Olympic Privacy-Preserving Blueprints: Faster Communication, Highly Functional, Stronger Security

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. Introduced by Kohlweiss, Lysyanskaya, and Nguyen (Eurocrypt'23), an f-privacy-preserving blueprint (PPB) system allows an auditor with secret input x to create a public encoding of the function $f(x,\cdot)$ that verifiably corresponds to a commitment C_x to x. The auditor will then be able to derive $f(x,y)$ from an escrow Z computed by a user on input the user's private data y corresponding to a commitment C_y. Z verifiably corresponds to the commitment C_y and reveals no other information about y.

PPBs provide an abuse-resistant escrow mechanism: 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. Yet, the auditor's x must correspond to a publicly available (and potentially authorized by a transparent, lawful process) C_x, and the auditor will learn nothing except $f(x,y)$.

In this paper, we build on the original PPB results in three ways: (1) We define and satisfy a stronger notion of security where a malicious auditor cannot frame a user in a transaction to which this user was not a party. (2) We provide efficient schemes for a bigger class of functions f; for example, for the first time, we show how to realize f that would allow the auditor to trace e-cash transactions of a criminal suspect. (3) 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.

Table of Contents

1 Introduction

Cryptography gives us powerful tools for balancing 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. Anonymous credentials [Cha90,LRSW99,CL01,Lys02,CL02,CV02,CL04,BCL04,BL13] 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 a 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.

A PPB system works as follows: 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 $Z = \mathsf{Escrow}(\mathsf{pk}, y, r)$ 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.

Kohlweiss, Lysyanskaya and Nguyen (KLN) showed that PPB were realizable for any efficiently computable f from either fully homomorphic encryption (FHE) or non-interactive secure computation (NISC); however, this general construction was not suitable for practical use. They additionally showed 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.

Motivating application. 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, for example, 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.

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[4]. 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 nevertheless see the size of the watchlist and the fact that there was a lawfully obtained warrant for placing a person on it, would 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.

[4] See page 17 of [Gov22].

Our contributions. Unfortunately, as we explain below, the original PPB system and its realization [KLN23] are not suitable for the above motivating application for several fundamental reasons. Our main contribution is to bridge the gap between the needs of this motivating application and the security, functionality and efficiency properties of PPBs.

Let us begin with security. Let us see why the KLN definition of security for PPBs is not strong enough to allow an auditor to soundly prove that a given value z indeed corresponds to the correctly computed $f(x, y)$. In other words, the definition of security 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 $z \neq f(x, y)$. Even worse, we show (in Section 1.1) that the KLN construction of $f_{watchlist}$-PPB also allows for this "framing" attack: a malicious auditor can cause an escrow to decrypt to the identity of an honest user y who is not a party to the transaction. Addressing these security issues is our first contribution.

> Our Contribution 1: 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. We also show how to modify the KLN constructions [KLN23] to achieve non-frameability.

Even with the above security improvement, the resulting construction of PPBs still falls short of the motivating application's needs as far as the functionality it offers is concerned. The problem is that, as defined above, $f_{watchlist}$ does not work for the application. 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 instead of recovering y, 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 Contribution 2: Appropriate functionality. 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 non-frameable 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.

In Section 8 we discuss other flavors of PPB that formalize and elaborate on the idea of granting the auditor access to a seed (or key) via a blueprint match. In particular, we discuss options that restrict the impact of such a match on the privacy of matching users.

Finally, the KLN construction [KLN23] is not suitable for our motivating application because, in CBDCs, we expect the watchlist x to be quite large. In the KLB construction, the size of the escrow Z was linear in the size of the watchlist x. We give a substantial efficiency improvement:

Our Contribution 3: Exponential improvement in the size of escrow Z. Let $f(x, y) = y$ if $y = (y_1, y_2)$, and $y_1 \in x$, and $\perp$ otherwise. In our practical instantiation of a non-frameable f-PPB construction, the size of Z is logarithmic in the size of x.

1.1 Non-Frameability and Why It Matters

Let us begin by explaining why the watchlist PPB scheme of Kohlweiss, Lysyanskaya, Nguyen [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 (modulo some integer τ) 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 from pk and the opening y to the commitment C_y. To decrypt Z, decrypt $\hat{Z}$ as long as the proof π verifies. Since r is random, the decryption of $\hat{Z}$ that's formed by the honest user with identity y will be random whenever $y \notin x$, but will equal y otherwise.

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 linear (in r) equation $rP(y) + y = y^*$, and will produce an escrow $Z = (\hat{Z}, \pi)$ by following the original algorithm, but just using this value for r instead of a truly random one.

Note that, of course, 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 7.

In Section 3, we improve the KLN definition of privacy-preserving blueprints by incorporating non-frameability. The gist of the new definition is that, even if the auditor is adversarial, if $\mathsf{VerPK}(\Lambda, \mathsf{pk}_\mathsf{A}, C_x)$ and $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C_y, Z)$ accept, then Z decrypts to $f(x, y)$. Moreover, we require that the decryption algorithm produce a proof π_z of correct decryption, and add a new algorithm, Judge that verifies this proof. The proof π_z is important when information garnered from blueprints is meant to be used as evidence in legal proceedings[5] or as input in a smart contract.

In order to satisfy our new definition, we need to modify the KLN construction. The general approach taken by KLN is to construct f-PPBs from "f-homomorphic-enough encryption" (f-HEC) and appropriate non-interactive zero-knowledge proof systems. An f-HEC scheme includes algorithms HECenc, HECeval, and HECdec (and a few additional ones, see Section 4). HECenc

[5] 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].

produces a public key X (and a secret decryption key d) that hides x but nevertheless allows the algorithm $\text{HECEVAL}(X, y)$ to output Z.[6] This Z will decrypt to $f(x, y)$ using algorithm $\text{HECDEC}(d, Z)$. KLN showed that an f-HEC can be transformed into an f-PPB using proof systems for ensuring that X and Z are computed correctly. In Section 4, we define *consistent* f-HEC where an adversary cannot produce the values x and y and randomness r and r_Z such that $(X, d) = \text{HECENC}(x; r)$ and $Z = \text{HECEVAL}(X, y; r_Z)$ but $\text{HECDEC}(d, Z) \neq f(x, y)$.

We then show that, in combination with appropriate non-interactive zero-knowledge proof systems, a consistent f-HEC yields a non-frameable f-PPB. We also show that the KLN construction of f-HEC from fully homomorphic encryption (FHE) already satisfy the definition of a consistent f-HEC; therefore, using our general construction of non-frameable f-PPBs from consistent f-HEC, we obtain non-frameable f-PPBs for all f. However, this general construction is not efficient because tools such as FHE are not (yet) practical, and the general NIZK proof systems invoked are not optimized for efficiency, either.

Thus, in order to obtain a practical non-frameable f-PPB for the watch-list function, we need a more efficient approach. We use the KLN construction described above as a starting point, except that 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.

Let us see why this fix works. Suppose y is present in the watchlist. Then for any r and r', we have that $r'P(y) = 0$ and $rP(y) + y = y$, and therefore even a maliciously formed Z will correctly decrypt to y. Suppose $y \notin x$. Then $r'P(y)$ cannot equal zero (since we require that $r' \neq 0$) and thus the decryption algorithm will correctly output $\perp$.

1.2 Our Techniques for Achieving Appropriate Functionality

Above, we already hinted at how our consistent f_{CDBC}-HEC scheme will work: $\text{HECENC}(x)$ will pick a polynomial $P(\chi)$ of degree n whose roots are values on the list x, and it will output encrypted coefficients of this polynomial; i.e. X will consist of these encrypted coefficients. The underlying encryption scheme will need to be additively homomorphic in order to make sure that $\text{HECEVAL}(X, y)$ can output $\hat{Z}$ and $\hat{Z}'$, the encryptions of $rP(y_{id}) + y$ and $r'P(y_{id})$ respectively, even though X leaks no information on x.

Prior work's limitation was that it used ElGamal over a group $\mathcal{G}$ of order q as an additively homomorphic cryptosystem for elements of $\mathbb{Z}_q$. This is non-standard, since traditionally we think of ElGamal's message space as $\mathcal{G}$, not $\mathbb{Z}_q$. The ElGamal decryption algorithm will be able to recover $g^{rP(y_{id})+y}$ from $\hat{Z}$, which is, in general, not sufficient for recovering y in the event that $y_{id} \in x$.

[6] Formally, a HEC scheme as defined by KLN supports a class of functions F and HECEVAL takes an additional input $f \in F$; we are being somewhat informal here.

The advantage of this approach, however, was that it was possible to prove correctness of $\hat{Z}$ using the well-understood Σ-protocols for proving relations between discrete logarithm representations of group elements.

In this paper, we adopt a more general approach: in Section 5, we construct both f_{CDBC}-HEC and define the requisite proof systems from additively homomorphic encryption that, in addition, comes 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: $c = c_1 \otimes (c_2 \odot a)$, where $\otimes$ is the homomorphic operation on ciphertexts, and $c_2 \odot a$ denotes that the homomorphic operation was applied to c_2 with itself a times.

Thanks to this more abstract way of designing f_{CDBC}-HEC, in Section 6 we are able to show that in addition to trivially instantiating it with ElGamal by splitting the seed into sufficiently small chunks, see Section 5.5 (which still has limitations as it heavily relies on range proofs), we can also instantiate it with a variant of the Camenisch-Shoup cryptosystem [CS03], which has the advantage that the decryption algorithm recovers $rP(y_{id}) + y$ from $\hat{Z}$.

Our commit-and-prove scheme for homomorphic cryptosystems may be of independent interest in other settings that involve proving and verifying correctness of computation on encrypted data (where privacy against the decryptor is important).

1.3 Our Techniques for Achieving Succinct Escrows

As explained above, an escrow will consist of $\hat{Z}$, $\hat{Z}'$ and NIZK proofs $\pi_{\hat{Z}}$ and $\pi_{\hat{Z}'}$ that they are encryptions of $rP(y_{id}) + y$ and $r'P(y_{id})$ respectively, where P is a polynomial whose encrypted coefficients $A_i = enc(a_i)$ are available to both the prover and the verifier (the encryptions of these coefficients are given as part of the public key pk). In other words, $\pi_{\hat{Z}}$ is a proof that $\hat{Z} = ((\otimes_{i=0}^{n} A_i^{y_{id}^i}) \odot r) \otimes enc(y)$ (and $\pi_{\hat{Z}'}$ is analogous).

The naïve way for computing $\pi_{\hat{Z}}$ is to form a commitment to each intermediate ciphertext $A_i^{y_{id}^i}$ computed on the way to obtaining $\hat{Z}$, and to prove that this commitment was formed correctly; this would be linear in n. To reduce the size of this proof to $O(\log n)$, we use a degree reduction technique similar to the variable reduction technique in Shamir's proof that IP=PSPACE [Sha90]. More recently, it was used in cryptography by Goldwasser, Kalai and Rothblum [GKR08] and later by Pietrzak [Pie19], who was the first to use it to halve the degree of the polynomial rather than to eliminate a variable, and follow-up work [HHKP23]. As far as we know, our paper is the first time that this technique is used in order to prove correctness of computation on encrypted data.

The overall idea, described in Section 5.3, is to use recursion such that each recursive step halves the degree of the polynomial. In other words, suppose that we need to prove that a committed ciphertext E is an encryption of $P(y) = \sum_{i=0}^{n} a_i y^i$, where the coefficients of P are not known to the prover and verifier,

but instead, their encryptions $A_i = enc(a_i)$ are known; further, the prover knows y (and thus can compute $E = (\otimes_{i=0}^{n} A_i^{y^i})$) and r, while the verifier knows $C_y = \mathsf{Com}(y; r)$. The recursive step is to reduce the proof of this statement to the proof that another ciphertext E' is an encryption of $P'(y)$, which is a polynomial of degree $n/2$ whose encrypted coefficients are known to both prover and verifier. This can be accomplished using the Schwartz-Zippel lemma, by setting $P'(\chi) = \sum_{i=0}^{(n+1)/2-1}(a_i + \alpha a_{i+(n+1)/2})\chi^i$ for a random α chosen by the verifier (or output by the random oracle).

2 Preliminaries

2.1 BB-PSL Non-Interactive Zero Knowledge (NIZK)

Non-interactive zero-knowledge (NIZK) proofs are an important building block in this paper. We follow the KLN notation and definitions (Section 2.1 of [KLN23]) of the completeness and ZK properties of NIZK proof system, provided in abbreviated form in Definition 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:

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

$\mathsf{NIZK}^{\mathsf{Adv},0}(1^\lambda)$	$\mathcal{O}_{\mathsf{S}}(m)$	$\mathcal{O}_{\mathsf{P}}(\mathbb{x}, \mathbb{w})$
return $\mathsf{Adv}^{\mathsf{S}(\cdot),\mathsf{P}^{\mathsf{S}}(\cdot,\cdot)}(1^\lambda)$	$\mathsf{st}, h, \tau_{\mathsf{Ext}} \leftarrow \mathsf{SimS}(\mathsf{st}, m)$	**if** $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$: **return** $\perp$
$\mathsf{NIZK}^{\mathsf{Adv},1}(1^\lambda)$	**return** h	$\mathsf{st}, \pi \leftarrow \mathsf{Sim}(\mathsf{st}, \mathbb{x})$
return $\mathsf{Adv}^{\mathcal{O}_{\mathsf{S}}(\cdot),\mathcal{O}_{\mathsf{P}}(\cdot,\cdot)}(1^\lambda)$		**return** π

Fig. 2.1: NIZK game

Let us review black-box partial straight line simulation extractable proof systems [KLN23] (Definition 2). In such a proof system, the straight-line extractor does not in fact extract the entire witness, but just some function of it; at the same time, a black-box extractor (that's allowed to rewind the adversary) can in fact extract the entire witness. In Section 2.2, we motivate this definition further.

Definition 2 (Black-box partial straight-line simulation extractability). $\mathsf{Adv}^{\mathsf{nisimbbpslextract}}_{\mathsf{Adv}}(\lambda) = \Pr\left[f\text{-}\mathsf{NISimBBPSLExtract}^{\mathsf{Adv}}(1^\lambda) = 1 \right] = \nu(\lambda)$ *for some negligible function* ν.

f-$\mathsf{NISimBBPSLExtract}^{\mathsf{Adv}}(1^\lambda)$

$\mathcal{Q}, \mathcal{Q}_\mathsf{S} \leftarrow [\,]$
$(\mathbb{x}, \pi) \leftarrow \mathsf{Adv}^{\tilde{\mathcal{O}}_\mathsf{S}(\cdot), \mathcal{O}_\mathsf{Sim}(\cdot)}(1^\lambda)$
$\mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{BB}(\mathsf{Adv})}(\mathcal{Q}_\mathsf{S}, \mathbb{x}, \pi)$
$\mathbb{w}' \leftarrow \mathsf{ExtSL}(\mathcal{Q}_\mathsf{S}, \mathbb{x}, \pi)$
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 π

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

2.2 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-knownthat, 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 [7]. 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.

2.3 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 Section 2.4.

Definition 3 (Relation for proof of equality of discrete logarithm representations in cyclic groups of prime order). *Let $R_{\mathsf{eqrep}-p}$ be the following relation: $R_{\mathsf{eqrep}-p}(\mathbb{x}, \mathbb{w})$ 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 $x_i = \prod_{j=1}^{m} g_{i,j}^{w_j}$.*

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 Appendix B. While using this protocol, we use Camenisch-Stadler notation. For example, the above example with C, B, and A would be written as: $\mathsf{PoK}[c, a, b, r_C, r_A, r_B : C = g^c h^{r_C} \wedge A = g^a h^{r_A} \wedge B = g^b h^{r_B} \wedge c = ab]$.

Definition 4 (Relation for proof of multiplication of witnesses over bases in cyclic groups of prime order). *Let $R_{\mathsf{eqrep}-p^*}$ be the following relation: $R_{\mathsf{eqrep}-p^*}(\mathbb{x}, \mathbb{w})$ 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 $[m]$.

[7] 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.

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 Appendix B. 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 $R_{\mathsf{eqrep}-n^*}$ be the following relation: $R_{\mathsf{eqrep}-n^*}(\mathbb{x}, \mathbb{w})$ 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 $[m]$.

2.4 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 prime order cyclic groups where the DDH and CDH assumptions are hard. We do so in Def. 6.

Definition 6 (Σ-protocol for proof of equality of discrete logarithm representations cyclic groups of prime order). *Let $R_{\mathsf{eqrep}-p}$ be the following relation: $R_{\mathsf{eqrep}-p}(\mathbb{x}, \mathbb{w})$ 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 $x_i = \prod_{j=1}^{m} g_{i,j}^{w_j}$.*

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

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 $1 \le i \le m$, and sends $\mathsf{res} = (s_1, \ldots, s_m)$ to the Verifier.*

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

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

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

3 Non-Frameable Privacy-Preserving Blueprints

In this section we first provide the formal definition of a blueprint scheme as introduced in [KLN23]. We then define the property of non-frameability for privacy preserving blueprints, and then focus our attention on proving it.

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

Definition 7. *For a non-interactive commitment scheme* $(\mathsf{CSetup}, \mathsf{Com})$*, an f-blueprint scheme consists of the following probabilistic polynomial time algorithms:*

$\mathsf{Setup}(1^\lambda, cpar) \to \Lambda$: Outputs the public parameters Λ which includes 1^λ and $cpar$.

$\mathsf{KeyGen}(\Lambda, x, r_x) \to (\mathsf{pk_A}, \mathsf{sk_A})$: The key generation algorithm for auditor A.

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

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

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

$\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C_y, Z) \to f(x, y)$ or $\perp$: Takes the auditor's secret key $\mathsf{sk_A}$, 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 $\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C_y, Z) = 1$.

[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 $(cpar, \mathsf{pk_A}, C_x, C_y, Z)$.

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

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 $\mathsf{pk_A}$ 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.

We provide more complete and formal definitions of the existing blueprint scheme and its security properties in the Appendix C.

3.1 Definition of Non-Frameability

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

Definition 8. *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* Decrypt *algorithm and an additional* Judge *algorithm, both probabilistic polynomial time algorithms:*

$\mathsf{Decrypt}(\Lambda, \mathsf{sk_A}, C_y, Z) \to (f(x,y), \pi_z)$: The algorithm 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. It 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 the Judge algorithm that $f(x,y)$ was decrypted correctly from Z.

$\mathsf{Judge}(\Lambda, \mathsf{pk_A}, C_x, C_y, Z, z, \pi_z) \to 0$ or 1: This is the algorithm which, on input all the inputs of VerPK, VerEscrow, and z and π, 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* is generated by $\mathsf{CSetup}(1^\lambda)$; (2) Λ is generated by $\mathsf{Setup}(1^\lambda, cpar)$; (3) $(\mathsf{pk_A}, \mathsf{sk_A})$ is the output of $\mathsf{KeyGen}(\Lambda, x, r_x)$; (4) $C_x = \mathsf{Com}_{cpar}(x; r_x)$; (5) $C_y = \mathsf{Com}_{cpar}(y; r_y)$; (6) Z is generated by $\mathsf{Escrow}(\Lambda, \mathsf{pk_A}, y, r_y)$; (7) (z, π) is generated by $\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C_y, Z) \to (z, \pi)$. We require that algorithm Judge accept with probability 1.

Definition 9 (Non-Frameability). *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.*

Let C_x and C_y be commitments computed from 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: $Pr[\mathsf{NonFraming}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda) = 1] < \nu(\lambda)$

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.

4 Consistent Homomorphic-Enough Encryption

In Kohlweiss et al. [KLN23], they use a "homomorphic-enough" encryption scheme. This encryption 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.

