Privacy-Preserving Blueprints

Markulf Kohlweiss[1], Anna Lysyanskaya[2], and An Nguyen[2]

[1] University of Edinburgh and Input Output, markulf.kohlweiss(at)ed.ac.uk
[2] Brown University, {anna_lysyanskaya,an_q_nguyen}(at)brown.edu

Abstract. If everyone were to use anonymous credentials for all access control needs, it would be impossible to trace wrongdoers, by design. This would make legitimate controls, such as tracing illicit trade and terror suspects, impossible to carry out. Here, we propose a privacy-preserving blueprint capability that allows an auditor to publish an encoding pk_A of the function $f(x,\cdot)$ for a publicly known function f and a secret input x. For example, x may be a secret watchlist, and $f(x,y)$ may return y if $y \in x$. On input her data y and the auditor's pk_A, a user can compute an escrow Z such that anyone can verify that Z was computed correctly from the user's credential attributes, and moreover, the auditor can recover $f(x,y)$ from Z. Our contributions are:

- We define secure f-blueprint systems; our definition is designed to provide a modular extension to anonymous credential systems.
- We show that secure f-blueprint systems can be constructed for all functions f from fully homomorphic encryption and NIZK proof systems. This result is of theoretical interest but is not efficient enough for practical use.
- We realize an optimal blueprint system under the DDH assumption in the random-oracle model for the watchlist function.

1 Introduction

Cryptography offers powerful answers on how to strike a balance between privacy and accountability. The study of anonymous credentials [27,55,17,54,18,19,2] has given us general practical tools that make it possible to obtain and prove possession of cryptographic credentials without revealing any additional information. In other words, users can obtain credentials without revealing who they are, and then prove possession of credentials in a way that is unlinkable to the session where these credentials were obtained and to other sessions in which they were shown. Anonymous credentials can be shown a limited number of times (compact e-cash) [15], or at a limited rate total or per verifier [14,16]. Anonymous credentials are compatible with identity escrow [51,3], where and appropriate trusted authority can establish the identity of the user when needed.

In this paper, we extend the state-of-the-art on anonymous credentials by adding a new desirable feature: that of a privacy-preserving blueprint capability: even a malicious authority cannot learn anything about a user other than what's revealed by comparing the blueprinted data with the user's data.

Consider anonymous e-cash with a secret watchlist as a motivating application. In anonymous e-cash [25,26,28,15], we have a Bank that issues e-coins (credentials), Users who withdraw and spend them, and Vendors (or Verifiers) that verify e-coins and accept them as payment in exchange for goods and services. Some small number of users are suspected of financial crimes, and, unbeknownst to them, a judge has placed them on a watchlist. We need a mechanism that allows an auditor to trace the transactions of these watchlisted users without violating the privacy of any other users, and also while keeping the contents of the watchlist confidential from everyone.

A high-level definition of a privacy-preserving blueprint. We have three types of participants: the users, the verifiers, and the de-anonymization auditor. On input x (for example, a watchlist), the auditor outputs a blueprint pk_A that the users and verifiers will need.

Next, the user and the verifier engage in an anonymous transaction; we don't actually care what else happens in this transaction; the user might be proving to the verifier that they are authorized, or it may be an e-cash transaction. What we do care about is that, as a by-product of this transaction, the user and the verifier have agreed on a cryptographic commitment C such that (1) the user is in possession of the opening of C; and (2) the transaction that just occurred guarantees that the opening of C contains user data y that is relevant for the auditor's needs. For example, imagine that x is a watchlist consisting of names of individuals of interest, and y contains a user's name; then this user is of interest to the auditor if $y \in x$.

To enhance this anonymous transaction with privacy-preserving blueprint capability, the user runs the algorithm Escrow to compute a value Z that is an escrow of the opening of the commitment C; from Z, the auditor will be able to recover the information relevant to him, and no other information about the user. Specifically, in the watchlist scenario, the auditor will recover y if $y \in x$, but will learn nothing about the user if $y \notin x$. More generally, in an f-blueprint scheme, the auditor will recover $f(x, y)$ and no additional information. The verifier's job is to verify the escrow Z against C using $\mathsf{VerEscrow}$ and only let the transaction go through if, indeed, it verifies.

It is important that even a malicious auditor cannot create a blueprint that corresponds to an unauthorized input x. To capture this, we also require that there is a publicly available cryptographic commitment C_A. Outside of our protocol, we expect a mechanism for arriving at an acceptable (but secret) input x and the commitment C_A to x. For example, a judge may publish a commitment to a secret watchlist, and privately reveal the opening to the auditor; or several authorities may be responsible for different components of a watchlist and the auditor aggregates them together in a publicly verifiable fashion; or another distributed protocol can be agreed upon for arriving at the commitment C_A such that its opening (i.e., x) is known to the auditor. To ensure that only such an authorized secret input x is blueprinted, a secure blueprint scheme must include an algorithm VerPK that verifies that pk_A indeed corresponds to the value to which C_A is a commitment.

Our security definition mandates that the following properties hold: (1) correctness, so that honestly created blueprints and escrows pass VerPK and $\mathsf{VerEscrow}$, respectively, and the escrow Z correctly decrypts; (2) soundness of $\mathsf{VerEscrow}$ that ensures that if, for a commitment C, escrow Z is accepted, then it correctly decrypts to $f(x, y)$ where x is the opening of C_{A} and y is the opening of C; (3) blueprint hiding, i.e., the blueprint pk_{A} does not reveal anything about x other than what the adversary can learn by forming valid escrows and submitting them for decryption; (4) privacy against a dishonest auditor that ensures that even if the auditor is malicious, an honest user's escrow contains no information beyond $f(x, y)$, where x is the opening of C_{A} and y is the opening of C; and finally (5) privacy with an honest auditor that ensures that an adversary who does not control the auditor learns nothing from the escrows. We give a precise formal definition of an f-blueprint scheme in Sect. 3.

Our results. Our first result is a blueprint scheme specifically for watchlists; more precisely, it is an f-blueprint scheme for

$$f(x, y) = \begin{cases} y & \text{if } y = y_1 \| y_2 \text{ and } y_2 \in x \\ \perp & \text{otherwise} \end{cases}$$

where y_1 denotes $O(\log \lambda)$ most significant bits of y. This first scheme is secure in the random-oracle model under the decisional Diffie-Hellman assumption. The size of pk_{A} is optimal at $O(\lambda n)$ where λ is the security parameter, which is linear in the number of bits needed to represent a group element; and the watchlist x consists of n elements of $\mathbb{Z}_q$, where q ($\log q = \Theta(\lambda)$) is the order of the group. The size of the escrow Z is also $O(\lambda n)$.

Our second result is an f-blueprint scheme for any f from fully homomorphic encryption (FHE) and non-interactive zero-knowledge proofs of knowledge. In the full version of this paper [52], we also show how to obtain an f-blueprint scheme for any f from non-interactive secure computation (NISC) [50].

Technical roadmap. We obtain the results above via the same general method: by first defining (Sect. 4) and then realizing (Sections 6 and 7) a homomorphic-enough cryptosystem (HEC) for the function f. We can think of a homomorphic-enough cryptosystem as a protocol between Alice and Bob that works as follows: first, Alice uses the HEC$_{\text{ENC}}$ algorithm to encode her input x into a value X, and she also obtains a decryption key d for future use; next, Bob uses HEC$_{\text{EVAL}}$ to compute an encryption Z of $z = f(x, y)$ from Alice's encoding X and his input y. Finally, Alice runs HEC$_{\text{DEC}}$ to recover z from Z. To be useful for our application, an HEC scheme must be correct even when the inputs to the algorithms are chosen maliciously, and it also must ensure that X hides x, and that X and Z together hide the inputs x and y. Additionally, it must allow for an algorithm HEC$_{\text{DIRECT}}$ that computes an encryption Z of z directly from X and $z = f(x, y)$, such that its output is indistinguishable from the output of HEC$_{\text{EVAL}}$, even if Alice is malicious.

A HEC combined with an appropriate non-interactive zero-knowledge (NIZK) proof system gives a generic construction of an f-blueprint:. The auditor obtains $(X, d) \leftarrow \text{HECENC}(x)$, and an NIZK proof π_{A} that X was computed correctly in a way that corresponds to the opening of C_{A}; he sets the blueprint as $\mathsf{pk}_{\mathsf{A}} = (X, \pi_{\mathsf{A}})$. Verifying this blueprint amounts to verifying π_{A}. To compute the escrow Z, the user obtains $Z' \leftarrow \text{HECEVAL}(X, y)$ and then a proof π_Z that Z' was computed correctly from X and the opening of C; then set $Z = (Z', \pi_Z)$. Verifying the escrow amounts to verifying π_Z. Finally, in order to recover $f(x, y)$ from the escrow Z, the auditor uses the decryption key d to run $\text{HECDEC}(d, Z')$.

Given this roadmap, our theoretical construction that works for any f is relatively straightforward: we show that HEC can be realized from circuit-private fully homomorphic encryption [46,60,35] which, in turn, can be realized from regular fully homomorphic encryption [46,10,9,47]. The circuit-privacy guarantee ensures that Z hides Bob's input y from a malicious Alice. Alternatively, as we explore in the full version of this paper [52], it can be realized for any f from a related primitive of non-interactive secure computation (NISC) [50]. Since here we don't aim for efficiency, general (inefficient) simulation-extractable NIZK PoK can be used for the proofs. This instantiation of our generic construction is presented in Sect. 7.

Our practical construction for watchlists under the decisional Diffie-Hellman assumption is not as straightforward: first, it requires that we construct a practical homomorphic enough cryptosystem based on DDH, and next we need efficient non-interactive zero-knowledge proof systems for computing and verifying π_{A} and π_Z. Let us give a brief overview.

Our HEC construction uses the ElGamal cryptosystem [36,65] as a building block. Suppose as part of setup we are given a group $\mathbb{G}$ of order q in which the decisional Diffie-Hellman assumption holds. Let g be a generator of $\mathbb{G}$. In order to encode her input $x = (a_1, \ldots, a_n)$, Alice's HECENC algorithm first generates an ElGamal key pair $(\mathsf{pk}, \mathsf{sk})$. She then picks a random $s \leftarrow \mathbb{Z}_q^*$ and computes the coefficients $c_0, \ldots, c_n$ of the n-degree polynomial $p(\mathbf{x}) = s\prod(\mathbf{x} - a_i)$ for which $a_1, \ldots, a_n$ are the n zeroes. The encoding X are ElGamal encryptions $C_0, \ldots, C_n$ of the values $g^{c_0}, \cdots, g^{c_n}$ under the ElGamal public key pk, so the output of HECENC is $X = (C_0, \ldots, C_n, \mathsf{pk})$, and $d = \mathsf{sk}$.

Bob's algorithm HECEVAL computes Z as follows: first, it parses $y = y_1 \| y_2$ (recall that y_1 denotes the first $O(\log \lambda)$ bits of y). Then Bob obtains an ElGamal encryption E of $g^{p(y_2)}$ from the encrypted coefficients $C_0, \ldots, C_n$: since the ElGamal cryptosystem is multiplicatively homomorphic, $E = C_0 C_1^{y_2} C_2^{y_2^2} \ldots C_n^{y_2^n}$ is the desired ciphertext (for an appropriate multiplication operation on ElGamal ciphertexts). Next, let F be an encryption of g^y; finally, Bob obtains the ciphertext $Z = FE^r$, i.e., Bob uses E to mask the encryption of g^y; if E is an encryption of 0, the mask won't work and Z will decrypt to g^y.

This is reminiscent of the private set intersection construction of Freedman, Nissim and Pinkas [42], but with a subtle difference: the polynomial encoded as part of X has an additional random coefficient, s. Thus, even if Bob knows Alice's entire input x, he still does not know $p(a)$ for $a \notin x$. This ensures that in

the event that $f(x, y) = \perp$, Bob cannot set r in such a way that Z will decrypt to a value of his choice; instead, it will decrypt to a random value.

