Privacy-Preserving Blueprints

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

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

Abstract. In a world where everyone uses anonymous credentials for all access control needs, it is impossible to trace wrongdoers, by design. This makes 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

It is a reasonable concern that information technology might enable totalitarian control and favor dystopian rather than utopian societies. Surprisingly, cryptography offers powerful answers on how to strike a balance between privacy and accountability. The study of anonymous credentials [23,43,15,42,16,17,2] 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.

It has been shown that this can be done at a limited rate. For example, N anonymous transactions total (also known as compact e-cash) [13], or N anonymous showings a day (compact e-tokens) [12], or at most N anonymous transactions with the same verifier (e-cash with money laundering control) [14] are all possible. It has also been shown that anonymous credentials are compatible with identity escrow [41,3]. This way, an appropriate authority can establish the identity of the user later, as needed. Of course, in an identity escrow scheme, the user is not in fact fully anonymous from such an authority. Such a system is not trustworthy if there is a possibility that this authority can be malicious or compromised.

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.

Let us describe a motivating application: anonymous e-cash with a secret watchlist. Consider standard anonymous e-cash [21,22,24,13]: 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 authorized that their transactions be traced by a de-anonymization auditor, and placed them on a watchlist. We need a mechanism that allows the 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.

A high-level definition of a privacy-preserving blueprint. In a privacy-preserving blueprint scheme, 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 or an e-token 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 Section 3.

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

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

where y_2 denotes the last $O(\log \lambda)$ 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.

Technical roadmap. We obtain both results above via the same general method: by first defining (Section 3) and 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 HECENC algorithm to encode her input x into a value X, and she also obtains a decryption key d for future use; next, Bob uses HECEVAL to compute an encryption Z of $z = f(x, y)$ from Alice's encoding X and his input y. Finally, Alice runs HECDEC 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 HECdirect 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 HECeval, even if Alice is malicious.

An HEC combined with an appropriate non-interactive zero-knowledge (NIZK) proof system gives us a generic construction of an f-blueprint as follows. 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 just verifying π_A. To compute the escrow Z, the user first 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 [38,47,30] which, in turn, can be realized from regular fully homomorphic encryption [38,9,8,39]. The circuit-privacy guarantee ensures that Z hides Bob's input y from a malicious Alice. Since here we don't aim for efficiency, general (inefficient) simulation-extractable NIZK PoK can be used for the proofs here. This instantiation of our generic construction is presented in Section 7.

Our practical construction for watchlists under the decision 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 ElGamal encryption [31,50] as a building block. Suppose as part of setup we are given a group G of order q in which the decisional Diffie-Hellman assumption holds. Let g be a generator of 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 a_0 and computes the coefficients $c_0, \ldots, c_n, c_{n+1}$ of the $n+1$-degree polynomial $p(\mathbf{x}) = \prod(\mathbf{x} - a_i)$ for which $a_0, a_1, \ldots, a_n$ are the n zeroes. The encoding X is an ElGamal encryptions $C_0, \ldots, C_{n+1}$ of the values $g^{c_0}, \cdots, g^{c_{n+1}}$ under the ElGamal public key pk $X = (C_0, \ldots, C_{n+1}, \mathsf{pk})$, and $d = \mathsf{sk}$.

Bob's algorithm HECeval computes Z as follows: first, it parses $y = y_1 \circ y_2$ (recall that y_2 denotes the last $O(\log \lambda)$ bits of y). Then Bob obtains an ElGamal encryption E of $g^{p(y_1)}$ from the encrypted coefficients $C_0, \ldots, C_{n+1}$. This is possible to do because the ElGamal cryptosystem is multiplicatively homomorphic, thus $E = C_0 C_1^{y_1} C_2^{y_1^2} \ldots C_{n+1}^{y_1^{n+1}}$ 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 [34], but there is a subtle difference: the polynomial encoded as part of X has an additional zero, a_0, that is unknown to Bob even in the event that he knows the rest of Alice's input x. This ensures that in the event that $f_w(x, y) = \perp$, Bob cannot set the randomness r in such a way that Z will decrypt to a value of his choice; instead, it will decrypt to a random value. Conversely, to prevent Alice from choosing a value a_0 that will match with one of Bob's values y_1 we require that the domains of a_0 and y_1 are disjoint.

Finally, $\text{HEC}\textsc{dec}(d, Z)$ decrypts the ElGamal ciphertext Z to some group element $u \in G$, and for each $a_i \in x$, and for all possible values for y_2, checks whether $g^{a_i \circ y_2} = u$. 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 [36,19,46], we can represent the statement that a given ElGamal ciphertext encrypts g^a such that a given Pedersen [49] 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 polynomials, 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 and equality of discrete logarithm representations in the random-oracle model are known [36,35,27], we can also efficiently instantiate the NIZK proof system in the random-oracle model.

One subtlety in using these random-oracle-based proof systems, however, is that in order to obtain a witness from an adversarial prover, generally such proof systems 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 a nested rewinding problem. In order to adapt these proof systems to our use, 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 some function of the witness; this gives the security reduction enough information to proceed. Although this approach is somewhat folklore, we believe our rigorous formulation and instantiation in the random-oracle model (Section 2.4) 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 he 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 [42,16,17,2], and therefore this modeling of the problem allows us to add this feature to anonymous credentials in a modular way. Moreover, our ElGama-based scheme is compatible with literature on anonymous credentials [42,16,17,2] and compact e-cash and variants [13,14,12] because Pedersen commitments are used everywhere.

Related work. Group signatures and identity escrow schemes [25,18,41,1,4] 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.

Another related line of work specifically for blueprints for watchlists is private set intersection (PSI) [34,20]. Although techniques from PSI are helpful here, in general PSI is not a solution to our problem because it's a two-party protocol where both parties are online and each participant has their list in the clear. 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 [48] allows an untrusted proxy to process streaming data using encrypted keywords such that, should any of the data match the encrypted keywords, it gets retained (in encrypted form, to be decrypted by the client), but otherwise it is not; the proxy does not learn anything about the keywords of interest to the client. The data that's either retained for the client's benefit or discarded 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.

Abuse-resistant law enforcement access systems (ARLEAS) [40] address what might be considered the encryption counterpart of our watchlist scenario. In ARLEAS, a law enforcement agency with a valid warrant to secretly place a user Alice under surveillance 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. Since ARLEAS concerns itself with encryption, while we worry about privacy-preserving authentication, our technical contributions are somewhat orthogonal.

2 Preliminaries

We recall the properties of public key encryption (PKE), non-interactive commitments and proof systems that we require for our construction.

2.1 PKE Security and ElGamal Encryption

Let KGen be an algorithm that, upon input security parameter 1^λ, outputs an ElGamal key pair $\mathsf{pk} = (g, y)$ and $\mathsf{sk} = (s)$ such that $\langle g \rangle = \langle h \rangle = \mathbb{G}$ and $g^s = y$. The encryption algorithm Enc encrypts a message $m \in \mathbb{G}$ by sampling $r \leftarrow\!\!\$\ \mathbb{Z}_q$ and computing the ciphertext $c = (g^r, my^r)$. The decryption algorithm Dec decrypts a ciphertext $c = (a, b) \in \mathbb{G}^2$ by computing ba^{-s}. We use $\mathsf{Enc}(\mathsf{pk}, m; r)$ to specify the randomness used in the encryption.

ElGamal is secure against Chosen Plaintext Attacks under the decisional Diffie-Hellman assumption. See Appendix A.1 for the definition of IND-CPA.

Let $\oplus : \mathbb{G}^2 \times \mathbb{G}^2 \rightarrow \mathbb{G}^2$ 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 composition rule on the group $\mathbb{G}$. We also write c^a as shorthand for repeated composition of c with itself a times.

2.2 Commitment Security and Pedersen Commitments

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 $D(cpar, m_0)$ are statistically close, where $D(cpar, m) = \{r \leftarrow R_{cpar} : \mathsf{Commit}_{cpar}(m; r)$; and (2) computational binding, i.e. for any probabilistic poly-time adversary $\mathcal{A}$, there exists a negligible ν such that*

$\Pr[cpar \leftarrow \mathsf{CSetup}(1^\lambda); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(cpar) : \mathsf{Commit}_{cpar}(m_0; r_0) = \mathsf{Commit}_{cpar}(m_1; r_1)] = \nu(\lambda)$

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 DL assumption in $\mathbb{G}$. We write $\mathsf{Commit}_{h_1,\ldots,h_n,g}(m_1, \ldots, m_n; r)$ to specify the randomness for the commitment.

2.3 Non-interactive Zero Knowledge

Let $\mathcal{R}$ be a polynomial-time verifiable binary relation. For 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 $\mathsf{failure}$ otherwise. V takes as input $(\mathbb{x}, \pi)$ and either outputs accept or reject.

Definition 2 (Simulation sound 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 $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $\Pr\left[\pi \leftarrow \mathsf{P}^{\mathsf{S}}(\mathbb{x}, \mathbb{w}) : \mathsf{V}^{\mathsf{S}}(\mathbb{x}, \pi) = \mathsf{reject}\right] = 0$.
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 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 Figure 2.1, the advantage function $\nu(\lambda)$ defined below is negligible:

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

```
$\mathsf{NIZK}^{\mathcal{A},0}(1^\lambda)$
  1: return $\mathcal{A}^{\mathsf{S}(\cdot),\mathsf{P}^{\mathsf{S}}(\cdot,\cdot)}(1^\lambda)$

$\mathsf{NIZK}^{\mathcal{A},1}(1^\lambda)$
  1: return $\mathcal{A}^{\mathsf{O}_\mathsf{S},\mathsf{O}_\mathsf{P}(\cdot,\cdot)}(1^\lambda)$

$\mathsf{O}_\mathsf{S}(m)$
  1: $\mathsf{state}, h, \tau_{\mathsf{Ext}} \leftarrow \mathsf{SimS}(\mathsf{state}, m)$
  2: return $h$

$\mathsf{O}_\mathsf{P}(\mathbb{x}, \mathbb{w})$
  1: if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$ : return $\mathsf{failure}$
  2: $\mathsf{state}, \pi \leftarrow \mathsf{Sim}(\mathsf{state}, \mathbb{x})$
  3: return $\pi$
```

Fig. 2.1: NIZK game

SimS shares state with Sim modeling both RO programming and CRS trapdoors. Additionally there are extraction trapdoors τ_{Ext} that are only used to define simulation extractability.

Simulation 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.

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. Φ provides simulation soundness if for any PPT adversary $\mathcal{A}$ participating in the game defined in Figure 2.2, the advantage function $\nu(\lambda)$ defined below is negligible. In Figure 2.2, $\mathcal{Q}$ denotes the query tape that records the instances $\mathbb{x}$ for which $\mathcal{A}$ has obtained proofs π via oracle access to the simulator Sim; this is explicitly recorded by $\mathsf{O}_{\mathsf{Sim}}$.

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

for some negligible function ν.

```
NISimSound^A(1^λ)
1 : $\mathcal{Q} \leftarrow [\,]$
2 : $(\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{S}}(\cdot), \mathsf{O}_{\mathsf{Sim}}(\cdot)}(1^{\lambda})$
3 : return $\mathsf{V}^{\mathsf{O}_{\mathsf{S}}}(\mathbb{x}, \pi) \wedge (\mathbb{x}, \pi) \notin \mathcal{Q} \wedge \forall \mathbb{w}.(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$

$\mathsf{O}_{\mathsf{S}}(m)$
1 : $\mathsf{state}, h, \tau_{\mathsf{Ext}} \leftarrow \mathsf{SimS}(\mathsf{state}, m)$
2 : return $h$

$\mathsf{O}_{\mathsf{Sim}}(\mathbb{x})$
1 : $\mathsf{state}, \pi \leftarrow \mathsf{Sim}(\mathsf{state}, \mathbb{x})$
2 : $\mathcal{Q}.\mathsf{add}((\mathbb{x}, \pi))$
3 : return $\pi$
```

Fig. 2.2: $\mathsf{NISimSound}$ game

Soundness and Extractability We omit a formal definition of soundness and extractability here; in order to obtain it, it is sufficient to remove oracle access to $\mathsf{O}_{\mathsf{Sim}}$ in the simulation soundness definition and the simulation extractability definitions below.