$\mathsf{NonFraming}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$

1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
3 : $(\mathsf{pk}_\mathsf{A}, x, r_x, y, r_y, Z, r, z, \pi_Z) \leftarrow \mathcal{A}(1^\lambda, \Lambda)$
4 : $C_x = \mathsf{Com}_{cpar}(x, r_x); C_y = \mathsf{Com}_{cpar}(y, r_y)$
5 : **return** $[(\mathsf{Judge}(\Lambda, \mathsf{pk}_\mathsf{A}, C_x, C_y, Z, z, \pi_Z) = 1) \wedge (f(x, y) \neq z)]$

Fig. 3.1: Experiments $\mathsf{NonFraming}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$

When creating the homomorphic ciphertexts, an encryptor specifies a specific function (f) from the function family as well as some secret information (x) to produce the ciphertext (X) and decryption key (d). The ciphertext does not necessarily hide f but does hide x. Another party can then use X to compute an encryption of $f(x, y)$. The original encryptor can then use d to retrieve this value. We provide a more formal definition in 10. We include the game for our contributed definition (HECconsistent) to Fig. 4.1 to save space though the original scheme [KLN23] does not consider this. In Kohlweiss et al. [KLN23] the HEC scheme is used to implement their generic blueprints construction.

Definition 10 (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 the set* HEC *of algorithms* (HECsetup, HECenc, HECeval, HECdec, HECdirect) *constitute a homomorphic-enough cryptosystem (HEC) for F if they satisfy the following input-output, correctness, and security requirements:*

HECsetup$(1^\lambda) \rightarrow hecpar$ takes the security parameter as input, outputs the parameters $hecpar$.
HECenc$(hecpar, f, x) \rightarrow (X, d)$ 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$(hecpar, f, X, y) \rightarrow Z$ 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$(hecpar, d, Z) \rightarrow z$ takes as input the parameters $hecpar$, the decryption key d, and a ciphertext Z, decrypts Z to obtain a value z.
HECdirect$(hecpar, X, z) \rightarrow Z$ on input $hecpar$, an encrypted representation X of some function, and a value z, outputs a ciphertext Z.

HEC correctness. For a given adversary Adv and HEC, let $\mathsf{Adv}_{\mathrm{HEC},\mathsf{Adv}}(\lambda)$ be the probability that the experiment HECcorrect in Fig. 4.1 accepts. HEC is *correct* if $\mathsf{Adv}_{\mathrm{HEC},\mathsf{Adv}}(\lambda)$ is negligible for all PPT algorithms Adv.

Definition 11 (Security of x, security of x and y from third parties, and security of DirectZ.). *Consider Fig. 4.1.* HEC *provides* security for x

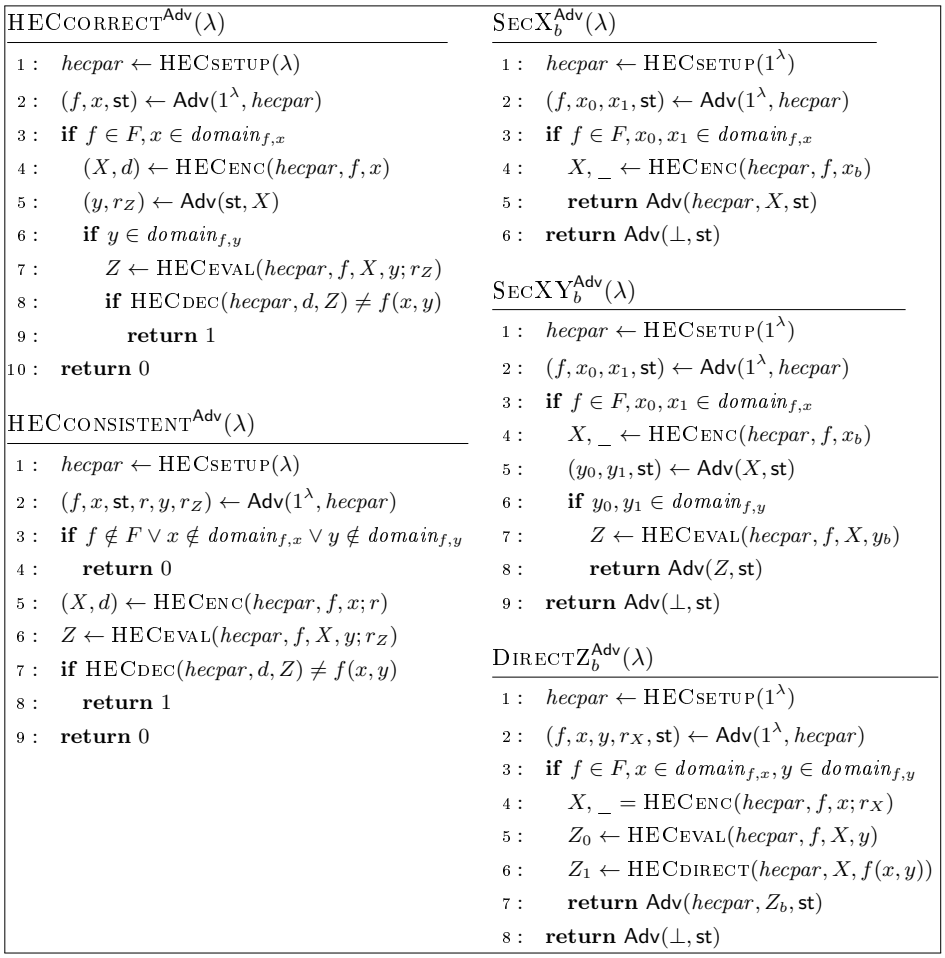

Fig. 4.1: HEC correctness, consistency and security games

if for any PPT Adv, $|p^{\text{SecX}}_{\mathsf{Adv},0}(\lambda) - p^{\text{SecX}}_{\mathsf{Adv},1}(\lambda)|$ *is negligible.* HEC *provides* security for x and y from third parties *if or any* PPT Adv, $|p^{\text{SecXY}}_{\mathsf{Adv},0}(\lambda) - p^{\text{SecXY}}_{\mathsf{Adv},1}(\lambda)|$ *is negligible.* HEC *provides* security of DirectZ *if or any* PPT Adv, $|p^{\text{DirectZ}}_{\mathsf{Adv},0}(\lambda) - p^{\text{DirectZ}}_{\mathsf{Adv},1}(\lambda)|$ *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 pre-image y cannot be computed from z because f is a One-Way Function.

Our main insight for adapting the generic construction of blueprints from homomorphic enough encryption, HEC, and NIZK is as follows: A HEC scheme must be such that even when an adversary fully controls x, y, and the randomness for the HEC scheme, the encryption and evaluation algorithms must not produce a ciphertext that decrypts to a plaintext other than $f(x, y)$. We refer to this strengthened correctness property with respect to adversarial inputs as *HEC consistency*. We formalize this in the HEC$_{\text{CONSISTENT}}$ game in Fig. 4.1.

After giving a definition of HEC consistency, we give an adapted generic construction for blueprints from HEC and NIZK and prove it secure.

4.1 Definition of Consistent HEC

We show our new game for Consistent HEC as HEC$_{\text{CONSISTENT}}$ in Fig. 4.1. The new definition of HEC consistency is stronger than the previous correctness definition since the adversary outputs 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 HEC$_{\text{ENC}}$.

4.2 Modifying the Generic Blueprint Scheme from HEC to Obtain Non-Frameability

We extend the existing generic construction of blueprint schemes to include the property of non-frameability. In order to do so, we modify the Dec algorithm and introduce a new algorithm Judge which we define as given below in Figure 4.2. The new Dec algorithm returns a proof of knowledge of correct decryption.

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 also including the condition on the additional NIZK PoK required, Ψ_3 and the property of non-frameability.

Theorem 1. *If* HEC *is a 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.*

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 1, we prove the non-frameability of the HEC scheme in 4.2.

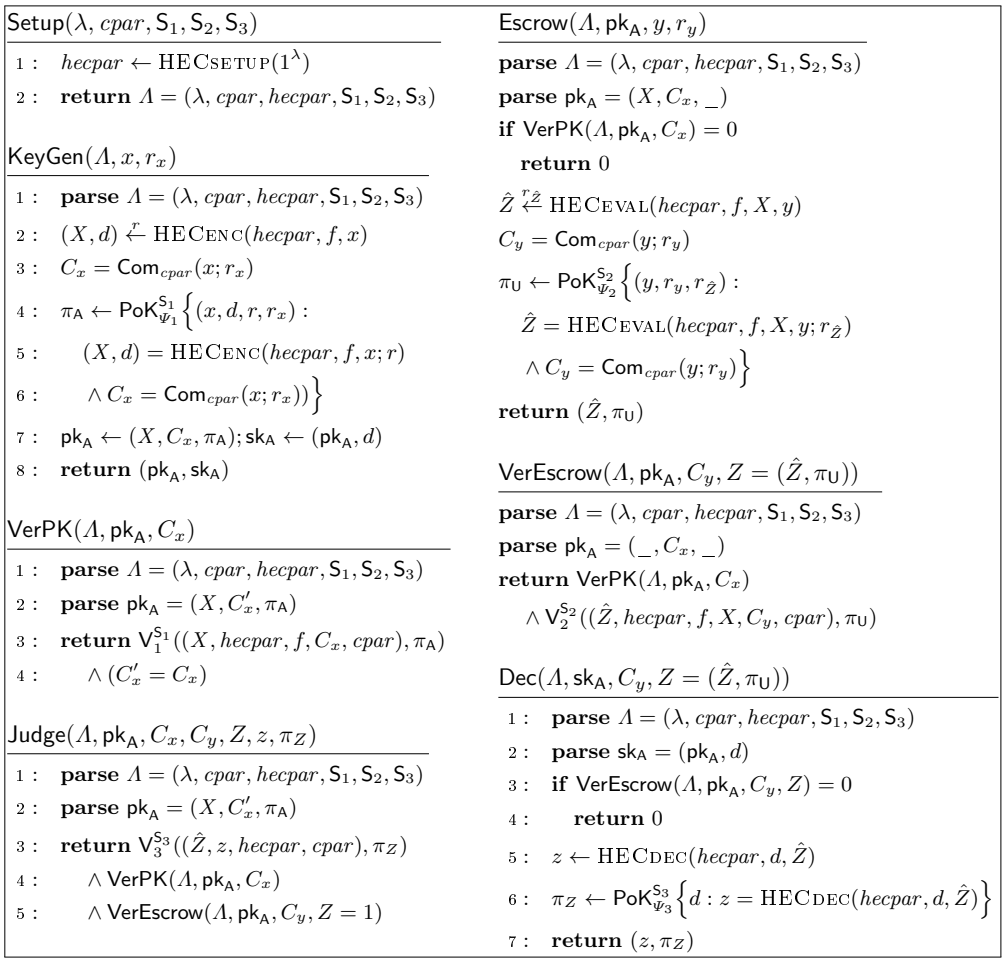

Fig. 4.2: Construction of generic f-blueprint scheme from HEC and NIZK PoKs Ψ_1, Ψ_2 and Ψ_3 with setup $\mathsf{S}_1, \mathsf{S}_2$, and S_3 respectively.

Lemma 1. *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. 4.2. Suppose this algorithm Judge returns 0 in the above mentioned experiment. This can happen if either VerEscrow returns a reject or if $\mathsf{V}^{\mathsf{S}_3}_3(Z, f_{xy}, hecpar, cpar) = 0$. From correctness of VerEscrow, we know that VerEscrow returns 1, so Judge only returns 0 if $\mathsf{V}^{\mathsf{S}_3}_3(Z, f_{xy}, hecpar, cpar) = 0$. 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.

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

Lemma 2. *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. 3.1. Suppose, for the sake of contradiction, that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{nonframing}}_{\mathcal{A},\mathsf{Blu}}(\lambda) = \nu(\lambda)$ 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 $\hat{Z} = \text{HECEVAL}(hecpar, f, X, y'; r_{\hat{Z}})$ for $y \neq y'$, (ii) when $C = \mathsf{Com}(y; r)$ and $\hat{Z} = \text{HECEVAL}(hecpar, f, X, y; r_{\hat{Z}})$ for some $r_{\hat{Z}}$ where in both (i) and (ii) X is a part of pk_{A}, (iii) the case where neither of these equalities holds and (iv) when $C = \mathsf{Com}(y; r)$ and $(X, d) = \text{HECENC}(hecpar, f, x; r)$.

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 $\hat{Z} = \text{HECEVAL}(hecpar, f, X, y'; r_{\hat{Z}})$ 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 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].

4.3 Consistent HEC from FHE

We provide an efficient construction for a secure, consistent HEC scheme for the watchlist function in Section 5.2. However, the existing construction of a HEC scheme for any function f from FHE, as provided in [KLN23] is also proven to be secure and consistent. The full details of the construction is provided in Appendix D.1.

Theorem 2. *If* (FHEKeyGen, FHEEnc, FHEDec, FHEEval) *is a fully-homomorphic public-key encryption scheme that is circuit-private for circuit family* $\{\mathcal{C}_j^f : f \in F\}$ *(as defined in [KLN23]), then our construction above constitutes a consistent and secure homomorphic-enough encryption 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 $Z \leftarrow \text{HECEVAL}(hecpar, f, x, y; r_Z)$ but $\text{HECDEC}(hecpar, d, Z) \neq f(x, y)$. We can then construct an adversary $\mathcal{A}'$ from adversary $\mathcal{A}$ which outputs x, y and Φ_y^f where the output of the circuit $\Phi_y^f(x) = f(x, y)$.

This gives us a tuple (x, y, Φ_y^f) for which, given $\mathsf{FHEKeyGen}(\lambda) \rightarrow (\mathsf{FHESK}, \mathsf{FHEPK})$, $c \leftarrow \mathsf{FHEEnc}(\mathsf{FHEPK}, x)$ and $c_\Phi = \mathsf{FHEEval}(\mathsf{FHEPK}, \Phi, c)$, but $\mathsf{FHEDec}(\mathsf{FHESK}, c_\Phi) \neq \Phi(x)$.Since the correctness of FHE (as provided in Appendix D.1) is defined over all possible inputs x and y and for all circuits Φ, the tuple (x, y, Φ_y^f) is clearly a violation of this FHE Correctness condition. This proves that the HEC construction is indeed consistent.

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.

5 Efficient PPBs for f_{CBDC} and Related Functions

In this section, we show how to efficiently instantiate privacy-preserving non-framing blueprints for watchlists. This includes a new HEC scheme as well as new proof systems for the Ψ_1, Ψ_2, and Ψ_3 proofs used in Fig. 4.2. Notably, our Ψ_2 proof produces a NIZK of size $O(\log(n))$ as opposed to $O(n)$ as was previously constructed [KLN23], where n is the length of the watchlist.

These constructions are generically obtained from the high-level ingredients listed in Section 5.1. In Section 6.2 we show how to instantiate our high-level ingredients under the decisional Diffie-Hellman assumption from the ElGamal cryptosystem; the limitation here is that, conditioned on x the value y can only come from a polynomial-size domain: the ElGamal cryptosystem only allows the auditor to directly decrypt g^y, so recovering y requires an auditor to do a brute-force search (using the list of identities x as a starting point). The scheme we give in Section 6.3, based on the Camenisch-Shoup cryptosystem, does not have this limitation, and is secure under same assumption as the Paillier cryptosystem [Pai99], the decisional composite residuosity (DCR) assumption.

Ψ_1 and Ψ_3 run in time independent of the watchlist and so are not as important to optimize. These can be implemented for ElGamal and Camenisch-Shoup with many known techniques [CL01,CS03].

5.1 Building Blocks Towards an Efficient Watchlist Construction

Commitment to user data and other scalars. Recall that a blueprint scheme is defined relative to a commitment scheme $(\mathsf{CSetup}, \mathsf{Com})$; the public parameters of the blueprint scheme, *cpar*, contain the public parameters for

this commitment. Our construction for watchlists works if, in this commitment scheme, the input domain for the values to be committed is a ring $\mathbb{Z}_\tau$ (in our instantiations, τ is either a prime q, or an RSA modulus N), and the opening can be parsed in a specific way. More precisely, Com takes as input an element x from $\mathbb{Z}_\tau$, a random value r sampled uniformly at random from $[R]$ for some integer R. For simplicity and efficiency, we'll use $\mathbb{Z}_\tau$ as the space for user data, message space for encryption, and some randomization scalars (such as r_1 and r_2 in r_z) used in our construction.

Proofs of correct modular addition and multiplication of committed values. In order to prove that an escrow was correctly computed, we'll need to add and multiply the values in our scalar commitments together (modulo τ). Let us define the following relations:

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

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

Additively homomorphic 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 $g(m)$, 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 12 (Semantically secure additively homomorphic 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 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 $m' = g_{\mathsf{pk}_{AH}}(m)$ 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 $c \in \mathsf{Enc}_{AH}(\mathsf{pk}, m)$, $\mathsf{Dec}_{AH}(\mathsf{sk}, c) = g_{\mathsf{pk}_{AH}}(m)$. I.e., the decryption algorithm correctly recovers $g_{\mathsf{pk}_{AH}}(m)$ from an encryption of m.*

Security *A semantically secure 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 Section 6.1 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), (1) we also need a way to sample new encryptions of messages, i.e., compute $c' \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, m)$ given any $c \in \mathsf{Enc}(\mathsf{pk}_{AH}, m)$; 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 $c' = c \oplus c_0$; and (2) 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 $a \in \mathbb{Z}_\tau$[8]. Our application to privacy-preserving blueprints requires that the user's input y is in the message space $\mathcal{M}_{\mathsf{pk}_{AH}} = \mathbb{Z}_\tau$. When generating the key pair for this cryptosystem, KeyGen_{AH} should take as input public parameters *params* generated by Setup that are also relevant for other algorithms we describe below.

Note that the function $g_{\mathsf{pk}_{AH}}$ 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 g is the identity function, a semantically secure additively homomorphic g-semi-encryption scheme is just a regular additively homomorphic semantically secure encryption scheme.

Notation. 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 $\{\boxed{a_i} : i \in [0...n]\}$.

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,

[8] 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.

we need a non-interactive statistically hiding, computationally binding commitment scheme Com_{AH} (parameterized by public parameters *params* generated by Setup) 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'll use $\mathsf{Com}_{AH}(\boxed{y}; r)$ to denote a commitment to a ciphertext.

Proofs of properties of 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).

We'll use the Camenisch-Stadler notation to describe an instance of these protocols. 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 $\pi = \mathsf{NIZK}[a, b, c, r_a, r_b, r_c : A = \mathsf{Com}_{AH}(\boxed{a}, r_a) \wedge B = \mathsf{Com}_{AH}(\boxed{b}, r_b) \wedge C = \mathsf{Com}(c; r_C, a_C) \wedge \boxed{a} = \boxed{b} \odot c]$. If π is accepted by the verification algorithm, then the prover can open commitments A, B and C to ciphertexts $\boxed{a}$, $\boxed{b}$ and scalar c respectively, such that $\boxed{a} = \boxed{b} \odot c$.

