can interleave his decryption queries and his random-oracle queries in such a way that he will force the security reduction to run in exponential time in the number q of queries. In order to respond to the i^{th} decryption query (c_i, π_i) where $\pi_i = (a_i, z_i)$, the reduction needs to rewind the adversary to the point in time where the adversary queried the random oracle to get $e_i = H(\mathsf{pk}, c_i, a_i)$. By first issuing all the random-oracle queried in reverse order, i.e. obtaining $e_q = H(\mathsf{pk}, c_q, a_q)$, and then $e_{q-1}, \ldots, e_1$ before issuing any decryption queries at all, and then querying for the decryptions of $(c_1, \pi_1), \ldots, (c_q, \pi_q)$, the adversary will ensure that the reduction will need to rewind $O(2^q)$ times [12]. This is because each time the reduction rewinds the adversary, they also need to rewind for each previous query to ensure the adversary receives the correct decryptions to run normally. Thus, each decryption query doubles the number of required rewinds.

There are two ways of fixing this problem. One is to use a straight-line extractable proof system that does not need to rewind at all; but that can be inefficient. The other way to fix it (implicitly in the spirit of Shoup and Gennaro) is to not require the straight-line extraction of the entire witness: the reduction does not need both m and r to proceed, just the message m alone is sufficient. The fact that, with rewinding, it is possible to extract the entire witness is still crucial since it guarantees that the adversary's interaction with the security reduction results in exactly the same view as in its interaction with the decryption oracle: if not, then a separate reduction would break the soundness of the proof system.

C Full Definitions for Privacy Preserving f-Blueprint Schemes

A blueprint scheme has three parties - an auditor, a set of users and a set of recipients. It is defined as follows:

Definition 10. *For a non-interactive commitment scheme* $(\mathsf{CSetup}, \mathsf{Com})$*, an f-blueprint scheme consists of the following probabilistic polynomial time algorithms:*

[12] The adversary must also base the first message of each Σ-protocol on the output of the random oracle from the last query to ensure rewinding is impossible.

$\mathsf{Setup}(1^\lambda, cpar) \rightarrow \Lambda$: This algorithm takes as input the security parameter 1^λ and the commitment parameters $cpar$ output by $\mathsf{CSetup}(1^\lambda)$. It outputs the public parameters Λ which includes 1^λ and $cpar$. For the remainder of the paper, Com is used synonymously with Com_{cpar} to reduce notational overhead.

$\mathsf{KeyGen}(\Lambda, x, r_x) \rightarrow (\mathsf{pk_A}, \mathsf{sk_A})$: The key generation algorithm for auditor A takes 1^λ, Λ, and commitment value and opening (x, r_x) as input, and outputs the key pair $(\mathsf{pk_A}, \mathsf{sk_A})$. The values (x, r_x) define a commitment C_x.

$\mathsf{VerPK}(\Lambda, \mathsf{pk_A}, C_x) \rightarrow 1$ or 0: This is the algorithm that, on input the auditor's public key $\mathsf{pk_A}$ and a commitment C_x, verifies that the auditor's public key was computed correctly for the commitment C_x.

$\mathsf{Escrow}(\Lambda, \mathsf{pk_A}, y, r_y) \rightarrow Z$: This algorithm takes Λ, $\mathsf{pk_A}$, and commitment value and opening (y, r_y) as input and outputs an escrow Z for commitment $C = \mathsf{Com}(y; r_y)$.

$\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z) \rightarrow 1$ or 0: This algorithm takes the auditor's public key $\mathsf{pk_A}$, a commitment C, and an escrow Z as input and verifies that the escrow was computed correctly for the commitment C.

$\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z) \rightarrow f(x, y)$ or $\perp$: This algorithm takes the auditor's secret key $\mathsf{sk_A}$, a commitment C and an escrow Z as input. It decrypts the escrow and returns the output $f(x, y)$ if C is a commitment to y and $\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z) = 1$.

[KLN23] also defines a *secure* f-blueprint scheme as one that possesses the following properties:

Correctness of VerPK **and** $\mathsf{VerEscrow}$: For honestly generated values $(cpar, \mathsf{pk_A}, C_x, C, Z)$, the algorithms $\mathsf{VerEscrow}$ and VerPK should accept with probability 1.

Correctness of Dec: For honestly generated values $(cpar, \mathsf{pk_A}, \mathsf{sk_A}, C, Z)$, $\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z) = f(x, y)$ should hold with overwhelming probability .

Soundness: For all PPT adversaries $\mathcal{A}$ involved in the experiment in Fig. C.1, there exists a negligible function ν such that:

$$\mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{Adv},\mathsf{Blu}} = \Pr\left[\mathsf{Sound}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda) = 1\right] = \nu(\lambda)$$

Definition 11 (Blueprint Hiding). *The blueprint-hiding property makes sure that* $\mathsf{pk_A}$ *just reveals that* x *is a valid first argument to* f*. Otherwise,* x *is hidden even from an adversary who (1) may already know a lot of information about* x *a-priori; and (2) has oracle access to* $\mathsf{Dec}(\Lambda, \mathsf{sk_A}, \cdot, \cdot)$.

This is formalized by requiring that there exist a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimKeygen}, \mathsf{SimDecrypt})$ *such that for any* PPT *adversary the following two games are indistinguishable:*

1. **Real Game**: Λ is chosen honestly, the public key $\mathsf{pk_A}$ is computed correctly for adversarially chosen x, r_x, and the adversary's decryption queries (C, Z) are answered with $\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z)$.

```
$\mathsf{Sound}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$
$cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
$\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
$x, r_x \leftarrow \mathsf{Adv}(1^\lambda, \Lambda)$
$(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_x)$
$(C, y, r_y, Z) \leftarrow \mathsf{Adv}(\mathsf{pk_A})$
$\textbf{return}\ [C = \mathsf{Com}(y; r_y) \wedge$
    $\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z) \wedge \mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z) \neq f(x, y)]$
```

Fig. C.1: Experiments $\mathsf{Sound}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$

2. **Ideal Game**: Λ is computed using $\mathsf{SimSetup}$, the public key $\mathsf{pk_A}$ is computed using $\mathsf{SimKeygen}$ independently of x (although with access to the commitment C_{A}), and the adversary's decryption query Z_i is answered by first running $\mathsf{SimDecrypt}$ to obtain enough information about the user's data y_i to be able to compute $f(x, y_i)$. "Enough information" means that for an efficiently computable f^* and a function g such that $f(x, y) = f^*(x, g(y))$ for all possible inputs (x, y), $\mathsf{SimDecrypt}$ obtains $y_i^* = g(y_i)$.