Finally, $\text{HEC}_{\text{DEC}}(d, Z)$ decrypts the ElGamal ciphertext Z to some group element $u \in \mathbb{G}$, and for each $a_i \in x$, and for all possible values for y_1, checks whether $g^{y_1 \| a_i} = u$. If it finds such a pair, it outputs it; else, it outputs $\perp$.

Plugging in this HEC scheme in our generic construction gives us an efficient blueprint scheme for watchlists as long as we can also find efficient instantiations of the NIZK proof systems for computing the proofs π_{A} and π_Z. As was already well-known [44,23,58], we can represent the statement that a given ElGamal ciphertext encrypts g^a such that a given Pedersen [62] commitment C is a commitment to a as a statement about equality of discrete logarithm representations; moreover, we can also represent statements about polynomial relationships between committed values (i.e., that C_p is a commitment to the value $p(a_1, \ldots, a_\ell)$ where p is a polynomial, and commitments $C_1, \ldots, C_\ell$ are to values $a_1, \ldots, a_\ell$) as statements about equality of representations. Using this fact, as well as the fact that efficient NIZK proofs of knowledge for equality of discrete logarithm representations in the random-oracle model are known [44,43,31], we can also efficiently instantiate the NIZK proof system in the random-oracle model.

A subtlety in using these random-oracle-based proof systems, however, is that generally such proof systems' knowledge extractors require black-box access to the adversary and involve rewinding it. In situations where the adversary expects to also issue queries to its challenger, and a security experiment or reduction must extract the adversary's witness in order to answer them, using such proof systems runs into the nested rewinding problem. One could opt to use straight-line extractable proofs instead (in such proofs, the knowledge extractor does not need to rewind the adversary); however, known techniques to achieve straight-line extraction come either at a $\omega(\log \lambda)$ multiplicative cost [13,39] or require cumbersome setup assumptions [21].

A more efficient technique is to have a common random string (CRS) and interpret it as an ElGamal public key. There is an efficient Σ-protocol [31] for proving that the contents of two ElGamal ciphertexts under two different keys are equal; it can be converted into a non-interactive zero-knowledge proof using the Fiat-Shamir heuristic [38] in the random-oracle model. If one of these public keys comes from the CRS, then the soundness of the proof system allows for straight-line extraction that uses the corresponding secret key as the extraction trapdoor. Here we give a formalization of this previously used (e.g. [20]) approach.

Specifically, we formulate a new flavor of NIZK proof of knowledge systems: black-box extractability with partial straight-line (BB-PSL) extraction, and give an efficient NIZK BB-PLS PoK proof system for equality of discrete-logarithm representations. This proof system allows straight-line extraction (i.e. extraction from the proof itself, without rewinding the adversary) of a function of the witness (for example, instead of extracting w the extractor computes g^w); this gives the security experiment enough information to proceed. Although this approach is somewhat folklore, we believe our rigorous formulation and instantiation in the random-oracle model (Sect. 2.2) may be of independent interest.

How our scheme builds on the anonymous credentials literature. Note that, as stated so far, neither the definitions nor the schemes concern themselves with credentials. Instead, the user and the verifier agree on a commitment C to the user's relevant attribute y. Out of band, the user may have already convinced the verifier that she has a credential from some third-party organization attesting that y is meaningful. For example, if y is the user's name, then the third-party organization might be the passport bureau. Indeed, this is how anonymous credentials work in general [54,18,19,2], and therefore this modeling of the problem allows us to add this feature to anonymous credentials in a modular way. Moreover, our ElGamal-based scheme is compatible with literature on anonymous credentials [54,18,19,2] and compact e-cash and variants [15,16,14] because Pedersen commitments are used everywhere.

Related work. Group signatures and identity escrow schemes [29,22,51,1,5] allow users to issue signatures anonymously on behalf of a group such that an anonymity-revoking trustee can discover the identity of the signer. The difference between this scenario and what we are doing here is that in group signatures the signer's identity is always recoverable by the trustee, while here it is only recoverable if it matches the watchlist.

Group signatures with message-dependent opening [63] and bifurcated [53], multimodal and related signatures [59,34,41] allow a tracing authority to recover a function of the user's private information that's known to the user at the time of group-signing or credential showing. In contrast, in a secure blueprinting scheme, the user knows only one of two inputs to this function.

Another related line of work specifically for watchlists is private set intersection (PSI) [42,24]. Although techniques from PSI are helpful here, in general PSI is an interactive two-party protocol, while here, the Auditor who knows the watchlist x is offline at the time when the user is forming the escrow. Private searching on streaming data [61] allows an untrusted proxy to process streaming data using encrypted keywords. The resulting encrypted data does not come with any assurance that it was *correct*. In contrast, in our scenario, the verifier and the auditor can both verify that Z was computed correctly.

A series of recent papers explored accountable law enforcement access system [48,40,64,49]. None of them, however, consider integration with anonymous credential systems for privacy-preserving authentication. Break-glass encryption by Scafuro [64] realizes a mechanism in which the auditor can decrypt Alice's ciphertexts simply be reliably revealing that he did so, i.e., that he broke the glass. The choice of which messages are to be leaked can happen even after Alice's public key is generated. Scafuro achieves this in certain strong models, such as those of hardware tokens and existence of a blockchain. In abuse-resistant law enforcement access systems (ARLEAS) [49], a law enforcement agency with a valid warrant can secretly place a user Alice under surveillance. They will be able to decrypt messages that are encrypted to Alice's public key, but not those encrypted to other users for whom surveillance has not been authorized. Moreover, ARLEAS make it possible for an email server to enforce compliance by verifying that an encrypted message indeed allows lawful access by law enforcement; and

(in a nutshell) all participants can verify the validity of all warrants even though they are unable to tell who is under surveillance. In our view, ARLEAS follows principles that are similar to ours: finding a way to reconcile the need to monitor illegal activity with privacy needs of the law-abiding public. However, since ARLEAS concerns itself with encryption, while we worry about privacy-preserving authentication, our technical contributions are somewhat orthogonal.

2 Preliminaries

The ElGamal Cryptosystem and Its Security. Let KGen be an algorithm that, on input a description of a group $\mathbb{G}$ with generator g of prime order q in which the discrete-logarithm problem is hard for PPT (in the security parameter 1^λ) adversaries, outputs (1) a public key pk consisting of an element $y \leftarrow\!\!\$\ \mathbb{G}$; and (2) a secret key $\mathsf{sk} = s$ such that $g^s = y$. The encryption algorithm Enc encrypts a message $m \in \mathbb{G}$ by sampling $r \leftarrow\!\!\$\ \mathbb{Z}_q$ and outputting the ciphertext $c = (g^r, my^r)$. The decryption algorithm Dec decrypts $c = (a, b) \in \mathbb{G}^2$ by computing ba^{-s}. We use $c \xleftarrow{r} \mathsf{Enc}(\mathsf{pk}, m)$ and $\mathsf{Enc}(\mathsf{pk}, m; r)$ to make randomness explicit.

ElGamal is semantically secure under the decisional Diffie-Hellman assumption [66]. In this paper, we use the equivalent notion of security against chosen plaintext attack (IND-CPA) formulated by Boneh and Shoup [7]. In their security game, the adversary continuously interacts with either the 0-encryption oracle that always encrypts the first of the two messages the adversary sends it, or with the 1-encryption oracle that always encrypts the second message. Their security definition is more convenient for us because it allows us to avoid an additional hybrid argument.

Let $\oplus : \mathbb{G}^2 \times \mathbb{G}^2 \to \mathbb{G}^2$ be the operator for the homomorphic composition of two ElGamal ciphertexts $c_1 = (a_1, b_1) \in \mathbb{G}^2, c_2 = (a_2, b_2) \in \mathbb{G}^2$ such that: $c_1 \oplus c_2 := (a_1 \cdot a_2, b_1 \cdot b_2)$ where $\cdot$ is the group operator of $\mathbb{G}$. We also write c^a as shorthand for repeated operation of c with itself a times.

Definition 1 (Statistically hiding non-interactive commitment). *A pair of algorithms* $(\mathsf{CSetup}, \mathsf{Commit})$ *constitute a statistically hiding non-interactive commitment scheme for message space* M_{cpar} *and randomness space* R_{cpar} *if they satisfy (1) statistical hiding, i.e., for any cpar output by* $\mathsf{CSetup}(1^\lambda)$*, for any* $m_0, m_1 \in M_{cpar}$*, the distributions* $D(cpar, m_0)$ *and* $D(cpar, m_1)$ *are statistically close, where* $D(cpar, m) = \{r \leftarrow R_{cpar} : \mathsf{Commit}_{cpar}(m; r)\}$*; and (2) computational binding, i.e. for any* PPT *adversary* $\mathcal{A}$*, there exists a negligible* ν *such that* $\Pr[cpar \leftarrow \mathsf{CSetup}(1^\lambda); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(cpar) : \mathsf{Commit}_{cpar}(m_0; r_0) = \mathsf{Commit}_{cpar}(m_1; r_1) \wedge m_0 \neq m_1] = \nu(\lambda)$

We will use the Pedersen commitment scheme which employs a cyclic group $\mathbb{G}$ of prime order q. Let $g, h_1, h_2, \ldots, h_n$ be generators of $\mathbb{G}$ and $m_1, m_2, \ldots, m_n \in \mathbb{Z}_q^n$, then $\mathsf{Commit}_{h_1,h_2,\ldots,h_n,g}(m_1, m_2, \ldots, m_n)$ samples $r \leftarrow\!\!\$\ \mathbb{Z}_q$ and computes $g^r \prod_{i=1}^n h_i^{m_i}$. This scheme is binding under the discrete logarithm assumption in $\mathbb{G}$. We write $\mathsf{Commit}_{h_1,\ldots,h_n,g}(m_1, \ldots, m_n; r)$ to make randomness explicit.

2.1 Non-interactive Zero Knowledge

Let $\mathcal{R}$ be a polynomial-time verifiable binary relation. For a pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, we refer to $\mathbb{x}$ as the statement and $\mathbb{w}$ as the witness. Let $\mathcal{L} = \{\mathbb{x} \mid \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$.

A non-interactive proof system for $\mathcal{R}$ consists of a prover algorithm P and verifier algorithm V both given access to a setup S. The setup can either be a random oracle or a reference string—we show later how we abstract over the differences in their interfaces. P takes as input a statement $\mathbb{x}$ and witness $\mathbb{w}$, and outputs a proof π if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\perp$ otherwise. V takes as input $(\mathbb{x}, \pi)$ and either outputs 1 or 0.

Definition 2 (NIZK). *Let S be the setup, and (P, V) be a pair of algorithms with access to setup S. $\Phi = (\mathsf{S}, \mathsf{P}, \mathsf{V})$ is a simulation-sound (optionally extractable) non-interactive zero-knowledge proof system for relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ if it has the following properties:*

Completeness: For all $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $\Pr\left[\pi \leftarrow \mathsf{P}^{\mathsf{S}}(\mathbb{x}, \mathbb{w}) : \mathsf{V}^{\mathsf{S}}(\mathbb{x}, \pi) = 0\right] = 0$.

S is a stateful oracle that captures both the common-random-string setting and the random-oracle setting. In the random-oracle setting, S responds to a query m by sampling a random string h of appropriate length ℓ (clear from context). In the common-reference-string (CRS) setup, it samples a reference string on the first invocation, and from then onward returns the same reference string to all callers.

Zero-knowledge: The zero-knowledge property requires that no adversary can distinguish the real game in which the setup is generated honestly and an honest prover computes proofs using the correct algorithm P, from the simulated game in which the proofs are computed by a simulator that does not take witnesses as inputs, and in which the setup is also generated by the simulator. More formally, there exist probabilistic polynomial time (PPT) simulator algorithms $(\mathsf{SimS}, \mathsf{Sim})$ such that, for any PPT adversary $\mathcal{A}$ interacting in the experiment in Fig. 2.1, the advantage function $\nu(\lambda)$ defined below is negligible:

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

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

Fig. 2.1: NIZK game

SimS shares state with Sim modeling both RO programming and CRS trapdoors. Additionally there is an extraction trapdoor τ_{Ext} that will be used below to define simulation extractability.