More precisely, let us define the following relations:

- $R_{params_{AH}}(C, (c, r)) = 1$ iff $C = \mathsf{Com}_{AH}(c; r)$;
- $R^{add}_{params_{AH}}((C_1, C_2, C_3), (c_1, r_1, c_2, r_2, c_3, r_3)) = 1$ iff $\forall i \in [3] : C_i = \mathsf{Com}_{AH}(c_i; r_i)$ and $c_3 = \mathsf{Op}_{AH}(c_1, c_2)$;
- $R^{mult}_{params_{AH}}((C_1, C_2, C_3), (c_1, r_1, c_2, r_2, x, r_3)) = 1$ iff $\forall i \in [2] : C_i = \mathsf{Com}_{AH}(c_i; r_i)$, $C_3 = \mathsf{Com}(x; r_3)$ and $c_3 = c_1 \odot x$.
- $R^{VE}_{params_{AH}}((C_1, C_2), (c_1, r_1, r_{c_1}, y, r_2)) = 1$ 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 $(\mathsf{Prove}_{AH}, \mathsf{Verify}_{AH})$ for the relation $R_{params_{AH}}$, $(\mathsf{Prove}^{add}_{AH}, \mathsf{Verify}^{add}_{AH})$ for the relation $R^{add}_{params_{AH}}$, $(\mathsf{Prove}^{mult}_{AH}, \mathsf{Verify}^{mult}_{AH})$ for the relation $R^{mult}_{params_{AH}}$, and $(\mathsf{Prove}^{VE}_{AH}, \mathsf{Verify}^{VE}_{AH})$ for the relation $R^{VE}_{params_{AH}}$; we usually refer to these proof systems indirectly via the Camenisch-Stadler notation. 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 also show how to construct them efficiently in Section 6.

Proof of ciphertext equality. Our construction further requires a proof system for proving that two commitments open to the same ciphertext; i.e. a proof system for the relation $R^{eq}_{params_{AH}}(C_1, C_2, (c, r_1, r_2)) = 1$ iff $C_1 = \mathsf{Com}_{AH}(c; r_1)$ and $C_2 = \mathsf{Com}_{AH}(c; r_2)$. This can be generically constructed by using $x = 1$ as the scalar in the proof for the relation $R^{mult}_{params_{AH}}$.

5.2 Instantiation of Consistent HEC Scheme

In this section, we provide a HEC scheme that satisfies Def. 9. In Section 5.3 we'll show a succinct proof system Ψ_2 which ensures escrows are created honestly.

To obtain a non-frameable watchlist scheme, we construct the algorithms HECEVAL and HECDEC in Fig. 5.1 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_{attr}, where the user's input consists of the user's identifier y_{id} and an attribute: $y = (y_{id}, y_{attr})$. $f_{n,k}$ 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 Section 1. y_{id} uniquely identifies a user and y_{attr} 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 Section 5.5.

Overview of the construction. The HECENC algorithm (Fig. 5.1) takes as input the list x of n watchlisted identities, and computes a polynomial $P(\chi) = a_i\chi^i$ 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 g-semi-encryption scheme (Def. 12), and outputs the public key $X = (\mathsf{pk}_{AH}, \{A_i = \mathsf{Enc}(\mathsf{pk}_{AH}, a_i)\}_{i\in[0\ldots n]})$ 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 $y = (y_{id}, y_{attr})$, HECEVAL will output the escrow $Z = (Z_{id}, Z_{attr}, Z_{nf})$ which consists of three ciphertexts under the key pk_{AH}; these will decrypt to the values $(y_{id}, y_{attr}, 0)$ 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. 5.1, 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_{attr}, yielding ciphertexts Y_{id} and Y_{attr}. 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_{attr} is $Z_{attr} = (r_2 \odot E) \oplus Y_{attr} = \boxed{r_2 P(y_{id}) + y_{attr}}$. 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 will take as input the HEC decryption key $d = (\mathsf{sk}_{AH}, x)$ and the escrow Z, and recovers y'_{id}, y'_{attr}, and y' by decrypting the escrows, $(Z_{id}, Z_{attr}, Z_{nf})$ using the secret key, sk_{AH}. Note that, by the correctness property the decryption algorithm for g-semi-encryption, we know that for $Z \in \text{HECENC}(X, y)$, $y' = g(r_3 P(y_{id})) = g(0)$ if and only if $y_{id} \in x$; so if $y' \neq g(0)$, HECDEC outputs $\perp$. Else, we know that $y_{id} \in x$, so HECDEC must somehow determine (1) y_{id} from $y'_{id} = g(y_{id})$, and (2) y_{attr} from $y'_{attr} = g(y_{attr})$. Let us explain how HECDEC can do so.

If g is the identity function (or an efficiently invertible one), then this step is trivial; we will show in Section 6 that we can achieve an additively homomorphic g-semi-encryption scheme where g is the identity function under the decisional composite residuosity assumption using the Camenisch-Shoup cryptosystem.

If, however, g is a one-way injective function, then (1) can be done by looking for $g(y_{id})$ on the list $g(x_1), \ldots, g(x_n)$ where $x_i \in x$ and (2) can only be done by exhaustive search, which is only possible if y_{attr} 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 g^y-semi-encryption scheme, and g^y is a one-way function, they could only achieve attributes from a small space. In Section 6.2 we also give the ElGamal-based formalization.

HECDEC$(hecpar, d, Z)$

1 : **parse** $d = (\mathsf{sk}_E, f_{n,k,\ell}, x)$,
$Z = (Z_{id}, Z_{attr}, Z_{nf})$
2 : $y'_{id} \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{id})$
3 : $y'_{attr} \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{attr})$
4 : $y' \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{nf})$
5 : **if** $y' \neq g(0)$
6 : **return** $\emptyset$
7 : **for** $y_{id} \in x$
8 : **if** $g(y_{id}) = y'_{id}$
9 : **return** (y_{id}, y_{attr})
where $y_{attr} \in domain_{f,y,attr}$
$\wedge\, g(y_{attr}) = y'_{attr}$
10 : **return** $\emptyset$

HECENC$(hecpar, f_{n,k}, x)$

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

HECEVAL$(hecpar, f_{n,k,\ell}, X, y; r_{\hat{Z}})$

1 : **parse** $X = (\mathsf{pk}_{AH}, A_1, ..., A_{n+1})$,
$y = (y_{id}, y_{attr})$,
$r_{\hat{Z}} = (r_{id}, r_{attr}, r_1, r_2, r_3)$
2 : **if** $r_3 = 0$, **return** $\perp$
3 : $E \leftarrow \bigoplus_{i=1}^{n+1} (A_i \odot y_{id}^i)$
4 : $Y_{id} \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, y_{id}; r_{id})$
5 : $Y_{attr} \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, y_{attr}; r_{attr})$
6 : $Z_{id} \leftarrow (r_1 \odot E) \oplus Y_{id}$
7 : $Z_{attr} \leftarrow (r_2 \odot E) \oplus Y_{attr}$
8 : $Z_{nf} = r_3 \odot E$
9 : **return** $Z = (Z_{id}, Z_{attr}, Z_{nf})$

HECDIRECT$(hecpar, X, z)$

1 : **parse** $X = (\mathsf{pk}_{AH}, A_1, \ldots, A_{n+1})$
2 : $z = (z_1, z_2, z_3)$
3 : **if** $z = \emptyset$
4 : $\beta_1 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
5 : $\beta_2 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
6 : $\beta_3 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
7 : **return** $(\mathsf{Enc}(\mathsf{pk}_{AH}, \beta_1),$
8 : $\mathsf{Enc}(\mathsf{pk}_{AH}, \beta_2), \mathsf{Enc}(\mathsf{pk}_{AH}, \beta_3))$
9 : **return** $(\mathsf{Enc}(\mathsf{pk}_{AH}, g(z_1)),$
10 : $\mathsf{Enc}(\mathsf{pk}_{AH}, g(z_2)), \mathsf{Enc}(\mathsf{pk}_{AH}, g(z_3)))$

Fig. 5.1: HEC algorithms

Theorem 3 (HEC consistency of Fig. 5.1). *Our construction in Fig. 5.1 achieves* HEC *consistency in Def. 4.1.*

Theorem 4 (Security of the construction in Fig. 5.1). *Our construction in Fig. 5.1 achieves security of* DIRECTZ *, **Security of** y, and **security of** x **and** y **from third parties**, defined in Definition 11.*

We provide the full proofs of Theorems 3 and 4 in Appendix D.2.

5.3 Efficient Instantiation of HEC Evaluation Proof Ψ_2

In this section we show how to efficiently instantiate a NIZK proof used in the Escrow algorithm in Fig. 4.2 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 $\hat{Z} = \text{HECEVAL}(hecpar, f_{n,k}, X, y; r_{\hat{Z}})$ $\wedge C_y = \mathsf{Com}(y; r_y)$ where $f_{n,k}$ is the watchlist blueprinting function described at the start of this section.

Overview. In Alg. 1, we give the construction of Ψ_2 for HECEVAL. The proof function for this system, PoK_{Ψ_2}, calls ProveRecursive (described in Alg. 2) to prove the correctness of commitment C to E. The input values, (C_z, r_t), on the first call from Alg. 1 to Alg. 2 are initialized to "default" values. For this first call, z takes the value $(y_{id})^{(n+1)/2-1}$ as it is left unspecified in the input. This allows the proof to recursively interact with itself while keeping the interface to PoK_{Ψ_2} simpler.

The most involved part of this process is to prove that ciphertext $E = \bigoplus_{i=0}^{n}(\boxed{a_i} \odot y_{id}^i)$ was computed correctly. The verifier will verify this proof while only knowing a commitment to E, $C = \mathsf{Com}_{AH}(E)$. Naively, this proof would require a correctness proof over the ciphertexts containing each coefficient, $\{\boxed{a_i}\}_{i\in[0\ldots n]}$ where n is the length of the watchlist. The correctness of the final ciphertexts Z_{id}, Z_{attr}, and Z_{nf} will follow from the correctness of E and the additional proofs about committed ciphertexts specified in Section 5.1. To make the proof of correctness of $C = \mathsf{Com}_{AH}(E)$ efficient (taking $O(\log(n))$ instead of $O(n)$ communication) we use a protocol similar to Shamir's [Sha90], recently used in cryptography by Pietrzak [Pie19] and follow-up work [HHKP23]. The idea is to use recursion, where each step halves the degree of the polynomial by computing the random linear combination of the lower and upper half of coefficients.

We will now explain how our ProveRecursive algorithm in Fig. 2 ensures that C is a correct commitment to E. The prover and verifier can both evaluate the polynomial $P(\chi)$ using the encrypted coefficients $\{A_i\}_{i\in[n]}$. We assume the watchlist is padded such that the number of coefficients $(n+1)$ is a power of 2. They can further break P into two parts such that $P(\chi) = P_1(\chi) + P_2(\chi)$ where P_1 contains the terms of P up to degree $(n+1)/2 - 1$ and P_2 contains the rest of the terms from degree $(n+1)/2$ to n. They both also represent P as $P(\chi) = P_1(\chi) + \chi^{(n+1)/2}P_3(\chi)$ such that $P_3(\chi) = P_2/\chi^{(n+1)/2}$. The prover then commits to ciphertexts $\boxed{e_1} = \boxed{P_1(y_{id})}$, $\boxed{e_2} = \boxed{P_2(y_{id})}$, $\boxed{e_3} = \boxed{P_3(y_{id})}$ and to the value $z = y_{id}^{(n+1)/2}$, and then proves that $\boxed{e} = \boxed{e_1} \oplus \boxed{e_2}$ and $\boxed{e_2} = z \odot \boxed{e_3}$. Thus, the prover has reduced the task of proving correctness of committed $\boxed{e} = \boxed{P(y_{id})}$ for a degree-n encrypted polynomial P to the task of proving correctness of (1) committed $\boxed{e_1} = \boxed{P_1(y_{id})}$ and committed $\boxed{e_3} = \boxed{P_3(y_{id})}$, where P_1 and P_3 are

both polynomials of degree $(n+1)/2-1$, and (2) the committed value z is indeed $y_{id}^{(n+1)}/2$. (2) is straightforward with $O(\log n)$ commitments and proofs of correct multiplication of committed values: the prover commits to y_{id} exponentiated by powers of 2 from 2^0 to 2^{n+1}, and uses a proof of multiplication of committed values to prove correctness of each subsequent commitment from the previous one.

For (1), we see that if the prover proved P_1 and P_3 individually, it would fork the proof which would be inefficient. Instead, the prover procures a Σ-protocol-style challenge, α, from the random oracle. The prover and verifier can then represent a fourth polynomial, $P'(\chi) = P_1(\chi) + \alpha P_3(\chi)$ using α and the encrypted coefficients. The prover then computes $\boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_3})$. The prover then computes a commitment (C') to $\boxed{e'}$. By the Schwartz-Zippel Lemma (Lemma 3), if committed $\boxed{e_1} \neq \boxed{P_1(y_{id})}$ or committed $\boxed{e_3} \neq \boxed{P_3(y_{id})}$, then with overwhelming probability over the choice of a random α, committed $\boxed{e'} = \boxed{e_1} \oplus \alpha \odot \boxed{e_2} \neq \boxed{P'(y_{id})}$. Thus, we have reduced proving correctness of committed ciphertext $\boxed{e}$ to proving that committed ciphertext $\boxed{e'} = \boxed{P'(y_{id})}$, where P' is a polynomial of degree $(n+1)/2-1$. This completes the recursive step: it has halved the degree of the encrypted polynomial. After the recursive step is applied $O(\log n)$ times, we will be left with a constant-degree encrypted polynomial P', and correctness of committed ciphertext $\boxed{e'}$ can be proved directly using the protocols for ciphertext commitments.

Our proof system must have the zero-knowledge and extractability properties needed for the proofs of both blueprint hiding (Definition 14) and user privacy (Definitions 15 and 16) for our construction in Fig. 4.2. 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 $g(y)$; g is some function such that $g(y)$, jointly with x is sufficient to compute $f(x,y)$ because there is some efficiently computable function f^* such that $f^*(x, g(y)) = f(x,y)$. In order to achieve straight-line extractability of $g(y)$, our proof system requires that the prover 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 g-semi-encryption scheme ($\Gamma_{sky} = \{\mathsf{KeyGen}_{sky}, \mathsf{Enc}_{sky}, \mathsf{Dec}_{sky}\}$). (Using our notation from Definition 2, the prover retrieves the public key in the sky by querying the setup S_2.)

We explain how we instantiate the NIZK proofs in Algs. 1 and 2 in Section 5.4.

We present the corresponding verification functions for PoK_{Ψ_2} and ProveRecursive ($V_{\Psi_2}^{S_2}$ and VerifyRecursive) in Algs. 3 and 4.

Theorem 5. *Our scheme in Algs. 2 and 1 are complete and ZK (Def. 1).*

Theorem 6. *The* ProveRecursive *function in Alg. 2 is black-box (BB) simulation extractable with respect to Def. 2 for the relation* $R((C, C_{y_{id}}, X, n), (O, O_{y_{id}}, y_{id})) = 1$ *iff* $C = \mathsf{Com}_{AH}(\mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, \bigoplus_{i=0}^{n}(\boxed{a_i} \odot y_{id}^i); O)) \wedge C_{y_{id}} = \mathsf{Com}(y, O_{y_{id}})$.

Theorem 7 (g^*-BB-PSL for Ψ_2). *If* ProveRecursive *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*

Algorithm 1 $\mathsf{PoK}^{S_2}_{\Psi_2}(hecpar, f, X, y, r_y, r_{\hat{Z}}) \rightarrow \pi$

parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in[0\ldots n]}); r_{\hat{Z}} = (r_1, r_2, r_3, r_{\mathsf{id}}, r_{\mathsf{attr}})$
$(y_{id}, y_{attr}) \leftarrow y; (C_{id}, O_{id}) = \mathsf{Com}(y_{id})$
$E \leftarrow \boxed{e} = \bigoplus_{i=0}^{n}(\boxed{a_i} \odot y_{id}^{i})$
$Z_{id} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{id}; r_{\mathsf{id}}) \oplus (r_1 \odot \boxed{e})$
$Z_{attr} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{attr}; r_{\mathsf{attr}}) \oplus (r_2 \odot \boxed{e})$
$Z_{nf} = r_3 \odot \boxed{e}$
$Z = (Z_{id}, Z_{attr}, Z_{nf})$
$C_y \leftarrow \mathsf{Com}(y; r_y)$
$\mathsf{pk}_{sky} \leftarrow S_2(1^{\lambda}); C_{sky} = \mathsf{Enc}_{sky}(\mathsf{pk}_{sky}, y; r_{sky})$
$\pi_{sky} = \mathsf{NIZK}[y, r_y, r_{sky} : C_{sky} = \mathsf{Enc}(\mathsf{pk}_{sky}, y; r_{sky})]$
$(C, O) \leftarrow \mathsf{Com}_{AH}(E)$
$\pi_{rec} \leftarrow \mathsf{ProveRecursive}(C, O, C_{id}, O_{id}, y_{id}, X, n)$
$Y_{\mathsf{id}} \leftarrow \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{id}; r_{\mathsf{id}})$
$Y_{\mathsf{attr}} \leftarrow \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{attr}; r_{\mathsf{attr}})$
$(C_{Y_{\mathsf{id}}}, O_{Y_{\mathsf{id}}}) \leftarrow \mathsf{Com}_{AH}(Y_{\mathsf{id}})$
$(C_{Y_{\mathsf{attr}}}, O_{Y_{\mathsf{attr}}}) \leftarrow \mathsf{Com}_{AH}(Y_{\mathsf{attr}})$
$\pi_{\hat{Z}} \leftarrow \mathsf{NIZK}[O, O_{Y_{\mathsf{id}}}, O_{Y_{\mathsf{attr}}}, r_{\hat{Z}}, E, Y, O_{id}, y :$
$\quad \wedge \mathsf{Com}(y_{id}, O_{id}) = C_{id} \wedge \mathsf{Com}(y, r_y) = C_y \wedge y = (y_{attr}, y_{id})$
$\quad \wedge \mathsf{Com}_{AH}(E, O) = C$
$\quad \wedge \mathsf{Com}_{AH}(Y_{\mathsf{id}}, O_{Y_{\mathsf{id}}}) = C_{Y_{\mathsf{id}}} \wedge \mathsf{Com}_{AH}(Y_{\mathsf{attr}}, O_{Y_{\mathsf{attr}}}) = C_{Y_{\mathsf{attr}}}$
$\quad \wedge Y_{\mathsf{id}} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{id}; r_{\mathsf{id}})$
$\quad \wedge Y_{\mathsf{attr}} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{attr}; r_{\mathsf{attr}})$
$\quad \wedge Z_{id} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{id}; r_{\mathsf{id}}) \oplus (r_1 \odot \boxed{e})$
$\quad \wedge Z_{attr} = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y_{attr}; r_{\mathsf{attr}}) \oplus (r_2 \odot \boxed{e})$
$\quad \wedge Z_{nf} = r_3 \odot \boxed{e} \wedge r_3 \neq 0]$
return $C, C_{id}, \pi_{rec}, \pi_{\hat{Z}}, \pi_{sky}, C_{sky}, C_{Y_{\mathsf{id}}}, C_{Y_{\mathsf{attr}}}$

Algorithm 2 $\mathsf{ProveRecursive}(C, O, C_y, r_y, y, X, n, (C_z, r_z)) \to \pi$

parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in[0\ldots n]})$,

$(C) = \mathsf{Com}_{AH}(\boxed{e}; O)$ where $\boxed{e} = \bigoplus_{i=0}^{n} \boxed{a_i} \odot y^i = \boxed{\sum_{i=0}^{n} y^i a_i}$,

and if recursing, $C_z = \mathsf{Com}(z; r_z)$, where $z = y^{(n+1)/2}$

If the degree of the polynomial is low enough, prove its computation directly

1: **if** $n = 1$,

2: $\quad \pi_2 \leftarrow \mathsf{NIZK}[y, r_z, y, O, r_y : \mathsf{Com}(y, r_y) = C_y$

3: $\quad\quad \wedge \mathsf{Com}_{AH}(\boxed{e}, O) = C \wedge \boxed{e} = \bigoplus_{i=0}^{n} \boxed{a_i} \odot y^i$

4: $\quad\quad \wedge \mathsf{Com}(y; r_z) = C_z]$

5: $\quad$ **return** π_2

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.