Formally, for all probabilistic poly-time adversaries Adv involved in the game described in Fig. C.2, the advantage function satisfies:

$$\mathsf{Adv}^{\mathsf{BH}}_{\mathsf{Adv},\mathsf{Sim}} = \left| \Pr\left[\mathsf{BHreal}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda) = 0 \right] - \Pr\left[\mathsf{BHideal}^{\mathsf{Adv}}_{\mathsf{Blu},\mathsf{Sim}}(\lambda) = 0 \right] \right| = \nu(\lambda)$$

for some negligible ν.

Definition 12 (Privacy against Dishonest Auditor). *There exists a simulator such that the adversary's views in the following two games are indistinguishable:*

1. ***Real Game**: The adversary generates the public key and the data x corresponding to this public key, honest users follow the* Escrow *protocol using adversarial inputs and openings.*
2. ***Privacy-Preserving Game**: The adversary generates the public key and the data x corresponding to this public key. Next, for adversarially chosen inputs and openings, the users run a simulator algorithm that depends only on the commitment and $f(x, y)$ but is independent of the commitment openings.*

More formally, there exists algorithms $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimEscrow})$ *such that, for any* PPT *adversary* Adv *involved in the game described in Fig. C.3, the following equation holds for some negligible function ν:*

$$\mathsf{Adv}^{\mathsf{PADA}}_{\mathsf{Adv},\mathsf{Blu},\mathsf{Sim}} = \left| \Pr\left[\mathsf{PADA}^{\mathsf{Adv},0}_{\mathsf{Blu},\mathsf{Sim}}(\lambda) = 1 \right] - \Pr\left[\mathsf{PADA}^{\mathsf{Adv},1}_{\mathsf{Blu},\mathsf{Sim}}(\lambda) = 1 \right] \right| = \nu(\lambda)$$

```
$\mathsf{BHreal}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$
    $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
    $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
    $(x, r_x, \mathsf{st}_{\mathsf{Adv}}) \leftarrow \mathsf{Adv}(1^\lambda, \Lambda)$

    $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_x)$
    return $\mathsf{Adv}^{\mathcal{O}_0(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}, \cdot, \cdot)}(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}_{\mathsf{Adv}})$

$\mathsf{BHideal}^{\mathsf{Adv}}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$
    $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
    $(\Lambda, \mathsf{st}) \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
    $(x, r_x, \mathsf{st}_{\mathsf{Adv}}) \leftarrow \mathsf{Adv}(1^\lambda, \Lambda)$
    $dsim \leftarrow (|x|, \mathsf{Com}(x; r_x))$
    $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}) \leftarrow \mathsf{SimKeygen}(1^\lambda, \mathsf{st}, dsim)$
    return $\mathsf{Adv}^{\mathcal{O}_1(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}, x, \cdot, \cdot)}(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}_{\mathsf{Adv}})$

$\mathcal{O}_0(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}, C, Z)$
    if $\neg\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C, Z)$
        return $\perp$
    return $\mathsf{Dec}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z)$

$\mathcal{O}_1(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}, x, C, Z)$
    if $\neg\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C, Z)$
        return $\perp$
    $y^* \leftarrow \mathsf{SimDecrypt}(\mathsf{st}, C, Z)$
    return $f(x, y) = f^*(x, y^*)$
```

Fig. C.2: Experiments $\mathsf{BHreal}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$ and $\mathsf{BHideal}^{\mathsf{Adv}}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$

```
$\mathsf{PADA}^{\mathsf{Adv},b}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$
    $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
    $\Lambda_0 \leftarrow \mathsf{Setup}(1^\lambda, cpar); (\Lambda_1, \mathsf{st}) \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
    $(x, r_{\mathsf{A}}, \mathsf{pk}_{\mathsf{A}}, \mathsf{st}_{\mathsf{Adv}}) \leftarrow \mathsf{Adv}(1^\lambda, \Lambda_b)$
    if $\mathsf{VerPK}(\Lambda_b, \mathsf{pk}_{\mathsf{A}}, \mathsf{Com}(x; r_{\mathsf{A}})) = 0$ : return $\perp$
    return $\mathsf{Adv}^{\mathcal{O}_b(\cdot,\cdot)}(\mathsf{st}_{\mathsf{Adv}})$

$\mathcal{O}_0(y, r_y)$
    return $\mathsf{Escrow}(\Lambda_0, pk_{\mathsf{A}}, y, r_y)$

$\mathcal{O}_1(y, r_y)$
    return $\mathsf{SimEscrow}(\mathsf{st}, \Lambda_1, \mathsf{pk}_{\mathsf{A}}, \mathsf{Com}(y; r_y),$
                            $f(x, y))$
```

Fig. C.3: Game $\mathsf{PADA}^{\mathsf{Adv},b}_{\mathsf{Blu}}(\lambda)$

Definition 13 (Privacy with Honest Auditor). *There exists a simulator* Sim *such that the adversary's views in the following two games are indistinguishable:*

1. ***Real Game****: The honest auditor generates the public key on input x provided by the adversary, and honest users follow the* Escrow *protocol on input adversarially chosen openings.*
2. ***Privacy-Preserving Game****: The honest auditor generates the public key on input x provided by the adversary. On input adversary-generated com-*

mitments and openings, the users run a simulator that is independent of y (although with access to the commitment C_y) to form their escrows.

In both of these games, the adversary has oracle access to the decryption algorithm.

We formalize these two games in Fig. C.4. We require that there exists a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimEscrow})$ *such that, for any* PPT *adversary* Adv *involved in the game described in the figure, the following equation holds:*

$$\mathsf{Adv}^{\mathsf{PWHA}}_{\mathsf{Blu},\mathsf{Sim}} = \left|\Pr\left[\mathsf{PWHA}^{\mathsf{Adv},0}_{\mathsf{Blu},\mathsf{Sim}}(\lambda) = 0\right] - \Pr\left[\mathsf{PWHA}^{\mathsf{Adv},1}_{\mathsf{Blu},\mathsf{Sim}}(\lambda) = 0\right]\right| = \nu(\lambda)$$

for some negligible function ν.

```
$\mathsf{PWHA}^{\mathsf{Adv},b}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$
    $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
    $\Lambda_0 \leftarrow \mathsf{Setup}(1^\lambda, cpar); \Lambda_1 \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
    $M \leftarrow [\,]$
    $x, r_x \leftarrow \mathsf{Adv}(1^\lambda, \Lambda_b)$
    $(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow \mathsf{KeyGen}(\Lambda_b, x, r_x)$
    return $\mathsf{Adv}^{\mathcal{O}^{\mathsf{Escrow}}_b(\cdot,\cdot), \mathcal{O}^{\mathsf{Dec}}(\Lambda_b, \mathsf{sk_A}, \cdot, \cdot)}(\mathsf{pk_A})$

$\mathcal{O}^{\mathsf{Escrow}}_0(y, r_y)$
    return $\mathsf{Escrow}(\Lambda_0, \mathsf{pk_A}, y, r_y)$

$\mathcal{O}^{\mathsf{Escrow}}_1(y, r_y)$
    $C = \mathsf{Com}(y; r_y)$
    $Z \leftarrow \mathsf{SimEscrow}(\mathsf{st}, \Lambda_1, \mathsf{pk_A}, C)$
    $M[C, Z] \leftarrow f(x, y)$
    return $Z$

$\mathcal{O}^{\mathsf{Dec}}(\Lambda_1, \mathsf{sk_A}, C, Z)$
    if $M[C, Z]$ is defined return $M[C, Z]$
    return $\mathsf{Dec}(\Lambda_1, \mathsf{sk_A}, C, Z)$
```

Fig. C.4: Game $\mathsf{PWHA}^{\mathsf{Adv},b}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$

D Number-Theoretic Building Blocks

D.1 Construction of Equality of (Linear) DL Representations Proof in Prime Order Groups

Using known techniques, e.g. KLM from which we took the following description, we can construct the protocol in Def. 3 in cyclic groups of prime order where the DDH and CDH assumptions are hard. We do so in Def. 14.