Soundness: A proof system is sound if no adversary can fool a verifier into accepting a proof of a false statement. It is simulation sound if the adversary cannot do so even given oracle access to the simulator — of course in that case the adversary is prohibited from outputting statement-proof pairs for which the proof was obtained from the simulator. It is a proof of knowledge if a knowledge extractor algorithm can compute the witness given appropriate access to the adversarial prover's algorithm. We explore various flavors of simulation soundness in the full version of this paper [52], but here we focus on just one of them: the flavor of a proof of knowledge that allows for (partial) straight-line extraction.

2.2 NIZK Proof of Knowledge

Simulation Extractability: A proof system is extractable (also often called a *proof of knowledge*, or PoK for short) if there exists a polynomial-time extractor algorithm that, on input a proof π for a statement $\mathbb{x}$ that passes verification, outputs the witness $\mathbb{w}$ for $\mathbb{x}$. In order to reconcile extractability with the zero-knowledge property, it is important that the extractor algorithm Ext have some additional information that is not available to any regular participants in the system. This information depends on the setup S: in the CRS setting, it is a trapdoor that corresponds to the CRS; in the random-oracle setting it comes from the ability to observe the adversary's queries to the random oracle. Note that, in addition, trapdoors can be embedded by programming the random oracle. Further, a proof system is simulation-extractable if the extractor algorithm works even when the adversary has oracle access to the simulator and can thus obtain simulated proofs.

Let $\mathcal{Q}$ denote the simulator's query tape that records all the queries the adversary $\mathcal{A}$ made to the simulator. $\mathcal{Q}_{\mathsf{S}}$ denotes the setup query tape that records the queries, replies, and embedded trapdoors of the simulated setup; this is explicitly recorded by O_{S} and $\tilde{\mathsf{O}}_{\mathsf{S}}$. As we will discuss below, $\tilde{\mathsf{O}}_{\mathsf{S}}$ additionally reveals to $\mathcal{A}$ the extraction trapdoor τ_{Ext}; this captures adaptive extraction from many proofs.

Ad attractive definition of simulation extractability is the one of straight-line extractability [39]: the extractor obtains the witness just from $\mathcal{Q}_{\mathsf{S}}$ and the pair $(\mathbb{x}, \pi)$. A weaker definition allows for black-box extractability, where the extractor additionally obtains black-box access to $\mathcal{A}$, i.e. it can reset it to a previous state. By $\mathsf{BB}(\mathcal{A})$ we denote this mode of access to $\mathcal{A}$, and by $\mathsf{Ext}^{\mathsf{BB}(\mathcal{A})}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi)$ we denote an extractor algorithm that, in addition to its inputs, also has this access to $\mathcal{A}$. See the full version of this paper for additional discussions and the definition of the black-box simulation extractability game $\mathsf{NISimBBExtract}$.We now propose a notion that falls between straight-line and black-box simulation extractability.

Black-Box with Partial Straight Line (BB-PSL) Simulation Extractability: Sometimes, it is good enough that a straight-line extractor be able to learn something about the witness, say some function $f(\mathbb{w})$, not necessarily the entire witness. For such a scenario, it is convenient to have two extractors: Ext that is a black-box extractor that extracts the entire witness given black-box access to the adversary, and ExtSL that extracts some function of that witness in a straight-line fashion. The reason this is good enough for some proofs of security is that, in a reduction, $f(\mathbb{w})$ may be enough information for the reduction to know how to proceed, without the need to reset the entire security experiment. This is similar to f-extractability [4].

Let us now formalize BB-PSL simulation extractability; let $\Phi = (\mathsf{S}, \mathsf{P}, \mathsf{V})$ be an NIZK proof system satisfying the zero-knowledge property above; let $(\mathsf{SimS}, \mathsf{Sim})$ be the simulator. Let f be any polynomial-time computable function. Φ is f-BB-PSL simulation-extractable if there exists a pair of polynomial-time extractor algorithms $(\mathsf{Ext}, \mathsf{ExtSL})$ such that for any PPT adversary $\mathcal{A}$ participating in the game defined in Fig. 2.2, the advantage function $\nu(\lambda)$ defined below is negligible. As mentioned before, $\mathcal{Q}$ denotes the query tape. $\mathcal{Q}_{\mathsf{Ext}}$ denotes the setup query tape that records the queries, replies, and embedded trapdoors of the simulated setup; this is explicitly recorded by O_{S}.

$$\mathsf{Adv}^{\mathsf{NISimBBPSLExtract}}_{\mathcal{A}}(\lambda) = \Pr\Big[f\text{-}\mathsf{NISimBBPSLExtract}^{\mathcal{A}}(1^\lambda) = 1\Big] = \nu(\lambda)$$

for some negligible function ν.

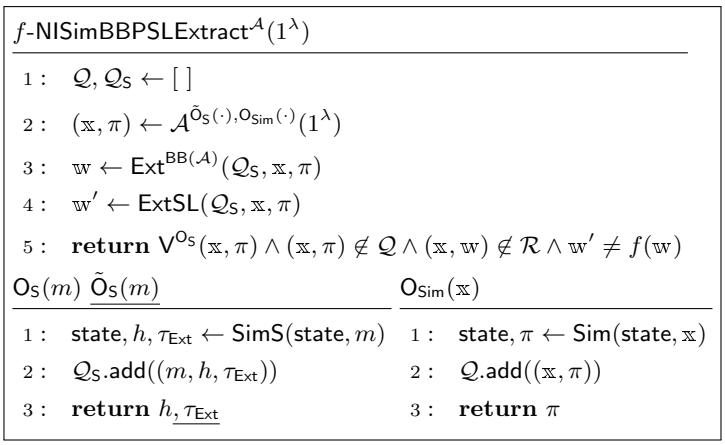

Fig. 2.2: f-NISimBBPSLExtract game

More on the simulator and extractor. In the games NIZK, NISimSound, and NISimBBPSLExtract the simulator initializes and updates the setup using SimS and then responds to queries from $\mathcal{A}$ for simulated proofs using Sim. Note

that the two halves of the simulator, SimS and Sim, share state information, and update it when queried. This captures both the CRS and the random-oracle settings. In the CRS setting, SimS computes the reference string S so that it can pass the corresponding simulation trapdoor to Sim via the shared state. In the random-oracle (RO) setting, SimS programs the random oracle (computes the value h that the random oracle will return when queried on m) and uses the shared state in order to memorize the information that Sim will need to use h in the future. Similarly, in the random-oracle mode, Sim has the ability to program the random oracle as well and memorize what it did using the state variable.

In the simulation extractability experiments, the extractor Ext takes $\mathcal{Q}_{\mathsf{S}}$ as input. In the CRS model, $\mathcal{Q}_{\mathsf{S}}$ will contain the extraction trapdoor corresponding to the CRS. In the RO model, $\mathcal{Q}_{\mathsf{S}}$ also contains information that the simulator algorithms SimS generated, such as how the RO was programmed and where the adversary queried it. It does not, however, contain the simulation trapdoor or give the extractor the ability to program the RO.

Our definition requires successful extraction even when all information in $\mathcal{Q}_{\mathsf{S}}$, in particular τ_{Ext}, is available to the adversary. This allows the adversary to run Ext itself, and thus allows for extraction from multiple proofs.

Instantiating Simulation Extractable proofs. While simulation-extractable proof systems exist for all NP relations [33], there are multiple ways to realize non-interactive zero-knowledge (NIZK) proof systems more efficiently. One of them is to start with Σ-protocols and convert them into a NIZK proof in the random oracle model, e.g. using the techniques of [38,37,56]. As we will elaborate below, Σ protocols are particularly suitable for proving knowledge of group isomorphisms such as discrete logarithm representations; see, e.g. [57]. They can also efficiently prove disjunctive statements [30]. This has been used for range proofs.

Bulletproofs [12] is a practically efficient NIZK proof system for arithmetic circuits, specifically optimized for range-proofs. Recent work shows that Bulletproofs are simulation extractable [45] and can be integrated with Σ-protocols [11].

Bernhard et al. [6, Theorem 1] state that Fiat-Shamir Σ-protocols are black-box simulation extractable with respect to expected polynomial-time adversaries. To show partial straightline extractability we use a theorem of [37, Theorem 2] that shows that Σ-protocols compiled using Fiat-Shamir are simulation-sound and adapt the theorem of [32, Theorem F.1] which shows how to transform simulation-sound into simulation-extractable NIZK, by encrypting the witness to the sky. Our approach differs from their approach in that we only encrypt a partial witness and can thus use groups for which computing discrete logarithms is hard.

In Sect. 2.5 we give a construction from Σ-protocols of a proof system Ψ for equality of discrete logarithm representation relations and prove that it is an f-BB-PSL simulation-extractable NIZK proof system in the random-oracle model for an appropriate f.

Notation. When using NIZK proofs of knowledge in a protocol, it is convenient to be able to compactly specify what exactly the prover is proving its knowledge

of. We shall use the notation:

$$\pi \leftarrow \mathsf{PoK}_{\Psi}\left\{ \mathbb{w} : R(\mathbb{x}, \mathbb{w}) \right\}$$

to indicate that the proof π was computed as follows: the proof system $\Psi = (\mathsf{S}, \mathsf{P}, \mathsf{V})$ for the relation R was used; the prover ran $\mathsf{P}^{\mathsf{S}}(\mathbb{x}, \mathbb{w})$; to verify π, the algorithm $\mathsf{V}^{\mathsf{S}}(\mathbb{x}, \pi)$ needs to be run. In other words, the value $\mathbb{w}$ in this notation is the witness the knowledge of which the prover is proving to the verifier, while $\mathbb{x}$ is known to the verifier. A helpful feature of this notation is that it describes what we need Ψ to be: it needs to be a NIZK PoK for the relation R.

2.3 Σ-protocol for proof of equality of discrete logarithm representations

Let R_{eqrep} be the following relation: $R_{\mathsf{eqrep}}(\mathbb{x}, \mathbb{w})$ accepts if $\mathbb{x} = (\mathbb{G}, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{n})$ where $\mathbb{G}$ is the description of a group of order q, and all the x_is and $g_{i,j}$s are elements of $\mathbb{G}$, 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 $(\mathbb{x}, \mathbb{w}) \in R_{\mathsf{eqrep}}$, 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 $1 \le i \le n$. 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 $1 \le i \le n$. 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 $1 \le j \le m$.

2.4 From Σ-protocols to BB simulation extractable NIZK PoK via Fiat-Shamir

Let $\Psi_{\mathsf{eqrep}} = (\mathsf{S}_{\mathsf{eqrep}}, \mathsf{P}_{\mathsf{eqrep}}, \mathsf{V}_{\mathsf{eqrep}})$ be the proof system we get from the Σ-protocol described in Sect. 2.3 via the Fiat-Shamir heuristic. Specifically, $\mathsf{S}_{\mathsf{eqrep}}$ is a random oracle.

We use a theorem of [37, Theorem 2] that shows that Σ-protocols compiled using Fiat-Shamir are simulation-sound; moreover, it follows from a theorem of [6, Theorem 1] and the proof of [37, Theorem 3] that it is in fact black-box simulation extractable.

Recall that the notation $\pi \leftarrow \mathsf{PoK}_{\Psi_{\mathsf{eqrep}}}\left\{ \mathbb{w} : R_{\mathsf{eqrep}}(\mathbb{x}, \mathbb{w}) \right\}$ denotes that the proof π is the output of $\mathsf{P}_{\mathsf{eqrep}}$.

2.5 g^x-BB-PSL simulation extractable NIZK from Ψ_{eqrep}

Now we want a BB-PSL simulation extractable proof system for R_{eqrep} such that, in a straight-line fashion, a function of $\mathbb{w}$ can be extracted. Specifically, recall that $\mathbb{x} = (\mathbb{G}, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^n)$ and $\mathbb{w} = \{w_j\}_{j=1}^m$ such that $x_i = \prod_{j=1}^m g_{i,j}^{w_j}$.

Consider the following proof system $\Psi = (\mathsf{S}, \mathsf{P}, \mathsf{V})$ for the relation R_{eqrep} and for the function $f(J, \cdot)$, defined as follows. Let g be the generator of $\mathbb{G}$ included in the description of $\mathbb{G}$. Let J be a subset of the set of indices $[m]$. Let $f(J, \mathbb{w}) = \{g^{w_j} \ : \ j \in J\}$.