6: $(C_1, O_1) \leftarrow \mathsf{Com}_{AH}(\boxed{e_1})$ where $\boxed{e_1} = \bigoplus_{i=0}^{(n+1)/2-1} \boxed{a_i} \cdot y^i = \boxed{\sum_{i=0}^{(n+1)/2-1} y^i a_i}$

Next, commit to the upper half of the polynomial

7: $(C_2, O_2) \leftarrow \mathsf{Com}_{AH}(\boxed{e_2})$ where $\boxed{e_2} = \bigoplus_{i=0}^{(n+1)/2-1} \boxed{a_{i+(n+1)/2}} \odot y^{i+(n+1)/2}$

$= \boxed{\sum_{i=0}^{(n+1)/2-1} y^{i+(n+1)/2} a_{i+(n+1)/2}}$

Lastly, commit to the upper half of the polynomial with the degree lowered by half

8: $(C_3, O_3) \leftarrow \mathsf{Com}_{AH}(\boxed{e_3})$ where $\boxed{e_3} = \bigoplus_{i=0}^{(n+1)/2-1} \boxed{a_{i+(n+1)/2}} \odot y^i$

$= \boxed{\sum_{i=0}^{(n+1)/2-1} y^i a_{i+(n+1)/2}}$

Put the current transcript (τ) of the proof (including all inputs) into the random oracle to get a value, α.

9: $\alpha \leftarrow H(\tau)$

Compute the encryptions of the new coefficients for a reduced degree polynomial

10: $X' \leftarrow (\mathsf{pk}_{AH}, \{\boxed{a'_i}\}_{i\in[(n+1)/2]})$ where $\boxed{a'_i} = \boxed{a_i} \oplus (\boxed{a_{i+(n+1)/2-1}} \odot \alpha)$

Compute a new evaluation over this reduced degree polynomial

11: $(C', O') \leftarrow \mathsf{Com}_{AH}(\boxed{e'})$ where $\boxed{e'} = \bigoplus_{i=0}^{(n+1)/2} \boxed{a'_i} \odot y^i$

If we're not recursing, compute the correct value to scale e_2 with..

12: **if** $C_z = O_z = \perp$,

13: $\quad (C_z, O_z) \leftarrow \mathsf{Com}(y^{(n+1)/2})$

Prove that this new commitment (C') is consistent with C,C_1,C_2, and C_3.

14: $\pi_\alpha \leftarrow \mathsf{NIZK}[O, O_1, O_2, O_3, O', r_y, y, O_z, z, r_z, \boxed{e}, \boxed{e_1}, \boxed{e_2}, \boxed{e_3} :$

15: $\quad \mathsf{Com}_{AH}(\boxed{e}, O) = C \wedge \mathsf{Com}_{AH}(\boxed{e'}, O') = C'$

16: $\quad\quad \wedge \forall 1 \le i \le 3 : \mathsf{Com}_{AH}(\boxed{e_i}, O_i) = C_i$

17: $\quad\quad \wedge \mathsf{Com}_{AH}(\boxed{e'}, O') = C'; \mathsf{Com}(z, O_z) = C_z;$

18: $\quad\quad \wedge \boxed{e} = \boxed{e_1} \oplus \boxed{e_2}$

19: $\quad\quad \wedge \boxed{e_2} = z \odot \boxed{e_3}$

20: $\quad\quad \wedge \boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_3})]$

Setup the recursion and call it.

21: $z' = y^{(n+1)/4}; (C'_z, r'_z) = \mathsf{Com}(z')$

22: $\pi_z \leftarrow \mathsf{NIZK}[z, z', r_z, r'_z : \mathsf{Com}(z, r_z) = C_z \wedge \mathsf{Com}(z', r'_z) = C'_z \wedge z = z' * z']$

23: **return** $\big(C, C_1, C_2, C_3, C_z, \pi_z, \pi_\alpha, \mathsf{ProveRecursive}(C', O', C_y, r_y, y, X', (n+1)/2 - 1, (C'_z, r'_z))\big)$

Algorithm 3 $\mathsf{V}_{\Psi_2}^{S_2}(hecpar, f, X, C_y, Z, \pi) \to \{0,1\}$

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

Algorithm 4 $\mathsf{VerifyRecursive}(C, C_y, X, n, (C_z)) \to \{0,1\}$

parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in[0\ldots n]})$,
1: **parse** $\pi = \left(C, C_1, C_2, C_3, C_z, \pi_z, \pi_C, \pi'\right)$
2: **if** $n = 1$,
3: Verify π_2
4: **return** 0 if π_2 didn't verify, otherwise, **return** 1
Random oracle hash current transcript (τ) of the proof (including all inputs)
5: $\alpha \leftarrow H(\tau)$
6: Verify π_C
7: Verify π_z
8: Verify π' by recursing into VerifyRecursive.
9: If any proof failed to verify, **return** 0, otherwise **return** 1

$C = \mathsf{Com}_{AH}(\mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, \bigoplus_{i=0}^{n}(\boxed{a_i} \odot y_{id}^i); O)) \wedge C_{y_{id}} = \mathsf{Com}(y, O_{y_{id}}))$ *and if* $\Gamma_{sky} = \{\mathsf{KeyGen}_{sky}, \mathsf{Enc}_{sky}, \mathsf{Dec}_{sky}\}$ *is a semantically secure g-semi-encryption scheme, our* Ψ_2 *proof is a* g^**-BB-PSL protocol, where* $g^*(y, r_{\hat{Z}}) = g(y)$.

We prove Thms. 5, 6, and 7 next. 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.

Proof of Thm. 5 (Correctness and ZK). Correctness 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 Section 5.1 and constructed in Section 6. 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. We show the simulator for PoK_{Ψ_2} and ProveRecursive in Algs. 6 and 5 for completeness in Section 5.3. 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 3. We will use this in our proof of black-box simulation extractability proof for Alg. 2 in Thm. 6

Lemma 3 (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 4

Lemma 4 (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 $(dp + dq)/n$ 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.*

Proof of Lemma 4. Using the Chinese remainder theorem, we know that $\mathbb{Z}_n$ decomposes into $\mathbb{Z}_q \times \mathbb{Z}_p$. For $r(\chi)$ to be equal to $r'(\chi)$ it must be that $P(\chi) = r(\chi) - r'(\chi) = 0$. We can decompose $P(\chi)$ into $\mathbb{Z}_p$ and $\mathbb{Z}_q$ by defining $P_{\mathbb{Z}_q}$ and $P_{\mathbb{Z}_p}$ such that $P_{\mathbb{Z}_p}(\chi)$ is in $\mathbb{Z}_p[\chi]$ and $P_{\mathbb{Z}_q}(\chi)$ is in $\mathbb{Z}_q[\chi]$ and $P(\chi) = P_{\mathbb{Z}_p}(\chi) * P_{\mathbb{Z}_q}(\chi)$. We know that if $P(\chi)$ equals 0 in $\mathbb{Z}_n$, then it there must be unique (x, y) in $\mathbb{Z}_q \times \mathbb{Z}_p$ such that the product of x and y is $0 \mod n$ and $x = P_{\mathbb{Z}_p}(\chi)$ and $y = P_{\mathbb{Z}_q}(\chi)$ $P_{\mathbb{Z}_p}$. Thus, $xy = kn$ for some k. Because $\mathbb{Z}_n$ and $\mathbb{Z}_q \times \mathbb{Z}_p$ are isomorphic, and $p|n, q|n$, we know that drawing randomly from $\mathbb{Z}_n$ gives us random points in $\mathbb{Z}_q$ and $\mathbb{Z}_p$. Let us assume that either x or y must be zero for $P(\chi)$ to be zero. There are only $d*p+d*q$ evaluations of $P(\chi)$ in $\mathbb{Z}_n$ that make either $P_{\mathbb{Z}_p}(\chi)$ or $P_{\mathbb{Z}_q}(\chi)$ equal to zero. Thus, with our assumption that one of these polynomials ($P_{\mathbb{Z}_p}(\chi)$ or $P_{\mathbb{Z}_q}(\chi)$) must be zero, the probability of choosing one of these randomly is $(dp + dq)/n = d/q + d/p$ which is negligible. We know that $P(\chi)$ cannot be zero without either $P_{\mathbb{Z}_p}(\chi)$ or $P_{\mathbb{Z}_q}(\chi)$ being zero because if $2q > p > q$, we know that for $P(\chi)$ to be zero, then $x * y = kn$. Because $p|kn$ and $q|kn$, we know that $p|xy$ and $q|xy$. But we know that $p \nmid y$ since $y \in \mathbb{Z}_q$ and $p > q$, and we know that $pq \nmid x$ since $n = pq > 2q$. Thus, either $P_{\mathbb{Z}_p}(\chi)$ or $P_{\mathbb{Z}_q}(\chi)$ must be zero and from our earlier result, only $d * p + d * q$ evaluations of $P(\chi)$ are zero and thus we get the probability from Lemma 4.

Proof of Thm. 6 (Simulation extractability of ProveRecursive*).* 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 Section 5.1 and constructed in Section 6. We can use the simulator in Alg. 6 in Section 5.3 for this reduction. Because our simulator are zero knowledge, the BB-simulation-extractability adversary gets no advantage when given these proofs.

First, we'll start by proving why each C_z commitment is correctly committed to $y^{(n+1)/2}$. We can prove this by induction. We can see that in the final recursion (when $n = 1$), C_z is proven to open to y (from C_y) with π_z (on line 2). This forms a base-case for our induction. If we can prove that a C_z is correctly computed at any given step based on the assumption that C'_z is correctly computed in the next

step, we'll have proven that each C_z is correctly computed. At each step before the last, we see that C_z is proven to open to $(z')^2$ where C'_z is committed to z'. If we assume that C'_z (the commitment for the next step) is correctly computed i.e. a commitment to $y^{(n+1)/4}$ then we know that the C_z in the current step is a commitment to $y^{(n+1)/2}$. Thus, by induction, at each step, C_z must be a commitment to $z = y^{(n+1)/2}$ at each step.

Next, 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 $n = 3$, we know that in the next recursive step, $n' = (n+1)/2 - 1 = 1$, which means we will prove that C' is correctly computed in line 3 of Alg. 2 (i.e.: $e' = P'(y) = \sum_{i=0}^{(n+1)/2-1} a'_i y^i$). 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 11 of Alg. 1) 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_2. 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_2. We now rewrite these polynomials and fix y to reform these as: $q(\chi) = e_1 + \chi e_3$ and $q'(\chi) = \sum_{i=0}^{(n+1)/2-1} y^i a_i + \sum_{i=0}^{(n+1)/2-1} \chi y^i a_{i+(n+1)/2}$. 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 3) that the probability of this occurring when $q(\chi)$ is distinct from $q'(\chi)$ is $1/q$. 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 $e_1 = \sum_{i=0}^{(n+1)/2-1} y^i a_i$ and $e_3 = \sum_{i=0}^{(n+1)/2-1} y^i a_{i+(n+1)/2}$. This is because e_1 is the 0-degree coefficient in $q(\chi)$ and $\sum_{i=0}^{(n+1)/2-1} y^i a_i$ is the 0-degree coefficient in $q'(\chi)$ (with similar reasoning for e_3 and $\sum_{i=0}^{(n+1)/2-1} y^i a_{i+(n+1)/2}$ for being the 1-st degree coefficient of $q(\chi)$ and $q'(\chi)$). We then see that π_C proves that $e_2 = e_3^z$. Thus, $e_2 = e_3^{y^{(n+1)/2}}$ and since we proved e_3 correctly with π_C, we now know that $e_2 = \sum_{i=0}^{(n+1)/2-1} \chi y^{i+(n+1)/2} a_{i+(n+1)/2}$. 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.

Proof of Thm. 7 (BB-PSL). We assume in this theorem that we can extract a witness for the relation R_P in a black-box way (Thm. 6). Thus, we know

that the ciphertext (C_{sky}) containing $g(y)$ is correct, and thus, we can extract $g(y) = g^*(y, r_{\hat{Z}})$ by decrypting this ciphertext.

Theorem 8 (g^*-BB-PSL for Ψ_2). *Our Ψ_2 proof is a g^*-BB-PSL protocol, where $g^*(y, r_{\hat{Z}}) = g(y)$.*

Proof of Thm. 8 (BB-PSL). Because we know that $\mathsf{ProveRecursive}$ 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 g-semi-encryption scheme from Thm. 6, this is clear from Thm. 7.

Algorithm 5 $\mathsf{SimPoK}_{\Psi_2}^{S_2}(hecpar, f, X, C_y, \hat{Z}) \rightarrow \pi$

parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in[0\ldots n]}); r_{\hat{Z}} = (r_1, r_2, r_3)$
$y \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
$y_{id} \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}; (C_{id}, O_{id}) = \mathsf{Com}(y_{id})$
$\mathsf{pk}_{sky} \leftarrow S_2(1^\lambda); C_{sky} = \mathsf{Enc}_{sky}(\mathsf{pk}_{sky}, y; r_{sky})$
$\pi_{sky} = \mathsf{Sim}[C_{sky} = \mathsf{Enc}(\mathsf{pk}_{sky}, y; r_{sky})]$
$E \leftarrow \mathsf{Enc}_{AH}(e_1)$ where $e_1 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
$(C, O) \leftarrow \mathsf{Com}_{AH}(E)$
$\pi_{rec} \leftarrow \mathsf{SimProveRecursive}(C, C_{id}, X, n)$
$Y \leftarrow \mathsf{Enc}_{AH}(e_2)$ where $e_2 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
$(C_Y, O_Y) \leftarrow \mathsf{Com}_{AH}(Y)$
$\pi_{\hat{Z}} \leftarrow \mathsf{Sim}[\mathsf{Com}(y_{id}, O_{id}) = C_{id} \wedge \mathsf{Com}(y, r_y) = C_y \wedge y = y_{attr}||y_{id}$
  $\wedge \mathsf{Com}_{AH}(E, O) = C \wedge \mathsf{Com}_{AH}(Y, O_Y) = C_Y$
  $\wedge Y = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y; r_3)$
  $\wedge\big((E \odot r_1) \oplus Y, E \odot r_2\big) = \hat{Z}]$
return $C, C_{id}, \pi_{rec}, \pi_{\hat{Z}}, \pi_{sky}, C_{sky}$

Algorithm 6 $\mathsf{SimProveRecursive}(C, C_y, X, n, (C_z)) \to \pi$

parse $X = (\mathsf{pk}_{AH}, \{\boxed{a_i}\}_{i\in\{0\ldots n\}})$,
if $n = 1$,
  $\pi_2 \leftarrow \mathsf{Sim}[\mathsf{Com}(y, r_y) = C_y$
    $\wedge\, \mathsf{Com}_{AH}(\boxed{\mathrm{e}}, O) = C \wedge \boxed{\mathrm{e}} = \bigoplus_{i=0}^{n} \boxed{a_i} \odot y^i]$
    $\wedge\, \mathsf{Com}(y; r_z) = C_z$
  return π_2
$(C_1, O_1) \leftarrow \mathsf{Com}_{AH}(\boxed{e_1})$ where $e_1 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
$(C_2, O_2) \leftarrow \mathsf{Com}_{AH}(\boxed{e_2})$ where $e_2 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
$(C_3, O_3) \leftarrow \mathsf{Com}_{AH}(\boxed{e_3})$ where $e_3 \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
$\alpha \leftarrow H(\tau)$
$X' \leftarrow (\mathsf{pk}_{AH}, \{\boxed{a'_i}\}_{i\in[0\ldots(n+1)/2]})$ where $\boxed{a'_i} = \boxed{a_i} \oplus \boxed{a_{i+(n+1)/2-1}} \odot \alpha$
$(C', O') \leftarrow \mathsf{Com}_{AH}(\boxed{e'})$ where $e' \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}$
if $C_z = O_z = \perp$,
  $z \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}(C_z, O_z) \leftarrow \mathsf{Com}(z)$
$\pi_C \leftarrow \mathsf{Sim}[$
  $\mathsf{Com}_{AH}(\boxed{e}, O) = C \wedge \mathsf{Com}_{AH}(\boxed{e'}, O') = C'$
  $\wedge\, \forall 1 \le i \le 3 : \mathsf{Com}_{AH}(\boxed{e_i}, O_i) = C_i$
  $\wedge\, \mathsf{Com}_{AH}(\boxed{e'}, O') = C'; \mathsf{Com}(z, O_z) = C_z$
  $\wedge\, \boxed{e} = \boxed{e_1} \oplus \boxed{e_3}$
  $\wedge\, \boxed{e_3} = zy \odot \boxed{e_2}$
  $\wedge\, \boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_2})]$
$z' \leftarrow\!\!\$; (C'_z, r'_z) = \mathsf{Com}(z')$
$\pi_z \leftarrow\!\!\$\ \mathcal{M}_{\mathsf{pk}_{AH}}\mathsf{Sim}[\mathsf{Com}(z, r_z) = \mathsf{Com}(z', r'_z) = C_z \wedge z = z' * z']$
return $(C, C_1, C_2, C_3, C_z, \pi_z, \pi_C, \mathsf{SimProveRecursive}(C', C_y, X', (n + 1)/2 - 1, (C'_z)))$

5.4 Construction of NIZKs in ψ_2 proof scheme

In Algs. 1 and 2, we perform the following four proofs:

$$\pi_{\hat{Z}} \leftarrow \mathsf{NIZK}[O, O_Y, r_1, r_2, r_3, E, Y, O, O_{id}, y_{id}, y :$$
$$\wedge\, \mathsf{Com}(y_{id}, O_{id}) = C_{id} \wedge \mathsf{Com}(y, r_y) = C_y \wedge y = (y_{attr}, y_{id})$$
$$\wedge\, \mathsf{Com}_{AH}(E, O) = C \wedge \mathsf{Com}_{AH}(Y, O_Y) = C_Y$$
$$\wedge\, Y = \mathsf{Enc}_{AH}(\mathsf{pk}_{AH}, y; r_3)$$
$$\wedge\, \big((E \odot r_1) \oplus Y, E \odot r_2\big) = \hat{Z}]$$

$$\pi_\alpha \leftarrow \mathsf{NIZK}[O, O_1, O_2, O_3, O', r_y, y, O_z, z, r_z, \boxed{e}, \boxed{e_1}, \boxed{e_2}, \boxed{e_3} :$$
$$\mathsf{Com}_{AH}(\boxed{e}, O) = C \wedge \mathsf{Com}_{AH}(\boxed{e'}, O') = C'$$
$$\wedge\, \forall 1 \le i \le 3 : \mathsf{Com}_{AH}(\boxed{e_i}, O_i) = C_i$$
$$\wedge\, \mathsf{Com}_{AH}(\boxed{e'}, O') = C'; \mathsf{Com}(z, O_z) = C_z;$$
$$\wedge\, \boxed{e} = \boxed{e_1} \oplus \boxed{e_3}$$
$$\wedge\, \boxed{e_3} = z \odot \boxed{e_2}$$
$$\wedge\, \boxed{e'} = \boxed{e_1} \oplus (\alpha \odot \boxed{e_2})]$$

$$\pi_2 \leftarrow \mathsf{NIZK}[y, r_z, y, O, r_y : \mathsf{Com}(y, r_y) = C_y$$
$$\wedge\, \mathsf{Com}_{AH}(\boxed{e}, O) = C \wedge \boxed{e} = \bigoplus_{i=0}^{n} \boxed{a_i} \odot y^i$$
$$\wedge\, \mathsf{Com}(y; r_z) = C_z]$$

$$\pi_z \leftarrow \mathsf{NIZK}[z, z', r_z, r'_z : \mathsf{Com}(z, r_z) = \mathsf{Com}(z', r'_z) = C_z \wedge z = z' * z']$$

For $\pi_{\hat{Z}}$, we see that we need to construct a proof for $\hat{Z} = \big((E \odot r_1) \oplus Y, E \odot r_2\big)$. We can prove each element in $\hat{Z}$ separately. Proving $\hat{Z}_1 = r_2 \odot E$ is straightforward. The prover creates a commitment to r_2 and then invokes our multiplication protocol $\mathsf{Prove}_{AH}^{\mathsf{multiply}}$ for our ciphertext commitments and scalar commitments.