Definition 14 (Σ-protocol for proof of equality of discrete logarithm representations cyclic groups of prime order). *Let $R_{\textsf{eqrep-}p}$ be the following relation: $R_{\textsf{eqrep-}p}(\mathbb{x}, \mathbb{w})$ accepts if $\mathbb{x} = (\mathcal{G}, \{x_i, \{g_{i,1}, \dots, g_{i,m}\}\}_{i=1}^k)$ where $\mathcal{G}$ is the description of a group of order q, and all the x_is and $g_{i,j}$s are elements of $\mathcal{G}$, and witness $\mathbb{w} = \{w_j\}_{j=1}^m$ such that $x_i = \prod_{j=1}^m g_{i,j}^{w_j}$.*

P→V *On input the $(\mathbb{x}, \mathbb{w}) \in R_{\textsf{eqrep-}p}$, the Prover chooses $e_j \leftarrow \mathbb{Z}_q$ for $1 \leq j \leq m$ and computes $d_i = \prod_{j=1}^m g_{i,j}^{e_j}$ for $1 \leq i \leq k$. Finally, the Prover sends to the Verifier the values $\mathsf{com} = (d_1, \dots, d_n)$.*

P←V *On input $\mathbb{x}$ and com, the Verifier responds with a challenge $\mathsf{chal} = c$ for $c \leftarrow \mathbb{Z}_q$.*

P→V *The Prover receives $\mathsf{chal} = c$ and computes $s_i = e_i + cw_i \bmod q$ for $1 \leq i \leq m$, and sends $\mathsf{res} = (s_1, \dots, s_m)$ to the Verifier.*

Verification *The Verifier accepts if for all $1 \leq i \leq n$, $d_i x_i^c = \prod_{j=1}^m g_{i,j}^{s_j}$; rejects otherwise.*

Simulation *On input $\mathbb{x}$ and $\mathsf{chal} = c$, the simulator chooses $s_j \leftarrow \mathbb{Z}_q$ for $1 \leq j \leq m$, and sets $d_i = (\prod_{j=1}^m g_{i,j}^{s_j})/x_i^c$ for $1 \leq i \leq k$. He then sets $\mathsf{com} = (d_1, \dots, d_n)$ and $\mathsf{res} = (s_1, \dots, s_m)$.*

Extraction *On input two accepting transcripts for the same $\mathsf{com} = (d_1, \dots, d_n)$, namely $\mathsf{chal} = c$, $\mathsf{res} = (s_1, \dots, s_m)$, and $\mathsf{chal}' = c'$, $\mathsf{res}' = (s'_1, \dots, s'_m)$, output $w_j = (s_j - s'_j)/(c - c') \bmod q$ for $1 \leq j \leq m$.*

D.2 Useful Lemmas for Composite-Order Groups

Lemma 6. $(n+1) \in QR_{n^2}$

Proof of Lemma 6. In Ireland and Rosen's textbook [IR90] Proposition 5.1.1 gives us that an element, a, in $\mathbb{Z}_{n^2}$ if a quadratic residue iff $a^{(p-1)/2} = 1(\mod p)$ and $a^{(q-1)/2} = 1(\mod q)$. We can see that $(n+1)^{(p-1)/2} = \sum_{i=0}^{(p-1)/2} 1^{(p-1)/2-i} \cdot$ $n^{(p-1)/2-i} = 1 + kn$ for some k. Since n is divisible by both p and q, this value is simply 1 mod p and q. Thus, $(n+1)$ is in QR_{n^2}.

Lemma 7. $(-1) \in \mathsf{QNR}_{n^2}$ *for RSA modulus, n.*

Proof of Lemma 7. Using Proposition 5.1.1 from Ireland and Rosen's textbook [IR90] again we see that $(-1)^{(p-1)/2} \mod p$ is equal to $(-1)^{(4k+2)/2} \mod p$ since we are working with primes that are equal to $3 \mod 4$. Thus, this equals $(-1)^{2k+1} \mod p$. Note that $2k+1$ is odd and thus this equals $(-1) \mod p$ thus failing the criteria in Proposition 5.1.1 and thus $(-1) \in \mathsf{QNR}_{n^2}$.

Lemma 8 (Any element to the 2-nd power likely generates QR_{n^2}). *Formally, no PPT algorithm can produce an element a such that $\langle a^2 \rangle \neq QR_{n^2}$. As a corollary, we know that sampling a random element in QR_{n^2} or squaring a random element in $\mathbb{Z}_{n^2}$ results in a generator of QR_{n^2}.*

Proof of Lemma 8. QR_{n^2} is cyclic and thus every element in QR_{n^2} can be represented as g^i for some g. We see that any g^i doesn't generate QR_{n^2} when $i|\#QR_{n^2}$. The order of QR_{n^2} is $pqp'q'$ and thus, this only occurs when i is a multiple of p, q, p', q'. Thus, there are at most $pqp' + pqq' + pp'q' + qp'q'$ elements that don't generate QR_{n^2}. When we compare this to the total elements, we see: $(pqp' + pqq' + pp'q' + qp'q')/pqp'q' = 1/q' + 1/p' + 1/p + 1/q$ which is negligible if p, q, p', q' are large.

Lemma 9. *If $2^B > ord(g)$ then no PPT adversary running in time polynomial to λ can distinguish distribution $\{g^s : s \leftarrow\!\!\$\ 2^{B+\lambda}\}$ from $\{u : u \leftarrow\!\!\$\ \langle g \rangle\}$ for any g such that $g \in \mathbb{Z}_{n^2}$ and $ord(g) > 2$.*

We refer to [DF02] for a proof of Lemma 9.