2.4 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. More formally, 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. Φ is simulation-extractable if there exists a polynomial-time extractor algorithm Ext such that for for any PPT adversary $\mathcal{A}$ participating in the game defined in Figure 2.3, the advantage function $\nu(\lambda)$ defined below is negligible.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{NISimExtract}}(\lambda) = \Pr\left[\mathsf{NISimExtract}^{\mathcal{A}}(1^\lambda) = 1\right] = \nu(\lambda)$$

In Figure 2.3 $\mathcal{Q}$ denotes the simulator query tape. $\mathcal{Q}_\mathsf{S}$ 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 discussed below, $\tilde{\mathsf{O}}_\mathsf{S}$ gives $\mathcal{A}$ additional power and allows the definition to captures adaptive extraction from many proofs.

```
$\mathsf{NISimExtract}^{\mathcal{A}}(1^\lambda)$
1 :  $\mathcal{Q}, \mathcal{Q}_\mathsf{S} \leftarrow [\,]$
2 :  $(\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\tilde{\mathsf{O}}_\mathsf{S}(\cdot), \mathsf{O}_{\mathsf{Sim}}(\cdot)}(1^\lambda)$
3 :  $\mathbb{w} \leftarrow \mathsf{Ext}(\mathcal{Q}_\mathsf{S}, \mathbb{x}, \pi)$
4 :  return $\mathsf{V}^{\mathsf{O}_\mathsf{S}}(\mathbb{x}, \pi) \wedge (\mathbb{x}, \pi) \notin \mathcal{Q} \wedge (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$

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

$\mathsf{O}_{\mathsf{Sim}}(\mathbb{x})$
1 :  $\mathsf{state}, \pi \leftarrow \mathsf{Sim}(\mathsf{state}, \mathbb{x})$
2 :  $\mathcal{Q}.\mathsf{add}((\mathbb{x}, \pi))$
3 :  return $\pi$
```

Fig. 2.3: $\mathsf{NISimExtract}$ game: τ_{Ext} is only returned by $\tilde{\mathsf{O}}_\mathsf{S}(m)$

Our definition of simulation extractability above is the one of straight-line extractability: the extractor obtains the witness just from Q_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 $\mathsf{Ext}^{\mathsf{BB}(\mathcal{A})}(Q_\mathsf{S}, \mathbb{x}, \pi)$ we denote an extractor algorithm that, in addition to its inputs, also has this access to $\mathcal{A}$. We further discuss and formally define a black-box simulation extractability game $\mathsf{NISimBBExtract}$ in Appendix A.2. 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.

Let us now formalize BB-PSL simulation extractability; as before, 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 Figure 2.4, the advantage function $\nu(\lambda)$ defined below is negligible. As 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}. In fact, the game here is almost identical to that in Figures 2.3 and A.2, except now we have two extractors.

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

for some negligible function ν.

```
$f$-$\mathsf{NISimBBPSLExtract}^{\mathcal{A}}(1^{\lambda})$
1 :  $\mathcal{Q}, \mathcal{Q}_{\mathsf{S}} \leftarrow [\,]$
2 :  $(\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\tilde{\mathsf{O}}_{\mathsf{S}}(\cdot), \mathsf{O}_{\mathsf{Sim}}(\cdot)}(1^{\lambda})$
3 :  $\mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{BB}(\mathcal{A})}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi)$
4 :  $\mathbb{w}' \leftarrow \mathsf{ExtSL}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi)$
5 :  return $\mathsf{V}^{\mathsf{O}_{\mathsf{S}}}(\mathbb{x}, \pi) \wedge (\mathbb{x}, \pi) \notin \mathcal{Q} \wedge (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \wedge \mathbb{w}' \neq f(\mathbb{w})$

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

$\mathsf{O}_{\mathsf{Sim}}(\mathbb{x})$
1 :  $\mathsf{state}, \pi \leftarrow \mathsf{Sim}(\mathsf{state}, \mathbb{x})$
2 :  $\mathcal{Q}.\mathsf{add}((\mathbb{x}, \pi))$
3 :  return $\pi$
```

Fig. 2.4: f-$\mathsf{NISimBBPSLExtract}$ game

More on the simulator and extractor. In the games NIZK, $\mathsf{NISimSound}$, $\mathsf{NISimExtract}$, $\mathsf{NISimBBExtract}$, and $\mathsf{NISimBBPSLExtract}$ the simulator initializes and updates the setup using SimS and then responds to queries from $\mathcal{A}$ for simulated proofs. 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 has only access to $\mathcal{Q}_{\mathsf{S}}$, which contains the information necessary to extract a witness from a proof produced by the adversary. In the CRS model, $\mathcal{Q}_{\mathsf{S}}$ will contain extraction trapdoor information created at the beginning, when S was first generated. 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, however, not 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 execute Ext itself, and thus allows for the extraction from multiple proofs.

Instantiating Simulation Extractable proofs. While simulation-extractable proof systems exist for all NP relations [29], there are multiple ways to realize non-interactive zero-knowledge (NIZK) proof systems more efficiently. One of which is to start with Σ-protocols and convert them into a NIZK proof in the random oracle model, e.g. using the techniques of [33,32,44]. As we will elaborate below, Σ protocols are particularly suitable for proving Group Homomorphisms such as discrete logarithm representations, see, e.g. [45], but can also efficiently prove disjunctive statements [26]. This has been used for range proofs.

Bulletproofs [11] is a practically efficient NIZK proof system for arithmetic circuits, specifically optimized for range-proofs. Recent work shows that Bulletproofs are simulation extractable [37] and can be integrated with Σ-protocols [10].

Bernhard et al. [5, 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 [32, Theorem 2] that shows that Σ-protocols compiled using Fiat-Shamir are simulation-sound and adapt the theorem of [28, 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 Section 2.7 we give a construction from Σ-protocols of a proof system Ψ for equality of discrete logarithm representation relations and prove the following theorem:

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.*

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:

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

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.

2.5 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 $\mathbb{x} = (G, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{n})$ where G is the description of a group of order q, and all the x_is and $g_{i,j}$s are elements of 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 $e_j \leftarrow Z_q$ for $1 \leq j \leq m$ and computes $d_i = \prod_{j=1}^{m} g_{i,j}^{e_j}$ for $1 \leq i \leq n$. 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 $c \leftarrow Z_q$.

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

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

Simulation On input $\mathbb{x}$ and $\mathsf{chal} = c$, the simulator chooses $s_j \leftarrow Z_q$ for $1 \leq j \leq m$, and sets $d_i = (\prod_{j=1}^{m} g_{i,j}^{s_j})/x_i^c$ for $1 \leq i \leq n$. 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 $1 \leq j \leq m$.

2.6 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 Section 2.5 via the Fiat-Shamir heuristic. Specifically, $\mathsf{S}_{\mathsf{eqrep}}$ is a random oracle.

We use a theorem of [32, Theorem 2] that shows that Σ-protocols compiled using Fiat-Shamir are simulation-sound; moreover, it follows from a theorem of [5, Theorem 1] and the proof of [32, Theorem 3] that it is in fact black-box simulation extractable.

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

2.7 g^x-BB-PLE 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 $\mathbb{x} = (G, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^n)$ 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 G included in the description of 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 G with generator g of order q, outputs a random element h of G; we can think of this h as the public key of the ElGamal cryptosystem.

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

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

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

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

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

Set $\mathbb{w}' = (w_1, \ldots, w_m, r_1, \ldots, r_k)$. Using the algorithm $\mathsf{P}^{\mathsf{S}}_{\mathsf{eqrep}}$, compute $\pi_{\mathsf{eqrep}} \leftarrow \mathsf{PoK}_{\Psi\mathsf{eqrep}}\Big\{\mathbb{w}' : R_{\mathsf{eqrep}}(\mathbb{x}', \mathbb{w}')\Big\}$, 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 $\mathsf{V}^{\mathsf{S}}_{\mathsf{eqrep}}(\mathbb{x}', \pi_{\mathsf{eqrep}})$.

We now sketch the proof of Theorem 1.

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

$\mathsf{SimS}(\mathsf{state}, m) \rightarrow \mathsf{state}, h', \tau_{\mathsf{Ext}}$: On input the description of a group G with generator g of order q, sample $\tau_{\mathsf{Ext}} \leftarrow Z_q$ and output the hash value that will be interpreted as the element $g^{\tau_{\mathsf{Ext}}}$ of 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.

$\mathsf{Sim}(\mathsf{state}, \mathbb{x}) \rightarrow \mathsf{state}, \pi$: On input $\mathbb{x}$, the simulator extends $\mathbb{x}$ with random ElGamal ciphertexts to $\mathbb{x}'$, chooses $c \leftarrow Z_y$, $s_j \leftarrow Z_q$ for $1 \leq j \leq m + |J|$, and sets $d_i = (\prod_{j=1}^{m+k} g_{i,j}^{s_j})/x_i^c$ for $1 \leq i \leq n + 2|J|$. 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 $(\mathsf{chal}, \mathsf{res})$.

$\mathsf{Ext}^{\mathsf{BB}(\mathcal{A})}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi) \rightarrow \mathbb{w}$: 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 $\mathbb{w}$.

$\mathsf{ExtSL}(\mathcal{Q}_{\mathsf{S}}, \mathbb{x}, \pi) \rightarrow f(J, \mathbb{w})$: Parse $\mathbb{x}$ as $(G, \{x_i, \{g_{i,1}, \ldots, g_{i,m}\}\}_{i=1}^{n+2|J|})$, obtain τ_{Ext} from the entry $(G, h, \tau_{\mathsf{Ext}})$ 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 warrants data 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.

Auditors create a warrant-specific public key and private key pair $(\mathsf{pk_A}, \mathsf{sk_A})$ and the user can escrow its private data under $\mathsf{pk_A}$ to obtain an escrow Z. We require, that Z decrypts (with the help of $\mathsf{sk_A}$) 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 $\mathsf{pk_A}$ and C. Similarly, a privacy-conscious user can verify that indeed $\mathsf{pk_A}$ 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]

$\mathsf{VerPK}(\Lambda, \mathsf{pk}_\mathsf{A}, C_\mathsf{A}) \to \mathsf{accept}$ or reject: 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 for commitment $C = \mathsf{Commit}(y; r)$.

$\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z) \to \mathsf{accept}$ or reject: is the algorithm that, on input the auditor's public key pk_A, a commitment C, and an 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 Z such that $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z) = \mathsf{accept}$, 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 a commitment whose opening (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 probabilistic polynomial-time adversaries $\mathcal{A}$ involved in the experiment in Figure 3.1, there exists a negligible function ν such that:

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

[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 : $\textbf{return}\ [C = \mathsf{Commit}(y; r) \wedge$
7 :       $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, C, Z) \wedge \mathsf{Decrypt}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z) \neq f(x, y)]$
```

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

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 $\mathsf{Decrypt}(\Lambda, \mathsf{sk}_{\mathsf{A}}, \cdot, \cdot)$.

We formalize this security property by requiring that there exist a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimKeygen}, \mathsf{SimDecrypt})$ such that a probabilistic polynomial-time 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 $\mathsf{Decrypt}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z)$; the "hiding" 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 Figure 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:

[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 $y^* = g(y)$.

```
$\mathsf{BHreal}^{\mathcal{A}}_{\mathsf{Blu}}(\lambda)$
1 :  $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 :  $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
3 :  $(x, r_\mathsf{A}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, \Lambda)$
4 :
5 :  $(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_\mathsf{A})$
6 :  return $\mathcal{A}^{\mathsf{O}_0(\mathsf{pk_A}, \mathsf{sk_A}, \cdot, \cdot)}(\mathsf{pk_A}, \mathsf{state}_\mathcal{A})$

$\mathsf{BHideal}^{\mathcal{A}}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$
1 :  $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 :  $(\Lambda, \mathsf{state}) \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
3 :  $(x, r_\mathsf{A}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, \Lambda)$
4 :  $dsim \leftarrow (|x|, \mathsf{Commit}(x; r_\mathsf{A}))$
5 :  $(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow \mathsf{SimKeygen}(1^\lambda, \mathsf{state}, dsim)$
6 :  return $\mathcal{A}^{\mathsf{O}_1(\mathsf{pk_A}, \mathsf{state}, x, \cdot, \cdot)}(\mathsf{pk_A}, \mathsf{state}_\mathcal{A})$

$\mathsf{O}_0(\mathsf{pk_A}, \mathsf{sk_A}, C, Z)$
1 :  if $\neg\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z)$
2 :      return $\perp$
3 :  return $\mathsf{Decrypt}(\Lambda, \mathsf{sk_A}, C, Z)$

$\mathsf{O}_1(\mathsf{pk_A}, simtrap, x, C, Z)$
1 :  if $\neg\mathsf{VerEscrow}(\Lambda, \mathsf{pk_A}, C, Z)$
2 :      return $\perp$
3 :  $y^* \leftarrow \mathsf{SimDecrypt}(\mathsf{state}, C, Z)$
4 :  return $f(x, y) = f^*(x, y^*)$
```

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

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

More formally, there exists algorithms $(\mathsf{SimSetup}, \mathsf{SimEscrow})$ such that, for any PPT adversary $\mathcal{A}$ involved in the game described in Figure 3.3, the following equation holds:

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

for some negligible function ν.
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.