For proving $\hat{Z}_2 = r_1 \odot E \oplus Y$, the prover needs to create intermediate commitments to $r_1 \odot E$ and Y, C_E and C_Y. The proof for $r_1 \odot E$ will be formed similar to our proof for $r_2 \odot E$, i.e. by committing to r_1 and then invoking our $\mathsf{Prove}_{AH}^{\mathsf{multiply}}$ protocol. The prover then proves that C is committed to the product of C_E and C_Y. This can be done using our $\mathsf{Prove}^{\mathsf{add}}$ function for commitments to ciphertexts.

For π_α, we have a number of addition and multiplication proofs. The prover is trying to show that $\boxed{e} = \boxed{e_1} \oplus \boxed{e_3}$ and $\boxed{e_3} = \boxed{ze_2}$. This is straightforward as the prover can apply $\mathsf{Prove}^{\mathsf{add}}$ and $\mathsf{Prove}^{\mathsf{multiply}}$ directly to C_1, C_2, C_3, and C_z. For the last proof $\boxed{e'} = \boxed{e_1} \oplus \boxed{\alpha e_2}$, the prover will need to create an intermediate commitment to $\boxed{\alpha e_2}$ and use $\mathsf{Prove}^{\mathsf{multiply}}$ to prove that it was computed with C_2, C_1 and α (we can create a canonical commitment to α to reuse our multiplication protocol as-is). We then compute $\mathsf{Prove}^{\mathsf{add}}$ on this intermediate commitment and C_1 to prove the final product contained in C'.

For π_2, the prover performs a number of intermediate commitments for each $i \in [n]$. The prover computes and proves intermediate commitments to each $H_i = y^i \odot \boxed{a_i}$ and then computes and proves intermediate commitments to larger products of the elements, $D_i = \bigoplus_{j=1}^{i} \boxed{a_i y^i}$ for $i \in [n]$, using the previous commitment to prove the next, i.e. $D_i = D_{i-1} \oplus H_i$.

Note that this could be done much more efficiently by having the verifier compute the homomorphic operation on the commitment themselves, but using intermediate commitments and proofs assumes less about our underlying commitments. While this is easy to do with Pedersen commitments, the size of value committed to by Damgård-Fujisaki commitments grows as homomorphic operations are performed on them. Having the prover use $\mathsf{Prove}^{\mathsf{add}}$ and $\mathsf{Prove}^{\mathsf{multiply}}$ ensures that our values stay low as discussed in Section 6.3 in Remark 1.

For π_z, the prover creates commitment C'_z to $y^{(n+1)/4}$ and proves via $\mathsf{Prove}^{\mathsf{multiply}}$ that the value in this commitment multiplied by itself is equivalent to C_z.

5.5 Multi-attribute HEC scheme

In this section, we provide a HEC scheme that satisfies Def. 9 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 (discussed in Section 8).

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:

$$f_{n,k,\ell}(\mathbf{x}, y) = \begin{cases} y & y_{id} \in \mathbf{x} \\ \emptyset & \text{otherwise} \end{cases} \tag{1}$$

We construct a HEC scheme for this function in Fig. 5.2. In our previous construction in Alg. 1 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. 5.2, 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 are only decryptable 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 Section 5 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.

Fig. 5.2: Multi-attribute HEC functions

HECEVAL$(hecpar, f_{n,k,\ell}, X, y; r_{\hat{Z}})$

1 : **parse** $X = (\mathsf{pk}_{AH}, A_1, ..., A_{n+1})$,
$y = (y_{id}, y_1^{attr}, ..., y_\ell^{attr})$,
$r_{\hat{Z}} = (\{r_{E,1}, ..., r_{E,\ell}\}, \{r_1, ..., r_\ell\}, r_{nf})$
2 : $E \leftarrow \bigoplus_{i=1}^{n+1}(A_i \odot y_{id}^i)$
3 : $\forall i \in [\ell] : Y_i \leftarrow \mathsf{Enc}(\mathsf{pk}_{AH}, y_i^{attr}; r_i)$
4 : $\forall i \in [\ell] : Z_i \leftarrow ((r_{E,i} \odot E) \oplus Y_i$
5 : $Z_{nf} = r_{nf} \odot E$
6 : **return** $Z = (\{Z_1, ..., Z_\ell\}, Z_{nf})$

HECDEC$(hecpar, d, Z)$

1 : **parse** $d = (\mathsf{sk}_E, f_{n,k,\ell}, x)$,
$Z = (\{Z_1, ..., Z_\ell\}, Z_{nf})$
2 : $\forall i \in [\ell] : y_{g,i}^{attr} \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_i)$
3 : $y_g \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_{nf})$
4 : **if** $y_g \neq g(0)$
5 : **return** $\emptyset$
6 : **for** $y_{id} \in x$
7 : **if** $g(y_{id}) = y_g$
8 : **return** $(y_{id}, y_1^{attr}, ..., y_\ell^{attr})$
where $\forall i \in [\ell] : y_i^{attr} \in domain_{f,y}$
$\wedge\, g(y_i^{attr}) = y_{g,i}^{attr}$
9 : **return** $\emptyset$

6 Constructions of Commitments to Ciphertexts

We first define variants of ElGamal and Camenisch-Shoup encryption, see 6.1. We then construct commitments to ciphertexts and associated proof systems for adding and multiplying ElGamal ciphertexts and Camenisch-Shoup ciphertexts. Note that (Lifted) ElGamal is a g-semi-encryption as defined in Section 5 with message space $\mathcal{M}_{\mathsf{pk}} = \mathbb{Z}_p$ and $g(x) = h^m \mod p$. 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 g-semi-encryption where 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 Section 6.2 we construct commitments to ElGamal ciphertexts. In Section 6.3 we construct commitments to Camenisch-Shoup ciphertexts.

6.1 Encryption Schemes

We review (Lifted) ElGamal encryption in Fig. 6.1a. We include an extra generator (h) for the lifting to exponents in ElGamal so that we can draw parallels between ElGamal and Camenisch-Shoup (Lifted ElGamal encryption generally uses the default generator, g). The scheme is still correct and secure if $g = h$. We also slightly modify Camenisch-Shoup encryption in Fig. 6.1b, replacing some values (parameter $g \in \mathbb{Z}_n$ and ciphertext c) with their absolute values.

Modifying Camenisch-Shoup allows us to ensure the elements of our Camenisch-Shoup ciphertexts exist in a "commitment-friendly" group $\mathcal{G}_{CS}$. We also define a similar group ($\mathcal{G}_{ElG}$) for ElGamal ciphertexts. The two commitment schemes are very similar at a high-level and only differ due to limitations with the *eqrep-n^** protocol which is the protocol we use to prove relations between the ciphertexts in Camenisch-Shoup commitments. Namely this limitation is that the *eqrep-n^** protocol only holds for the absolute values of group elements. The *eqrep-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.

Description of $\mathcal{G}_{ElG}$ We can see that g and h in ElGamal are both elements of $\mathcal{G}_{ElG}$ where $\mathcal{G}_{ElG}$ is a cyclic group of prime order p in which the decisional and computational Diffie-Hellman assumptions are hard. This can be realized by taking the group of quadratic residues of $\mathbb{Z}_q$ where q is a safe prime, or using elliptic curves. Note that this is an important advantage of the ElGamal approach in practice, as g and h can be created by hashing into the group. As commonly the case secure setup ceremonies for RSA-like groups are a lot more involved. See for instance [KMV23].

Description of $\mathcal{G}_{CS}$ To understand $\mathcal{G}_{CS}$ we have to define the absolute value function, shown in Equation 2:

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

Subgroup Generators g and h are both in the group $\mathcal{G}_{\mathsf{CS}} = \{|x| : x \in QR_{n^2} \wedge x < n^2/2\}$ We see that g is in $\mathcal{G}_{CS}$ 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 $\mathcal{G}_{CS}$. We prove that h is $\in QR_{n^2}$ using Lemma 6 and $h \in \mathcal{G}_{CS}$ follows from the fact that $|1+n| = 1+n$. We also see that $\mathcal{G}_{CS}$ comprises 1/4 of $\mathbb{Z}^*_{n^2}$ with an equal number of elements (disregarding 1) in QR_{n^2} and $\mathsf{QNR}^{+1}_{n^2}$. From Lemma 6 in Appendix A and the fact that g is in QR_{n^2}, we can see that both elements of our modified Camenisch-Shoup ciphertexts are in $\mathcal{G}_{CS}$. Thus, if we can create commitments to elements of $\mathcal{G}_{CS}$, we can use them to commit to our modified Camenisch-Shoup ciphertexts and construct the associated protocols for multiplication and exponentiation.

In Section 6.3, we'll see that ensuring that Camenisch-Shoup ciphertexts are in $\mathcal{G}_{CS}$ is useful because $\mathcal{G}_{CS}$ is cyclic (which helps with our hiding and ZK proofs) and also $(-1)x = x$ for elements in $\mathcal{G}_{CS}$. This is important because it means that using *eqrep-n^** (as defined in 2.3) to prove relations between $\mathcal{G}_{CS}$ 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. 6.1b 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 $m = (((c_1^2/(c_0^2)^x)^t \mod n^2) - 1)/n$ where the first operation the decryptor does it square both elements of the ciphertext and the fact that $|x|^2 = x^2 \in \mathbb{Z}_{n^2}$.

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 $m + m'$ in both encryption schemes (after taking the absolute value in Camenisch-Shoup). Also, exponentiation is similar, i.e. $((g^r)^y, (k^r h^m)^y)$ is a valid encryption of ym in both encryption schemes. Thus, if we can commit to elements of $\mathcal{G}_{ElG}$ and $\mathcal{G}_{CS}$ 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 Section 6.2 and commitments to ciphertexts in Camenisch-Shoup ciphertexts in Section 6.3.

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

Correctness of simplified Camenisch-Shoup in Fig. 6.1a. 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.

Fig. 6.1: Encryption schemes

$\mathsf{Setup}(1^\lambda) \to params$

1 : $g, h \leftarrow\$ \mathcal{G}$

2 : **return** $g, h, \mathcal{G}$

$\mathsf{KeyGen}(params) \to (\mathsf{pk}, \mathsf{sk})$

1 : $x \leftarrow\$ \mathbb{Z}_p$;

2 : **return** $\mathsf{pk} \leftarrow g^x, \mathsf{sk} \leftarrow x$;

$\mathsf{Enc}(\mathsf{pk} = k, m) \to c$

1 : $r \leftarrow\$ \mathbb{Z}_p$;

2 : **return** $c = (g^r, k^r h^m)$

$\mathsf{Dec}(\mathsf{sk}, c = (c_0, c_1)) \to M$

1 : $z = c_0^{\mathsf{sk}} = k^r$

2 : **return** $c_1/z = M = h^m$

(a) Lifted ElGamal

$\mathsf{Setup}(1^\lambda) \to params$

1 : Sample p', q' be two $O(\lambda)$-bit SG primes.

2 : $p = 2p' + 1, q = 2q' + 1, n = pq, g' \leftarrow\$ \mathbb{Z}_{n^2}$,

3 : $g = |(g')^{2n}|, h = (1 + n)$,

4 : **return** $params = (n, g, h)$

$\mathsf{KeyGen}(params) \to (\mathsf{pk}, \mathsf{sk})$

1 : $\mathsf{sk} = x \leftarrow\$ [n^2/4], \mathsf{pk} = k = g^x \mod n^2$

2 : **return** pk, sk

$\mathsf{Enc}(\mathsf{pk}, m \in [n]) \to c$

1 : $r \leftarrow\$ [n/4]$,

2 : **return** $c = (|g^r|, |k^r h^m|) \mod n^2$

$\mathsf{Dec}(\mathsf{sk}, c = (c_0, c_1)) \to m$

1 : $t = 2^{-1} \mod n$

2 : **return** $m = (((c_1/c_0^x)^{2t} \mod n^2) - 1)/n$

(b) Simplified Camenisch-Shoup

CPA security of simplified Camenisch-Shoup in Fig. 6.1a. 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 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.

6.2 Commitments to $\mathcal{G}_{ElG}$ elements and ElGamal ciphertexts

In this section, we introduce commitments to group elements (in $\mathcal{G}_{ElG}$) and then construct a commitment scheme to ElGamal ciphertext in Fig. 6.3 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 Section 6.1. In this section, g and h refer to commitment bases for a Pedersen commitment.

Commitments to $\mathcal{G}_{ElG}$ group elements. In Alg. 6.2 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 $M, C_1, C_2 \in \mathcal{G}_{ElG}$, 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 $C = (C_1, C_2) = \mathsf{Com}_{\mathcal{G}_{ElG}}(M)$ 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-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: $\mathsf{PoK}_{eqrep\text{-}p^*}[s_a, s_b, s_c, r_a, r_b, r_c, \beta_1, \beta_2 : D_1 = g^{\beta_1} \wedge D_2 = g^{\beta_1} h^{\beta_2}]$ 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 $\mathcal{G}_{ElG}$ commitment using a scalar in a Pedersen commitment. This can be done by using the *eqrep-p** relation described in Section 2.3. An exponentiation proof takes group element commitments C_a to $\mathcal{G}_{ElG}$ 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 $\mathsf{PoK}_{eqrep\text{-}p^*}[y, r_y, \beta_1, \beta_2 : C_y = g^y h^{r_y} \wedge C_{a,1} = C_{b,1}^y g^{\beta_1} \wedge C_{a,2} = C_{b,2}^y g^{\beta_1} h^{\beta_2}]$ where $\beta_1 = s_a - y s_b$ and $\beta_2 = r_a - y r_b$ 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 $C = \mathsf{Com}_{\mathcal{G}_{ElG}}(M; (s, r))$ and $C' = \mathsf{Com}_{\mathcal{G}_{ElG}}(M'; (s', r'))$, then $C \cdot C' = \mathsf{Com}_{\mathcal{G}_{ElG}}(MM'; (s + s', r + r'))$.

Fig. 6.2: Commitments to $\mathcal{G}_{ElG}$ elements

$\mathsf{Setup}_{\mathcal{G}_{ElG}}(1^\lambda) \to params$

1: Generate a group of prime order $p, \mathcal{G}_{ElG} = \langle g \rangle$.
(or using an existing group e.g. from a bilinear pairing)
2: Generate a random element $h \in \mathcal{G}_{ElG}$ as the base for opening.
3: **return** $params = (\mathcal{G}_{ElG}, g, h)$

$\mathsf{Commit}_{\mathcal{G}_{ElG}}(params, M \in \mathcal{G}_{ElG}) \to C, O$

4: $s \leftarrow\$ \mathbb{Z}_p; r \leftarrow\$ \mathbb{Z}_p$
5: $C \leftarrow (C_1, C_2) = (Mg^s, g^s h^r)$
6: **return** $C, O = (s, r)$

Theorem 9. *Our construction in Fig. 6.2 is binding.*

Proof of Thm. 9 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 Pedersen commitments relies on the computational Diffie-Hellman assumption and so our $\mathcal{G}_{ElG}$ commitments are computationally binding.

Theorem 10. *Our construction in Fig. 6.2 is hiding.*

Proof of Thm. 10 For any $M, C_1, C_2 \in \mathcal{G}_{ElG}$, 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 $\mathcal{G}_{ElG}$ 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 $\mathcal{G}_{ElG}$ 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. 6.3. Verifying these proofs is a direct application of the *eqrep-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 $\mathcal{G}_{ElG}$ 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 $\mathcal{G}_{ElG}$ commitments are perfect hiding and computationally binding to elements of $\mathcal{G}_{ElG}$, 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 ($\mathsf{Prove}^{\mathsf{open}}_{ElG}, \mathsf{Prove}^{\mathsf{add}}_{ElG}, \mathsf{Prove}^{\mathsf{multiply}}_{ElG}$) 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 $\mathcal{G}_{ElG}$ 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 $\mathcal{G}_{ElG}$ 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, $\mathsf{Prove}^{\mathsf{multiply}}_{ElG}$ 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 Section 6.3 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 11 (Hiding of the commitments in Fig. 6.3). *Our commitments to ElGamal ciphertexts in Fig. 6.3 are statistically hiding.*

Proof (Proof of Thm. 11). We can see that (C_1, C_2) is identical to a $\mathcal{G}_{ElG}$ commitment to c_1 and (C_3, C_4) is identical to a $\mathcal{G}_{ElG}$ commitment to c_2, we can see that they statistically hide c_1 and c_2.

Theorem 12 (Binding of the commitments in Fig. 6.3). *Our commitments to ElGamal ciphertexts in Fig. 6.3 are computationally binding.*

Fig. 6.3: Commitments to ElGamal ciphertexts

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

parse $params_{ElG} = (\mathcal{G}_{ElG}, g', h')$
1: $(g, h) \leftarrow\$ \mathcal{G}_{ElG}$
2: $params = (g, h, params_{ElG})$
3: **return** $params$

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

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

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

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

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

parse $params = (g, h, params_{ElG})$
$params_{ElG} = (\mathcal{G}_{ElG}, g', h')$
$O_c = (s_{c,1}, s_{c,2}, r_{c,1}, r_{c,2})$
$c = ((g')^{\rho_c}, k^{\rho_c}(h')^y)$,
$O_y = (r_y)$
1: $\pi = \mathsf{NIZK}[$
$s_{c,1}, s_{c,2}, s_y, \rho_c, r_{c,1}, r_{c,2}, r_y, y :$
2: $C_y = g^y h^{r_y}$
3: $\wedge C_{c,1} = (g')^{\rho_c} g^{s_{c,1}}$
4: $\wedge C_{c,2} = g^{s_{c,1}} h^{r_{c,1}}$
5: $\wedge C_{c,3} = k^{\rho_c}(h')^y g^{s_{c,2}}$
6: $\wedge C_{c,4} = g^{s_{c,2}} h^{r_{c,2}}]$
7: **return** π

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

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

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

parse $O_a = (s_{a,1}, s_{a,2}, r_{a,1}, r_{a,2})$
$O_b = (s_{b,1}, s_{b,2}, r_{b,1}, r_{b,2})$
$O_c = (s_{c,1}, s_{c,2}, r_{c,1}, r_{c,2})$
1: $D_1 \leftarrow C_{c,1}/(C_{a,1} * C_{b,1})$
2: $D_2 \leftarrow C_{c,2}/(C_{a,2} * C_{b,2})$
3: $D_3 \leftarrow C_{c,3}/(C_{a,3} * C_{b,3})$
4: $D_4 \leftarrow C_{c,4}/(C_{a,4} * C_{b,4})$
5: $\beta_1 = s_{c,1} - s_{a,1} - s_{b,1}$
6: $\beta_2 = r_{c,1} - r_{a,1} - r_{b,1}$
7: $\beta_3 = s_{c,2} - s_{a,2} - s_{b,2}$
8: $\beta_4 = r_{c,2} - r_{a,2} - r_{b,2}$
9: $\pi = \mathsf{NIZK}[\beta_1, \beta_2, \beta_3, \beta_4 :$
10: $D_1 = g^{\beta_1}$
11: $\wedge D_2 = g^{\beta_1} h^{\beta_2}$
12: $\wedge D_3 = g^{\beta_3}$
13: $\wedge D_4 = g^{\beta_3} h^{\beta_4}]$
14: **return** π

Proof (Proof of Thm. 12). We can see that (C_1, C_2) is identical to a $\mathcal{G}_{ElG}$ commitment to c_1 and (C_3, C_4) is identical to a $\mathcal{G}_{ElG}$ commitment to c_2, thus, if a PPT adversary can produce a double opening such that one of these commit-

ments opens to some c_1' or c_2' in $\mathcal{G}_{ElG}$, we obtain a double opening for our $\mathcal{G}_{ElG}$ commitments.