Lemma 10. *If $x, x' \in \mathsf{QR}_p$ and $y, y' \in \mathsf{QNR}_p$ then $xy \in \mathsf{QNR}_p$, $xx', yy' \in \mathsf{QR}_p$.*

Lemma 11. *For $n = pq$ where p, q are safe primes, if $x, x' \in \mathsf{QR}_{n^2}$ and $y, y' \in \mathsf{QNR}_{n^2}$ then $xy \in \mathsf{QNR}_{n^2}$, $xx', yy' \in \mathsf{QR}_{n^2}$*

Lemma 12. *$\#QR_{n^2} = \mathbb{Z}_{n^2}/4$*

Proofs of Lemmas 11, 10, and 12 are present in [KL20] (deriving Lemma 12 from [KL20] is a trivial exercise and stems from the fact that $QR_{n^2} \cong QR_p \times QR_q \times QR_{p'} \times QR_{q'}$).

D.3 How to Prove Equality and Other Relations of Committed Values

Constructing *eqrep*-$\mathbb{G}_p$* We gave a construction to prove *eqrep*-$\mathbb{G}_p$ relations in 14, though this is not fully general as it does not allow for arbitrary multiplication of witnesses. In this section, we give a construction of an example relation for the *eqrep*-${\mathbb{G}_p}^*$ protocol. In Alg. 9 we show how to implement a *eqrep*-${\mathbb{G}_p}^*$ protocol from an underlying *eqrep*-$\mathbb{G}_p$ protocol by construction intermediate Pedersen commitments. In this example, we are proving that a Pedersen commitment C_a is committed to the product of the values in three other Pedersen commitments, C_b, C_c, and C_d. Formally, Alg. 9 proves the following relation: $R((C_a, C_b, C_c, C_d), (a, b, c, d, r_a, r_b, r_c, r_d)) = 1$ iff $C_a = g^a h^{r_a} \wedge C_b = g^b h^{r_b} \wedge C_c = g^c h^{r_c} \wedge C_d = g^d h^{r_d} \wedge a = bcd$. Because E is a commitment to bc with fresh randomness, revealing it to the verifier does not affect the zero knowledge of the scheme. The only other communication in this proof for *eqrep*-$\mathbb{G}_p$ is the proof for an *eqrep* relation. Thus this scheme is zero knowledge. We can see that the relation proves that $E = g^{bc} h^{c\beta_2}$ which is a valid Pedersen commitment to bc. Thus, because the prover also proves that $C_a = E^d h^{\beta_2}$, the verifiers knows that $C_a = g^{bcd} h^{d\beta_2}$ which is a valid Pedersen commitment to bcd and thus, $a = bcd$. This means we've proven soundness with extraction for this protocol. Using the notation from Def. 4, the map μ would be $\mu(a) = \{b, c, d\}$ (and $\mu(x) = \{x\}$ otherwise). This would ensure that the witness $a = bcd$ with no constraints on the other witnesses. To build an *eqrep*-$\mathbb{G}_p$ protocol for more multiplications of witnesses, more commitments for intermediate values would be used. It should be clear from the example how to do this for any map μ from Def. 4.

Algorithm 9 Example *eqrep*-$\mathbb{G}_p$ proof

```
$\rho \leftarrow\$ \mathbb{Z}_p; E = g^{bc}h^{\rho}$
$\beta_1 = \rho - cr_b; \beta_2 = r_a - d\rho$
Send $E$ to the verifier
Prove the following relation via eqrep
$\mathsf{PoK}_{eqrep}[a, b, c, r_a, r_b, r_c, \beta_1, \beta_2 :$
    $C_a = g^a h^{r_a} \wedge C_b = g^b h^{r_b} \wedge C_c = g^c h^{r_c} \wedge C_d = g^d h^{r_d}$
    $\wedge E = C_b^c h^{\beta_1} \wedge C_a = E^d h^{\beta_2}]$
```

Constructing *eqrep*-$\mathbb{Z}_{n^2}$ Construction 1 shows an example construction of a proof of a relation for *eqrep*-$\mathbb{Z}_{n^2}$ defined in Sec. 2. We note that to reduce a construction of *eqrep*-$\mathbb{Z}_{n^2}$ to the soundness of Damgård-Fujisaki commitments, we need to create Damgård-Fujisaki commitments to each witness in the relation and use a proof of opening in the protocol to ensure we can extract the witnesses. This step is not necessarily required, but is sufficient to realize *eqrep*-$\mathbb{Z}_{n^2}$ and allows us to reduce to the auxiliary proofs for Damgård-Fujisaki commitments rather than number theoretic lemmas. In this example, we'll use Damgård-Fujisaki commitments in $\mathbb{Z}_{n^2}$ which we prove are secure in Sec. 4.3. In this example, we prove the exponentiation of an element in a $|QR_{n^2}|$ commitment (which we define in Sec. 4.3) by a scalar committed to by a Damgård-Fujisaki commitment. This proof can be seen as proving the relation $R((c_1, c_2, t, d_1, d_2), (x_1, r_1, x_2, r_2, x_3, r_3, M, N, x_1, x_2, x_3)) = 1$ iff $c_2 = g^{x_1}h^{r_1} \wedge t = g^{x_2}h^{r_2} \wedge d_2 = g^{x_3}h^{r_3} \wedge c_1 = Mg^{x_1} \wedge d_1 = Ng^{x_3} \wedge N = M^{x_2}$.

For this proof, both the prover P and the verifier V have a scalar commitment t to value x_2 along with two $|QR_{n^2}|$ commitments $c = (c_1, c_2)$ and $d = (d_1, d_2)$ to two $\mathbb{Z}_{n^2}$ elements, M, and N. The prover wants to show that $N = M^{x_2}$. Damgård and Fujisaki [DF02] give a multiplication protocol which yields a commitment scheme for integers in any group that satisfies certain properties. We prove in Sec. 4.3 that QR_{n^2} and $\mathbb{Z}_{n^2}$ both satisfy these properties. We can see that the second elements of both of our $|QR_{n^2}|$ commitments (c_2 and d_2) are exactly Damgård-Fujisaki commitments. We also note that our commitments to scalars (the commitment t in this example) are simply Damgård-Fujisaki commitments. The Damgård-Fujisaki exponentiation proof is a Σ-protocol and thus has transcripts a, e, z. If the prover uses the z value from a proof of opening of the scalar commitment (t) and reuses this z value in a relation to the $|QR_{n^2}|$ commitments, the prover can prove this exponentiation property for the c, and d commitments. We construct this exponentiation protocol in Construction 1. This example should give the reader enough intuition to build a proof for any *eqrep*-$\mathbb{Z}_{n^2}$ relation by adding more Damgård-Fujisaki commitments to witnesses similar to the extension of *eqrep*.

The prover must also prove knowledge of the opening of each commitment in addition to running this protocol.

Construction 1 ($|QR_{n^2}|$-commitments - proof of exponentiation) *Goal: Prove that the $|QR_{n^2}|$-commitment d is committed to $N = M^{x_2}$ where c is a*

$|QR_{n^2}|$-commitment to M and t is a Damgård-Fujisaki commitment to the integer x_2.