```
$\mathsf{PADA}^{\mathcal{A},b}_{\mathsf{Blu},\mathsf{Ext},\mathsf{Sim}}(\lambda)$
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 : $\textbf{if } \mathsf{VerPK}(\Lambda_b, \mathsf{pk}_\mathsf{A}, \mathsf{Commit}(x; r_\mathsf{A})) = \mathsf{reject} : \textbf{return } \perp$
5 : $\textbf{return } \mathcal{A}^{\mathsf{O}_b(y,r)}(\mathsf{state}_\mathcal{A})$

$\mathsf{O}_0(y, r)$
1 : $\textbf{return } \mathsf{Escrow}(\Lambda_0, pk_\mathsf{A}, y, r)$

$\mathsf{O}_1(y, r)$
1 : $\textbf{return } \mathsf{SimEscrow}(\mathsf{state}, \Lambda_1, \mathsf{pk}_\mathsf{A}, \mathsf{Commit}(y; r),$
2 :         $f(x, y))$
```

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

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

We formalize these two games in Figure 3.4. We require that there exists an simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimEscrow})$ such that, for any PPT adversary $\mathcal{A}$ 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^λ) 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($hecpar, f, x$) 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 (garbled, obfuscated) representation X of the function $f(x, \cdot)$, and a decryption key d.

HECEVAL($hecpar, f, X, y$) 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 $f(x, y)$.

```
$\mathsf{PWHA}^{\mathcal{A},b}_{\mathsf{Blu},\mathsf{Sim}}(\lambda)$
1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda_0 \leftarrow \mathsf{Setup}(1^\lambda, cpar); \Lambda_1 \leftarrow \mathsf{SimSetup}(1^\lambda, cpar)$
3 : $M \leftarrow [\,]$
4 : $x, r_{\mathsf{A}} \leftarrow \mathcal{A}(1^\lambda, \Lambda_b)$
5 : $(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow \mathsf{KeyGen}(\Lambda_b, x, r_{\mathsf{A}})$
6 : $\textbf{return } \mathcal{A}^{\mathsf{O}^{\mathsf{Escrow}}_b(\cdot,\cdot,\cdot), \mathsf{O}^{\mathsf{Decrypt}}(\Lambda_b, \mathsf{sk_A}, \cdot, \cdot)}(\mathsf{pk_A})$

$\mathsf{O}^{\mathsf{Escrow}}_0(y, r)$
1 : $\textbf{return } \mathsf{Escrow}(\Lambda_0, \mathsf{pk_A}, y, r)$

$\mathsf{O}^{\mathsf{Escrow}}_1(y, r)$
1 : $C = \mathsf{Commit}(y; r)$
2 : $Z \leftarrow \mathsf{SimEscrow}(\mathsf{state}, \Lambda_1, \mathsf{pk_A}, C)$
3 : $M[C, Z] = f(x, y)$
4 : $\textbf{return } Z$

$\mathsf{O}^{\mathsf{Decrypt}}(\Lambda_1, \mathsf{sk_A}, C, Z)$
1 : $\textbf{if } M[C, Z] \quad \textbf{return } M[C, Z]$
2 : $\textbf{return } \mathsf{Decrypt}(\Lambda_1, \mathsf{sk_A}, C, Z)$
```

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

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

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

HEC correctness. HEC = (HECSETUP, HECENC, HECEVAL, HECDEC) is correct with adversarial evaluation randomness if for any PPTadversary $\mathcal{A}$, the following advantage function:

$$\mathsf{Adv}^{\text{HECCORRECT}}_{\mathcal{A}}(\lambda) = \Pr\big[\text{HECCORRECT}^{\mathcal{A}}(\lambda) = \mathsf{accept}\big] = \nu(\lambda)$$

holds for a negligible function ν and for experiment HECCORRECT shown in Figure 4.1.

Security of x. For any PPT algorithm $\mathcal{A}$, $|p_{\mathcal{A},0} - p_{\mathcal{A},1}| = \nu(\lambda)$ for a negligible ν, where for $b \in \{0,1\}$, $p_{\mathcal{A},b}$ is the probability that $\mathcal{A}$ outputs 0 in experiment SECX in Figure 4.1.

Security of x and y from third parties. For any PPT algorithm $\mathcal{A}$, $|p_{\mathcal{A},0} - p_{\mathcal{A},1}| = \nu(\lambda)$ for a negligible ν, where for $b \in \{0,1\}$, $p_{\mathcal{A},b}$ is the probability that $\mathcal{A}$ outputs 0 in the SECXY game in Figure 4.1.

```
HECcorrect^A(λ)
1 :  $hecpar \leftarrow \text{HECSetup}(\lambda)$
2 :  $(f, x, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^{\lambda}, hecpar)$
3 :  if $f \in F, x \in domain_{f,x}$
4 :    $(X, d) \leftarrow \text{HECEnc}(hecpar, f, x)$
5 :    $(y, r_Z) \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, X)$
6 :    if $y \in domain_{f,y}$
7 :      $Z \leftarrow \text{HECEval}(hecpar, f, X, y; r_Z)$
8 :      if $\text{HECDec}(hecpar, d, Z) \neq f(x, y)$
9 :        return accept
10 :      return reject
11 :    return reject
12 :  return reject
```

```
SecX_b^A(λ)
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 :    $X, \_ \leftarrow \text{HECEnc}(hecpar, f, x_b)$
5 :    return $\mathcal{A}(hecpar, X, \mathsf{state})$
6 :  return $\mathcal{A}(\perp, \mathsf{state})$
```

```
SecXY_b^A(λ)
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 :    $X, \_ \leftarrow \text{HECEnc}(hecpar, f, x_b)$
5 :    $(y_0, y_1, \mathsf{state}) \leftarrow \mathcal{A}(X, \mathsf{state})$
6 :    if $y_0, y_1 \in domain_{f,y}$
7 :      $Z \leftarrow \text{HECEval}(hecpar, f, X, y_b)$
8 :      return $\mathcal{A}(Z, \mathsf{state})$
9 :    return $\mathcal{A}(\perp, \mathsf{state})$
10 :  return $\mathcal{A}(\perp, \mathsf{state})$
```

```
DirectZ_b^A(λ)
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 :    $X, \_ = \text{HECEnc}(hecpar, f, x; r_X)$
5 :    $Z_0 \leftarrow \text{HECEval}(hecpar, f, X, y)$
6 :    $Z_1 \leftarrow \text{HECDirect}(hecpar, X, f(x, y))$
7 :    return $\mathcal{A}(hecpar, Z_b, \mathsf{state})$
8 :  return $\mathcal{A}(\perp, \mathsf{state})$
```

Fig. 4.1: HEC correctness and security games

Suppose that we want to form a ciphertext Z that will decrypt to a specific value z — how do we do that? 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 forming Z consists of first computing $(X, d) = \text{HECEnc}(hecpar, f, x)$ and then computing $Z = \text{HECEval}(hecpar, f, X, y)$. But in general, it may be 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. $z = \text{HECDec}(hecpar, d, Z)$.

Security of DirectZ Even when the decryption key is known. For any PPT algorithm $\mathcal{A}$, $|p_{\mathcal{A},0} - p_{\mathcal{A},1}| = \nu(\lambda)$ for a negligible ν, where for $b \in \{0, 1\}$, $p_{\mathcal{A},b}$ is the probability that $\mathcal{A}$ outputs 0 in the DirectZ experiment in Figure 4.1.

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 $\mathsf{pk_A}$ 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 Figure 5.1 uses VerPK as a subroutine in Escrow and VerEscrow. To be consistent with the syntax we add C_A to $\mathsf{pk_A}$. Similarly, we use VerEscrow in Decrypt and add $\mathsf{pk_A}$ to $\mathsf{sk_A}$.

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 of VerEscrow, blueprint hiding and binding, and privacy against dishonest auditor and with honest auditor in separate lemmas.

Correctness of VerEscrow and VerPK follows from the completeness of the NIZK proof system. Correctness of Decrypt additionally requires correctness of HEC.

Lemma 1. *If the NIZK PoKs Ψ_1 and Ψ_2 are complete, then the generic blueprint scheme satisfies correctness of* VerEscrow *and* VerPK*.*

Proof. Consider VerPK as defined in Figure 5.1. Suppose the same VerPK returns reject in the experiment in Figure B.1. Then either $(C_\mathsf{A} \neq C'_\mathsf{A})$ or $\mathsf{V}_1^{\mathsf{S}_1}(\mathsf{pk_A}) =$ reject. Since $C_\mathsf{A} = C'_\mathsf{A} = \mathsf{Commit}_{cpar}(y; r)$, the later must be true. However, this contradicts completeness of the NIZK scheme because the proof π_A in $\mathsf{pk_A}$ is generated by KeyGen on a valid statement and witness pair.

Similarly, consider VerEscrow as defined in Figure 5.1. We know that VerPK returns accept, so VerEscrow only returns reject if $\mathsf{V}_2^{\mathsf{S}_2}(Z, hecpar, f, X, C, cpar) =$ reject. However, similar to in the case of VerPK, Z was generated using $\mathsf{P}_2^{\mathsf{S}_2}$ on a valid statement and witness pair. This again contradicts completeness of the NIZK PoK schemes.

```
$\mathsf{Setup}(\lambda, cpar)$
$hecpar \leftarrow \mathrm{HECSETUP}(1^\lambda)$
$\textbf{return } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2)$

$\mathsf{KeyGen}(\Lambda, x, r_\mathsf{A})$
$\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2)$
$(X, d) \overset{r_X}{\leftarrow} \mathrm{HECENC}(hecpar, f, x)$
$C_\mathsf{A} = \mathsf{Commit}_{cpar}(x; r_\mathsf{A})$
$\pi_\mathsf{A} \leftarrow \mathsf{PoK}^{\mathsf{S}_1}_{\Psi_1}\Big\{(x, d, r_X, r_\mathsf{A}) :$
    $(X, d) = \mathrm{HECENC}(hecpar, f, x; r_X)$
    $\wedge\, C_\mathsf{A} = \mathsf{Commit}_{cpar}(x; r_\mathsf{A}))\Big\}$
$\mathsf{pk}_\mathsf{A} \leftarrow (X, C_\mathsf{A}, \pi_\mathsf{A}); \mathsf{sk}_\mathsf{A} \leftarrow (\mathsf{pk}_\mathsf{A}, d)$
$\textbf{return } (\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$

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

$\mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, y, r)$
$\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2)$
$\textbf{parse } \mathsf{pk}_\mathsf{A} = (X, C_\mathsf{A}, \_)$
$\textbf{if } \mathsf{VerPK}(\Lambda, \mathsf{pk}_\mathsf{A}, C_\mathsf{A}) = \mathsf{reject}$
    $\textbf{return } \mathsf{reject}$
$\hat{Z} \overset{r_{\hat{Z}}}{\leftarrow} \mathrm{HECEVAL}(hecpar, f, X, y)$
$C = \mathsf{Commit}_{cpar}(y; r)$
$\pi_\mathsf{U} \leftarrow \mathsf{PoK}^{\mathsf{S}_2}_{\Psi_2}\Big\{(y, r, r_{\hat{Z}}) :$
    $\hat{Z} = \mathrm{HECEVAL}(hecpar, f, X, y; r_{\hat{Z}})$
    $\wedge\, C = \mathsf{Commit}_{cpar}(y; r)\Big\}$
$\textbf{return } (\hat{Z}, \pi_\mathsf{U})$