Theorem 13 (Zero-knowledge of Fig. 6.3). *Our protocols in Fig. 6.3 (*$\mathsf{Prove}^{\mathsf{open}}_{ElG}$, $\mathsf{Prove}^{\mathsf{enc}}_{ElG}$, $\mathsf{Prove}^{\mathsf{multiply}}_{ElG}$, *and* $\mathsf{Prove}^{\mathsf{add}}_{ElG}$*) are zero-knowledge against any PPT adversary.*

Proof (Proof of Thm. 13). 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.

Theorem 14 (Black box knowledge extraction of Fig. 6.3). *Given a PPT adversary that can produce a proof that verifies for our protocols in Fig. 6.3 (*$\mathsf{Prove}^{\mathsf{open}}_{ElG}$, $\mathsf{Prove}^{\mathsf{enc}}_{ElG}$, $\mathsf{Prove}^{\mathsf{multiply}}_{ElG}$, *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. 14). Similar to our proof of zero-knowledge for these protocols, because these protocols simply return $eqrep\text{-}p^*$ proofs, we can use the black-box extractor for these proofs to extract the witnesses. This extractor is described in Section 2.3.

6.3 Commitments to $\mathcal{G}_{CS}$ Elements and Camenisch-Shoup Ciphertexts

We will now explain our commitments to Camenisch-Shoup ciphertext. To construct commitments to Camenisch-Shoup ciphertexts, we need to construct commitments to the group in which elements of 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 commitments [DF02]. Similar to Pedersen commitments for ElGamal, we adapt them to commit to elements of $\mathcal{G}_{CS}$.

Modifications to Damgård-Fujisaki Damgård and Fujisaki [DF02] construct a commitment scheme to integers which works over a generic group $\mathcal{G}$ as long as $\mathcal{G}$ is efficiently recognizable and sampleable and has certain properties. Mainly, $\mathcal{G}$ must have hidden order. They then prove that the group $\mathbb{Z}_n$ satisfies these properties. We prove that QR_{n^2} also has these properties and then use the Damgård-Fujisaki commitment scheme over QR_{n^2} as a building block to commit to group elements of $\mathcal{G}_{CS}$ to construct commitments to Camenisch-Shoup ciphertexts. We also need to prove that $\mathbb{Z}_{n^2}$ satisfies the properties outlined in [DF02] in order for the proofs to be extractable. This is because QR_{n^2} is not efficiently recognizable and thus a malicious prover may submit a commitment in this group. We do this in Section 6.3.

Fig. 6.4: Simplified Damgård-Fujisaki commitments in QR_{n^2}

$\mathsf{Setup}(1^\lambda) \to params$:

1: Sample $O(\lambda)$-bit SG primes p', q' and compute $p = 2p' + 1, q = 2q' + 1, n = pq$.
2: Sample a random $h' \in \mathbb{Z}_{n^2}$ and compute $h = (h')^2$.
3: Compute $g = h^\alpha$ where $\alpha \leftarrow\!\!\$\ [2^{B+\lambda}]$.
4: **return** $params = (g, h)$

$\mathsf{Commit}(params, m) \to (C, O)$:

1: To commit to integer, m, compute: $C = g^m h^r b$
where $r \leftarrow\!\!\$\ [2^{B+\lambda}]$ and $b = 1$.
2: Compute the opening as $O = (r, b)$.
3: **return** (C, O)

Proving that Damgård-Fujisaki commitments are secure for $\mathcal{G} = \mathbb{Z}_{n^2}$ 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 QR_{n^2} to ensure our commitment scheme is secure.

Proof that Construction in Figure 6.4 is secure. We can see that the only difference between [DF02] and our scheme is that g and h are always in QR_{n^2}. In our modified setup, we simply square g and h, ensuring that they are in QR. In the original scheme, h is sampled randomly from $\mathbb{Z}_{n^2}$ and $|QR_{n^2}|/|\mathbb{Z}_{n^2}| = 1/4$, thus, there is a $1/4$ chance in the original scheme that h is in QR_{n^2}. In this case, our modified scheme is identical to the original scheme. Thus, if an adversary can defeat any security property of this hybrid scheme, they can defeat the original scheme with non-negligible probability.

The assumptions required in [DF02] to prove their integer commitment scheme secure are shown below. In [DF02] they 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'.

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.

In [DF02], they 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. p, q are not given to the adversary in these assumptions.

There is an additional property that Damgård and Fujisaki do not explicitly mention: the group needs to be efficiently decidable, i.e., given any encoding, a, there must exist an algorithm such that $f(a) = 1$ if and only if a is an encoding of an element of QR_{n^2}. We see that QR_{n^2} does not have this property, and thus, we must also prove that these commitments are secure for $\mathbb{Z}_{n^2}$, which does have this property. Because all proofs include an extraction of $g^s h^r b$, then, because the properties hold for $\mathbb{Z}_{n^2}$, and $g, h \in QR_{n^2}$, no adversary can produce a commitment and open or complete a proof such that $g^s, h^r \notin QR_{n^2}$. Thus ensuring that g^s is in QR_{n^2}.

We now prove that these properties hold for $\mathbb{Z}_{n^2}$ with n formed the same way as in Damgård-Fujisaki [DF02]. If these properties hold in $\mathbb{Z}_{n^2}$, they will also hold for QR_{n^2} (except for DF Property 4). This is because QR_{n^2} is a subgroup of $\mathbb{Z}_{n^2}$ and so any elements of QR_{n^2} output by an adversary would still work for $\mathbb{Z}_{n^2}$. In DF Property 4, since the element is chosen randomly, we don't immediately get that property holding in $\mathbb{Z}_{n^2}$ means it holds in QR_{n^2}. But, because $\#QR_{n^2}/\#\mathbb{Z}_{n^2} = 1/4$, there is a non-negligible change that a random element chosen from $\mathbb{Z}_{n^2}$ will also be in QR_{n^2}, thus if DF Property 4 holds for $\mathbb{Z}_{n^2}$, it also holds for QR_{n^2}.

We review the strong RSA assumption as shown in [DF02] in Assumption 1. And prove a useful lemma (Lemma 5).

One more useful property of this modified scheme is that we see that an adversary cannot double open a commitment for distinct (r, b) and (r', b'). This is notable since binding only requires that the adversary cannot open a commitment to distinct s, s'. We can see that we can modify the scheme such that $h = g^\alpha$ and the two schemes are indistinguishable. This is because both h and g are random elements in QR_{n^2} and thus they generate QR_{n^2} with high probability. Since α is much larger than the order of QR_{n^2}, g and h are indistinguishable from random elements in QR_{n^2} and thus we can simply swap them in setup to make the scheme binding on r. Since s and r are now binding, b is uniquely determined.

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

Proof of Lemma 5 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 5. 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

Fig. 6.5: $\mathcal{G}_{CS}$-DF-Commitments

$\mathsf{Setup}_{\mathcal{G}}(1^{\lambda}) \rightarrow (params, \tau)$

1: Sample $O(\lambda)$-bit SG primes p', q' and compute $p = 2p' + 1, q = 2q' + 1, n = pq$.
2: Sample a random $h' \in \mathbb{Z}_{n^2}$ and compute $h = (h')^2$.
3: Compute $g = h^{\alpha}$ where $\alpha \leftarrow\!\!\$\ [2^{B+\lambda}]$.
4: **return** $params = (n, g, h), \tau = (p, q, p', q')$

$\mathsf{Com}_{\mathcal{G}}(params, M) \rightarrow C, O$

1: $s \leftarrow\!\!\$\ [2^{B+\lambda}]; r \leftarrow\!\!\$\ [2^{B+\lambda}]; a \leftarrow\!\!\$\ \{-1, 1\}; b \leftarrow\!\!\$\ \{-1, 1\}$
2: $C = (C_1, C_2) = (Mg^s a, bg^s h^r)$
3: **return** $C, O = (s, r, a, b)$

divide the order of the group and thus we have a contradiction and a^{2t} cannot have even order.

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.

Next, by employing Damgård-Fujisaki commitments, we can construct a scheme for committing to elements of $\mathcal{G}_{CS}$. We show this scheme in Fig. 6.5. We refer to such commitments as $\mathcal{G}_{CS}$-DF-Commitments since they commit to elements of $\mathcal{G}_{CS}$ using Damgård-Fujisaki commitments as a building block. We can see that these $\mathcal{G}_{CS}$ commitments are multiplicatively homomorphic, i.e. if you take two $\mathcal{G}_{CS}$ commitments, $c = (c_1, c_2)$ committed to element, M and $d = (d_1, d_2)$ committed to element, N then if you compute their pair-wise multiplication: $e = (c_1 * d_1, c_2 * d_2)$, the resulting commitment will be committed to $M * N$ with opening information $s_c + s_d, r_c + r_d, a_c * a_c, b_c * b_c$.

Proofs of hiding and binding for $\mathcal{G}_{CS}$-DF-commitments in Fig. 6.5 We provide number theory background in Appendix A.

Hiding proof for Fig. 6.5. 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 7. Let's create a hybrid scheme where instead of choosing s, we choose $u \leftarrow\!\!\$\ QR_{n^2}$. The hybrid scheme keeps Setup identical.

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 10). In $\mathsf{Com}_{\mathcal{G}}^{hybrid}$, a challenger needs to

Fig. 6.6: $\mathcal{G}_{CS}$-DF-Commitments - hybrid scheme

$\mathsf{Com}_{\mathcal{G}}^{hybrid}(params, M) \rightarrow C, O$

1: $u \leftarrow\$ QR_{n^2}; r \leftarrow\$ [2^{B+\lambda}]; a \leftarrow\$ \{-1, 1\}$
2: $C = (C_1, C_2) = (Mua, uh^r)$
3: **return** $C, O = (u, r)$

know the factorization of n to sample randomly from QR_{n^2} but because the indistinguishiability holds statistically, it doesn't matter that our hybrid must know the factorization of n. Thus our hybrid commitment algorithm, $\mathsf{Com}_{\mathcal{G}_{CS}}^{hybrid}$ is indistinguishable from our original algorithm, $\mathsf{Com}_{\mathcal{G}_{CS}}$ in Fig. 6.5.

To prove that our commitments are hiding, we need to prove that for any $M_1, M_2 \in \mathcal{G}_{CS}$, the chance that this commitment, C is created from an honest invocation of $\mathsf{Com}_{\mathcal{G}_{CS}}$ is equal, i.e. $|\Pr[C|\mathsf{Com}_{\mathcal{G}_{CS}}^{hybrid}(M_1) = C] - \Pr[C|\mathsf{Com}_{\mathcal{G}}^{hybrid}(M_2) = C]| \leq negl(\lambda)$.

First, we'll prove that for an honestly created C_1, either $C_1 \in QR_{n^2}$ or $C_1 \in \mathsf{QNR}_{n^2}^{+1}$. If $C_1 \notin QR_{n^2}$, then we know that if $M \notin QR_{n^2}$, then $M \in \mathsf{QNR}_{n^2}^{+1}$ due to Lemma 8 and our number theory background. We also know that $a \in QR_{n^2}$ or $a \in \mathsf{QNR}_{n^2}^{+1}$ due to Lemma 9 (and because $1 \in QR_{n^2}$). Thus, due to Lemma 12, Mua is either in QR_{n^2} or $\mathsf{QNR}_{n^2}^{+1}$ and thus if C_1 is honestly created and $C_1 \notin QR_{n^2}$, then $C_1 \in \mathsf{QNR}_{n^2}^{+1}$.

Next, we'll prove that a (u, a) pair exists for any M such that $C_1 = Mua$. To do this, we'll examine possible subgroups of $\mathbb{Z}_{n^2}$ in which C_1/M lies and find a (u, a) for any possible group. If $C_1, M \in QR_{n^2}$ then $C_1/M \in QR_{n^2}$ due to the fact that QR_{n^2} is a group and thus $1/M \in QR_{n^2}$ and Lemma 12. If $C_1 \in \mathsf{QNR}_{n^2}^{+1}$ and $M \in QR_{n^2}$, we see that C_1/M is in $\mathsf{QNR}_{n^2}^{+1}$ due to Lemma 12. If $C_1 \in QR_{n^2}$ and $M \in \mathsf{QNR}_{n^2}^{+1}$, then $-M \in QR_{n^2}$ due to Lemmas 8 and 9 and thus $(-1)C_1/M \in QR_{n^2}$ and due to Lemma 8, $C_1/M \in \mathsf{QNR}_{n^2}^{+1}$.

Now we'll look at two cases: (1): $C_1/M \in QR_{n^2}$ and (2): $C_1/M \in \mathsf{QNR}_{n^2}^{+1}$. In case (1), we set $u = C_1/M$ and $a = 1$ and find that this ensures that $C_1 = Mua$. In case (2), we set $u = (-1)C_1/M$ and $a = -1$. We know that $(-1)C_1/M \in QR_{n^2}$ due to Lemmas 8 and 9. Thus, we find that $C_1 = Mua$. Thus, a (u, a) pair exists such that $C_1 = Mua$ for any M.

Thus far, we've proven that a (u, a) exists to make any C, M a valid commitment/message pair. Next, we'll prove that the probability of any C_1 being output by an honest invocation of $\mathsf{Com}_{\mathcal{G}}^{hybrid}$ is equally probable for a given M. We can see that the possible values of C_1 are $QR_{n^2} \cup \mathsf{QNR}_{n^2}^{+1}$. We can see that if $Mua = Mu'a'$ and $(u, a) \neq (u', a')$ then we have that $aM/u = a'M/u'$. If $a = a'$, then $u = u'$ and thus for $(u, a) \neq (u', a')$ to be true it must be that $a \neq a'$. Thus $-M/u = M/u'$. This implies that $u' = -u$, but we know that $-u$ is not in

$\mathsf{QNR}_{n^2}^{+1}$ (since it is sampled randomly from QR_{n^2}) and thus, by contradiction, no $u, u' \in QR_{n^2}$ can be chosen such that u, u', a, a' exists and $Mua = Mu'a'$ but $(u, a) \neq (u', a')$. There are $2 * \#QR_{n^2}$ choices for (u, a) and we see that $\#QR_{n^2} = \#\mathsf{QNR}_{n^2}^{+1}$ due to Lemma 13. Thus, $\mathsf{Com}_{\mathcal{G}}^{hybrid}(M; \cdot)$ must be bijective over the space of possible values of C_1 and since (u, a) are chosen uniformly, we see that $\Pr[C_1 | \mathsf{Com}_{\mathcal{G}}^{hybrid}(M_1) = C] = \Pr[C_1 | \mathsf{Com}_{\mathcal{G}}^{hybrid}(M_2) = C]$. Finally, because the element C_2 in our scheme is a Damgård-Fujisaki commitment, we know that it hides any s and thus our hybrid commitment (in Construction 6.6) hides any u value. Thus $|\Pr[C | \mathsf{Com}_{\mathcal{G}}^{hybrid}(M_1) = C] - \Pr[C | \mathsf{Com}_{\mathcal{G}}^{hybrid}(M_2) = C]| \leq negl(\lambda)$.

Binding proof for $\mathcal{G}_{CS}$-DF-commitments in Fig. 6.5. If a PPT adversary can open a commitment $C = (C_1, C_2)$ to two values $M, M' \in \{|x| : x \in \mathcal{G}_{CS}\}$ (providing openings, $s, s', a, a', r, r', b, b'$) 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 $C_2, (s, r, b), (s', r', b')$ is a double opening Damgård-Fujisaki commitment scheme. Thus, $s = s'$, we see that it must be that $a \neq a'$. Because $a \in \{-1, 1\}$ we see that $M = -M'$. But, we see that both $M, M' \in \{|x| : x \in \mathcal{G}_{CS}\}$ where $\mathcal{G}_{CS}$ is $\{|x| : x \in QR_{n^2}\}$. We see that that either $M \notin \mathcal{G}_{CS}$ or $M' \notin \mathcal{G}_{CS}$ thus contradicting the assumption that both M, M' are both in $\mathcal{G}_{CS}$. Thus, no $a \neq a'$ exists when $s = s'$ such that $C_1 = Mua = Mu'a'$ and thus it is impossible for a PPT adversary to double open our $\mathcal{G}_{CS}$ commitments without double opening a Damgård-Fujisaki commitment.

Auxiliary proofs for commitments to $\mathcal{G}_{CS}$ In this section, we describe protocols that we can use to create proofs of opening, multiplication, and exponentiation of elements in $\mathcal{G}_{CS}$ which can be verified using only their commitments.

Proof of knowledge of opening for $\mathcal{G}_{CS}$-DF-commitments We can see that the second element of a $\mathcal{G}_{CS}$ commitment is simply an integer commitment from Damgård-Fujisaki [DF02]. If we use their opening protocol to create a proof of opening of the second part of the commitment, this suffices as a proof of opening for a $\mathcal{G}_{CS}$ commitment as we can extract s, r, b from C_2 and compute: $M = C_1 / (g^s)$.

Proof of multiplication of $\mathcal{G}_{CS}$-DF-commitments. We can perform a proof of knowledge for multiplication of $\mathcal{G}_{CS}$ elements over their commitments by utilizing homomorphic property of the commitments. This proof operates over C_M, C_N, C_P committed to $\mathcal{G}_{CS}$ elements M, N, P, and proves that $M = N * P \in \mathcal{G}_{CS}$. Examining these commitments, we see that $C_M = (C_{M,1}, C_{M,2}) = (Ma_M g^{s_M}, b_M g^{s_M} h^{r_M})$, $C_N = (C_{N,1}, C_{N,2}) = (Na_N g^{s_N}, b_N g^{s_N} h^{r_N})$, and $C_P = (C_{P,1}, C_{P,2}) = (Pa_P g^{s_P}, b_P g^{s_P} h^{r_P})$. To begin this proof, both the prover and the verifier compute $D_1 = C_{M,1} / (C_{N,1} C_{P,1})$ and $D_2 = C_{M,2} / (C_{N,2} C_{P,2})$. The prover then uses $egrep{-}n^*$ (from Section 2.3) to prove the relation $R((\gamma_1, \gamma_2, \beta_1, \beta_2), (D_1, D_2)) = 1$ iff $D_1 = \beta_1 g^{\gamma_1} \wedge D_2 = \beta_2 g^{\gamma_1} h^{\gamma_2} \wedge \beta_1 \in \{-1, 1\} \wedge \beta_2 \in \{-1, 1\}$. The Prover uses $\gamma_1 = s_M - s_N - s_P$ and $\gamma_2 = r_M - r_N - r_P$ to satisfy this relation.

We see that this implies that D_1 is a commitment to $1 \in \mathcal{G}_{CS}$. This is because $\beta_1, \beta_2 \in \{-1, 1\}$ which does not flip the sign of the element that $\mathcal{G}_{CS}$ is committed to (because $-1 \notin \mathcal{G}_{CS}$). Thus, this proves that dividing C_M by $C_N * C_P$ cancels out the elements they are committed to and thus proves they are equal.

Proof of exponentiation of $\mathcal{G}_{CS}$-DF-commitments with Damgård-Fujisaki commitments. We can again prove this with *eqrep*$-n^*$ from Section 2.3. This proof operates over two $\mathcal{G}_{CS}$ commitments C_M, C_N to $\mathcal{G}_{CS}$ elements M, N and one scalar commitment C_y to scalar y and proves that $N = M^y$. Examining these commitments, we see that C_M and C_N are formed just like in our multiplication proof above and $C_y = g^y h^{r_y}$. This can be proven with the relation $R((\gamma_1, \gamma_2, \beta_1, \beta_2), (C_M, C_N, C_P)) = 1$ iff $C_{M,1} = \beta_1 C_{N,1}^y g^{\gamma_1} \wedge C_{M,2} = \beta_2 C_{N,2}^y g^{\gamma_1} h^{\gamma_2} \wedge \beta_1 \in \{-1, 1\} \wedge \beta_2 \in \{-1, 1\}$. The Prover uses $\gamma_1 = s_M - y s_N$ and $\gamma_2 = r_M - y r_N$ to satisfy this relation. Because $C_{M,1} = \beta_1 C_{N,1}^y g^{\gamma_1}$ and $C_{M,2} = \beta_2 C_{N,2}^y g^{\gamma_1} h^{\gamma_2}$, we know that $M = \pm N^y$. Thus, $M = N^y \in \mathcal{G}_{CS}$ since only either N^y or $-N^y$ exists in $\mathcal{G}_{CS}$.