Public values: $c = (c_1, c_2), t, d = (d_1, d_2)$ where $c_2 = g^{x_1}h^{r_1}$, $t = g^{x_2}h^{r_2}$, $d_2 = g^{x_3}h^{r_3}$, $c_1 = Mg^{x_1}$, $d_1 = M^{x_2}g^{x_3}$.

Secret values: $x_1, x_2, x_3, r_1, r_2, r_3, M$.

First, the prover uses the proof of knowledge of commitment opening from Damgård and Fujisaki [DF02] to prove that $t = g^{x_2}h^{r_2}$. The prover then shows that the prover can open c and d such that $M = \pm c_1/g^{x_1}$ and $N = \pm d_1/g^{x_3}$. The prover and verifier then engage in the following sigma protocol:

$P \qquad\qquad\qquad\qquad \leftrightarrow \quad V$

ρ_1 *will hide* ex_2

$\rho_1 \leftarrow\!\!\$\ [CT2^{\lambda}]$

ρ_2 *will hide* er_2

$\rho_2 \leftarrow\!\!\$\ [C2^{B+2\lambda}]$

ρ_3 *will hide* $e(-x_2x_1 + x_3)$

$\rho_3 \leftarrow\!\!\$\ [CT^2 2^{\lambda}]$

ρ_4 *will hide* $e(-r_1x_1 + r_3)$

$\rho_4 \leftarrow\!\!\$\ [CT2^{B+2\lambda}]$

$a_1 = g^{\rho_1}h^{\rho_2}$

$a_2 = c_1^{\rho_1}g^{\rho_3}$

$a_3 = c_2^{\rho_1}g^{\rho_3}h^{\rho_4}$

$a_1, a_2, a_3 \rightarrow$

$e \leftarrow\!\!\$\ [C]$

$\leftarrow e$

$z_1 = \rho_1 + ex_2$

$z_2 = \rho_2 + er_2$

$z_3 = \rho_3 + e(-x_1x_2 + x_3)$

$z_4 = \rho_4 + e(-r_1x_2 + r_3)$

$z_1, z_2, z_3 \rightarrow$

$g^{z_1}h^{z_2} = a_1t^e$

$c_1^{z_1}g^{z_3} = a_2d_1^e$

$c_2^{z_1}g^{z_3}h^{z_4} = a_3d_2^e$

Lemma 13 (Strong special soundness property of [DF02]). *If we find $a, e, e', z_1, z_1', z_2, z_2'$ such that a, e, z_1, z_2 and a, e', z_1', z_2' are both valid transcripts for a Damgård-Fujisaki opening protocol. If $g^{z_1}h^{z_2} = ac^e$ and $g^{z_1'}h^{z_2'} = ac^{e'}$, where c is a Damgård-Fujisaki commitment, then we know that $(e-e')|(z_1 - z_1')$ and $(e-e')|(z_2-z_2')$ and we can extract a b such that $bg^{(z_1-z_1)/(e-e')}h^{(z_2-z_2')/(e-e')} = c$*

Proof of Lemma 13 can be found in [DF02]. This is stronger than simple extraction as it ensures that $e - e'$ divides both $z_1 - z_1'$ and $z_2 - z_2'$.

Theorem 24. *Our exponentiation protocol in Construction 1 has special soundness i.e. given two accepting transcripts, there exists an efficient extractor that extracts an opening of d to M^{x_2}, c to M and t to x_2.*

Special soundness proof overview. Over the course of the proof, we'll extract $\Delta_e = e - e'$ as well as $\forall i \in [4], \Delta_{z_i} = z_i - z_i', \delta_{z_i} = \Delta_{z_i}/\Delta_e \forall i \in [4]$ along with β_1, β_2, and β_3 such that: $b_1 g^{\delta_{z_1}} h^{\delta_{z_2}} = t$, $b_2 c_1^{\delta_{z_1}} g^{\delta_{z_3}} = d_1$, and $b_3 c_2^{\delta_{z_1}} g^{\delta_{z_3}} h^{\delta_{z_4}} = d_2$.

Our proof will proceed as follows: First, we'll extract the opening of t, then we'll extract the values from the third equation, $c_2^{z_1} g^{z_3} h^{z_4} = a_3 d_2^e$, and use our knowledge of the opening of t to help us. Lastly, we'll extract values from the second equation ($c_1^{z_1} g^{z_3} = a_2 d_1^e$) using our knowledge of the last two extractions (from the first and third equations). Using these extracted values, we'll be able to prove that the commitments are sound. We need to proceed in this order to ensure we've extracted enough values to compute $(z_3 - z_3')/(e - e')$ and $(z_4 - z_4)/(e - e')$. Without knowing previously extracted values, we cannot trivially reduce to the soundness of the proof of knowledge of opening protocol in [DF02] because c_1 and c_2 are used as the bases for verification in the second two equations. We will see that we can carefully craft final messages s_1, s_2 to give to the [DF02] challenger which will allow us to compute $(z_3 - z_3')/(e - e')$ and $(z_4 - z_4')/(e - e')$ in the final two equations to prove them secure. In the proof, we'll use Δ and δ to refer to values used in the extraction. For example, Δ_{z_1} will refer to $z_1 - z_1'$ after rewinding a prover and δ_{z_1} will refer to $(z_1 - z_1')/(e - e')$.

Proof of special soundness. Since we have the prover prove they know the openings of t, c, and d individually, our extractor can compute $c = (Mg^{x_1}a_d, g^{x_1}h^{r_1})$, $d = (Ng^{x_3}a_d, g^{x_3}h^{r_3})$, and $t = g^{x_2}h^{r_2}b_t$.

Using rewinding, we can extract $\Delta_{z_1} = z_1 - z_1'$, $\Delta_{z_2} = z_2 - z_2'$, $\Delta_{z_2} = z_2 - z_2'$, $\Delta_{z_3} = z_3 - z_3'$, $\Delta_{z_4} = z_4 - z_4'$, and $\Delta_e = e - e'$. We can see that the first equality, $g^{z_1}h^{z_2} = a_1 t^e$ appears exactly like a proof of opening for Damgård-Fujisaki commitments, and thus, we can extract $\delta_{z_1} = \Delta_{z_1}/\Delta_e, \delta_{z_2} = \Delta_{z_2}/\Delta_e$, b_1, from this due to Lemma 13. To show why we can extract, we can create a reduction to the soundness of proof of opening of [DF02].