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

$\mathsf{Decrypt}(\Lambda, \mathsf{sk}_\mathsf{A}, C, Z = (\hat{Z}, \pi_\mathsf{U}))$
$\textbf{parse } \Lambda = (\lambda, cpar, hecpar, \mathsf{S}_1, \mathsf{S}_2)$
$\textbf{parse } \mathsf{sk}_\mathsf{A} = (\mathsf{pk}_\mathsf{A}, d)$
$\textbf{if } \mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z) = \mathsf{reject}$
    $\textbf{return } \mathsf{reject}$
$\textbf{return } \mathrm{HECDEC}(hecpar, d, \hat{Z})$
```

Fig. 5.1: Construction of generic f-blueprint scheme

Lemma 2. *If the NIZK PoKs Ψ_1 and Ψ_2 are complete and the HEC is correct, then the generic blueprint scheme satisfies correctness of* $\mathsf{Decrypt}$.

Proof. Consider Figure B.2. By Lemma 1, we get that Escrow and in extension $\mathsf{Decrypt}$ will not return reject, as that requires $\mathsf{VerEscrow}$ and VerPK to reject on correct inputs. This tells us that any parts of $\mathsf{DecCorrect}$ that relates to the NIZK does not affect the output of the algorithm. Omitting these lines results in:

```
DecCorrect′(λ, x, r_A, y, r)
1 :  $hecpar \leftarrow \mathrm{HECSETUP}(1^\lambda)$
2 :  $(X, d) \overset{r_X}{\leftarrow} \mathrm{HECENC}(hecpar, f, x)$
3 :  $\hat{Z} \overset{r_{\hat{Z}}}{\leftarrow} \mathrm{HECEVAL}(hecpar, f, X, y)$
4 :  $m \leftarrow \mathrm{HECDEC}(hecpar, d, \hat{Z})$
5 :  return $[m = f(x, y)]$
```

Fig. 5.2: Experiment $\mathsf{DecCorrect}_{\mathsf{Blu}}(\lambda, x, r_{\mathsf{A}}, y, r)$ with all NIZK parts removed

which is the definition for correctness of our HEC scheme for adversaries that hardcode f, x, y and sample $r_{\hat{Z}}$ at random. Assuming HEC is correct, $\mathsf{DecCorrect}'$ returns 1 with overwhelming probability, and so does $\mathsf{DecCorrect}$.

Lemma 3. *Let Ψ_2 be a BB extractable NIZK scheme, let* $(\mathsf{CSetup}, \mathsf{Commit})$ *be a computationally binding commitment scheme, and* HEC *be correct with adversarial evaluation randomness, then our proposed scheme achieves Soundness.*

Proof. Consider Figure 3.1. Suppose, for the sake of contradiction, that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Sound}}_{\mathcal{A},\mathsf{Blu}}(\lambda) = \nu(\lambda)$ is non negligible. Let Z, one of the adversary's output in the experiment, be divided into $\hat{Z}$ and a proof π to validate $\hat{Z}$.

The events where $\mathcal{A}$ outputs 1 can be divided into three: (i) when $C = \mathsf{Commit}(y; r)$, $C = \mathsf{Commit}(y'; r')$ and $\hat{Z} = \mathrm{HECEVAL}(hecpar, f, X, y'; r_{\hat{Z}})$ for $y \neq y'$, (ii) when $C = \mathsf{Commit}(y; r)$ and $\hat{Z} = \mathrm{HECEVAL}(hecpar, f, X, y; r_{\hat{Z}})$ for some $r_{\hat{Z}}$ where in both (i) and (ii) X is a part of pk_{A}, and (iii) the case where neither of these equalities holds. Let the probability of these events be expressed with functions $\nu_0(\lambda)$, $\nu_1(\lambda)$, and $\nu_2(\lambda)$ respectively. Since $\nu(\lambda)$ is non negligible and these three events covers all cases where $\mathcal{A}$ would output 1, at least one of $\nu_0(\lambda)$, $\nu_1(\lambda)$, or $\nu_2(\lambda)$ must be non negligible.

Suppose $\nu_2(\lambda)$ is non negligible. The adversary produced a proof of a false statement and we can construct a reduction $\mathcal{B}$ to the BB extractable NIZK system. $\mathcal{B}$ runs $\mathcal{A}$ the same way as Sound, but outputs $(\hat{Z}, hecpar, f, X, C, cpar), \pi_{\mathsf{U}})$

instead. By BB extractability of the NIZK, $\Pr[\mathcal{B}$ wins$]$ of extraction failure is negligible, which contradicts our assumption that $\nu_2(\lambda)$ is non negligible.

We now assume that the BB extractor extracts a witness $(y', r', r_{\hat{Z}})$, such that $\hat{Z} = \text{HEC}\textsc{Eval}(hecpar, f, X, y'; r_{\hat{Z}})$ and $C = \mathsf{Commit}_{cpar}(y'; r')$.

Suppose $\nu_0(\lambda)$ is non negligible. In this event, we break the computational binding property using a reduction that outputs (y, r, y', r').

Suppose $\nu_1(\lambda)$ is non negligible. In this event, we get a situation where both $\mathsf{pk_A}$ and Z were generated correctly with adversarial randomness $r_{\hat{Z}}$, but the output of decrypt is incorrect. We can construct a reduction $\mathcal{B}$ using $\mathcal{A}$ to HEC correctness with adversarial evaluation randomness. $\mathcal{B}$ runs $\mathcal{A}$, in the same way as Sound, but instead of returning a bit at the end, it outputs the tuple $(y, r_{\hat{Z}})$.

Lemma 4. *Let Ψ_1 be a NIZK with simulation algorithms* $(\mathsf{SimS}, \mathsf{Sim})$ *and Ψ_2 be a g^*-BB-PSL extractable NIZK with extraction algorithms* $(\mathsf{Ext}, \mathsf{ExtSL})$ *along with an efficiently computable f^* such that $f^*(x, g^*(y)) = f(x, y)$. Let HEC be correct with adversarial evaluation randomness and satisfy security of x. Then our proposed scheme achieves Blueprint Hiding.*

Proof. Consider the game in Figure 3.2. We shall define $\mathsf{SimSetup}$, $\mathsf{SimKeygen}$, and $\mathsf{SimDecrypt}$ as follows:

- $\mathsf{SimSetup}$ runs $\text{HEC}\textsc{Setup}$, but also sets up state_i to contain the state and query list $\mathcal{Q}_{\mathsf{S}_i}$ of the simulated PoK setup $\mathsf{O}_{\mathsf{S}_i}$ available to $\mathcal{A}$, P_i, and V_i, for $i \in \{1, 2\}$. The state_1 will additionally be programmed by Sim while $\mathcal{Q}_{\mathsf{S}_2}$ will be used for extraction.
- $\mathsf{SimKeygen}$ first samples $x' \leftarrow domain_{f,x}$ with the same size as x and computes $X \leftarrow \text{HEC}\textsc{Enc}(hecpar, f, x')$. Then it runs $\mathsf{Sim}(\mathsf{state}_1, (X, \mathsf{Commit}(x; r_\mathsf{A})))$ to produce a simulated proof π_A that X is generated honestly and corresponds to $\mathsf{Commit}(x; r_\mathsf{A})$. Finally, it produces $\mathsf{pk_A}$ that contains Λ, X, $C_\mathsf{A} = \mathsf{Commit}(x; r_\mathsf{A})$, and the simulated proof.
- $\mathsf{SimDecrypt}$ splits Z into $\hat{Z}$ and the proof π_U, then runs $\mathsf{ExtSL}(Q_{\mathsf{S}_2}, (\hat{Z}, C), \pi_\mathsf{U})$ to obtain y^*.

Consider the following games:

- **Game 0**: The left blueprint hiding game BHreal as described in Figure 3.2.
- **Game 1**: In this game, we replace Setup with $\mathsf{SimSetup}$, and run a modified version of KeyGen that generates the proof π_A using Sim instead.
- **Game 2**: Functions identically to the last game, but with $\mathsf{Decrypt}$ swapped with $\mathsf{SimDecrypt}$ that outputs y^* and returns $f^*(x, y^*)$.
- **Game 3**: We replace our modified KeyGen with $\mathsf{SimKeygen}$, that is we replace x with a a randomly sampled x'. This game is identical to the game $\mathsf{BHideal}$.

Let $\mathcal{A}$ be a fixed probabilistic polynomial time adversary. We denote the interaction with the adversary $\mathcal{A}$ in Game i as $\mathsf{BH}_i^{\mathcal{A}}$. We claim that

$$\left|\Pr\left[\mathsf{BH}_j^{\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{BH}_{j+1}^{\mathcal{A}}(\lambda) = 0\right]\right|$$

is negligible with respect to λ for $j \in [0, 2]$.

For $j = 0$, we can show this via a reduction to the Zero-Knowledge property of the NIZK PoK system. We can construct a reduction $\mathcal{B}$ to the NIZK game described in Figure 2.1. $\mathcal{B}$ receives two oracles from its challenger, denoted $(\mathsf{S}', \mathsf{P}')$ and uses S' to answer queries to the S_1 oracle, and P' to generate the proof in the modified KeyGen. Otherwise, $\mathcal{B}$ interacts with $\mathcal{A}$ as if it is $\mathcal{A}$'s challenger in the blueprint security games, and output whatever $\mathcal{A}$ outputs. Notice that, when in the experiment $\mathsf{NIZK}^{\mathcal{B},0}$, $\mathcal{B}$ functions identically to $\mathcal{A}$'s challenger in Game 0. Similarly, when in experiment $\mathsf{NIZK}^{\mathcal{B},1}$, $\mathcal{B}$ acts identically as $\mathcal{A}$'s challenger in Game 1. Therefore, we get that:

$$\begin{aligned}&\left|\Pr\left[\mathsf{BH}_0^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{BH}_1^{\mathcal{A}}(1^\lambda) = 0\right]\right| \\ &= \left|\Pr\left[\mathsf{NIZK}^{\mathcal{B},0}(1^\lambda) = 0\right] - \Pr\left[\mathsf{NIZK}^{\mathcal{B},1}(1^\lambda) = 0\right]\right| = \nu(\lambda)\end{aligned}$$

for negligible function ν, as required.

For $j = 1$, let us propose a Game 1.5 that is identical to Game 1, but instead of running only $\mathsf{Decrypt}$ or $\mathsf{SimDecrypt}$, it runs both. When $\mathsf{Decrypt}$ and $\mathsf{SimDecrypt}$ agrees, the decryption oracle returns their output. Otherwise, the experiment is halted and $\emptyset$ is returned. We claim that this event occurs with negligible probability.

Let $\nu(\lambda) = \Pr\left[\mathsf{BH}_{1.5}^{\mathcal{A}}(1^\lambda) = \emptyset\right]$. We can separate this into the event E_0 with probability $\nu_0(\lambda)$ that captures the event where the output is $\emptyset$ and extraction fails (i.e. when the output of ExtSL used in $\mathsf{SimDecrypt}$ does returns a y^* for which there does not exists valid witness y such that $g^*(y) \neq y^*$ for the relation in the proof in π_{A}), and event E_1 with probability $\nu_1(\lambda)$ that captures all other cases where the experiment would output $\emptyset$.

$\nu_0(\lambda)$ must be negligible, otherwise we can construct a reduction $\mathcal{B}$ to the g^*-BB-PSL extractability property of the NIZK described in Figure 2.4. $\mathcal{B}$ receives two oracles $\tilde{\mathsf{O}}_{\mathsf{S}}$ and $\mathsf{O}_{\mathsf{Sim}}$ and uses $\tilde{\mathsf{O}}_{\mathsf{S}}$ to answer $\mathcal{A}$'s queries to the $\mathsf{O}_{\mathsf{S}_2}$ oracle. Otherwise, $\mathcal{B}$ acts identically as $\mathcal{A}$'s challenger in Game 1.5, except that it outputs the statement $\hat{Z}$ and proof π_{A} when of the first decryption query for which $\mathsf{Decrypt}$ and $\mathsf{SimDecrypt}$ disagrees.

If the event E_0 occurs, then $\mathcal{B}$ succeeds in its reduction. However, by g^*-BB-PSL simulation extractability, we know that $\mathcal{B}$ succeeds with negligible probability. This gives us:

$$\Pr[E_1] = \nu_0(\lambda) \leq \Pr\left[\mathsf{NISimBBPSLExtract}^{\mathcal{B}}(1^\lambda) = 1\right] = \mu(\lambda)$$

for some negligible μ.

Assume the BB extractor succeeds in extracting a witness $(y, r, r_{\hat{Z}})$, such that $\hat{Z} = \text{HEC}\textsc{Eval}(hecpar, f, X, y; r_{\hat{Z}})$ and $C = \mathsf{Commit}_{cpar}(y; r)$.

$\nu_1(\lambda)$ must also be negligible, otherwise we get a situation where both pk_{A} and Z were generated correctly with adversarial randomness $r_{\hat{Z}}$, but the output of decrypt is incorrect. We can construct a reduction $\mathcal{B}$ using $\mathcal{A}$ to HEC correctness with adversarial evaluation randomness. $\mathcal{B}$ first outputs f, x that it receives from $\mathcal{A}$, and uses the X, d that it receives from its challenger to simulate KeyGen. Then it behaves in the same way as $\mathsf{BH}_{1.5}$, but instead of answering an invalid decryption oracle, it outputs the tuple $(y, r_{\hat{Z}})$.

For $j = 2$, we can construct a reduction $\mathcal{B}$ to the HEC Security of x game in Figure 4.1. $\mathcal{B}$ functions identically to the challenger in Game 2, but instead of running KeyGen, it generates the public key pk_A using by first sampling $x' \leftarrow\!\!\$\ domain_{f,x}$. It then outputs (f, x, x') to its challenger and receives X back. $\mathcal{B}$ then obtains the simulated π_A by running $\mathsf{Sim}(X)$. At the end, it outputs what the adversary outputs.

We see that $\mathcal{B}$ acts identically to $\mathcal{A}$'s challenger in Game 2 if its challenger chose $b = 0$, and identically to $\mathcal{A}$'s challenger in Game 3 otherwise. Since the encryption scheme is HEC secure, we get that $\mathcal{B}$ has negligible distinguishing advantage, and so does $\mathcal{A}$.

Lemma 5. *Let* $(\mathsf{CSetup}, \mathsf{Commit})$ *be a computationally binding non-interactive commitment scheme. Let* Ψ_1 *be a BB extractable NIZK with extraction algorithms* Ext*, and* Ψ_2 *be a NIZK with simulation algorithms* $(\mathsf{SimS}, \mathsf{Sim})$*. Let HEC satisfy security of* HECdirect*. Then our proposed construction achieves privacy against dishonest auditor.*

Proof. We shall construct $\mathsf{SimSetup}$ and Sim for the PADA game as follows:

1. $\mathsf{SimSetup}(cpar, \mathsf{state})$ runs HECsetup, but also sets up state_i to contain the state and query list $\mathcal{Q}_{\mathsf{S}_i}$ of the simulated PoK setup $\mathsf{O}_{\mathsf{S}_i}$ available to $\mathcal{A}$, P_i, and V_i, for $i \in \{1, 2\}$. The state_2 will additionally be programmed by Sim while $\mathcal{Q}_{\mathsf{S}_1}$ will be used for extraction.
2. $\mathsf{Sim}(\mathsf{state}, \Lambda_1, \mathsf{pk}_\mathsf{A}, \mathsf{Commit}(y; r), f(x, y))$ first runs $\hat{Z} \leftarrow$ HECdirect$(hecpar, X, f(x, y))$. Afterwards, it creates π_U using the ZK simulator. Finally it outputs $Z = (\hat{Z}, \pi_\mathsf{U})$.

Consider the following games:

1. **Game 0**: The privacy against dishonest auditor game defined in Figure 3.3 where $b = 0$.
2. **Game 1**: In this game, we change from using Setup to using $\mathsf{SimSetup}$. Additionally, in the Escrow function, instead of using P_2 to generate π_U, we use Sim.
3. **Game 2**: Instead of using the modified Escrow procedure, we use $\mathsf{SimEscrow}$. This is effectively the game defined in Figure 3.3 with $b = 1$.

Let $\mathcal{A}$ be a fixed PPT adversary, and let $\mathsf{PADA}_i^\mathcal{A}$ denote $\mathcal{A}$'s interaction in Game i. We claim that $\left|\Pr\left[\mathsf{PADA}_0^\mathcal{A}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PADA}_2^\mathcal{A}(1^\lambda) = 0\right]\right| = \nu(\lambda)$ for some negligible function ν.

First, we show that $\left|\Pr\left[\mathsf{PADA}_0^\mathcal{A}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PADA}_1^\mathcal{A}(1^\lambda) = 0\right]\right|$ is negligible by a reduction $\mathcal{B}$ to the NIZK zero-knowledge game in Figure 2.1. $\mathcal{B}$ functions as follows:

1. $\mathcal{B}$ obtains two oracles $(\mathsf{S}', \mathsf{P}')$ from its challenger in the NIZK game. It then functions as $\mathcal{A}$'s adversary in the PADA game, with minor tweaks.
2. In setup, $\mathcal{B}$ runs $\mathsf{SimSetup}$ but assigns $\mathsf{S}_2 \leftarrow \mathsf{S}'$.
3. When its escrow oracle is queried, it generates $\hat{Z}$ using HECeval similar to Escrow, but obtains π_U using P' with input $\hat{Z}$.

Now $\mathcal{B}$ functions identically to $\mathcal{A}$'s challenger in Game b for b defined in the NIZK game. We know that NIZK_2 is zero-knowledge, therefore $\mathcal{B}$'s advantage is negligible and we get:

$$\left|\Pr\left[\mathsf{PADA}_0^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PADA}_1^{\mathcal{A}}(1^\lambda) = 0\right]\right| =$$
$$\left|\Pr\left[\mathsf{NIZK}^{\mathcal{B},0}(1^\lambda) = 0\right] - \Pr\left[\mathsf{NIZK}^{\mathcal{B},1}(1^\lambda) = 0\right]\right| = \nu(\lambda)$$

for some negligible function ν, as desired.

Further, we have to show that $\left|\Pr\left[\mathsf{PADA}_1^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PADA}_2^{\mathcal{A}}(1^\lambda) = 0\right]\right|$ is negligible.

The event where $\mathcal{A}$ outputs $x, r_\mathsf{A}, \mathsf{pk}_\mathsf{A}$ can be divided into three: (i) when $C_\mathsf{A} = \mathsf{Commit}(x; r_\mathsf{A})$, $C_\mathsf{A} = \mathsf{Commit}(x'; r'_\mathsf{A})$ and $X, \_ = \text{HECEnc}(\mathit{hecpar}, f, x'; r_X)$ for $x \neq x'$, (ii) when $C_\mathsf{A} = \mathsf{Commit}(x; r_\mathsf{A})$ and $X, \_ = \text{HECEnc}(\mathit{hecpar}, f, x; r_X)$ for some r_x where in both (i) and (ii) X is a part of pk_A, and (iii) the case where neither of these equalities holds.

Let the probability of these events be expressed with functions $\nu_0(\lambda)$, $\nu_1(\lambda)$, and $\nu_2(\lambda)$ respectively. Since $\nu(\lambda)$ is non negligible and these three events covers all cases of $\mathcal{A}$ outputting these values, at least one of $\nu_0(\lambda)$, $\nu_1(\lambda)$, or $\nu_2(\lambda)$ must be non negligible.

Suppose $\nu_2(\lambda)$ is non negligible. The adversary produced a proof of a false statement and we can construct a reduction $\mathcal{B}$ to the BB extractable NIZK system. $\mathcal{B}$ runs $\mathcal{A}$, but outputs $(X, \mathit{hecpar}, C_\mathsf{A}, \mathit{cpar}), \pi_\mathsf{A})$ instead of continuing the game. By BB extractability of the NIZK, $\Pr[\mathcal{B}$ wins$]$ is negligible, which contradicts our assumption that $\nu_2(\lambda)$ is non negligible.

Assume the BB extractor succeeds in extracting a witness $(x', d, r_X, r'_\mathsf{A})$, such that $C_\mathsf{A} = \mathsf{Commit}(x'; r'_\mathsf{A})$ and $X, d = \text{HECEnc}(\mathit{hecpar}, f, x'; r_X)$.

Suppose $\nu_0(\lambda)$ is non negligible. In this event, we break the computational binding property of the commitment scheme using a reduction that outputs $(x, r_\mathsf{A}, x', r'_\mathsf{A})$.

Suppose that it is only $\nu_1(\lambda)$ that is non negligible. Further, suppose $\mathcal{A}$ is allowed $\ell(\lambda)$ queries to the encryption oracle where ℓ is a polynomial. We use a hybrid reduction $\mathcal{B}$ to the security of HECdirect that shows that

$$\left|\Pr\left[\mathsf{PADA}_1^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PADA}_2^{\mathcal{A}}(1^\lambda) = 0\right]\right|$$

is negligible. In hybrid i, the first i queries the adversary makes to the oracle are answered as it would in PADA_1, and the rest are answered as it would be in PADA_2.

The reduction functions as follows:

1. $\mathcal{B}$ obtains *hecpar* from its challenger, It then uses it in the generation of Λ in $\mathsf{SimSetup}$.
2. When given $(x, r_{\mathsf{A}}, \mathsf{pk_A})$ by $\mathcal{A}$, it runs the BB extractor to obtain (x, r_X)
3. It samples $j \leftarrow\!\!\$\ \{0, 1, \ldots, \ell(\lambda) - 1\}$.
4. For the first j queries that $\mathcal{A}$ makes, $\mathcal{B}$ answers as it would in PADA_2.
5. For query j, $\mathcal{B}$ sends (f, x, r_X, y) to its challenger and obtains $\hat{Z}$ and runs $\mathsf{Sim}(\hat{Z})$ for π_{U}.
6. For the rest of the queries, $\mathcal{B}$ answers as it would in PADA_1.

If we are in DirectZ_0 and extraction succeeds, then the adversary's view is identical to hybrid j. Similarly, the adversary's view is identical to hybrid $j+1$ if $\mathcal{B}$ is in DirectZ_1. Note that hybrid 0 is PADA_1 and hybrid $\ell(\lambda)$ is PADA_2. By the hybrid argument, we get that:

$$\begin{aligned}\nu_1(\lambda) &= \ell(\lambda)\big|\Pr\big[\text{DirectZ}^{\mathcal{B},0}(1^\lambda) = 0\big] - \Pr\big[\text{DirectZ}^{\mathcal{B},1}(1^\lambda) = 0\big]\big| \\ &= \ell(\lambda)\nu(\lambda)\end{aligned}$$

for some negligible function ν, since HECDirect is secure. This is still negligible.

Lemma 6. *Let $\Psi_1 = (\mathsf{S}_1, \mathsf{P}_1, \mathsf{V}_1)$ be a NIZK proof system and $\Psi_2 = (\mathsf{S}_2, \mathsf{P}_2, \mathsf{V}_2)$ be a g^*-BB-PSL simulation extractable NIZK along with an efficiently computable f^* such that $f^*(x, g^*(y)) = f(x, y)$. Let* HEC *be correct with adversarial evaluation randomness and satisfy secure of x and y from third parties. Then our proposed construction achieves privacy with honest auditor.*