S is a random oracle, but we interpret its output as follows: On input the description of a group $\mathbb{G}$ with generator g of order q, outputs a random element h of $\mathbb{G}$; we can think of this h as the public key of the ElGamal cryptosystem.

P works as follows: on input $\mathbb{x} = (\mathbb{G}, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^n)$ and $\mathbb{w} = \{w_j\}_{j=1}^m$, it first obtains $h = \mathsf{S}(\mathbb{G})$ and then forms the ElGamal ciphertexts of $g^{w_{j_k}}$ for each $j_k \in J$: $(c_{k,1}, c_{k,2}) = (g^{r_k}, g^{w_{j_k}} h^{r_k})$, for $1 \leq k \leq |J|$.

It then forms $\mathbb{x}'$ and $\mathbb{w}'$ that allow us to express the following relation R as a special case of R_{eqrep}:

$$\begin{aligned} R = \{\mathbb{x}', \mathbb{w}' \mid \mathbb{x}' = (\mathbb{x}, \{(c_{j_k,1}, c_{j_k,2})\}) \text{ and} \\ \mathbb{w}' = (\mathbb{w}, \mathbb{w}'') \text{ where } \mathbb{w}'' = (r_1, \ldots, r_{|J|}) \text{ such that} \\ \text{for } 1 \leq k \leq |J|, (c_{k,1}, c_{k,2}) = (g^{r_k}, g^{w_{j_k}} h^{r_k}) \end{aligned}$$

In order to express $\mathbb{x}'$ and $\mathbb{w}'$ as a statement and witness for R_{eqrep}, form them as follows: $\mathbb{x}' = (\mathbb{G}, \{x'_i, \{g'_{i,1}, \ldots, g'_{i,m'}\}\}_{i=1}^{n'})$, where

$n' = n + 2|J|$, $m' = m + |J|$
For $1 \leq i \leq n$, $x'_i = x_i$, and for $1 \leq j \leq m$, $g'_{i,j} = g_{i,j}$, and for $m < j \leq m + |J|$, $g'_{i,j} = 1$.
For $1 \leq k \leq |J|$, $x'_{n+2(k-1)+1} = c_{k,1}$, $g'_{n+2(k-1)+1,m+k} = g$, and for $\ell \neq m + k$, $1 \leq \ell \leq m + |J|$, $g'_{n+2(k-1)+1,\ell} = 1$.
For $1 \leq k \leq |J|$, $x'_{n+2k} = c_{k,2}$, $g'_{n+2k,j_k} = g$, $g'_{n+2k,m+k} = h$, and for $\ell \notin \{j_k, m + k\}$, $1 \leq \ell \leq m + |J|$, $g'_{n+2(k-1)+1,\ell} = 1$.

Set $\mathbb{w}' = (w_1, \ldots, w_m, r_1, \ldots, r_k)$. Using the algorithm $\mathsf{P}^{\mathsf{S}}_{\mathsf{eqrep}}$, compute $\pi_{\mathsf{eqrep}} \leftarrow \mathsf{PoK}_{\Psi_{\mathsf{eqrep}}}\left\{\mathbb{w}' : R_{\mathsf{eqrep}}(\mathbb{x}', \mathbb{w}')\right\}$, and output $\pi = (\{(c_{k,1}, c_{k,2})\}, \pi_{\mathsf{eqrep}})$.

V works as follows: on input the statement $\mathbb{x}$, and the proof $\pi = (\{(c_{k,1}, c_{k,2})\}, \pi_{\mathsf{eqrep}})$, first compute $\mathbb{x}'$ exactly the same way as the prover's algorithm P did. Then output $\mathsf{V}^{\mathsf{S}}_{\mathsf{eqrep}}(\mathbb{x}', \pi_{\mathsf{eqrep}})$.

Theorem 1. *Let the relation $R_{\mathsf{eqrep}} = \{(\mathbb{x}, \mathbb{w})\}$ be an equality of discrete logarithm representations relation. For any $J \subseteq [m]$, let $f(J, \mathbb{w}) = \{g^{w_j} \ : \ j \in J\}$. The proof system $\Psi = (\mathsf{S}, \mathsf{P}, \mathsf{V})$ is an $f(J, \cdot)$-BB-PSL simulation-extractable NIZK proof system in the random-oracle model.*

Proof (Sketch). We need to describe the setup simulator, the proof simulator, the extractor trapdoor and the two extractors.

$\mathsf{SimS}(\mathsf{state}, m) \to (\mathsf{state}, h', \tau_{\mathsf{Ext}})$: On input the description of a group $\mathbb{G}$ with generator g of order q, sample $\tau_{\mathsf{Ext}} \leftarrow \mathbb{Z}_q$ and output the hash value that will be interpreted as the element $g^{\tau_{\mathsf{Ext}}}$ of $\mathbb{G}$; we can think of this as the public key of the ElGamal cryptosystem for secret key τ_{Ext}. On other inputs simulate the random oracle faithfully.

$\mathsf{Sim}(\mathsf{state}, \mathbb{x}) \to (\mathsf{state}, \pi)$: On input $\mathbb{x}$, the simulator extends $\mathbb{x}$ with random ElGamal ciphertexts to $\mathbb{x}'$, chooses $c \leftarrow \mathbb{Z}_q$, $s_j \leftarrow \mathbb{Z}_q$ for $1 \le j \le m + |J|$, and sets $d_i = (\prod_{j=1}^{m+k} g_{i,j}^{s_j})/x_i^c$ for $1 \le i \le n + 2|J|$. He then sets $\mathsf{com} = (d_1, \ldots, d_{n+2|J|})$, stores $H[\mathbb{x}, \mathsf{com}] = c$ in state, sets $\mathsf{chal} = c$, and $\mathsf{res} = (s_1, \ldots, s_m)$ and return $(\mathsf{chal}, \mathsf{res})$.

$\mathsf{Ext}^{\mathsf{BB}(\mathcal{A})}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi) \to \mathbb{w}$: Parse π as $(\mathsf{chal}, \mathsf{res})$ and compute com as Sim. Rewind $\mathsf{BB}(\mathcal{A})$ to the point where it queried the random oracle on $(\mathbb{x}, \mathsf{com})$ and provide it fresh random results. Repeat until it obtains two accepting transcripts for the same $\mathsf{com} = (d_1, \ldots, d_{n+2|J|})$ and then run the extractor of the Σ-protocol to obtain $\mathbb{w}'$. Remove the last k elements to obtain $\mathbb{w}$.

$\mathsf{ExtSL}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi) \to f(J, \mathbb{w})$: Parse $\mathbb{x}$ as $(\mathbb{G}, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{n+2|J|})$, obtain τ_{Ext} from the entry $(\mathbb{G}, h, \tau_{\mathsf{Ext}})$ of $\mathcal{Q}_{\mathsf{S}}$. Interpret the last $2|J|$ elements x_i as ElGamal ciphertext and decrypt them to obtain $f(J, \mathbb{w})$. □

3 Definition of Security of f-Blueprint Scheme

Our scheme features three parties: an auditor, a set of users, and a set of recipients. It is tied to a non-interactive commitment scheme $(\mathsf{CSetup}, \mathsf{Commit})$; let *cpar* be the parameters of the commitment scheme output by CSetup. The auditor A has private input x and publishes a commitment $C_{\mathsf{A}} = \mathsf{Commit}_{cpar}(x)$. The user has private data y and publishes a commitment $C = \mathsf{Commit}_{cpar}(y)$. For example, x could be a list and y could be the user's attributes in a credential system. The auditor creates a key pair $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}})$ corresponding to its input x, and the user can escrow its private data y under pk_{A} to obtain an escrow Z. We require that Z decrypts (with the help of sk_{A}) to $f(x, y)$ for a function f that all parties have agreed upon in advance. In the definition, we do not restrict f: it can be any efficiently computable function. Moreover, an escrow recipient R can verify that indeed Z was computed correctly for the given pk_{A} and C. Similarly, a privacy-conscious user can verify that indeed pk_{A} was computed correctly for the given warrants data commitment C_{A}.

Definition 3. *An f-blueprint scheme tied to a non-interactive commitment scheme* $(\mathsf{CSetup}, \mathsf{Commit})$ *consists of the following probabilistic polynomial time algorithms:*

$\mathsf{Setup}(1^\lambda, cpar) \to \Lambda$: is the algorithm that sets up the public parameters Λ. It takes as input the security parameter 1^λ and the commitment parameters *cpar* output by $\mathsf{CSetup}(1^\lambda)$; to reduce the number of inputs to the rest of

the algorithms, Λ includes 1^λ and *cpar*; we will also write Commit instead of Commit_{cpar} to reduce notational overhead.

$\mathsf{KeyGen}(\Lambda, x, r_\mathsf{A}) \to (\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$: is the key generation algorithm for auditor A. It takes in input 1^λ, parameters Λ, and values (x, r_A), and outputs the key pair $(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$. The values (x, r_A) define a commitment C_A. This allows to integrate KeyGen into larger systems.[3]

$\mathsf{VerPK}(\Lambda, \mathsf{pk}_\mathsf{A}, C_\mathsf{A}) \to 1$ or 0: is the algorithm that, on input the auditor's public key pk_A and a commitment C_A, verifies that the warrant public key was computed correctly for the commitment C_A.

$\mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, y, r) \to Z$: is the algorithm that, on input the values (y, r) outputs an escrow Z for commitment $C = \mathsf{Commit}(y; r)$.

$\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z) \to 1$ or 0: is the algorithm that, on input the auditor's public key pk_A, a commitment C, and an escrow Z, verifies that the escrow was computed correctly for the commitment C.

$\mathsf{Decrypt}(\Lambda, \mathsf{sk}_\mathsf{A}, C, Z) \to f(x, y)$ or $\perp$: is the algorithm that, on input the auditor's secret key sk_A, a commitment C and an escrow Z such that $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z) = 1$, decrypts the escrow. Our security properties will ensure that it will output $f(x, y)$ if C is a commitment to y.

Definition 4 (Secure blueprint). *An f-blueprint scheme* $\mathsf{Blu} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{VerPK}, \mathsf{Escrow}, \mathsf{VerEscrow}, \mathsf{Decrypt})$ *tied to commitment scheme* $(\mathsf{CSetup}, \mathsf{Commit})$ *constitutes a secure f-blueprint scheme if it satisfies the following properties:*

Correctness of VerPK **and** $\mathsf{VerEscrow}$: Values $(cpar, \mathsf{pk}_\mathsf{A}, C_\mathsf{A}, C, Z)$ are generated honestly if: (1) *cpar* is generated by $\mathsf{CSetup}(1^\lambda)$; (2) Λ is generated by $\mathsf{Setup}(1^\lambda, cpar)$; (3) pk_A is the output of $\mathsf{KeyGen}(\Lambda, x, r_\mathsf{A})$; (4) $C_\mathsf{A} = \mathsf{Commit}_{cpar}(x; r_\mathsf{A})$; (5) $C = \mathsf{Commit}_{cpar}(y; r)$; (6) Z is generated by $\mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, y, r)$. For honestly generated values $(cpar, \mathsf{pk}_\mathsf{A}, C_\mathsf{A}, C, Z)$, we require that algorithms $\mathsf{VerEscrow}$ and VerPK accept with probability 1.

Correctness of $\mathsf{Decrypt}$: Similarly, we require for honestly generated $(cpar, \mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A}, C, Z)$ that with overwhelming probability $\mathsf{Decrypt}(\Lambda, \mathsf{sk}_\mathsf{A}, C, Z) = f(x, y)$.