Our reduction will take t from our adversary, then claim to the [DF02] opening soundness challenger that we can open this. We can discard all other values from the adversary when doing this. Then, we also pass a_1 to the challenger and we receive the challenge, e from the challenger and pass this to the adversary. The adversary will then produce z_1, z_2, and we can discard the other z values and simply pass the first two to the challenger. We see that this satisfies $g^{z_1}h^{z_2} = a_1 t^e$ and thus is a valid proof and thus we can rewind and use the same algorithm as the challenger in the knowledge proof of [DF02] to extract $\delta_{z_1}, \delta_{z_2}, b_1$ such that $t = g^{\delta_{z_1}} h^{\delta_{z_2}} b_1$ and $b_1^2 = 1$.

The rest of our proof will create more reductions to the soundness game in [DF02], but the details will be omitted.

Next, we observe that we can continue rewinding until we obtain an even $e - e'$. See that any subset of $[C]$ must be at least half even or odd and the adversary must be able to answer a super polynomial subset of $[C]$. Thus, with probability at least $1/4$ it will be the case that e and e' will both be even or both be odd, thus ensuring that $e - e'$ is even. Let us focus on the case where $e - e'$ is even, knowing that we'll only reduce our chance of breaking soundness in this case by $1/4$ which is still efficient.

Next, we'll prove that because our extractor can open c_2, if we can't extract $\delta_{z_3} = \Delta_{z_3}/\Delta_e$, $\delta_{z_4} = \Delta_{z_4}/\Delta_e$, and β_3 such that $c_2^{\delta_{z1}} g^{\delta_{z3}} h^{\delta_{z4}} \beta_3 = d_2$, we can reduce to the proof of opening protocol. We can see that this is true with another reduction similar to our reduction for t. We pass d_2, a_3 to the challenger to receive e to pass back to the adversary. After our adversary proves they can open c_2, we receive x_1, r_1, b_3 such that $g^{x_1} h^{r_1} b_3 = c_2$ and $b_3^2 = 1$. We see that the verifier accepts, so, $c_2^{z_1} g^{z_3} h^{z_4} = a_3 d_2^e$ and thus, $c_2^{\Delta_{z1}} g^{\Delta_{z3}} h^{\Delta_{z4}} = a_3 d_2^{\Delta_e}$. We can replace this with $(\beta_3)^{\Delta_{z1}} g^{x_1 \Delta_{z1}} h^{r_1 \Delta_{z1}} g^{\Delta_{z3}} h^{\Delta_{z4}} = a_3 d_2^{\Delta_e}$. Since $e - e'$ is even and we know that $e - e'$ divides Δ_{z_1}, we know that Δ_{z_1} is even. Because $b^2 = 1$ and Δ_{z_1} is even, we see that $g^{x_1 \Delta_{z1}} h^{r_1 \Delta_{z1}} g^{\Delta_{z3}} h^{\Delta_{z4}} = a_3 d_2^{\Delta_e}$. We then give: $s_1 = x_1 \Delta z_1 + \Delta z_3, s_2 = r_1 \Delta z_1 + \Delta z_4$ to the challenger, which satisfies $g^{s_1} h^{s_2} = a_3 d_2^e$. Thus, because of the knowledge extractor for proof of opening, we know we can rewind the adversary and compute $\delta_{s_1} = (s_1 - s_1')/(e - e')$ as well as $\delta_{s_2} = (s_2 - s_2')/(e - e')$ and β_3. Because the adversary proved opening of d_2, we have x_3, r_3, b_{d_2} such that $\delta_{s_1} = x_3, \delta_{s_2} = r_3, b_{d_2} = \beta_3$. We can then extract δ_{z_3} with the following equation: $\delta_{z_3} = x_3 - x_1 \delta_{z_1} = (z_3 - z_3')/(e - e')$
This is because $\delta_{s_1} = x_1$ implies that:
$x_3(e - e') = s_1 - s_1' = x_1 z_1 + z_3 - x_1 z_1' - z_3'$
$x_3(e - e') - x_1 z_1 + x_1 z_1' = z_3 - z_3'$
$x_3(e - e') - x_1(z_1 - z_1') = z_3 - z_3'$
$x_3(e - e') - x_1 \delta_{z_1} * (e - e') = z_3 - z_3'$
$x_3 - x_1 \delta_{z_1} = (z_3 - z_3')/(e - e')$
We then know that:
$\delta_{s_2} = (s_2 - s_2')/(e - e') = (r_1 z_1 + z_4 - r_1 z_1' - z_4')/(e - e')$
And that $r_3 = \delta_{s_2}$ and thus:
$r_3(e - e') = (r_1 z_1 + z_4 - r_1 z_1' - z_4')$
$r_3(e - e') - r_1 z_1 + r_1 z_1' = (z_4 - z_4')$
$r_3(e - e') - r_1(z_1 + z_1') = (z_4 - z_4')$
And we know that $\delta_{z_1} = (z_1 + z_1')/(e - e')$, so:
$r_3(e - e') - \delta_{z_1} * (e - e') = (z_4 - z_4')$
$\delta_{z_4} = r_3 - \delta_{z_1} = (z_4 - z_4')/(e - e')$
This gives us that $d_2 = g^{x_1 \delta_{z_1} + \delta_{z_3}} h^{r_1 \delta_{z_1} + \delta z_4} \beta_3$. Which must agree with x_3, r_3, b_{d_2}. Because we know that $\delta_{z_1} = x_2$ from the opening of t, we know that $d_2 = g^{x_1 x_2 + \delta_{z_3}} h^{r_1 x_2 + \delta_{z_4}} b_{d_2}$.

We will now rewind the second equation, $c_1^{2z_1} g^{2z_3} = a_2 d_1^{2e}$ to extract values and prove them sound. We know that $g^{x_1} = c_1/M$ from the opening of c.

Since we know that Δ_{z_1} and Δ_{z_3} are divisible by Δ_e, we can proceed to extract the structure of d_1.

$c_1^{z_1} g^{z_3} = a_2 d_1^e$
$M^{z_1} g^{x_1 z_1} g^{z_3} = a_2 d_1^e$
$M^{z_1 - z_1'} g^{x_1(z_1 - z_1')} g^{z_3 - z_3'} = d_1^{e - e'}$
$M^{(z_1 - z_1')} g^{x_1(z_1 - z_1')} g^{(z_3 - z_3')} = d_1^{(e - e')}$

$$M^{(z_1-z'_1)/(e-e')}g^{x_1(z_1-z'_1)/(e-e')}g^{(z_3-z'_3)/(e-e')} = d_1$$
$$bM^{\delta_{z_1}}g^{x_1\delta_{z_1}}g^{\delta_{z_3}} = d_1$$
$$bM^{x_2}g^{x_1x_2}g^{\delta_{z_3}} = d_1$$
$$bg^{x_1x_2+\delta_{z_3}} = d_1/M^{x_2}$$

We can see that $b \in \{-1, 1\}$ since $b^{e-e'} = 1$ and thus, d is a correct commitment to $|M^{x_2}|$.

Honest verifier zero knowledge. If the ranges are adjusted correctly, our construction achieves this, similar to [DF02].