Remark 1 (Reducing the size of scalars.). Our protocols for commitments must have a have a maximum size of the witnesses (the committed values). We label this as T. This bound ensures that our ptocotols 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 $\mathcal{G}_{CS}$ commitments that we didn't have with $\mathcal{G}_{ElG}$ 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 moduluar 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 equivalency $\mod n$. As an example, let a prover have two $\mathcal{G}_{CS}$ 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 $\mathcal{G}_{CS}$ 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 Section 2.3. 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 $\mathcal{G}$ along with their associated proof protocols, we can use these commitments to build the construction in Fig. 6.7. This figure uses

*eqrep-n** from Section 2.3 to prove relations. We also leave b and a values out of our Camenisch-Stadler witnesses when is it clear from context (the b values in relations are combinations of the a and b values from witnesses). While we don't explicitly invoke $\mathsf{Com}_{\mathcal{G}_{CS}}$, we see that in the Com_{CS} scheme, C_1, C_2 is exactly a $\mathcal{G}_{CS}$ commitment to the first element of the ciphertext, c_1, and C_3, C_4 is a $\mathcal{G}_{CS}$ commitment to the second element, c_2. Realizing this, we can then see that our proof protocols ($\mathsf{Prove}_{CS}^{\mathsf{enc}}$, $\mathsf{Prove}_{CS}^{\mathsf{multiply}}$, $\mathsf{Prove}_{CS}^{\mathsf{exp}}$, and $\mathsf{Prove}_{CS}^{\mathsf{open}}$) are similar to the proofs over $\mathcal{G}_{CS}$ commitments that we discussed earlier in this section. We prove Theorems 15, 16, and 17 in Section 6.3.

Theorem 15 (Hiding and binding of the commitments in Fig. 6.7). *Our commitments to Camenisch-Shoup ciphertexts in Fig. 6.7 are statistically hiding and computationally binding.*

Theorem 16 (Zero-knowledge of proofs in Fig. 6.7). *Our protocols in Fig. 6.7 ($\mathsf{Prove}_{CS}^{\mathsf{open}}$, $\mathsf{Prove}_{CS}^{\mathsf{enc}}$, $\mathsf{Prove}_{CS}^{\mathsf{multiply}}$, and $\mathsf{Prove}_{CS}^{\mathsf{add}}$) are zero-knowledge against any PPT adversary.*

Theorem 17 (Black box knowledge extraction of proofs in Fig. 6.7). *Given a PPT adversary that can produce a proof that verifies for our protocols in Fig. 6.7 ($\mathsf{Prove}_{CS}^{\mathsf{open}}$, $\mathsf{Prove}_{CS}^{\mathsf{enc}}$, $\mathsf{Prove}_{CS}^{\mathsf{multiply}}$, 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.*

Proofs for commitments to Camenisch-Shoup ciphertexts We split Thm. 15 into two theorems, Thm. 18 and Thm. 19.

Theorem 18 (Hiding of the commitments in Fig. 6.7). *Our commitments to Camenisch-Shoup ciphertexts in Fig. 6.7 are statistically hiding.*

Proof (Proof of Thm. 18). We can see that (C_1, C_2) is identical to a $\mathcal{G}_{CS}$ commitment to c_1 and (C_3, C_4) is identical to a $\mathcal{G}_{CS}$ commitment to c_2, we can see that they statistically hide c_1 and c_2.

Theorem 19 (Binding of the commitments in Fig. 6.7). *Our commitments to Camenisch-Shoup ciphertexts in Fig. 6.7 are computationally binding.*

Proof (Proof of Thm. 19). We can see that (C_1, C_2) is identical to a $\mathcal{G}_{CS}$ commitment to c_1 and (C_3, C_4) is identical to a $\mathcal{G}_{CS}$ 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 $\mathcal{G}_{CS}$, we obtain a double opening for our $\mathcal{G}_{CS}$ commitments.

Proof (Proof of Thm. 16). 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. 6.7: Commitments to Camenisch-Shoup ciphertexts

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

parse $params_{CS} = (\mathcal{C}_p, g', h')$
$g, h \leftarrow\!\!\$\ \mathcal{G}_{CS}$
$params = (\mathcal{G}, g, h, params_{CS})$
return $params$

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

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

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

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

$\mathsf{Prove}^{\mathsf{open}}_{CS}(params, C, [M, O]) \to \pi$

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

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

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

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

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

Proof (Proof of Thm. 17). Similar to our proof of zero-knowledge for these protocols, because these protocols simply return *eqrep-p** proofs, we can use the black-box extractor for these proofs to extract the witnesses. This extractor is described in Section 2.3.

7 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.

8 Retrospective Blueprints

The blueprint scheme of [KLN23] requires that the auditor fixes the value x and posts the resulting pk_A to the user before any targeted transactions occur.

In this section we address a (weak) version of the retrospective case. In this new scheme, let x_t be the auditors information at time t. As before, users have long-term identity information y_{id}, but users now also have transaction information y_t at time t. In the watchlist use-case of retrospective blueprints, the

watchlist, x_t can evolve, and future matches with it at time t can reveal information $y_{t'}$ about transactions that have occurred earlier, say, at time t', for $t' \leq t$.

Intuition for definition. More formally, in a (P, f)-retrospective blueprint scheme the identity information y_{id} and transaction information y_t of a user are evaluated on both a predicate $P(x_t, y_{id}, y_t)$ and a function $f(y_{id}, y_t)$.

Let n be the considered lifetime of the system. The guarantees for the user are that, if none of the predicates $P(x_1, y_{id}, y_1), \ldots, P(x_n, y_{id}, y_n)$ are true, no information about y_{id} or any of the transactions $y_1, \ldots, y_n$ is revealed (except that these predicate evaluations are false). However, if any $P(x_t, y_{id}, y_t)$, $1 \leq t \leq n$ is true, then only $f(y_{id}, y_1), \ldots, f(y_{id}, y_n)$ is revealed.[9] The additional secondary filtering process offered by f is in contrast to leaking the seed of a CBDC scheme, which would immediately deanonymize all of a users transactions.[10]

We construct a scheme that satisfies this informal definition in the following section.

Construction of Generic Retrospective (P, f)-blueprint scheme We now provide a construction of a retrospective (P, f)-blueprint scheme using a secure f'-blueprint scheme $\mathcal{BP} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{VerPK}, \mathsf{Escrow}, \mathsf{VerEscrow}, \mathsf{Dec})$ (used in a black-box manner), an authenticated symmetric-key encryption scheme $E = (\mathsf{SymEnc}, \mathsf{SymDec})$ and a signature scheme $S = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$.

Retrospective Blueprints can be realized from Blueprints by adding a symmetric key to the users identity information and encoding P and f into a function $f'(x, (y_{id}, k), y_i) = P(x, y_{id}, y_i) * k \| \mathsf{SymEnc}_k(f(y_{id}, y_i))$ that encrypts $f(y_{id}, y_i)$ using k with k being conditionally released based on $P(x, y_{id}, y_i)$.

On signing up for the retrospective blueprint scheme, the user grants the certificate-issuing authority access to their identity. In the scheme, the user first sends their identity y_{id} and a commitment $C_k = \mathsf{Com}_{cpar}(k; r_k)$ to their secret key k to the issuing authority.[11] This authority could also be the auditor depending on the use case. The issuer then issues a single certificate on (y_{id}, C_k) per user identity y_{id} using their secret key sk_I as $\sigma = \mathsf{Sign}(sk_I, (y_{id}, C_k))$ and lets the user have this certificate to add it to their state $st = (k, r_k, y_{id}, \sigma)$.

[9] In order to achieve strong privacy via a simulation-based notion, escrows of non-suspicious users must be independent of the user's transaction information. To avoid the need for expensive non-committing encryption, we provide the simulator either with no information and a *promise that P is always 0*, or always provide $f(y_{id}, y_t)$. We refer to this notion as non-adaptive privacy against dishonest auditors.

[10] The function $f(y_{id}, y_t)$ is a relatively more specific function that only reveals information based on the user identity and transactions. A more general way to express the information revealed to the auditor once the key is revealed is $f(x_t, y_{id}, y_t)$ - the information revealed would be dependent on both the auditor input x_t as well as the user information y_{id} and y_t. We instead use $f(y_{id}, y_t)$ here for ease of understanding and presentation.

[11] Alternatively, one could employ a dedicated signature scheme with efficient protocols for signing committed messages such as [CL03].

$\mathsf{EscrowRet}(\Lambda, \mathsf{st}, \mathsf{pk}_\mathsf{A}, \mathsf{pk}_I, y_i, r_i)$

parse $\mathsf{st} = (k, r_k, y_{id}, \sigma)$

$C_i \leftarrow \mathsf{Com}_{cpar}(y_i; r_i)$

$C_y \overset{r_y}{\leftarrow} \mathsf{Com}_{cpar}((k, y_{id}, y_i))$

$(\hat{Z}, \pi_\mathsf{U}) \leftarrow \mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, (k, y_{id}, y_i), r_y)$

$\pi_{cert} \leftarrow \mathsf{PoK}^{\mathsf{S}_4}_{\Psi_4}\Big\{(k, y_{id}, y_i, \sigma, r_y, r_{y'}) :$

$\quad \mathsf{Ver}(\mathsf{pk}_I, (y_{id}, C_k), \sigma) = 1$

$\quad \wedge\, C_i = \mathsf{Com}_{cpar}(y_i; r_i)$

$\quad \wedge\, C_y = \mathsf{Com}_{cpar}((k, y_{id}, y_i); r_y)\Big\}$

return $(\hat{Z}, \pi_\mathsf{U}, \pi_{cert})$

$\mathsf{VerEscrowRet}(\Lambda, \mathsf{pk}_\mathsf{A}, \mathsf{pk}_I, C_{y'}, Z = (\hat{Z}, \pi_\mathsf{U}, \pi_{cert}))$

parse $\Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4)$

parse $\mathsf{pk}_\mathsf{A} = (\_, C_x, \_)$

return $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C_y, Z' = (\hat{Z}, \pi_\mathsf{U}))$

$\quad \wedge\, \mathsf{V}^{\mathsf{S}_4}_4((cpar, \mathsf{pk}_I, C_{y'}), \pi_{cert})$

$\mathsf{DecRet}(\Lambda, \mathsf{sk}_\mathsf{A}, \mathsf{pk}_I, C_{y'}, Z = (\hat{Z}, \pi_\mathsf{U}, \pi_{cert}))$

parse $\mathsf{sk}_\mathsf{A} = (\mathsf{pk}_\mathsf{A}, d, K = \{k_i\})$

if $\mathsf{VerEscrowRet}(\Lambda, \mathsf{pk}_\mathsf{A}, \mathsf{pk}_I, C_y, Z) = 0$

$\quad$ **return** 0

$(z', \pi_Z) \leftarrow \mathsf{Dec}(\Lambda, \mathsf{sk}_\mathsf{A}, C_{y'}, Z' = (\hat{Z}, \pi_\mathsf{U}))$

$(k, c) \leftarrow z'$

if $k = \perp$

$\quad z \leftarrow \mathsf{SymDec}_{k_i}(c)$ s.t. $\mathsf{SymDec}_{k_i}(c) \neq \perp \wedge k_i \in K$

else

$\quad K = K \cup \{k\}$

$\quad z \leftarrow \mathsf{SymDec}_k(c)$

$\pi'_Z \leftarrow \mathsf{PoK}^{\mathsf{S}_4}_{\Psi_4}\Big\{k : z = \mathsf{SymDec}_k(c)\Big\}$

return (z, π_Z, π'_Z)

Fig. 8.1: Construction of generic retrospective (P, f)-blueprint scheme from a generic f'-blueprint scheme, a signature scheme $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$, a symmetric encryption scheme $E = (\mathsf{SymEnc}, \mathsf{SymDec})$, and NIZK PoK Ψ_4 with setup S_4.

As shown in Fig. 8.1, the escrow algorithm of a retrospective blueprint scheme, EscrowRet() takes as input the user state st and y_i the i^{th} transaction information of the user, and combines it into a tuple (k, y_{id}, y_i) for which it computes a fresh commitment C_y with opening r_y. The escrow $\hat{Z}$ is computed in the usual manner by calling the old Escrow() routine on this committed input. Additionally, EscrowRet() also uses a NIZK proof π_{cert} to prove that the certificate provided is valid for the user identity and key commitment (y_{id}, C_k) and that the two commitments C_k and C_y are consistent.

In DecRet, the auditor will verify that the escrow was formed correctly and decrypt the key. If this decryption failed (the predicate was zero) then the auditor will search through a list of previously decrypted keys to check if any of these keys can decrypt the ciphertext. This is why we need symmmetric authenticated encryption for this as the auditor must be able to ensure that they do not decrypt a ciphertext with the wrong key. How this system will be used in practice is that the auditor will store any escrows that they cannot decrypt. Then, each time

they learn a new key, they will go back through these old escrows and check if they can now decrypt them with this new key.

Note: The user key k is not available outside the retrospective blueprint scheme and thus hidden from the external user (and the adversary trying to break the scheme). It is only available at the level of the retrospective blueprint scheme system. Thus the external commitment randomness r_i is only for a commitment C_i to y_i. In a similar vein, the transactions information y_i of (most) users would not be available to the adversary trying to break the anonymity of the transaction system employing the retrospective blueprinting system for accountability, e.g., of an e-cash scheme that tries to hide who is spending how much with whom.

VerEscrowRet is similar to VerEscrow - it only needs to verify the new π_{cert} proof aside from the rest of the checks in VerEscrow. Similarly, the new retrospective decrypt functionality DecRet only calls Dec of $\mathcal{BP}$ on the escrow Z.

Proof intuition. If none of the predicates $P(x, y_{id}, y_i)$ are true, the low bits of $f'(x, (y_{id}, k), y_i)$ are indistinguishable from random and uncorrelated with the key k, thus revealing no extra information to the auditor.

8.1 Achieving Post-investigation Privacy for retrospective blueprints

One downside of our retrospective blueprint construction is that even if suspected users turn out to be innocent after investigation, and their identity subsequently removed from the suspect list, their long-term key k would nevertheless remain under the control of the auditor, meaning that they remain traceable indefinitely. When a scheme protects former suspects' privacy after they have been removed from the watchlist, we say it has *post-investigation privacy*.

While the retrospective property of blueprints is a tool for auditors to scrutinize past transactions of individuals flagged on the watchlist, conversely, post-investigation privacy ensures the protection of future messages or transactions if the suspect recovers from a compromise or is removed from the watchlist after investigation. Although seemingly contradictory, achieving both retrospective access and post-investigation privacy is feasible within our proposed blueprint scheme. To realize this, users need to update keys in a manner that only allows for the derivation of old keys. If the last time the predicate is true is t, then a blueprint scheme only reveals $f(y_{id}, y_1), ..., f(y_{id}, y_t)$ about the user and their transaction.

Non-adaptivity of the escrow design. In order to achieve the basic tenets of retrospectivity and post-investigation privacy, escrows of non-suspicious users must be independent of the user transactions that the escrow was formed using. This must be done in order to stop the leakage of information from a compromised escrow about future transactions which will be used to build uncompromised escrows. Since all escrows are released together, they are required to be non-adaptive to avoid the violation of post-compromise security.

One-Way Trapdoor Permutation Integration One potential way to remedy this weakness and achieve post-investigation privacy involves the incorporation of a one-way trapdoor permutation within the protocol. By leveraging a trapdoor, users can generate the key for a specific epoch from the key of the preceding epoch by employing their trapdoor. Note that this requires updating the certificate to replace C_k with a commitment to the newly derived key. Upon discovering the key (k_i) for epoch i, the auditor can employ the permutation function to derive keys from previous epochs, facilitating retrospective analysis without compromising the privacy of future transactions.

One-Way Permutation and Merkle Tree Implementation One way to avoid updating the certificate, and indeed the need for a trapdoor is to precompute all keys starting from the final epoch (this restricts the number of epochs to be polynomial) and committing to all keys using a Merkle tree. During certification the user proofs wellformedness of the committed tree. Then the user can efficiently prove that they used the correct key k_i by proving knowledge of a merkle tree path.

Key Derivation Tree Implementation Alternatively, a key derivation tree can be employed to emulate a similar mechanism. In this implementation, the keys for each epoch serve as the leaves of a key derivation tree. The auditor, on revealing some nodes in the key derivation tree, gain access to all derivable present and past epoch keys. Crucially, this does not divulge the hashes of higher up subtree-roots that open the leaves containing keys of future epochs, thereby preserving the confidentiality of upcoming transactions. This is, in fact, quite similar to the construction in Section 4.6 of [BS15] where a PRF is built using a PRG.

We can also utilize this mechanism to attain a Blueprint Scheme where the retrospectivity is *flexible*, implying that the auditor can selectively be allowed to access user transactions from a specific range of epochs. This can be utilized in pertinent real-world applications where the auditor needs access to the transactions of a user from only a specific period, and not outside of this set period.

9 Acknowledgements

Anna Lysyanskaya and Scott Griffy were supported by NSF Grants 2312241, 2154170, and 2247305 as well as the Peter G. Peterson 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 Kenmoe Youdom for his helpful discussions.

References

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, 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, Heidelberg, August 2016.

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, Heidelberg, 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.

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.

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.

BS15. Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2015.

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.

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, 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, Heidelberg, 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, 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, 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, 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, Heidelberg, August 2002.

CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

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, Heidelberg, August 2004.

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, 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. `https://eprint.iacr.org/2022/495`.

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

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, Heidelberg, 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.

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.

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, 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, Heidelberg, May 2023.

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

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

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

KMV23. Dimitris Kolonelos, Mary Maller, and Mikhail Volkhov. Zero-knowledge arguments for subverted RSA groups. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 512–541. Springer, Heidelberg, May 2023.

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, Heidelberg, August 2014.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

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.

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

Sha90. Adi Shamir. IP=PSPACE. In *31st FOCS*, pages 11–15. IEEE Computer Society Press, October 1990.

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, 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. `https://eprint.iacr.org/2022/452`.

A Number theory background

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 $a^{(p-1)/2} = 1 \pmod p$ and $a^{(q-1)/2} = 1 \pmod q$. 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}.

We label the Jacobi value of an element in $\mathbb{Z}_m$ as $J_m(x)$. The Jacobi symbols is always $J_m(x) \in \{-1, 1\}$ for all m, x. The Jacobi symbol behaves differently if m is a prime or not. **If m is a prime,** we can compute this as $J_m(x) = x^{(m-1)/2} \mod m$. Further, x is in QR_m if and only if $J_m(x) = 1$. **If m is composite,** then $J_m(x)$ is efficiently computable without knowledge of its prime factors. Though, computing $J_p(x)$ for an RSA modulus $n = pq$ is hard. Further, $J_m(x) = \prod J_{p_i}(x)$ where $\{p_i^{a_i}\}_{i \in [k]}$ is the prime factorization of m. Lastly, $x \in \mathsf{QR}_m$ if and only if $\forall i \in [k] : J_{p_i}(x) = 1$. We label the set of elements such that $J_m(x) = 1$ but $x \notin \mathsf{QR}_m$ as $\mathsf{QNR}_{n^2}^{+1}$.