Soundness: Let C_A and C be commitments whose openings (x, r_A) and (y, r) are known to the adversary. Let $(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_\mathsf{A})$ be honestly derived keys. Soundness guarantees that any $\mathsf{pk}_\mathsf{A}, Z$ pair that passes $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z)$ will decrypt to $f(x, y)$ with overwhelming probability. More formally, for all PPT adversaries $\mathcal{A}$ involved in the experiment in Fig. 3.1, there exists a negligible function ν such that: $\mathsf{Adv}^{\mathsf{Sound}}_{\mathcal{A},\mathsf{Blu}}(\lambda) = \Pr\left[\mathsf{Sound}^{\mathcal{A}}_{\mathsf{Blu}}(\lambda) = 1\right] = \nu(\lambda)$

Blueprint Hiding: We want to make sure that pk_A just reveals that x is a valid first argument to f (i.e. this may possibly reveal the size of x or an upper bound on its size). 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 $\mathsf{Decrypt}(\Lambda, \mathsf{sk}_\mathsf{A}, \cdot, \cdot)$.

[3] E.g., A can prove that x does not contain journalists, but does contain all Russian oligarchs on the OFAC's sanctions list. `https://home.treasury.gov/policy-issues/financial-sanctions`

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

1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
3 : $x, r_{\mathsf{A}} \leftarrow \mathcal{A}(1^\lambda, \Lambda)$
4 : $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_{\mathsf{A}})$
5 : $(C, y, r, Z) \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{A}})$
6 : **return** $[C = \mathsf{Commit}(y; r) \wedge$
7 : $\quad \mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C, Z) \wedge \mathsf{Decrypt}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z) \neq f(x, y)]$

Fig. 3.1: Experiments $\mathsf{Sound}^{\mathcal{A}}_{\mathsf{Blu}}(\lambda)$

We formalize this security property by requiring that there exist a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimKeygen}, \mathsf{SimDecrypt})$ such that a PPT adversary cannot distinguish between the following two games: the "real" game in which Λ is chosen honestly, the public key pk_{A} is computed correctly for adversarially chosen x, r_{A}, and the adversary's decryption queries (C, Z) are answered with $\mathsf{Decrypt}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z)$; and the "ideal" game in which Λ is computed using $\mathsf{SimSetup}$, the public key 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 $f(x, y_i)$. When we say "enough information," we mean that $\mathsf{SimDecrypt}$ obtains $y_i^* = g(y_i)$ for some function g such that $f(x, y) = f^*(x, g(y))$ for an efficiently computable f^*, for all possible inputs (x, y)[4].

More formally, for all probabilistic poly-time adversaries $\mathcal{A}$ involved in the game described in Fig. 3.2, the advantage function satisfies:

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

for some negligible ν.

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

[4] For example, if x is a list $(x_1, \ldots, x_n)$ and $f(x, y)$ checks if $y = x_i$ for some i, $g(y)$ can be a one-way permutation: in order to determine whether y is on the list, it is sufficient to compute $g(x_j)$ and compare it to $y^* = g(y)$.

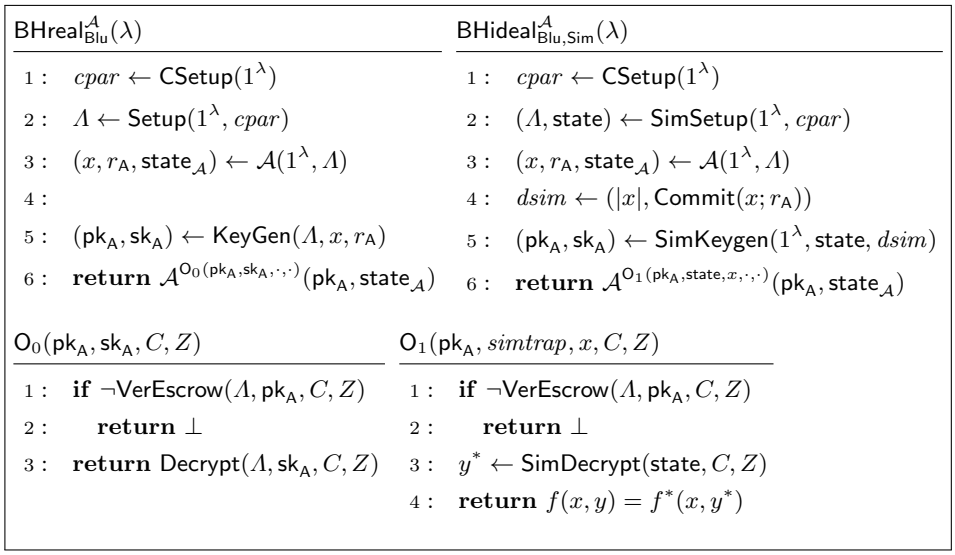

Fig. 3.2: Experiments $\mathsf{BHreal}^{\mathcal{A}}_{\mathsf{Blu}}(\lambda)$ and $\mathsf{BHideal}^{\mathcal{A}}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$

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 $\mathcal{A}$ involved in the game described in Fig. 3.3, the following equation holds for some negligible function ν:

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

Privacy with Honest Auditor: There exists a simulator Sim such that the adversary's views in the following two games are indistinguishable:

1. **Real Game**: The honest auditor generates the public key on input x provided by the adversary, and honest users follow the Escrow protocol on input adversarially chosen openings.
2. **Privacy-Preserving Game**: The honest auditor generates the public key on input x provided by the adversary. On input adversary-generated commitments and openings, the users run a simulator that is independent of y (although with access to the commitment C) 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. 3.4. We require that there exists a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimEscrow})$ such that, for any PPT adversary $\mathcal{A}$

$\mathsf{PADA}^{\mathcal{A},b}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$

1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda_0 \leftarrow \mathsf{Setup}(1^\lambda, cpar); (\Lambda_1, \mathsf{state}) \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
3 : $(x, r_\mathsf{A}, \mathsf{pk}_\mathsf{A}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, \Lambda_b)$
4 : **if** $\mathsf{VerPK}(\Lambda_b, \mathsf{pk}_\mathsf{A}, \mathsf{Commit}(x; r_\mathsf{A})) = 0$: **return** $\perp$
5 : **return** $\mathcal{A}^{\mathsf{O}_b(y,r)}(\mathsf{state}_\mathcal{A})$

$\mathsf{O}_0(y, r)$

1 : **return** $\mathsf{Escrow}(\Lambda_0, pk_\mathsf{A}, y, r)$

$\mathsf{O}_1(y, r)$

1 : **return** $\mathsf{SimEscrow}(\mathsf{state}, \Lambda_1, \mathsf{pk}_\mathsf{A}, \mathsf{Commit}(y; r),$
2 : $f(x, y))$

Fig. 3.3: Game $\mathsf{PADA}^{\mathcal{A},b}_{\mathsf{Blu}}(\lambda)$

involved in the game described in the figure, the following equation holds:

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

for some negligible function ν.

4 Homomorphic Enough Encryption

Definition 5 (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$(1^\lambda) \to hecpar$ is a PPT algorithm that, on input the security parameter, outputs the parameters *hecpar*; in case there is no HECSETUP algorithm, $hecpar = 1^\lambda$.

HECENC$(hecpar, f, x) \to (X, d)$ is a PPT algorithm that, on input the parameters *hecpar*, a function $f \in F$, and a value $x \in domain_{f,x}$, outputs an encrypted representation X of the function $f(x, \cdot)$, and a decryption key d.

HECEVAL$(hecpar, f, X, y) \to Z$ is a PPT algorithm that, on input the parameters *hecpar*, a function $f \in F$, an encrypted representation of $f(x, \cdot)$, and a value $y \in domain_{f,y}$, outputs a ciphertext Z, an encryption of $f(x, y)$.

HECDEC$(hecpar, d, Z) \to z$ is a polynomial-time algorithm that, on input the parameters *hecpar*, the decryption key d, and a ciphertext Z, decrypts Z to obtain a value z.

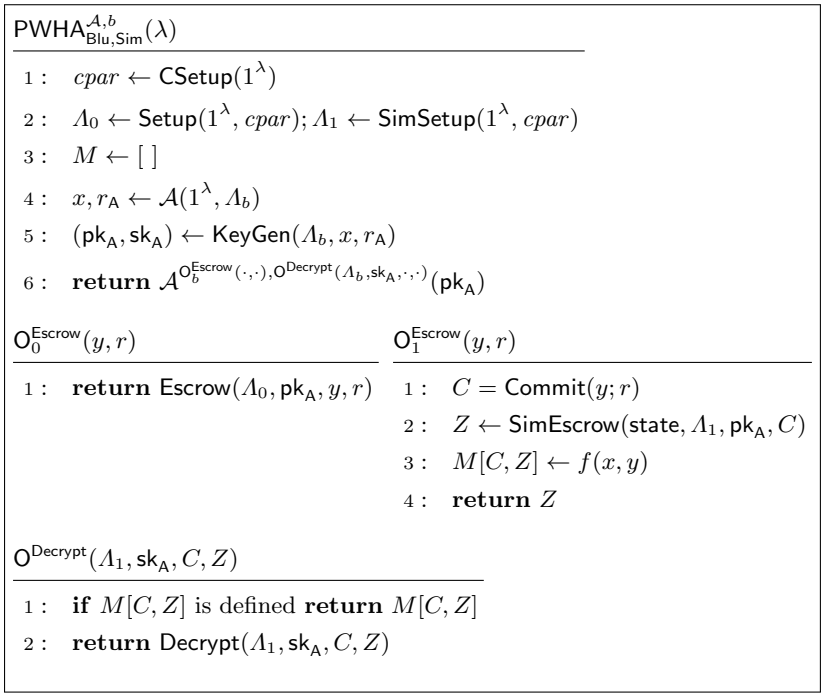

Fig. 3.4: Game $\mathsf{PWHA}^{\mathcal{A},b}_{\mathsf{Blu,Sim}}(\lambda)$

HECDIRECT$(hecpar, X, z) \to Z$ is a PPT algorithm that, on input *hecpar*, an encrypted representation X of some function, and a value z, outputs a ciphertext Z.

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

Security of x, security of x and y from third parties, and security of DirectZ. Consider Fig. 4.1. For a given HEC and an adversary $\mathcal{A}$, and for $b \in \{0, 1\}$, let $p^{\mathrm{SECX}}_{\mathcal{A},b}(\lambda)$ be the probability that $\mathcal{A}$ outputs 0 in experiment $\mathrm{SECX}^{\mathcal{A}}_b$, let $p^{\mathrm{SECXY}}_{\mathcal{A},b}(\lambda)$ be the probability that $\mathcal{A}$ outputs 0 in experiment $\mathrm{SECXY}^{\mathcal{A}}_b$, and let $p^{\mathrm{DIRECTZ}}_{\mathcal{A},b}(\lambda)$ be the probability that $\mathcal{A}$ outputs 0 in experiment $\mathrm{DIRECTZ}^{\mathcal{A}}_b$.

HEC provides *security for x* if or any PPT $\mathcal{A}$, $|p^{\mathrm{SECX}}_{\mathcal{A},0}(\lambda) - p^{\mathrm{SECX}}_{\mathcal{A},1}(\lambda)|$ is negligible. HEC provides *security for x and y from third parties* if or any PPT $\mathcal{A}$, $|p^{\mathrm{SECXY}}_{\mathcal{A},0}(\lambda) - p^{\mathrm{SECXY}}_{\mathcal{A},1}(\lambda)|$ is negligible. HEC provides *security of* DIRECTZ if or any PPT $\mathcal{A}$, $|p^{\mathrm{DIRECTZ}}_{\mathcal{A},0}(\lambda) - p^{\mathrm{DIRECTZ}}_{\mathcal{A},1}(\lambda)|$ is negligible.