E Additional HEC definitions, constructions, and proofs

E.1 Security Properties of HEC Scheme

In this section, we provide formal definitions for the security properties of the HEC scheme which are unchanged from [KLN23].

```
$\textsc{SecX}^{\mathsf{Adv}}_b(\lambda)$
$hecpar \leftarrow \mathrm{HEC}\textsc{setup}(1^\lambda)$
$(f, x_0, x_1, \mathsf{st}) \leftarrow \mathsf{Adv}(1^\lambda, hecpar)$
if $f \in F, x_0, x_1 \in domain_{f,x}$
  $X, \_ \leftarrow \mathrm{HEC}\textsc{Enc}(hecpar, f, x_b)$
  return $\mathsf{Adv}(hecpar, X, \mathsf{st})$
return $\mathsf{Adv}(\perp, \mathsf{st})$
```

```
$\textsc{SecXY}^{\mathsf{Adv}}_b(\lambda)$
$hecpar \leftarrow \mathrm{HEC}\textsc{setup}(1^\lambda)$
$(f, x_0, x_1, \mathsf{st}) \leftarrow \mathsf{Adv}(1^\lambda, hecpar)$
if $f \in F, x_0, x_1 \in domain_{f,x}$
  $X, \_ \leftarrow \mathrm{HEC}\textsc{Enc}(hecpar, f, x_b)$
  $(y_0, y_1, \mathsf{st}) \leftarrow \mathsf{Adv}(X, \mathsf{st})$
  if $y_0, y_1 \in domain_{f,y}$
    $Z \leftarrow \mathrm{HEC}\textsc{Eval}(hecpar, f, X, y_b)$
    return $\mathsf{Adv}(Z, \mathsf{st})$
return $\mathsf{Adv}(\perp, \mathsf{st})$
```

```
$\textsc{DirectZ}^{\mathsf{Adv}}_b(\lambda)$
$hecpar \leftarrow \mathrm{HEC}\textsc{setup}(1^\lambda)$
$(f, x, y, r_X, \mathsf{st}) \leftarrow \mathsf{Adv}(1^\lambda, hecpar)$
if $f \in F, x \in domain_{f,x}, y \in domain_{f,y}$
  $X, \_ = \mathrm{HEC}\textsc{Enc}(hecpar, f, x; r_X)$
  $Z_0 \leftarrow \mathrm{HEC}\textsc{Eval}(hecpar, f, X, y)$
  $Z_1 \leftarrow \mathrm{HEC}\textsc{direct}(hecpar, X, f(x, y))$
  return $\mathsf{Adv}(hecpar, Z_b, \mathsf{st})$
return $\mathsf{Adv}(\perp, \mathsf{st})$
```

Fig. E.1: HEC correctness, consistency and security games

Definition 15 (Security of x, security of x and y from third parties, and security of DirectZ.). *Consider Fig. E.1.* HEC *provides* security for x *if for any* PPT Adv*,* $|p^{\textsc{SecX}}_{\mathsf{Adv},0}(\lambda) - p^{\textsc{SecX}}_{\mathsf{Adv},1}(\lambda)|$ *is negligible.* HEC *provides* security for x and y from third parties *if or any* PPT Adv*,* $|p^{\textsc{SecXY}}_{\mathsf{Adv},0}(\lambda) - p^{\textsc{SecXY}}_{\mathsf{Adv},1}(\lambda)|$ *is* *negligible.* HEC *provides* security of DirectZ *if or any* PPT Adv, $|p^{\textsc{DirectZ}}_{\mathsf{Adv},0}(\lambda) - p^{\textsc{DirectZ}}_{\mathsf{Adv},1}(\lambda)|$ *is negligible.*

Explanation for DirectZ. This is an algorithm we need in order to use a HEC in our construction of PPBs. Intuitively, recall that the security of PPBs requires that there be a simulator that can simulate the output of Escrow just given $z = f(x, y)$, without knowledge of x or y. DirectZ allows the simulator to compute the encryption of z directly. For example, if $z = f(x, y)$ where f is a one-way function of y for any fixed x, then access to just the Eval function is not sufficient to compute the encryption of z, since Eval requires y as input, and no such preimage y cannot be computed from z because f is a One-Way Function.

E.2 Constructions of HEC Schemes

KLN Construction of HEC from Fully Homomorphic Encryption (FHE)

Definition 16 (Circuit-private fully homomorphic encryption). *Algorithms* (FHEKeyGen, FHEEnc, FHEDec, FHEEval) *form a secure fully homomorphic public-key encryption scheme [Gen09,BV11,BGV12,GSW13] if:*

Input-output specification: $\mathsf{FHEKeyGen}(1^\lambda, \Lambda)$ *takes as input the security parameter and possibly system parameters* Λ *and outputs a secret key* FHESK *and a public key* FHEPK. $\mathsf{FHEEnc}(\mathit{FHEPK}, b)$ *takes as input the public key and a bit* $b \in \{0,1\}$ *and outputs a ciphertext* c. $\mathsf{FHEDec}(\mathit{FHESK}, c)$ *takes as input a ciphertext* c *and outputs the decrypted bit* $b \in \{0,1\}$. $\mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}, c_1, \ldots, c_n)$ *takes as input a public key, a Boolean circuit* $\mathcal{C} : \{0,1\}^n \mapsto \{0,1\}$, *and* n *ciphertexts and outputs a ciphertext* $c_{\mathcal{C}}$*; correctness (below) ensures that* $c_{\mathcal{C}}$ *is an encryption of* $\mathcal{C}(b_1, \ldots, b_n)$ *when* c_i *encrypts* b_i.

Correctness of evaluation: *For any integer* n *(polynomial in* λ*) for any circuit* $\mathcal{C}$ *with* n *inputs of size that is polynomial in* λ, *for all* $x \in \{0,1\}^n$, *the event that* $\mathsf{FHEDec}(\mathit{FHESK}, C) \neq \mathcal{C}(x)$ *where* $(\mathit{FHESK}, \mathit{FHEPK})$ *are outputs of* FHEKeyGen, *ciphertexts* c_i *are outputs of* $\mathsf{FHEEnc}(\mathit{FHEPK}, x_i)$, *and* $c_{\mathcal{C}}$ *is output of* $\mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}, c_1, \ldots, c_n)$, *has probability* 0.