Lemma 7 (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 7. 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 8. *If $x \neq |x| \pmod m$ then $J_p(x) = -J_p(|x|)$ for any prime p such that $p|m$ and $p = 3 \mod 4$.*

Proof of Lemma 8. For any x, if $|x| \neq x$, then $|x| = m - x$. We see that $J_p(|x|) = (m-x)^{(p-1)/2} = \sum_{i=0}^{(p-1)/2} \binom{(p-1)/2}{i} m^i (-x)^{(p-1)/2-i} = (-x)^{(p-1)/2} + \binom{(p-1)/2}{1}(-x)^{(p-1)/2-1}m + ...$ In this binomial expansion, we can see that all terms for $i > 0$ are a multiple of m and thus vanish mod p since $p|m$. Thus, the Jacobi symbol for $m - x$ is $(-x)^{(p-1)/2} = (-1)^{(p-1)/2}(x)^{(p-1)/2} = (-1)^{(p-1)/2}b$ where $b = x^{(p-1)/2}$. Since $p = 3 \mod 4$, we see that this equals $(-1)^{(k4+2)/2}b = (-1)^{k2+1}b = -b$. Thus $J_p(x) = -J_p(|x|)$.

As a corollary of Lemma 8, we see that $J_{n^2}(x)$ for RSA modulus $n = pq$ is the same for x and $|x|$ since if $|x| \neq x$, $J_n(x) = J_p(x)J_q(x) = (-1)J_p(|x|)(-1)J_q(|x|) = J_n(|x|)$.

Lemma 9. *$(-1) \in QNR_{n^2}^{+1}$ for RSA modulus, n.*

Proof of Lemma 9. Since $p = 3 \mod 4$, we can see that $(-1)^{(p-1)/2} = (-1)^{(k4+2)/2} = (-1)^{k2+1} = -1$ for some k. Thus, $J_p(-1) = J_q(-1) = -1$ and thus $(-1) \in \mathsf{QNR}_{n^2}^{+1}$.

Lemma 10. *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 10.

Lemma 11. *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 12. *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 $xy \in \mathsf{QNR}_{n^2}^{+1}$, $xx', yy' \in \mathsf{QR}_{n^2}$*

Lemma 13. *$\#QR_{n^2} = \mathbb{Z}_{n^2}/4$ and $\#\mathsf{QNR}_{n^2}^{+1} = \mathbb{Z}_{n^2}/4$.*

Proofs of Lemmas 12, 11, and 13 are present in [KL20] (deriving Lemma 13 from [KL20] is a trivial exercise).

Lemma 14. *The map: $f : QR_{n^2} \to \mathsf{QNR}_{n^2}^{+1}$ defined by $f : x \mapsto (-1)x$ is bijective.*

Proof of Lemma 14. We can see that $(-1)x$ is $n^2 - x$ since $n^2 - x + x = 0$. Thus, this maps an element to it's additive inverse. Because $\mathbb{Z}_{n^2}$ is a group over addition, inverses must exist and be unique for any non-zero element in $\mathbb{Z}_{n^2}$. Thus, this map must be bijective since $QR_{n^2}, \mathsf{QNR}_{n^2}^{+1} \subset \mathbb{Z}_{n^2}^*$.

B Examples of using eqrep

B.1 Constructing *eqrep-p**

In Alg. 7 we show how to implement a *eqrep-p** protocol from an underlying *eqrep* 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. 7 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-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-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 7 Example *eqrep-p** proof

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

B.2 Constructing *eqrep-n**

Construction 1 shows an example construction of a proof os a relation for *eqrep-n** defined in Section 2.3. We note that to reduce a construction of *eqrep-n** 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-n** 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 Section 6.3. In this example, we prove the exponentiation of an element in a $\mathcal{G}_{CS}$ commitment (which we define in Section 6.3) 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 $\mathcal{G}_{CS}$ 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 Section 6.3 that QR_{n^2} and $\mathbb{Z}_{n^2}$ both satisfy these properties. We can see that the second elements of both of our $\mathcal{G}_{CS}$ 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 $\mathcal{G}_{CS}$ 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-n^**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 ($\mathcal{G}_{CS}$-DF-commitments - proof of exponentiation) *Goal: Prove that the $\mathcal{G}_{CS}$-DF-commitment d is committed to $N = M^{x_2}$ where c is a $\mathcal{G}_{CS}$-DF-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:

$$
\begin{array}{lcl}
P & \leftrightarrow & V \\
\rho_1 \textit{ will hide } ex_2 & & \\
\rho_1 \leftarrow\!\$\ [CT2^{\lambda}] & & \\
\rho_2 \textit{ will hide } er_2 & & \\
\rho_2 \leftarrow\!\$\ [C2^{B+2\lambda}] & & \\
\rho_3 \textit{ will hide } e(-x_2x_1+x_3) & & \\
\rho_3 \leftarrow\!\$\ [CT^2 2^{\lambda}] & & \\
\rho_4 \textit{ will hide } e(-r_1x_1+r_3) & & \\
\rho_4 \leftarrow\!\$\ [CT2^{B+2\lambda}] & & \\
a_1 = g^{\rho_1}h^{\rho_2} & & \\
a_2 = c_1^{\rho_1}g^{\rho_3} & & \\
a_3 = c_2^{\rho_1}g^{\rho_3}h^{\rho_4} & & \\
 & a_1,a_2,a_3 \rightarrow & \\
 & & e \leftarrow\!\$\ [C] \\
 & \leftarrow e & \\
z_1 = \rho_1 + ex_2 & & \\
z_2 = \rho_2 + er_2 & & \\
z_3 = \rho_3 + e(-x_1x_2+x_3) & & \\
z_4 = \rho_4 + e(-r_1x_2+r_3) & & \\
 & z_1,z_2,z_3 \rightarrow & \\
 & & g^{z_1}h^{z_2} = a_1t^e \\
 & & c_1^{z_1}g^{z_3} = a_2d_1^e \\
 & & c_2^{z_1}g^{z_3}h^{z_4} = a_3d_2^e
\end{array}
$$

Lemma 15 (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 15 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 20. *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 15. 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 $c_2^{\delta_{z_1}} g^{\delta_{z_3}} h^{\delta_{z_4}} \beta_3 = d_2$, 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, $c_2^{\Delta_{z_1}}g^{\Delta_{z_3}}h^{\Delta_{z_4}} = a_3 d_2^{\Delta_e}$. We can replace this with $(\beta_3)^{\Delta_{z_1}}g^{x_1\Delta_{z_1}}h^{r_1\Delta_{z_1}}g^{\Delta_{z_3}}h^{\Delta_{z_4}} = a_3 d_2^{\Delta_e}$. 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 $g^{x_1\Delta_{z_1}}h^{r_1\Delta_{z_1}}g^{\Delta_{z_3}}h^{\Delta_{z_4}} = a_3 d_2^{\Delta_e}$. 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.

$c_1^{z_1}g^{z_3} = a_2 d_1^e$
$M^{z_1}g^{x_1 z_1}g^{z_3} = a_2 d_1^e$
$M^{z_1 - z_1'}g^{x_1(z_1 - z_1')}g^{z_3 - z_3'} = d_1^{e - e'}$
$M^{(z_1 - z_1')}g^{x_1(z_1 - z_1')}g^{(z_3 - z_3')} = d_1^{(e - e')}$
$M^{(z_1 - z_1')/(e - e')}g^{x_1(z_1 - z_1')/(e - e')}g^{(z_3 - z_3')/(e - e')} = d_1$
$bM^{\delta_{z_1}}g^{x_1\delta_{z_1}}g^{\delta_{z_3}} = d_1$
$bM^{x_2}g^{x_1 x_2}g^{\delta_{z_3}} = d_1$
$bg^{x_1 x_2 + \delta_{z_3}} = d_1/M^{x_2}$

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].

C Definition of an f-Blueprint Scheme

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

Definition 13. *For a non-interactive commitment scheme* $(\mathsf{CSetup}, \mathsf{Com})$*, an f-blueprint scheme consists of the following probabilistic polynomial time algorithms:*

$\mathsf{Setup}(1^\lambda, cpar) \to \Lambda$: 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.

$\mathsf{KeyGen}(\Lambda, x, r_x) \to (\mathsf{pk_A}, \mathsf{sk_A})$: 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.

$\mathsf{VerPK}(\Lambda, \mathsf{pk_A}, C_x) \to 1$ 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.

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

$\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z) \to 1$ 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.

$\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z) \to f(x, y)$ 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:

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

$\mathsf{Sound}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$

1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
3 : $x, r_x \leftarrow \mathsf{Adv}(1^\lambda, \Lambda)$
4 : $(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_x)$
5 : $(C, y, r_y, Z) \leftarrow \mathsf{Adv}(\mathsf{pk_A})$
6 : **return** $[C = \mathsf{Com}(y; r_y) \wedge$
7 : $\quad \mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z) \wedge \mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z) \neq f(x, y)]$

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

Definition 14 (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 $\mathsf{Dec}(\Lambda, \mathsf{sk_A}, C, Z)$.
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:

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

for some negligible ν.

$\mathsf{BHreal}^{\mathsf{Adv}}_{\mathsf{Blu}}(\lambda)$

1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
3 : $(x, r_x, \mathsf{st}_{\mathsf{Adv}}) \leftarrow \mathsf{Adv}(1^\lambda, \Lambda)$
4 :
5 : $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_x)$
6 : **return** $\mathsf{Adv}^{\mathcal{O}_0(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}, \cdot, \cdot)}(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}_{\mathsf{Adv}})$

$\mathsf{BHideal}^{\mathsf{Adv}}_{\mathsf{Blu,Sim}}(\lambda)$

1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $(\Lambda, \mathsf{st}) \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
3 : $(x, r_x, \mathsf{st}_{\mathsf{Adv}}) \leftarrow \mathsf{Adv}(1^\lambda, \Lambda)$
4 : $dsim \leftarrow (|x|, \mathsf{Com}(x; r_x))$
5 : $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}) \leftarrow \mathsf{SimKeygen}(1^\lambda, \mathsf{st}, dsim)$
6 : **return** $\mathsf{Adv}^{\mathcal{O}_1(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}, x, \cdot, \cdot)}(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}_{\mathsf{Adv}})$

$\mathcal{O}_0(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}, C, Z)$

1 : **if** $\neg\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C, Z)$
2 : **return** $\perp$
3 : **return** $\mathsf{Dec}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z)$

$\mathcal{O}_1(\mathsf{pk}_{\mathsf{A}}, \mathsf{st}, x, C, Z)$

1 : **if** $\neg\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C, Z)$
2 : **return** $\perp$
3 : $y^* \leftarrow \mathsf{SimDecrypt}(\mathsf{st}, C, Z)$
4 : **return** $f(x, y) = f^*(x, y^*)$

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

Definition 15 (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:*

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

Definition 16 (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.*

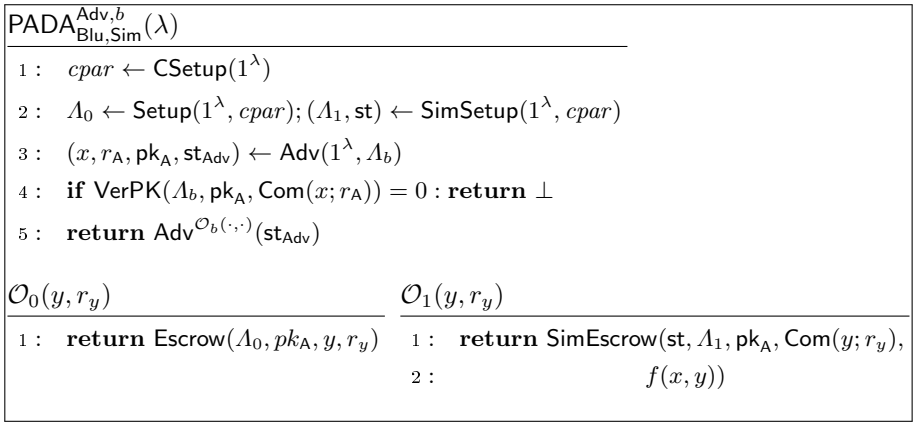

Fig. C.3: Game $\mathsf{PADA}^{\mathsf{Adv},b}_{\mathsf{Blu}}(\lambda)$

2. ***Privacy-Preserving Game:*** *The honest auditor generates the public key on input x provided by the adversary. On input adversary-generated commitments 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:*

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

for some negligible function ν.

D Constructions of HEC Schemes

D.1 KLN construction of HEC from Fully Homomorphic Encryption (FHE)

Definition 17 (Circuit-private fully homomorphic encryption). *Algorithms* $(\mathsf{FHEKeyGen}, \mathsf{FHEEnc}, \mathsf{FHEDec}, \mathsf{FHEEval})$ *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 $FHEPK$.* $\mathsf{FHEEnc}(FHEPK, b)$ *takes as input the public key and a bit $b \in \{0,1\}$ and outputs a ciphertext c.* $\mathsf{FHEDec}(FHESK, c)$ *takes as*

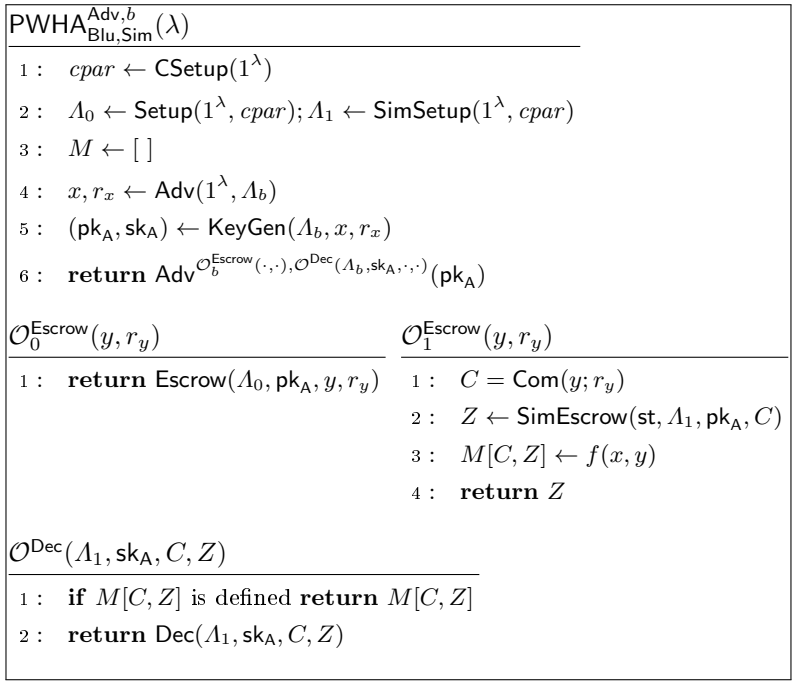

Fig. C.4: Game $\mathsf{PWHA}^{\mathsf{Adv},b}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$

input a ciphertext c and outputs the decrypted bit $b \in \{0,1\}$. $\mathsf{FHEEval}(FHEPK, \mathcal{C}, c_1, \ldots, c_n)$ *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 $\mathcal{C}(b_1, \ldots, b_n)$ where c_i is an encryption of 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* $\mathsf{FHEDec}(FHESK, C) \neq \mathcal{C}(x)$ *where $(FHESK, FHEPK)$ are output by* $\mathsf{FHEKeyGen}$*, $c_1, \ldots, c_n$ are ciphertexts where $c_i \leftarrow \mathsf{FHEEnc}(FHEPK, x_i)$, and $c_\mathcal{C} = \mathsf{FHEEval}(FHEPK, \mathcal{C}, c_1, \ldots, c_n)$, 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 $(FHESK, FHEPK)$ output by* $\mathsf{FHEKeyGen}(\lambda)$ *and for all input ciphertexts $c_1, \ldots, c_n$ generated by* $\mathsf{FHEEnc}(FHEPK, \cdot)$*, $c_\mathcal{C}$ generated by* $\mathsf{FHEEval}(FHEPK, \mathcal{C}, c_1, \ldots, c_n)$ *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*:*

$\mathsf{FHECircHideExpt}(1^\lambda)$

$(R, \mathcal{C}_0, \mathcal{C}_1, (x_1, r_1), \ldots, (x_n, r_n)) \leftarrow \mathsf{Adv}(1^\lambda)$
if $\mathcal{C}_0 \notin \mathcal{C} \vee \mathcal{C}_1 \notin \mathcal{C} \vee \mathcal{C}_0(x_1, \ldots, x_n) \neq \mathcal{C}_1(x_1, \ldots, x_n)$: **reject**
$(FHEPK, FHESK) = \mathsf{FHEKeyGen}(1^\lambda; R)$
for $i \in \{1, \ldots, n\}$:
  $c_i = \mathsf{FHEEnc}(FHEPK, x_i; r_i)$
$Z_0 \leftarrow \mathsf{FHEEval}(FHEPK, \mathcal{C}_0, c_1, \ldots, c_n)$
$Z_1 \leftarrow \mathsf{FHEEval}(FHEPK, \mathcal{C}_1, c_1, \ldots, c_n)$
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: $\mathcal{C}_j = \{\mathcal{C}^{f_j}_{y,z}(x) : y \in \{0,1\}^{\ell_y}, z \in \{0,1\}^2\}$.

$\mathrm{HEC}_{\mathrm{SETUP}}(1^\lambda) \to \Lambda$: Generate the FHE parameters Λ, if needed.

$\mathrm{HEC}_{\mathrm{ENC}}(1^\lambda, \Lambda, f, x) \to (X, d)$: Generate $(FHESK, FHEPK) \leftarrow \mathsf{FHEKeyGen}(1^\lambda, \Lambda)$. Let $|x| = n$; set $c_i \leftarrow \mathsf{FHEEnc}(FHEPK, x_i)$. Output $X = (FHEPK, c_1, \ldots, c_n)$, and decryption key $d = FHESK$.

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

$\mathrm{HEC}_{\mathrm{DEC}}(hecpar, d, Z) \to z$: Output $(\mathsf{FHEDec}(d, Z_1), \ldots, \mathsf{FHEDec}(d, Z_\ell))$.

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

Theorem 21. *If* $(\mathsf{FHEKeyGen}, \mathsf{FHEEnc}, \mathsf{FHEDec}, \mathsf{FHEEval})$ *is a fully-homomorphic public-key encryption scheme that is circuit-private for circuit family* $\{\mathcal{C}^f_j : f \in F\}$ *defined above, then our construction above constitutes a homomorphic-enough encryption for the family* F*.*

The proof of Theorem 21 is provided in [KLN23].

D.2 Additional proofs for consistent HEC scheme

Proof of Thm. 3 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 $Z \leftarrow \text{HECEVAL}(hecpar, f, X, y; r_Z)$ but $\text{HECDEC}(hecpar, d, Z) \neq f(x, y)$, 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 HECEVAL 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 4 into Theorems 22, 23, and 24.

Theorem 22 (Security of DIRECTZ for Fig. 5.1). *Our construction in Fig. 5.1 achieves security of* DIRECTZ *defined in Definition 11.*

Proof of Thm. 22 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_{attr}) 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 escrower 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, $Z_{id} = r_1 P(y_{id}) + y_{id}$, $Z_{attr} = r_2 P(y_{id}) + attr$ and $Z_{nf} = r_3 P(y_{id})$ because $P(y_{id}) \neq 0$. This makes the three ciphertext values indistiguishable 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 23 (Security of x and y for Fig. 5.1). *Our construction in Fig. 5.1 achieves* ***security of*** x ***and*** y ***from third parties****.*

Proof of Thm. 23. 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. 24, 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 24 (Security of x for Fig. 5.1). *Our construction in Fig. 5.1 achieves* ***Security of*** y

Proof of Thm. 24. 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.