Lemma 6. *Let $\Psi_1 = (\mathsf{S}_1, \mathsf{P}_1, \mathsf{V}_1)$ be a NIZK proof system and $\Psi_2 = (\mathsf{S}_2, \mathsf{P}_2, \mathsf{V}_2)$ be a g^*-BB-PSL simulation extractable NIZK along with an efficiently computable f^* such that $f^*(x, g^*(y)) = f(x, y)$. Let* HEC *be correct with adversarial evaluation randomness and satisfy secure of x and y from third parties. Then our proposed construction achieves privacy with honest auditor.*

Proof. Let $(\mathsf{SimS}_1, \mathsf{Sim}_1)$ be the setup and simulator functions that satisfies soundness and zero-knowledge for Ψ_1, and $(\mathsf{SimS}_2, \mathsf{Sim}_2)$ be the setup and simulator functions along with extractor pairs $(\mathsf{Ext}_2, \mathsf{ExtSL}_2)$ that satisfies black-box with partial straight line simulation extractability for g^* for Ψ_2 where ExtSL_2 is the straightline extractor for g^*. Recall that f^* is an efficiently computable function such that $f^*(x, g^*(y)) = f(x, y)$.

We shall construct $\mathsf{SimSetup}$ and Sim as follows:

1. $\mathsf{SimSetup}(cpar, \mathsf{state})$ runs HECSetup, but also sets up state_i to contain the state and query list $\mathcal{Q}_{\mathsf{S}_i}$ of the simulated PoK setup $\mathsf{O}_{\mathsf{S}_i}$ available to $\mathcal{A}$, P_i, and V_i, for $i \in \{1, 2\}$. The state_i will additionally be programmed by Sim_i while $\mathcal{Q}_{\mathsf{S}_2}$ will be used for extraction.
2. $\mathsf{SimEscrow}(\mathsf{state}, \Lambda_1, \mathsf{pk_A}, \mathsf{Commit}(y; r))$ first samples $y \leftarrow\!\!\$\ domain_{f,y}$ and obtains $\hat{Z} \leftarrow \text{HECEval}(hecpar, f, X, y)$. Afterwards, it runs Sim_2 to generate π_{U}. Finally it outputs $Z = (\hat{Z}, \pi_{\mathsf{U}})$.

Consider the following games:

- **Game 0**: The privacy with honest auditor game defined in Figure 3.4 where $b = 0$.
- **Game 1**: Here, we change $\mathsf{O}_0^{\mathsf{Escrow}}(y, r)$ to do the following:
  1. First, it runs $Z \leftarrow \mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, y, r)$
  2. Then, it sets $M[\mathbf{C}, Z] = f(x, y)$ and returns Z

  Essentially, this caches every $\mathsf{O}^{\mathsf{Escrow}}$ result for decryption.
- **Game 2**: In this game, we exchange Setup with $\mathsf{SimSetup}$. Then, we change oracle $\mathsf{O}_0^{\mathsf{Escrow}}$ of the game to run an algorithm identical to Escrow, but instead of generating π_U with P_2, it generates it by running Sim_2.
- **Game 3**: We change $\mathsf{O}^{\mathsf{Decrypt}}$ to an alternate algorithm $\mathsf{O}'^{\mathsf{Decrypt}}$ similar to O_1 in the blueprint hiding game described in Figure 3.2:
  1. First check if $M[C, Z]$ exists, and if so return $M[C, Z]$. This handles the simulated escrows.
  2. Then check if $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, \mathbf{C}, Z) = \mathsf{reject}$, and if so return $\perp$.
  3. Then it obtains α using ExtSL
  4. Finally, it returns $f^*(x, \alpha)$.
- **Game 4**: In this game, we use Sim_1 to simulate the proof π_A. Everything else stays the same.
- **Game 5**: In this game we change to using $\mathsf{O}_1^{\mathsf{Escrow}}$.
- **Game 6**: Now, we change to using P_1 to generate the proof π_A.
- **Game 7**: Now, we change back to using $\mathsf{O}^{\mathsf{Decrypt}}$. Notice that this is identical to the privacy with honest auditor game defined in Figure 3.4 where $b = 1$.

Let $\mathcal{A}$ be a fixed PPT adversary, and let $\mathsf{PWHA}_i^{\mathcal{A}}$ denote $\mathcal{A}$'s interaction in Game i. We claim that $\left|\Pr\left[\mathsf{PWHA}_0^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PWHA}_2^{\mathcal{A}}(1^\lambda) = 0\right]\right| = \nu(\lambda)$ for some negligible function ν.

For Game 0 and Game 1, we claim that the output of $\mathsf{O}^{\mathsf{Decrypt}}$ in both games agree with overwhelming probability, which can be shown via a reduction to soundness of the construction defined in Figure 3.1. Consider a Game 0.5 where we use the following decrypt oracle:

1. Runs $g \leftarrow \mathsf{Decrypt}(\Lambda, \mathsf{sk}_\mathsf{A}, \mathbf{C}, Z)$
2. Check if $M[\mathbf{C}, \mathsf{Escrow}]$, and if $g \neq M[\mathbf{C}, \mathsf{Escrow}]$ then halts and return $\emptyset$.
3. Otherwise, return g.

Additionally, instead of only recording $f(x, y)$ in M, it also records y, r but does not use this in any computation and $M[\mathbf{C}, Z]$ should only return $f(x, y)$. We see that if Game 0.5 does not end with $\emptyset$, then its behavior is identical to Game 0 or Game 1. We claim that the probability $\Pr\left[\mathsf{PWHA}_{0.5}^{\mathcal{A}}(1^\lambda) = \emptyset\right]$ is negligible by a reduction $\mathcal{B}$ to the the soundness game in Figure 3.1. $\mathcal{B}$ functions identically to $\mathcal{A}$'s challenger in Game 0.5 with the following modifications:

1. Upon receiving Λ from its challenger, it passes that to $\mathcal{A}$.
2. When receiving (x, r_A) from $\mathcal{A}$, it passes it to the challenger.

3. When about to return $\emptyset$, instead give its challenger the values $(\mathbf{C}, y, r, Z)$ that was used to call $\mathsf{O}^{\mathsf{Decrypt}}$.

We see that $\mathcal{A}$'s view of this game is identical to that of Game 0.5. Additionally, $\mathcal{B}$ succeeds in the reduction (produce an escrow where verification passes but decrypts to an incorrect value) if it would have outputted $\emptyset$ in Game 0.5. However, we know that $\mathcal{B}$'s probability of success must be negligible by Lemma 3, and so does $\Pr\left[\mathsf{PWHA}_{0.5}^{\mathcal{A}}(1^\lambda) = \emptyset\right]$.

For Game 1 and Game 2, we can construct a reduction $\mathcal{B}$ to the zero knowledge game for NIZK_2 as described in Figure 2.1. $\mathcal{B}$ acts as the challenger in Game 1 except the following modifications.

1. $\mathcal{B}$ receives $(\mathsf{S}', \mathsf{P}')$ from its challenger and in S, it assigns $\mathsf{S}_2 \leftarrow \mathsf{S}'$.
2. When performing Escrow in $\mathsf{O}^{\mathsf{Escrow}}$, instead of computing π_{U} with P_2, $\mathcal{B}$ runs $\pi_{\mathsf{U}} \leftarrow \mathsf{P}'((\mathsf{pk}_{\mathsf{A}}, \hat{Z}, \mathsf{Commit}(y; r), \Lambda), (y, r, r'))$ where r' is the value used to compute $\hat{Z}$.

If $\mathcal{B}$ is interacting in $\mathsf{NIZK}^{\mathcal{A},0}$, then $\mathcal{A}$ is interacting with its challenger in Game 1. Similarly, when $\mathcal{B}$ is interacting in $\mathsf{NIZK}^{\mathcal{A},1}$, then $\mathcal{A}$ is interacting with its challenger in Game 2. Note that, although the challenger in Game 2 simply runs Sim_2, the challenger for $\mathcal{B}$ will always pass the if check in O_{P} because we always provide it a valid witness-statement pair. We get that:

$$\left|\Pr\left[\mathsf{PWHA}_1^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PWHA}_2^{\mathcal{A}}(1^\lambda) = 0\right]\right| =$$
$$\left|\Pr\left[\mathsf{NIZK}^{\mathcal{B},0}(1^\lambda) = 0\right] - \Pr\left[\mathsf{NIZK}^{\mathcal{B},1}(1^\lambda) = 0\right]\right| = \nu(\lambda)$$

for some negligible function ν as required, since NIZK_2 is zero-knowledge.

For Game 2 and Game 3, we claim that $\mathcal{A}$ cannot cause the output of $\mathsf{O}^{\mathsf{Decrypt}}$ and $\mathsf{O}'^{\mathsf{Decrypt}}$ to disagree with non negligible probability. Consider a Game 2.5 where the challenger instead of running just $\mathsf{O}^{\mathsf{Decrypt}}$ or $\mathsf{O}'^{\mathsf{Decrypt}}$, it runs both and outputs the answer if both agrees, or ends the game and output $\emptyset$ otherwise. We claim that $\Pr\left[\mathsf{PWHA}_{2.5}^{\mathcal{A}}(1^\lambda) = \emptyset\right]$ is negligible.

For the sake of simplicity, we assume that all queries $\mathcal{A}$ made to the decryption oracle contains $(\mathbf{C}, Z)$ such that $\mathsf{VerEscrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, \mathbf{C}, Z) = \mathsf{accept}$. This is something that $\mathcal{A}$ itself can check, and if a query is made that does not fit this description, then the O will return with $\perp$ as both $\mathsf{O}^{\mathsf{Decrypt}}$ and $\mathsf{O}'^{\mathsf{Decrypt}}$ runs this verification and return $\perp$ when it fails. Consider two cases where $\emptyset$ is the output: when extraction succeeds and when it fails.

We know by simulation extractability of Ψ_2 that the later occurs with negligible probability. Otherwise, we can construct a reduction $\mathcal{B}$ to black-box partial straightline simulation extractability of Ψ_2. $\mathcal{B}$ runs exactly as the challenger in Game 2.5, but when $\mathsf{O}'^{\mathsf{Decrypt}}$ returns $\mathsf{failure}$, it halts execution and return the corresponding $(\Lambda, Z, \mathbf{C}, \mathsf{pk}_{\mathsf{A}})$ tuple that cases extraction to fail. We know Ψ_2 is simulation extractible, so $\mathcal{B}$'s advantage is negligible, and so is the probability of failure in Game 2.5 as long as the adversary is probabilistic polynomial time.

Consider the case where $\emptyset$ is returned and extraction does not fail, denoted $E_{badproof}$. This happens when $\mathsf{Decrypt}$ does not return $f(x, y)$ where y is the result of running Ext in $\mathsf{O}'^{\mathsf{Decrypt}}$. We claim that this happens with negligible probability. Otherwise, we can perform a reduction $\mathcal{B}$ to the correctness of HEC. $\mathcal{B}$ functions the same as the challenger in Game 2.5, with the following modifications:

1. It receives *hecpar* from its challenger, and uses it in generating Λ.
2. It receives (x, r_{A}), passes f, x to its challenger and uses the returned X, d to generate $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}})$.
3. Upon a decryption oracle query with values (C, Z) where extraction does not fail but the outputs disagrees, it pauses execution of $\mathcal{A}$. Further, $\mathcal{B}$ uses Ext_2 to extract values $(y, r, r_{\hat{Z}})$ from Z via rewinding, Then, it returns $(y, r_{\hat{Z}})$ to the challenger.

Notice that, up to the point that the event in interest occurs, $\mathcal{B}$ acts identically as $\mathcal{A}$'s challenger in Game 2.5. Additionally, when the event occurs, $\mathcal{B}$ returns a valid reduction. By correctness of HEC, we get that:

$$\Pr[E_{badproof}] = \Pr\big[\text{HEC}\textsc{correct}^{\mathcal{B}}(1^\lambda) = \mathsf{failure}\big] = \nu(\lambda)$$

for some negligible function ν, as required.

Therefore, the decryption oracle in Game 2.5 agrees with that of Game 2 and Game 3 with overwhelming probability.

The logic for Game 3 and 4 is similar to that of Game 1 and Game 2 – a reduction to the zero knowledge property of Ψ_1 described in Figure 2.1. Here, $\mathcal{B}$ receives $(\mathsf{S}', \mathsf{P}')$ and assigns $\mathsf{S}_1 \leftarrow \mathsf{S}'$ in $\mathsf{SimSetup}$. When computing the proof π_{A}, it uses P'. At the end, it outputs whatever $\mathcal{A}$ outputs.

When $\mathcal{B}$ is in $\mathsf{NIZK}^{\mathcal{B},0}$, $\mathcal{B}$ acts as $\mathcal{A}$'s challenger in Game 3. Likewise, it acts as $\mathcal{A}$'s challenger in Game 4 if $\mathcal{B}$ is in $\mathsf{NIZK}^{\mathcal{B},1}$. Therefore, we get:

$$\begin{aligned}\left|\Pr\left[\mathsf{PWHA}_3^{\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{PWHA}_4^{\mathcal{A}}(\lambda) = 0\right]\right| = \\ \left|\Pr\left[\mathsf{NIZK}^{\mathcal{B},0}(\lambda) = 0\right] - \Pr\left[\mathsf{NIZK}^{\mathcal{B},1}(\lambda) = 0\right]\right| = \nu(\lambda)\end{aligned}$$

for some negligible function ν as required, since NIZK_1 is zero-knowledge.

For Game 4 and Game 5, we can show the adversary performs negligibly different by a series of hybrid arguments. Suppose $\mathcal{A}$ is allowed to submit $\ell(\lambda)$ queries to the Escrow oracle where ℓ is a polynomial. Let Hybrid i, denoted $\mathsf{PWHA}_{4,i}$, defined for $i \in [0, \ell(k)]$ be the same as Game 5, but for the first i calls to $\mathsf{O}^{\mathsf{Escrow}}$, the challenger answers with $\mathsf{O}^{\mathsf{Escrow}}$, and for subsequent calls it answers with $\mathsf{O}'^{\mathsf{Escrow}}$. We see that Game 4 is the same as $\mathsf{PWHA}_{4,0}$ and Game 5 is the same as $\mathsf{PWHA}_{4,\ell(\lambda)}$.

Consider the following reduction $\mathcal{B}$ to the security of (x, y) as described in Figure 4.1:

1. $\mathcal{B}$ receives *hecpar* and uses it in generating Λ.

2. $\mathcal{B}$ outputs (f, x, x) to the challenger when it receives (x, r_A) and receives X. It then uses Sim_1 to generate π_A corresponding to X and computes $\mathsf{pk}_\mathsf{A} = (\Lambda, X, C_\mathsf{A} = \mathsf{Commit}(x; r_\mathsf{A}), \pi_\mathsf{A})$.
3. $\mathcal{B}$ then samples $j \leftarrow\$ [0, \ell(\lambda))$ uniform randomly.
4. For the first j queries to $\mathsf{O}^{\mathsf{Escrow}}$, it computes $\hat{Z}$ as $\textsc{HECEval}(\mathit{hecpar}, f, X, y)$.
5. For the $(j+1)^{th}$ query, $\mathcal{B}$ samples $y' \leftarrow\$ \mathit{domain}_{f,y}$ and outputs (y', y) to its challenger. It then receives $\hat{Z}$ from the challenger, and computes π_U using Sim_2 on this $\hat{Z}$.
6. For the next queries, it computes Z with Sim the same way as in $\mathsf{O}_1^{\mathsf{Escrow}}$.

We see that, in $\textsc{SecXY}_0^{\mathcal{B}}$, $\mathcal{B}$ is identical to $\mathcal{A}$'s challenger in $\mathsf{PWHA}_{4,j}$. In $\textsc{SecXY}_1^{\mathcal{B}}$, $\mathcal{B}$ is identical to $\mathcal{A}$'s challenger in $\mathsf{PWHA}_{4,j+1}$. By a hybrid argument, we get that:

$$\left|\Pr\left[\mathsf{PWHA}_4^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PWHA}_5^{\mathcal{A}}(1^\lambda) = 0\right]\right| =$$
$$\left|\Pr\left[\mathsf{PWHA}_{4,0}^{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{PWHA}_{4,\ell(\lambda)}^{\mathcal{A}}(1^\lambda) = 0\right]\right| =$$
$$\ell(\lambda)\left|\Pr\left[\textsc{SecXY}_0^{\mathcal{B}}(1^\lambda) = 0\right] - \Pr\left[\textsc{SecXY}_1^{\mathcal{B}}(1^\lambda) = 0\right]\right| = \ell(\lambda)\nu(\lambda)$$

for some negligible function ν since HEC is secure. This is still negligible, as required.

The argument for Game 5 and 6 is identical to that of Game 3 and Game 4.

The argument for Game 6 and 7 is identical to that of Game 2 and Game 3, except the reduction would make use of the simulator SimS used in $\mathsf{O}_1^{\mathsf{Escrow}}$. Everything else about the reduction stays the same.

6 Construction of HEC for f_k from ElGamal

Let $\mathit{domain}_{f,y} = \{0,1\}^{l_y}$. We require, for the sake of security, that 2^{l_y} be super-polynomial in terms of our security parameter λ. Additionally, we require the group we use in ElGamal has prime cardinality $q \geq 2^{l_y+1}$. Let function $\mathsf{lobits} : \{0,1\}^{l_y} \to \{0,1\}^k$ be defined as follows:

$$\mathsf{lobits}_k(y) = y \bmod 2^k$$

There, $\mathsf{lobits}_k(y)$ can be interpreted as an identifier for a user with private input y. $\mathsf{lobits}_k(y)$ returns the k least significant bits of y if it is interpreted as a binary integer. We assume implicit conversion between binary strings and integers.

We will construct a Homomorphic Enough Encryption using ElGamal for the function family $F = \{f_k\}$ with $\mathbf{x} \subseteq \mathit{domain}_{f,y}$.

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

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

We use bold font to indicate that $\mathbf{x}$ is a set of values. We sometimes interpret it as a vector $x_1, \ldots, x_{|\mathbf{x}|}$ by imposing an arbitrary order. $\mathbf{x}$ can be interpreted as a watchlist that reveals information about the users when used with f_k. As a caveat, for our construction, particularly HECDEC, to be polynomial time, we require that $2^{l_y - k}$ is polynomial to our security parameter λ. Additionally, lists in $domain_{f,\mathbf{x}}$ must have a fixed length, otherwise it is possible to distinguish between encryptions of lists of different lengths.

The idea is that with ElGamal, HECENC can encrypt a polynomial P that has roots at the elements of $\mathbf{x}$ as well as another element $s \in \mathbb{Z}_q \backslash \{0, 1, \ldots, 2^k - 1\}$. Then, HECEVAL can compute an encryption of y that is additively blinded by a randomized evaluation of the encrypted P at point $\mathsf{lobits}_k(y)$. If the values match, the blinding value is an encryption of 0. Thus, HECDEC can decrypt this encrypted evaluation and either obtain a random value or y.

6.1 ElGamal construction

Figure 6.1 describes our ElGamal construction for a HEC for the aforementioned functions f_k. Here, $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ are the key generation, encryption, and decryption algorithms respectively for the ElGamal cryptosystem. Recall that $\oplus$ is the homomorphic operator for ElGamal ciphertexts.

Our construction will employ ciphertexts $\mathbf{C}_i \leftarrow \mathsf{Enc}(g^{P_i})$ that encrypt the coefficients P_i of the polynomial $P = (\chi - s) \prod_{i=1}^{|\mathbf{x}|} (\chi - x_i)$, that is $P = \sum_{i=0}^{|\mathbf{x}|+1} P_i \chi^i$.

Theorem 3. *Under the decisional Diffie-Hellman assumption, our construction above constitutes a homomorphic-enough encryption for f_k.*

We prove each of the required security properties in a separate lemma.

Lemma 7. *Under the decisional Diffie-Hellman assumption, our* HEC *scheme for f_k satisfies correctness with adversarial evaluation randomness.*

Proof. Let $\mathcal{A}$ be any fixed PPT adversary. We can construct the reduction $\mathcal{B}$ to the CPA security of ElGamal as described in Figure A.1. $\mathcal{B}$ creates two polynomials $P_j \leftarrow (\chi - s_0) \prod_{i=1}^{|\mathbf{x}|} (\chi - x_i)$, $j \in \{0, 1\}$. 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 Figure C.1. Observe that, regardless of b, $\mathcal{B}$ functions identical to $\mathcal{A}$'s challenger.

Let $\mathsf{extend}(\mathbf{x})$ denote the set $\{y | y \in domain_{f,y} \wedge \mathsf{lobits}_k(y) \in \mathbf{x}\}$. Elements y of $\mathsf{extend}(\mathbf{x})$ have the property that $P(\mathsf{lobits}_k(y)) = 0$, and decrypts to y if it is used in HECEVAL.

Here, we suppose that HECEVAL uses r_Z by partitioning it into r used for randomizing the encrypted polynomial evaluation and r_{Enc} to run Enc on y.

Consider the case where $\mathcal{A}$ succeeds in the HECCORRECT experiment. We know that, for y chosen by $\mathcal{A}$, it is not the case that $\mathsf{lobits}_k(y) \in \mathbf{x}$. Otherwise, $P_b(\mathsf{lobits}_k(y)) = 0$ and HECEVAL's output always decrypts to y no matter the r value chosen. As $s_b > 2^k$, $\mathsf{lobits}_k(y)$ cannot be s_b.