Security: *FHE must satisfy the standard definition of semantic security.*

Compactness: *What makes fully homomorphic encryption non-trivial is the property that the ciphertext* $c_{\mathcal{C}}$ *should be of a fixed length that is independent of the size of the circuit* $\mathcal{C}$ *and of* n. *More formally, there exists a polynomial* $s(\lambda)$ *such that for all circuits* $\mathcal{C}$, *for all* $(\mathit{FHESK}, \mathit{FHEPK})$ *output by* $\mathsf{FHEKeyGen}(\lambda)$ *and for all input ciphertexts* $c_1, \ldots, c_n$ *generated by* $\mathsf{FHEEnc}(\mathit{FHEPK}, \cdot)$, $c_{\mathcal{C}}$ *generated by* $\mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}, c_1, \ldots, c_n)$ *is at most* $s(\lambda)$ *bits long.*

Circuit-privacy: *As defined by [Gen09,OPP14,BdMW16,DD22] an FHE scheme is circuit private for a circuit family* $\mathcal{C}$ *if for any* PPT *algorithm* Adv $|p_{\mathsf{Adv},0} - p_{\mathsf{Adv},1}| = \nu(1^\lambda)$ *for a negligible* ν, *where for* $b \in \{0,1\}$, $p_{\mathsf{Adv},b}$ *is the probability that the following experiment outputs* 0*:*

```
$\mathsf{FHECircHideExpt}(1^\lambda)$
$(R, \mathcal{C}_0, \mathcal{C}_1, (x_1, r_1), \ldots, (x_n, r_n)) \leftarrow \mathsf{Adv}(1^\lambda)$
$\textbf{if } \mathcal{C}_0 \notin \mathcal{C} \vee \mathcal{C}_1 \notin \mathcal{C} \vee \mathcal{C}_0(x_1, \ldots, x_n) \neq \mathcal{C}_1(x_1, \ldots, x_n) : \textbf{reject}$
$(\mathit{FHEPK}, \mathit{FHESK}) = \mathsf{FHEKeyGen}(1^\lambda; R)$
$\textbf{for } i \in \{1, \ldots, n\}:$
  $c_i = \mathsf{FHEEnc}(\mathit{FHEPK}, x_i; r_i)$
$Z_0 \leftarrow \mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}_0, c_1, \ldots, c_n)$
$Z_1 \leftarrow \mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}_1, c_1, \ldots, c_n)$
$\textbf{return } \mathsf{Adv}(Z_b)$
```

Construction of HEC for any f from CP-FHE. For a Boolean function $g : \{0,1\}^{\ell_x} \times \{0,1\}^{\ell_y} \mapsto \{0,1\}$, an ℓ_y-bit string y and a value $z \in \{0,1\}^2$, let $\mathcal{C}^g_{y,z}(x)$ be the Boolean circuit that outputs $g(x,y)$ if $z_1 = 0$, and z_2 otherwise.

Recall that our goal is to construct a secure f-HEC scheme with a direct encryption algorithm; suppose that the length of the output of f is ℓ; for $1 \leq j \leq \ell$, let $f_j(x,y)$ be the Boolean function that outputs the j^{th} bit of $f(x,y)$. Suppose we are given an FHE scheme that is circuit-private for the families of circuits $\{\mathcal{C}_j\}$ defined as follows: $\mathcal{C}_j = \{\mathcal{C}^{f_j}_{y,z}(x) \; : \; y \in \{0,1\}^{\ell_y}, z \in \{0,1\}^2\}$.

$\text{HEC}\textsc{setup}(1^\lambda) \rightarrow \mathit{hecpar}$: Generate the FHE parameters hecpar, if needed.

$\text{HEC}\textsc{Enc}(\mathit{hecpar}, f, x) \rightarrow (X, d)$: Generate $(\mathit{FHESK}, \mathit{FHEPK}) \leftarrow \mathsf{FHEKeyGen}(1^\lambda, \mathit{hecpar})$. Let $|x| = n$; set $c_i \leftarrow \mathsf{FHEEnc}(\mathit{FHEPK}, x_i)$. Output $X = (\mathit{FHEPK}, c_1, \ldots, c_n)$, and decryption key $d = \mathit{FHESK}$.

$\text{HEC}\textsc{Eval}(\mathit{hecpar}, f, X, y) \rightarrow Z$: Parse $X = (\mathit{FHEPK}, c_1, \ldots, c_n)$. For $j = 1$ to ℓ, compute $Z_j \leftarrow \mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}^{f_j}_{y,00}, c_1, \ldots, c_n)$. Output $Z = (Z_1, \ldots, Z_\ell)$.

$\text{HEC}\textsc{Dec}(\mathit{hecpar}, d, Z) \rightarrow z$: Output $(\mathsf{FHEDec}(d, Z_1), \ldots, \mathsf{FHEDec}(d, Z_\ell))$.

$\text{HEC}\textsc{Direct}(\mathit{hecpar}, X, z) \rightarrow Z$: Parse $X = (\mathit{FHEPK}, c_1, \ldots, c_n)$. For $j = 1$ to ℓ, compute $Z_j \leftarrow \mathsf{FHEEval}(\mathit{FHEPK}, \mathcal{C}^{f_j}_{0^\ell,1z_j}, c_1, \ldots, c_n)$. Output $Z = (Z_1, \ldots, Z_\ell)$.

Theorem 7. For a FHE scheme, $(\mathsf{FHEKeyGen}, \mathsf{FHEEnc}, \mathsf{FHEDec}, \mathsf{FHEEval})$ with the Correctness property, for a circuit family $\{\mathcal{C}^f_j \; : \; f \in F\}$ (as defined in [KLN23]), the construction in [KLN23] is a consistent HEC for the family F.

Proof. Let us assume the existence of an adversary $\mathcal{A}$ that is able to produce a $(f, x, \mathsf{st}, r, y, r_Z)$ such that $Z \leftarrow \text{HEC}\textsc{Eval}(\mathit{hecpar}, f, X, y; r_Z)$ but $\text{HEC}\textsc{Dec}(\mathit{hecpar}, d, Z) \neq f(x,y)$. We can then construct an adversary $\mathcal{A}'$ from adversary $\mathcal{A}$ which outputs x, y and Φ^f_y where the output of the circuit $\Phi^f_y(x) = f(x,y)$.

This gives us a tuple (x, y, Φ^f_y) for which the keys $\mathsf{FHESK}, \mathsf{FHEPK} \in \mathsf{FHEKeyGen}(\lambda)$, $c \in \mathsf{FHEEnc}(\mathsf{FHEPK}, x)$ from the output of $\text{HEC}\textsc{Enc}(\mathit{hecpar}, f, x; r)$ and $c_\Phi \in \mathsf{FHEEval}(\mathsf{FHEPK}, \Phi, c)$ from Z are as required, but $\mathsf{FHEDec}\ (\mathsf{FHESK}, c_\Phi) \neq \Phi(x)$. Since the correctness of FHE (as provided in Appx. E.2) is defined over all possible inputs x and y, all randomness tapes, and for all circuits Φ, the tuple (x, y, Φ^f_y) is clearly a violation of the correctness condition. This proves that the HEC construction is indeed consistent.

As shown by [KLN23] both Security of x, SecX and the security of x and y from third parties, SecXY is obtained by the semantic security of the FHE. The security of DirectZ follows from the circuit privacy.