Remark. Why do we need HECDIRECT? It allows us to directly form a ciphertext Z that will decrypt to a specific value z. If the function f is not one-way and it is easy, given z, to sample x and y such that $z = f(x, y)$, then we can de-

$\text{HECCORRECT}^{\mathcal{A}}(\lambda)$

1 : $hecpar \leftarrow \text{HECSETUP}(\lambda)$
2 : $(f, x, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$
3 : **if** $f \in F, x \in domain_{f,x}$
4 : $\quad (X, d) \leftarrow \text{HECENC}(hecpar, f, x)$
5 : $\quad (y, r_Z) \leftarrow \mathcal{A}(\mathsf{state}, X)$
6 : $\quad$ **if** $y \in domain_{f,y}$
7 : $\quad\quad Z \leftarrow \text{HECEVAL}(hecpar, f, X, y; r_Z)$
8 : $\quad\quad$ **if** $\text{HECDEC}(hecpar, d, Z) \neq f(x, y)$
9 : $\quad\quad\quad$ **return** 1
10 : $\quad\quad$ **return** 0
11 : $\quad$ **return** 0
12 : **return** 0

$\text{SECX}_b^{\mathcal{A}}(\lambda)$

1 : $hecpar \leftarrow \text{HECSETUP}(1^\lambda)$
2 : $(f, x_0, x_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$
3 : **if** $f \in F, x_0, x_1 \in domain_{f,x}$
4 : $\quad X, \_ \leftarrow \text{HECENC}(hecpar, f, x_b)$
5 : $\quad$ **return** $\mathcal{A}(hecpar, X, \mathsf{state})$
6 : **return** $\mathcal{A}(\perp, \mathsf{state})$

$\text{SECXY}_b^{\mathcal{A}}(\lambda)$

1 : $hecpar \leftarrow \text{HECSETUP}(1^\lambda)$
2 : $(f, x_0, x_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$
3 : **if** $f \in F, x_0, x_1 \in domain_{f,x}$
4 : $\quad X, \_ \leftarrow \text{HECENC}(hecpar, f, x_b)$
5 : $\quad (y_0, y_1, \mathsf{state}) \leftarrow \mathcal{A}(X, \mathsf{state})$
6 : $\quad$ **if** $y_0, y_1 \in domain_{f,y}$
7 : $\quad\quad Z \leftarrow \text{HECEVAL}(hecpar, f, X, y_b)$
8 : $\quad\quad$ **return** $\mathcal{A}(Z, \mathsf{state})$
9 : $\quad$ **return** $\mathcal{A}(\perp, \mathsf{state})$
10 : **return** $\mathcal{A}(\perp, \mathsf{state})$

$\text{DIRECTZ}_b^{\mathcal{A}}(\lambda)$

1 : $hecpar \leftarrow \text{HECSETUP}(1^\lambda)$
2 : $(f, x, y, r_X, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$
3 : **if** $f \in F, x \in domain_{f,x}, y \in domain_{f,y}$
4 : $\quad X, \_ = \text{HECENC}(hecpar, f, x; r_X)$
5 : $\quad Z_0 \leftarrow \text{HECEVAL}(hecpar, f, X, y)$
6 : $\quad Z_1 \leftarrow \text{HECDIRECT}(hecpar, X, f(x, y))$
7 : $\quad$ **return** $\mathcal{A}(hecpar, Z_b, \mathsf{state})$
8 : **return** $\mathcal{A}(\perp, \mathsf{state})$

Fig. 4.1: HEC correctness and security games

rive such Z by computing $(X, d) = \text{HECENC}(hecpar, f, x)$ and then computing $Z = \text{HECEVAL}(hecpar, f, X, y)$. But in general, it is helpful (for some applications) to have a separate algorithm $\text{HECDIRECT}(hecpar, X, z)$ such that, if $X = \text{HECENC}(hecpar, f, x)$, then $Z = \text{HECDIRECT}(hecpar, X, z)$ decrypts to z using the decryption key that corresponds to X, i.e. $z = \text{HECDEC}(hecpar, d, Z)$.

5 A Generic f-Blueprint scheme from HEC

We construct a privacy-preserving blueprint scheme using a commitment scheme, a homomorphic-enough cryptosystem, as well as two NIZK proof systems as building blocks. The scheme consists of the following six algorithms:

Setup takes λ and a commitment setup as input and generates *hecpar* and assigns the NIZK oracles S_1 and S_2. Note that when instantiated using real hash functions or reference strings both RO and CRS setups can be represented as bit-strings in implementations. KeyGen uses the HEC scheme to compute an encrypted representation of the function $f(x, \cdot)$ and proves that it was computed correctly. VerPK verifies that pk_A was computed correctly with respect to the auditor's commitment C_A. Escrow homomorphically evaluates $f(x, \cdot)$ on y to

obtain a ciphertext and proves that it was formed correctly. VerEscrow verifies the ciphertext with respect to the user's commitment C, and Decrypt decrypts.

Our construction in Fig. 5.1 uses VerPK as a subroutine in Escrow and VerEscrow. To be consistent with the syntax we add C_{A} to pk_{A}. Similarly, we use VerEscrow in Decrypt and add pk_{A} to sk_{A}.

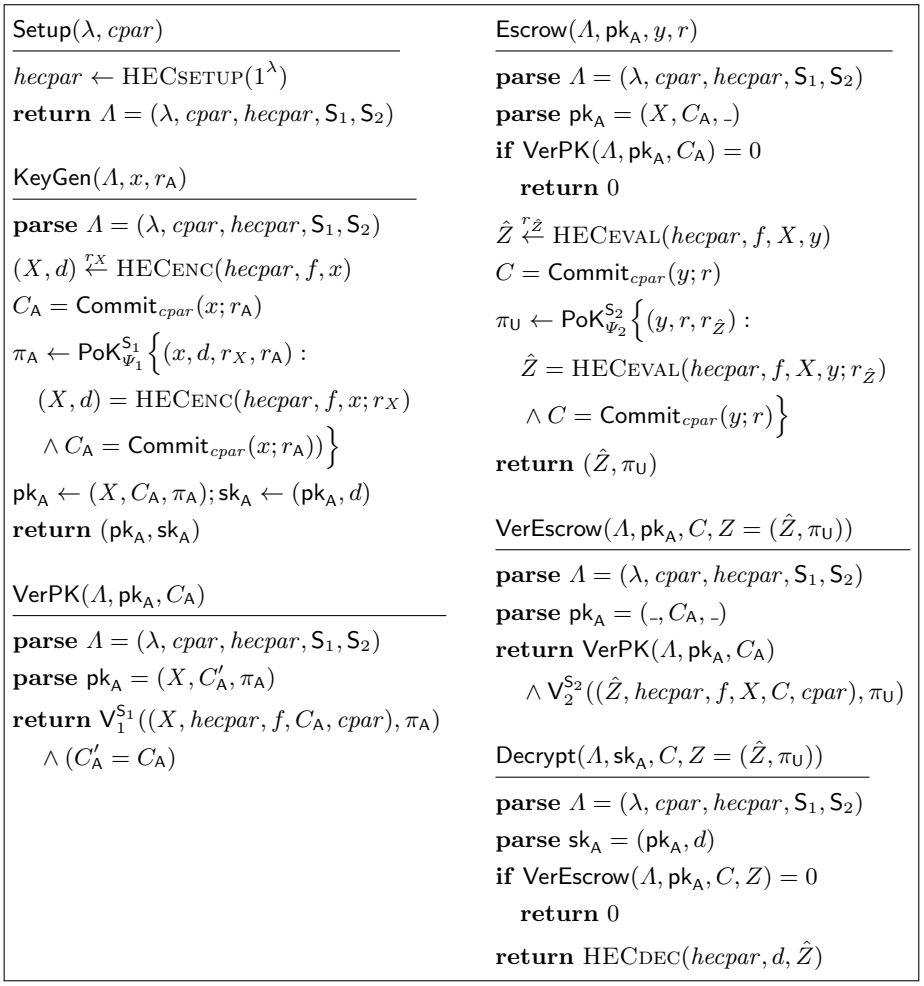

Fig. 5.1: Construction of generic f-blueprint scheme

Theorem 2. *If* HEC *is a secure homomorphic-enough cryptosystem, the commitment scheme is binding, and the NIZK PoKs Ψ_1 and Ψ_2 are zero-knowledge and BB-PSL simulation extractable then our generic blueprint scheme is a secure f-blueprint scheme.*

Note that, our formal security theorem does not require the commitment to be hiding. It only shows, using simulation, that no additional information besides the commitment is revealed. To benefit from the hiding and privacy properties of the blueprint scheme it is, however, crucial that the transaction system employing it uses a hiding commitment scheme.

We prove correctness of VerEscrow and VerPK, correctness of Decrypt, soundness, blueprint hiding, privacy against dishonest auditor, and privacy with honest auditor in separate lemmas. The statement and proof of these lemmas are in the full version [52].

6 HEC from the ElGamal Cryptosystem

For a binary string y (or an integer which can be interpreted as a binary string) and an integer k, let $\mathsf{lobits}_k(y) = y \bmod 2^k$; i.e., $\mathsf{lobits}_k(y)$ denotes the k least significant bits of y (or, equivalently, the corresponding integer). Let $domain_{f,y} = \{0,1\}^{l_y}$. Let us use bold font to indicate that $\mathbf{x}$ is a set of values; let $W_\ell = \{\mathbf{x} \mid \mathbf{x} \subseteq domain_{f,y}, |\mathbf{x}| = \ell\}$. Let the function family $F_\ell = \{f_k\}$, where $f_k : W_\ell \times domain_{f,y} \mapsto domain_{f,y}$ is defined as follows:

$$f_k(\mathbf{x}, y) = \begin{cases} y & \mathsf{lobits}_k(y) \in \mathbf{x} \\ \emptyset & \text{otherwise} \end{cases}$$

In other words, the function reveals y if $\mathsf{lobits}_k(y) \in \mathbf{x}$, and nothing otherwise.

In this section, we will use the ElGamal cryptosystem in order to construct an HEC for $f_\ell \in F_\ell$ for any k, ℓ such that ℓ and $2^{l_y - k}$ are polynomial in λ. Our cryptosystem will use a group G of prime order $q > 2^{l_y}$.

6.1 The ElGamalHEC Construction and Its Security

The idea of our construction ElGamalHEC for a HEC for functions $f_k \in F_\ell$, is that HECenc outputs the ElGamal ciphertexts of the coefficients of a random polynomial P of degree $\ell = |\mathbf{x}|$ whose roots are elements of $\mathbf{x}$. More precisely, $P = s\prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$, that is $P = \sum_{i=0}^{|\mathbf{x}|} P_i \chi^i$. The randomness in P comes from the choice of the leading coefficient s. HECenc outputs the ciphertexts $\mathbf{C}_i \leftarrow \mathsf{Enc}(g^{P_i})$ that encrypt the coefficients P_i of P; these ciphertext are part of X.

Using these ciphertexts $\{C_i\}$ and the homomorphic properties of ElGamal, HECeval computes an encryption of $g^{rP(\mathsf{lobits}_k(y))+y}$ for a random r. Note that if $\mathsf{lobits}_k(y) \in \mathbf{x}$, this is just an encryption of g^y; otherwise, it is an encryption of a random element of G. Thus, HECdec can use the ElGamal decryption algorithm to obtain some group element g^z, and then use the fact that ℓ and $2^{l_y - k}$ to either recover y with exhaustive search, or determine that $f_k(\mathbf{x}, y) = \emptyset$.

Fig. 6.1 describes our construction, ElGamalHEC. Here, $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ are the key generation, encryption, and decryption algorithms of ElGamal. Recall that $\oplus$ is the homomorphic operator for ciphertexts.