```
HECEnc($hecpar, f_k, \mathbf{x}$)
    $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow \mathsf{KGen}(1^\lambda)$
    $s \leftarrow\$ \mathbb{Z}_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
    $P \leftarrow (\chi - s)\prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$
    for $i$ in $\{0, \ldots, |\mathbf{x}| + 1\}$
        $\mathbf{C}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_E, g^{P_i})$
    return $(X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}|+1}),$
      $d = (\mathsf{sk}_E, f_k, \mathbf{x})))$

HECDec($hecpar, d = (\mathsf{sk}_E, f_k, \mathbf{x}), Z$)
    $D \leftarrow \mathsf{Dec}(\mathsf{sk}_E, Z_1)$
    for $y$ in $domain_{f,y} \wedge \mathsf{lobits}_k(y) \in \mathbf{x}$
      if $g^y = D$
        return $y$
    return $\emptyset$

HECEval($hecpar, f_k, X, y$)
    parse $X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}|+1})$
    $eval \leftarrow \bigoplus_{i=0}^{|\mathbf{x}|+1} (\mathbf{C}_i)^{\mathsf{lobits}_k(y)^i}$
    $enc \leftarrow \mathsf{Enc}(\mathsf{pk}_E, g^y)$
    $r \leftarrow\$ \mathbb{Z}_q$
    return $Z = eval^r \oplus enc$

HECdirect($hecpar, X, z$)
    parse $X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, C_{|\mathbf{x}|+1})$
    if $z = \emptyset$
      $\beta \leftarrow\$ \mathbb{Z}_q$
      return $\mathsf{Enc}(\mathsf{pk}_E, g^\beta)$
    return $\mathsf{Enc}(\mathsf{pk}_E, g^z)$
```

Fig. 6.1: ElGamal instantiation of a Homomorphic Enough Cryptosystem

Therefore, y must be a value for which $P_b(\mathsf{lobits}_k(y)) \neq 0$. Additionally, since $\mathcal{A}$ succeeds, it must be that $\mathsf{Decrypt}$ returns something other than $\emptyset$. Therefore, we know that $\mathsf{lobits}_k(P_b(\mathsf{lobits}_k(y))r + y) \in \mathbf{x}$.

Let's denote $\bar{b}$ as $1 - b$. Since $P_{\bar{b}}$ is generated independently of the adversary's view, we can imagine a different scenario where $s_{\bar{b}}$ and $P_{\bar{b}}$ are generated after all of $\mathcal{A}$'s responses. Here, $\mathsf{lobits}_k(P_{\bar{b}}(\mathsf{lobits}_k(y))r + y) \in \mathbf{x}$ occurs with the same probability as in our HEC correctness experiment. We claim that this probability is negligible.

Suppose $\mathcal{A}$ chose values $y, r, \mathbf{x}$. Consider the event E:

$$\begin{aligned} E &= [\mathsf{lobits}_k(P_b(\mathsf{lobits}_k(y))r + y) \in \mathbf{x}] \\ &= [P_b(\mathsf{lobits}_k(y))r + y \in \mathsf{extend}(\mathbf{x})] \end{aligned}$$

There are at most $2^{l_y - k}|\mathbf{x}|$ values for $P_{\bar{b}}(\mathsf{lobits}_k(y))$ such that the equation corresponding to E holds. Fixing $P_{\bar{b}}(\mathsf{lobits}_k(y))$, along with $\mathbf{x}$ also uniquely identifies a polynomial $P_{\bar{b}}$ since $y \notin \mathsf{extend}(\mathbf{x})$. This is because doing so gives us $|\mathbf{x}| + 1$ points on the polynomial, and we know a degree n polynomial can be uniquely identified by n unique points on the polynomial. But we know that there are $q - 2^k$ unique polynomials $P_{\bar{b}}$ can be due to how we generate the root $s_{\bar{b}}$, and each occurs with the same probability. Therefore, the probability that the event E happens is at most: $\frac{2^{l_y - k}|\mathbf{x}|}{q - 2^k}$. We know that the numerator is a product of two

```
IND-CPA$_{\mathcal{C},b}(1^\lambda)$
1 : $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
2 : **return** $\mathcal{C}^{\mathsf{O}_b(\mathsf{pk}_E,\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$

$\mathsf{O}_b(\mathsf{pk}_E, m_0, m_1)$
1 : **return** $\mathsf{Enc}(\mathsf{pk}_E, m_b)$

$\mathcal{B}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$
1 : $hecpar \leftarrow \text{HECSETUP}(\lambda)$
2 : $(f, \mathbf{x}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$
3 : **if** $f \in F, \mathbf{x} \in domain_{f,\mathbf{x}}$
4 :     $s_0 \leftarrow\$ Z_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
5 :     $s_1 \leftarrow\$ Z_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
6 :     $P_0 \leftarrow (\chi - s_0) \prod_{i=1}^{|\mathbf{x}|} (\chi - x_i)$
7 :     $P_1 \leftarrow (\chi - s_1) \prod_{i=1}^{|\mathbf{x}|} (\chi - x_i)$
8 :     **for** $i$ **in** $\{0, \ldots, |x| + 1\}$
9 :         $\mathbf{C}_i \leftarrow \mathsf{O}_b(g^{P_{0,i}}, g^{P_{1,i}})$
10 :    $X \leftarrow (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}|+1})$
11 :    $(y, r_Z) \leftarrow \mathcal{A}(\mathsf{state}_\mathcal{A}, X)$
12 :    **if** $y \in domain_{f,y}$
13 :        **parse** $r_Z = (r, r_{\mathsf{Enc}})$
14 :        $y_0 \leftarrow P_0(\mathsf{lobits}_k(y))r + y$
15 :        $y_1 \leftarrow P_1(\mathsf{lobits}_k(y))r + y$
16 :        **for** $b'$ **in**$\{1, 0\}$
17 :            **if** $(\mathsf{lobits}_k(y) \notin \mathbf{x} \cup \{s_{b'}\}) \wedge (\mathsf{lobits}_k(y_{b'}) \in \mathbf{x})$
18 :                **return** $b'$
19 :        **return** 0
20 :    **return** 0
21 : **return** 0
```

Fig. 6.2: Reduction from HEC correctness to IND-CPA

values that are polynomial to the security parameter λ, and the denominator is at least 2^k, which is super polynomial in λ. Therefore, the above probability is negligible.

Suppose that $\mathcal{A}$ succeeds with probability $p(\lambda)$. With $b = 0$, $\mathcal{B}$ outputs 1 with probability equal to some negligible $\nu(\lambda)$ as argued above. With $b = 1$,

$\mathcal{B}$ outputs 1 with probability $p(\lambda)$. By CPA security of ElGamal, we know that $|p(\lambda) - \nu(\lambda)|$ is also negligible, so p is a negligible function. Therefore, $\mathcal{A}$ succeeds with negligible probability, as required.