Theorem 3. *Under the decisional Diffie-Hellman assumption, ElGamalHEC constitutes a homomorphic-enough encryption for f_k any k, ℓ such that ℓ and $2^{l_y - k}$ are polynomial in λ, for any $f_k \in F_\ell$.*

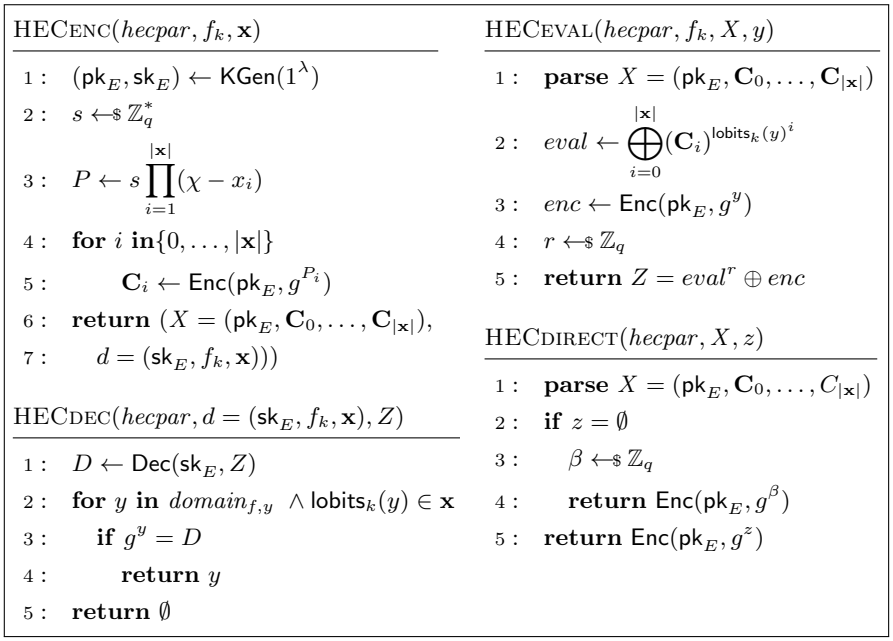

Fig. 6.1: Our Construction ElGamalHEC

We prove each of the required security properties in a separate lemma in the full version [52]. As surprisingly, one of the most challenging lemmas is **HECCorrectness** due to the adversaries control over the evaluation randomness, we reproduce it here.

Lemma 1. *Under the decisional Diffie-Hellman assumption, ElGamalHEC satisfies the correctness property of* HEC *for* f_k.

Proof. Let $\mathcal{A}$ be a PPT adversary playing the HEC correctness game with ElGamalHEC. Let $\epsilon_{\mathcal{A}}(1^\lambda)$ be the probability that the challenger accepts. Below, we (1) provide modified games G_0 and G_1 such that the probability that the challenger in G_0 accepts is also $\epsilon_{\mathcal{A}}(1^\lambda)$; (2) prove that the probability $\epsilon'_{\mathcal{A}}(1^\lambda)$ that the challenger in G_1 accepts is negligible; (3) give a reduction $\mathcal{B}_{\text{HEC}}$ that breaks the security of the ElGamal cryptosystem with advantage $\epsilon_{\mathcal{A}}(1^\lambda) - \epsilon'_{\mathcal{A}}(1^\lambda)$. Since the ElGamal cryptosystem is secure under the DDH assumption, it follows that, under the DDH assumption, $\epsilon_{\mathcal{A}}(1^\lambda)$ is negligible.

(1) First, consider the following game G_0, which is the same as the HEC correctness game with our ElGamal instantiation, except that actual polynomial evaluation instead of homomorphic evaluation. G_0 first obtains $(f, \mathbf{x}, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$; if $f \in F, \mathbf{x} \in domain_{f,\mathbf{x}}$, then it computes X as HECENC($hecpar$, $f_k, \mathbf{x}$), except that it renames P into P_0 and s into s_0, i.e., it computes a polynomial $P_0(\chi) = s_0 \prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$, where $\{x_i\} = \mathbf{x}$. Next, it invokes $\mathcal{A}$ again to

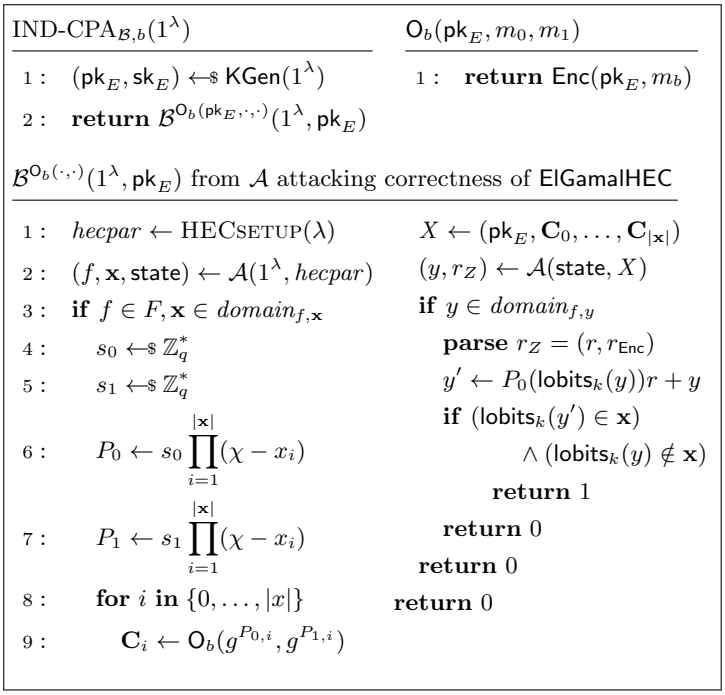

Fig. 6.2: Reduction for part (3) of the proof of Lemma 1

receive $(y, r_Z) \leftarrow \mathcal{A}(\mathsf{state}, X)$; from it, it computes $y' \leftarrow P_0(\mathsf{lobits}_k(y))r_Z + y$. If $(\mathsf{lobits}(y') \in \mathbf{x}) \wedge (\mathsf{lobits}(y) \notin \mathbf{x})$, accept. That is, instead of a homomorphic evaluation of P_0 using the ciphertexts $C_0, \ldots, C_{|\mathbf{x}|}$, followed by decrypting the resulting ciphertext, it performs an actual evaluation of P_0. Observe that the probability that the challenger in G_0 accepts is the same as in the original correctness game due to the correctness of homomorphic polynomial evaluation.

Second, consider the game G_1 that proceeds similarly to G_0: in addition to polynomial P_0 it computes a polynomial $P_1(\chi) = s_1 \prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$ with its own random value s_1 that it uses within HECENC (instead of P_0). Thus, X consists of the ciphertexts that correspond to coefficients of P_1. Running HECEVAL followed by HECDEC on input y would correspond to homomorphically evaluating $P_1(y)$; instead, G_1 (like G_0) uses P_0 to compute $y' \leftarrow P_0(\mathsf{lobits}_k(y))r_Z + y$ and accepts if $(\mathsf{lobits}(y') \in \mathbf{x}) \wedge (\mathsf{lobits}(y) \notin \mathbf{x})$.

(2) Let us prove that the probability $\epsilon'_{\mathcal{A}}(1^\lambda)$ that the challenger in G_1 accepts is negligible. The challenger will accept only if $\mathsf{lobits}(y) \notin \mathbf{x}$, so let us consider this case. Then $P_0(\mathsf{lobits}_k(y)) \neq 0$, and, since s_0 is random, $y' = P_0(\mathsf{lobits}_k(y)) \neq 0$ is independent of $\mathcal{A}$'s view. Thus, for any $x \in \mathbf{x}$, $\Pr[\mathsf{lobits}(y') = x] \approx 2^{-k}$, thus the probability that G_1 accepts is $\approx |\mathbf{x}|2^{-k}$.

(3) We construct the reduction $\mathcal{B}_{\mathrm{HEC}}$ to the security of the ElGamal cryptosystem. Recall that, under the DDH assumption, the ElGamal cryptosystem

is CPA-secure using the formulation of Boneh and Shoup (see Sect. 2); we use this version of (multi-instance) CPA-security in our reduction (this makes the proof simpler as it avoids the hybrid argument). $\mathcal{B}_{\text{HEC}}$ creates both polynomials $P_j \leftarrow s_j \prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$, $j \in \{0,1\}$. Let $P_{j,i}$ be their coefficients. It obtains the encryption of the coefficients of one of these polynomials via the ElGamal challenger: $\mathbf{C}_i \leftarrow \mathsf{O}_b(g^{P_{0,i}}, g^{P_{1,i}})$. This is described in more detail in Fig. 6.2. Observe that, $\mathcal{B}_{\text{HEC}}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$ creates the same view for $\mathcal{A}$ as G_b. Therefore, $\Pr\big[\text{IND-CPA}_{\mathcal{B}_{\text{HEC}},0}(1^\lambda) = 0\big] = \epsilon_{\mathcal{A}}(1^\lambda)$ and $\Pr\big[\text{IND-CPA}_{\mathcal{B}_{\text{HEC}},1}(1^\lambda) = 0\big] = \epsilon'_{\mathcal{A}}(1^\lambda)$. Since ElGamal is CPA-secure under the DDH assumption, $\epsilon_{\mathcal{A}}(1^\lambda) - \epsilon'_{\mathcal{A}}(1^\lambda) \leq \Pr\big[\text{IND-CPA}_{\mathcal{B}_{\text{HEC}},0}(1^\lambda) = 0\big] - \Pr\big[\text{IND-CPA}_{\mathcal{B}_{\text{HEC}},1}(1^\lambda) = 0\big]$ is negligible as required. □

6.2 From ElGamalHEC to an Efficient Secure Blueprint Scheme

In order to use our HEC construction in Fig. 6.1 to construct our Generic f-blueprint scheme in Fig. 5.1, we need a BB simulation extractable proof system for Ψ_1 to prove knowledge of the witness in the following relation:

$$\left\{ \begin{matrix} \mathbb{x} = (X, \mathit{hecpar}, f, C_{\mathsf{A}}, \mathit{cpar}), \\ \mathbb{w} = (\mathbf{x}, d, r_X, r_{\mathsf{A}}) \end{matrix} \middle| \begin{matrix} (X, d) = \text{HEC}_{\text{ENC}}(\mathit{hecpar}, f, \mathbf{x}; r_X) \wedge \\ C_{\mathsf{A}} = \mathsf{Commit}_{\mathit{cpar}}(\mathbf{x}; r_{\mathsf{A}}) \end{matrix} \right\}$$

The building blocks of this relation are statements about the message and randomness of ElGamal encryption and the opening of Pedersen commitments that can be expressed as statements about discrete logarithms representations in R_{eqrep}. By Theorem 1, we have a BB simulation-extractible NIZK proof system for R_{eqrep} and in extension Ψ_1.

For our specific construction, we assume that the auditor's commitment C_{A} contains commitments to coefficients of the polynomial $P' = \prod_{i=1}^{|\mathbf{x}|}(\chi - \mathbf{x}_i)$. To prove that we encrypted some polynomial $P = sP'$ involves proving that $P = sP'$. We first prove that we have properly encrypted the coefficients of P'. Then, we can exponentiate these encrypted values by s, effectively multiplying the coefficients by s. See the full version [52] for more details.

Additionally, we require that there exists a f'-BB-PSL simulation extractable proof system for Ψ_2 such that there exists an efficiently computable function f^* where $f^*(\mathbf{x}, f'(y)) = f(\mathbf{x}, y)$ for all $(f, \mathbf{x}, y) \in F \times \mathit{domain}_{f,\mathbf{x}} \times \mathit{domain}_{f,y}$. Recall that Ψ_2 is used to prove the following relation:

$$\left\{ \begin{matrix} \mathbb{x} = (\hat{Z}, \mathit{hecpar}, f, X, C, \mathit{cpar}), \\ \mathbb{w} = (y, r, r_{\hat{Z}}) \end{matrix} \middle| \begin{matrix} \hat{Z} = \text{HEC}_{\text{EVAL}}(\mathit{hecpar}, f, X, y; r_{\hat{Z}}) \wedge \\ C = \mathsf{Commit}_{\mathit{cpar}}(y; r) \end{matrix} \right\}$$

We need a range proof to prove that $\mathsf{lobits}_k(y)$ is used to generate Eval in $\hat{Z}$. This can be done using Bulletproofs [12]. The rest of the building blocks for the relation involves statements about ElGamal encryption and Pedersen commitments, we can again be expressed as eqrep relation statement.

Theorem 1 guarantees a $f(J, \cdot)$-BB-PSL simulation extractable NIZK system for eqrep, and in extension Ψ_2. Recall that $f(J, \mathbb{w}) = \{g^{\mathbb{w}_j} : j \in J\}$. Here, if we

choose J to be a singleton containing just the index corresponding to y in $\mathbb{w}$, we get a g^y-BB-PSL simulation extractable NIZK system. Luckily, knowing $\mathbf{x}$ and y is sufficient to compute $f(\mathbf{x}, y)$. Here, $f^*(\mathbf{x}, g^y)$ can be computed similar to HECDEC in Fig. 6.1. We first iterate over all y' values such that $\mathsf{lobits}_k(y') \in \mathbf{x}$. If $g^{y'} = g^y$, we return y'. If no such value exists, we return $\emptyset$. Since $|\mathbf{x}|2^{l_y - k}$ is polynomial in λ, f^* is efficiently computable. See full version [52] for more details.

7 HEC for any f from Fully Homomorphic Encryption

Definition 6 (Circuit-private (CP) fully homomorphic encryption (FHE)). *A tuple of algorithms* $(\mathsf{FHEKeyGen}, \mathsf{FHEEnc}, \mathsf{FHEDec}, \mathsf{FHEEval})$ *constitute a secure fully homomorphic public-key encryption scheme [46,10,9,47] 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 input a ciphertext c and outputs the decrypted bit* $b \in \{0,1\}$. $\mathsf{FHEEval}(FHEPK, \Phi, c_1, \ldots, c_n)$ *takes as input a public key, a Boolean circuit* $\Phi : \{0,1\}^n \mapsto \{0,1\}$*, and n ciphertexts and outputs a ciphertext* c_Φ*; correctness (below) ensures that* c_Φ *is an encryption of* $\Phi(b_1, \ldots, b_n)$ *where* c_i *is an encryption of* b_i*.*

Correctness of evaluation: *For any integer n (polynomial in* λ*) for any circuit* Φ *with n inputs of size that is polynomial in* λ*, for all* $x \in \{0,1\}^n$*, the event that* $\mathsf{FHEDec}(FHESK, C) \neq \Phi(x)$ *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* $c_\Phi = \mathsf{FHEEval}(FHEPK, \Phi, c_1, \ldots, c_n)$*, 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_Φ *should be of a fixed length that is independent of the size of the circuit* Φ *and of n. More formally, there exists a polynomial* $s(\lambda)$ *such that for all circuits* Φ*, 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_Φ *generated by* $\mathsf{FHEEval}(FHEPK, \Phi, c_1, \ldots, c_n)$ *is at most* $s(\lambda)$ *bits long.*

An FHE scheme is, additionally, ***circuit-private*** *[46,60,8,35] for a circuit family* $\mathcal{C}$ *if for any* PPT *algorithm* $\mathcal{A}$ *that outputs* $(R, \Phi_0, \Phi_1, (x_1, r_1), \ldots, (x_n, r_n))$*, the probability of distinguishing the homomorphic evaluation of* Φ_0 *on* $\{c_i = \mathsf{FHEEnc}(FHEPK, x_i; r_i)\}_{i \in [n]}$ *where FHEPK is computed as* $\mathsf{FHEKeyGen}(1^\lambda; R)$ *cannot be distinguished from the corresponding evaluation of* Φ_1 *on the same ciphertexts, as long as* $\Phi_0(x_1, \ldots, x_n) = \Phi_1(x_1, \ldots, x_n)$*.*

Bibliographic note. Definitions of circuit-privacy in the literature come in different flavors; we chose the formulation that makes it easiest to prove Theorem 4

below. The strongest, malicious circuit-privacy [60,35], is strictly stronger than what we give here; therefore, constructions that achieve it automatically achieve the definition here. Constructions of circuit-private FHE from regular FHE have been given by Ostrovsky et al. [60] and by Döttling and Dujmović [35].

Similarly, we chose to formulate correctness as perfect correctness, rather than allowing a negligible probability (over the randomness for the key generation, encryption, and evaluation) of a decryption error. Our construction below also achieves HEC from schemes that are strongly correct, i.e. where the probability of a decryption error is non-zero, but with high probability, no efficient adversary can find a public key and a set of ciphertexts and a circuit that will cause a decryption error. Achieving strong correctness from the more standard notion of correctness with overwhelming probability can be done with standard techniques, see the full version [52].

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 $\Phi^g_{y,z}(x)$ 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: $\mathcal{C}_j = \{\Phi^{f_j}_{y,z}(x) : y \in \{0,1\}^{\ell_y}, z \in \{0,1\}^2\}$.

$\text{HECSETUP}(1^\lambda) \to \Lambda$: Generate the FHE parameters Λ, if needed.

$\text{HECENC}(1^\lambda, \Lambda, f, x) \to (X, d)$: 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 $d = FHESK$.

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

$\text{HECDEC}(hecpar, d, Z) \to z$: Output $(\mathsf{FHEDec}(d, Z_1), \ldots, \mathsf{FHEDec}(d, Z_\ell))$.

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

Theorem 4. *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.

Proof. (Sketch) Correctness follows from the perfect correctness of FHE. Security of x by semantic security of FHE. Security of x and y from third parties is also by semantic security. Finally, the security of the direct encryption algorithm follows by circuit privacy.

Combining the fact that circuit-private FHE exists if and only FHE exists, and (as we saw earlier) the fact that HEC and simulation-extractable NIZK [33] give us a secure blueprint scheme, we have the following result:

Corollary 1. *If fully homomorphic encryption and simulation extractable NIZK exist, then for any function f, secure f-blueprint scheme is realizable.*

Acknowledgments

We thank Scott Griffy and Peihan Miao for helpful discussions, and the anonymous referees for constructive feedback. This research was supported by NSF awards #2154170 and #2154941, and by grants from Meta.

References

1. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (Aug 2000). https://doi.org/10.1007/3-540-44598-6_16
2. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 1087–1098. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516687
3. Bangerter, E., Camenisch, J., Lysyanskaya, A.: A cryptographic framework for the controlled release of certified data. In: Security Protocols Workshop. Lecture Notes in Computer Science, vol. 3957, pp. 20–42. Springer (2004)
4. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8_20
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_38
6. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_38
7. Boneh, D., Shoup, V.: A Graduate Course in Applied Cryptography. https://toc.cryptobook.us/
8. Bourse, F., del Pino, R., Minelli, M., Wee, H.: FHE circuit privacy almost for free. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 62–89. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_3
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). https://doi.org/10.1145/2090236.2090262
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 97–106. IEEE Computer Society Press (Oct 2011). https://doi.org/10.1109/FOCS.2011.12
11. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 423–443. Springer, Heidelberg (Feb 2020). https://doi.org/10.1007/978-3-030-51280-4_23

12. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00020
13. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_25
14. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) Proc. 13th ACM Conference on Computer and Communications Security. pp. 201–210. ACM (2006)
15. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-cash. In: Cramer, R. (ed.) Advances in Cryptology — Eurocrypt 2005. LNCS, vol. 3494, pp. 302–321. Springer (2005)
16. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Balancing accountability and privacy using e-cash (extended abstract). In: Prisco, R.D., Yung, M. (eds.) Proceedings of the 5th International Conference on Security and Cryptography for Networks (SCN). Lecture Notes in Computer Science, vol. 4116, pp. 141–155. Springer (2006)
17. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer Verlag (2001)
18. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: SCN 2002. LNCS, vol. 2576, pp. 268–289 (2003)
19. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: CRYPTO 2004. LNCS, vol. 3152, pp. 56–72 (2004)
20. Camenisch, J., Lysyanskaya, A., Neven, G.: Practical yet universally composable two-server password-authenticated secret sharing. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 525–536. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382252
21. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_8
22. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (Aug 1997). https://doi.org/10.1007/BFb0052252
23. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. Ph.D. thesis, ETH Zürich (1998)
24. Chase, M., Miao, P.: Private set intersection in the internet setting from lightweight oblivious PRF. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 34–63. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1_2
25. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO '82. pp. 199–203. Plenum Press (1982)
26. Chaum, D.: Blind signature systems. In: CRYPTO '83. pp. 153–156. Plenum (1983)
27. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM **28**(10), 1030–1044 (Oct 1985)
28. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: CRYPTO '90. LNCS, vol. 403, pp. 319–327 (1990)

29. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6_22
30. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48658-5_19
31. Damgård, I.: On σ-protocols. Available at http://www.daimi.au.dk/~ivan/Sigma.ps (2002)
32. Damgård, I., Ganesh, C., Khoshakhlagh, H., Orlandi, C., Siniscalchi, L.: Balancing privacy and accountability in blockchain identity management. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 552–576. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75539-3_23
33. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_33
34. Diaz, J., Lehmann, A.: Group signatures with user-controlled and sequential linkability. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 360–388. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75245-3_14
35. Döttling, N., Dujmovic, J.: Maliciously circuit-private FHE from information-theoretic principles. Cryptology ePrint Archive, Report 2022/495 (2022), https://eprint.iacr.org/2022/495
36. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (Aug 1984)
37. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_5
38. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
39. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_10
40. Frankle, J., Park, S., Shaar, D., Goldwasser, S., Weitzner, D.J.: Practical accountability of secret processes. In: Enck, W., Felt, A.P. (eds.) USENIX Security 2018. pp. 657–674. USENIX Association (Aug 2018)
41. Fraser, A., Garms, L., Lehmann, A.: Selectively linkable group signatures—stronger security and preserved verifiability. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security. pp. 200–221. Springer International Publishing, Cham (2021)
42. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_1
43. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: CRYPTO '97. LNCS, vol. 1294, pp. 16–30 (1997)

44. Fujisaki, E., Okamoto, T.: Witness hiding protocols to confirm modular polynomial relations. In: The 1997 Symposium on Cryptograpy and Information Security. The Institute of Electronics, Information and Communcation Engineers, Fukuoka, Japan (Jan 1997), sCSI97-33D
45. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 397–426. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_14
46. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of STOC 2009. pp. 169–178 (2009)
47. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_5
48. Goldwasser, S., Park, S.: Public accountability vs. secret laws: Can they coexist? Cryptology ePrint Archive, Report 2018/664 (2018), https://eprint.iacr.org/2018/664
49. Green, M., Kaptchuk, G., Laer, G.V.: Abuse resistant law enforcement access systems. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 553–583. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77883-5_19
50. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_23
51. Kilian, J., Petrank, E.: Identity escrow. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 169–185. Springer, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055727
52. Kohlweiss, M., Lysyanskaya, A., Nguyen, A.: Privacy-preserving blueprints. Cryptology ePrint Archive, Paper 2022/1536 (2022), https://eprint.iacr.org/2022/1536, https://eprint.iacr.org/2022/1536
53. Libert, B., Nguyen, K., Peters, T., Yung, M.: Bifurcated signatures: Folding the accountability vs. anonymity dilemma into a single private signing scheme. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 521–552. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77883-5_18
54. Lysyanskaya, A.: Signature schemes and applications to cryptographic protocol design. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (Sep 2002)
55. Lysyanskaya, A., Rivest, R., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H., Adams, C. (eds.) Selected Areas in Cryptography. LNCS, vol. 1758 (1999)
56. Lysyanskaya, A., Rosenbloom, L.N.: Universally composable sigma-protocols in the global random-oracle model. Cryptology ePrint Archive, Report 2022/290 (2022), https://eprint.iacr.org/2022/290
57. Maurer, U.M.: Unifying zero-knowledge proofs of knowledge. In: Preneel, B. (ed.) AFRICACRYPT 09. LNCS, vol. 5580, pp. 272–286. Springer, Heidelberg (Jun 2009)
58. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proc. 8th ACM Conference on Computer and Communications Security. pp. 116–125. ACM press (Nov 2001)

59. Nguyen, K., Guo, F., Susilo, W., Yang, G.: Multimodal private signatures. In: CRYPTO 2022, Part II. pp. 792–822. LNCS, Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_27
60. Ostrovsky, R., Paskin-Cherniavsky, A., Paskin-Cherniavsky, B.: Maliciously circuit-private FHE. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 536–553. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_30
61. Ostrovsky, R., Skeith III, W.E.: Private searching on streaming data. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 223–240. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_14
62. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9
63. Sakai, Y., Emura, K., Hanaoka, G., Kawai, Y., Matsuda, T., Omote, K.: Group signatures with message-dependent opening. In: Abdalla, M., Lange, T. (eds.) PAIRING 2012. LNCS, vol. 7708, pp. 270–294. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-36334-4_18
64. Scafuro, A.: Break-glass encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 34–62. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_2
65. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_18
66. Tsiounis, Y., Yung, M.: On the security of ElGamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC'98. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (Feb 1998). https://doi.org/10.1007/BFb0054019