```
IND-CPA_{B,b}(1^λ)
1 :  $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
2 :  return $\mathcal{B}^{\mathsf{O}_b(\mathsf{pk}_E,\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$

O_b(pk_E, m_0, m_1)
1 :  return $\mathsf{Enc}(\mathsf{pk}_E, m_b)$

B^{O_b(·,·)}(1^λ, pk_E)
1 :  $hecpar \leftarrow \text{HECSETUP}(\lambda)$
2 :  $(f, \mathbf{x}_0, \mathbf{x}', \mathsf{state}) \leftarrow \mathcal{A}(hecpar)$
3 :  $s_0 \leftarrow\!\!\$\ \mathbb{Z}_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
4 :  $s_1 \leftarrow\!\!\$\ \mathbb{Z}_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
5 :  $P = (\chi - s_0) \prod_{i=1}^{|\mathbf{x}|} (\chi - \mathbf{x}_i)$
6 :  $P' = (\chi - s_1) \prod_{i=1}^{|\mathbf{x}'|} (\chi - \mathbf{x}'_i)$
7 :  for $i$ in $\{0, |\mathbf{x}| + 1\}$
8 :      $\mathbf{C}_i = \mathsf{O}_b(g^{P_i}, g^{P'_i})$
9 :  $X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, C_{|\mathbf{x}|+1})$
10 : return $\mathcal{A}(hecpar, X, \mathsf{state})$
```

Fig. 6.3: Reduction from security of $\mathbf{x}$ to IND-CPA

Lemma 8. *Our ElGamal construction satisfies* ***Security of*** $\mathbf{x}$

Proof. Let $\mathcal{A}$ be any probabilistic polynomial algorithm. We shall construct a reduction $\mathcal{B}$ to CPA security of ElGamal as described in figure C.2. Since ElGamal is CPA secure, we get that:

$$\Pr\left[\text{IND-CPA}_{\mathcal{B},0}(1^\lambda) = 0\right] - \Pr\left[\text{IND-CPA}_{\mathcal{B},1}(1^\lambda) = 0\right] | = \nu(\lambda)$$
$$\Pr\left[\mathsf{SecX}_0^{\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{SecX}_1^{\mathcal{A}}(\lambda) = 0\right] | = \nu(\lambda)$$

for some negligible function ν, as required.

```
IND-CPA$_{\mathcal{C},b}(1^\lambda)$
1 : $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
2 : return $\mathcal{C}^{\mathsf{O}_b(\mathsf{pk}_E,\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$

$\mathsf{O}_b(\mathsf{pk}_E, m_0, m_1)$
1 : return $\mathsf{Enc}(\mathsf{pk}_E, m_b)$

$\mathcal{C}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$
1 : $hecpar \leftarrow \text{HECSETUP}(1^\lambda)$
2 : $(f, \mathbf{x}, \mathbf{x}_1, \mathsf{state}) \leftarrow \mathcal{A}(hecpar)$
3 : if $f \in F, \mathbf{x}_0, \mathbf{x}_1 \in domain_{f,\mathbf{x}}$
4 :     $s_0 \leftarrow\$ \mathbb{Z}_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
5 :     $s_1 \leftarrow\$ \mathbb{Z}_q \setminus \left\{0, 1, \ldots, 2^k - 1\right\}$
6 :     $P_0 \leftarrow (\chi - s_0) \prod_{i=1}^{|\mathbf{x}|} (\chi - \mathbf{x}_{0,i})$
7 :     $P_1 \leftarrow (\chi - s_1) \prod_{i=1}^{|\mathbf{x}|} (\chi - \mathbf{x}_{1,i})$
8 :     for $i$ in $\{0, |\mathbf{x}_0| + 1\}$
9 :         $\mathbf{C}_i \leftarrow \mathsf{O}_b(g^{P_{0,i}}, g^{P_{1,i}})$
10 :    $X \leftarrow (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}_0|+1})$
11 :    $(y_0, y_1, \mathsf{state}) \leftarrow \mathcal{A}(X, \mathsf{state})$
12 :    if $y_0, y_1 \in domain_{f,y}$
13 :        $eval \leftarrow \mathsf{O}_b(\mathsf{pk}_E, g^{P_0(\mathsf{lobits}_k(y_0))}, g^{P_1(\mathsf{lobits}_k(y_1))})$
14 :        $enc \leftarrow \mathsf{O}_b(\mathsf{pk}_E, y_0, y_1)$
15 :        $r \leftarrow\$ \mathbb{Z}_q$
16 :        $Z \leftarrow eval^r \oplus enc$
17 :        return $\mathcal{A}(Z, \mathsf{state})$
18 :    return $\mathcal{A}(\bot, \mathsf{state})$
19 : return $\mathcal{A}(\bot, \mathsf{state})$
```

Fig. 6.4: Reduction from security of XY to IND-CPA

Lemma 9. *Our ElGamal construction achieves* ***Security of*** *x* ***and*** *y* ***from third parties***

Proof. Let $\mathcal{A}$ be a fixed p.p.t. algorithm. We shall construct a reduction $\mathcal{B}$ as described in figure C.3. Note that $\mathcal{B}$ acts identically as $\mathcal{A}$'s challenger in the SECXY game in Figure 4.1. In particular, X is computed identically as how $\mathcal{A}$'s challenger would, and we claim that Z is distributed identically as if $\mathcal{A}$'s challenger would have computed it.

The distribution of Z in our reduction $\mathcal{B}$ is:

$$\begin{aligned}&\Big\{(\alpha,\beta,r)\leftarrow\$\ \mathbb{Z}_q^3 : (g^{\alpha r}, g^{P_b(\mathsf{lobits}_k(y_b))r}h^{\alpha r})\oplus(g^{\beta}, g^{y}h^{\beta})\Big\}\\ =&\Big\{(\alpha,\beta,r)\leftarrow\$\ \mathbb{Z}_q^3 : (g^{\alpha r+\beta}, g^{P_b(\mathsf{lobits}_k(y_b))r+y}h^{\alpha r+\beta})\Big\}\\ =&\Big\{(\alpha,\beta,r)\leftarrow\$\ \mathbb{Z}_q^3 : (g^{\beta}, g^{P_b(\mathsf{lobits}_k(y_b))r+y}h^{\beta})\Big\}\end{aligned}$$

since β is sampled uniformly over $\mathbb{Z}_q$ and so is the distribution of $\beta+\alpha r$.

Let r_i denote the randomness used to encrypt $\mathbf{C}_i$ for X_b. The distribution of Z in SecXY can be expressed as:

$$\Big\{(\beta,r)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{(\sum_{i=0}^{|\mathbf{x}|+1} r_i\mathsf{lobits}_k(y)^i)^r+\beta}, g^{(\sum_{i=0}^{|\mathbf{x}|+1} P_i\mathsf{lobits}_k(y)^i)^r+y}h^{(\sum_{i=0}^{|\mathbf{x}|+1} r_i\mathsf{lobits}_k(y)^i)^r+\beta})\Big\}$$

And since β is sampled uniformly from $\mathbb{Z}_q$, then so is the distribution of $\beta+(\sum_{i=0}^{|\mathbf{x}|+1} r_i\mathsf{lobits}_k(y)^i)r$. So we get:

$$\begin{aligned}&\Big\{(\beta,r)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{\beta}, g^{(\sum_{i=0}^{|\mathbf{x}|+1}(P_b)_i\mathsf{lobits}_k(y)^i)r+y}h^{\beta})\Big\}\\ &\Big\{(\beta,r)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{\beta}, g^{P_b(\mathsf{lobits}_k(y))r+y}h^{\beta})\Big\}\end{aligned}$$

which is identical to our distribution of Z in $\mathcal{B}$.

Since $\mathcal{B}$ is identical to $\mathcal{A}$'s challenger, we get:

$$\begin{aligned}\Big|\Pr\big[\text{IND-CPA}_{\mathcal{B},0}(1^\lambda)=0\big]-\Pr\big[\text{IND-CPA}_{\mathcal{B},1}(1^\lambda)=0\big]\Big|&=\nu(\lambda)\\ \Big|\Pr\Big[\mathsf{SecXY}_0^{\mathcal{A}}(\lambda)=0\Big]-\Pr\Big[\mathsf{SecXY}_1^{\mathcal{A}}(\lambda)=0\Big]\Big|&=\nu(\lambda)\end{aligned}$$

as required.

Lemma 10. *Our ElGamal construction achieves* ***Security of*** DIRECTZ.

Proof. Here, we can show that Z_0 and Z_1 in the experiment in Figure 4.1 are identically distributed, regardless of the value of X and $\mathbf{x}$. From the proof in Theorem 13, we get that the distribution of HECEVAL is as follows:

$$\begin{aligned}&\Big\{(\beta,r)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{\beta}, g^{(\sum_{i=1}^{|\mathbf{x}|+1}P_i\mathsf{lobits}_k(y)^i)r+y}h^{\beta})\Big\}\\ &\Big\{(\beta,r)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{\beta}, g^{P(\mathsf{lobits}_k(y))r+y}h^{\beta})\Big\}\end{aligned}$$

Similarly, we get that the distribution of HECDIRECT when $\mathsf{lobits}_k(y)\in X$ is as follows:

$$\Big\{(\beta)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{\beta}, g^{f(\mathbf{x},y)}h^{\beta})\Big\}$$

And otherwise, it is an encryption of a random value, distributed as:

$$\Big\{(\alpha,\beta)\leftarrow\$\ \mathbb{Z}_q^2 : (g^{\beta}, g^{\alpha}h^{\beta})\Big\}$$

Now suppose we fix a y value. If $\mathsf{lobits}_k(y) \in X$, then $f(\mathbf{x}, y) = y$, and $P(\mathsf{lobits}_k(y)) = 0$, so we get that both distributions are uniform over all ElGamal encryptions of g^y.

Otherwise, if $\mathsf{lobits}_k(y) \notin X$, then $f(\mathbf{x}, y) = \emptyset$. In $\textsc{Direct}\mathrm{Z}_0$, we know that $P(\mathsf{lobits}_k(y)) \neq 0$ since $\mathsf{lobits}_k(y)$ cannot be one of the $|\mathbf{x}| + 1$ roots of $\mathbf{x}$. This is because we explicitly restrict the root to $\mathbb{Z}_q \setminus \{0..2^k - 1\}$ which excludes the domain of lobits_k. We know that r is chosen uniform randomly, so $P(\mathsf{lobits}_k(y))r + y$ is also uniformly distributed over $\mathbb{Z}_q$ and the two distributions are identical.

6.2 NIZK for ElGamal Construction

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

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

For proving that the value s used to generate the polynomial P in $\text{HEC}\textsc{Enc}$ is sampled in the appropriate range, we employ Bulletproofs [11] . Apart from this, the remaining 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 $P = (\chi - s)P'$ involves proving that $P = \chi P' - sP'$. 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. Since ElGamal is additively homomorphic in the exponent, we can subtract the encrypted $P's$ from a shifted encryption of P' to form P. See Appendix D 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:

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

Similar to Ψ_1, 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 [11]. 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 Figure 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 Appendix D for more details.

7 Construction of HEC for any *f* from Fully Homomorphic Encryption

Definition 6 (Circuit-private (CP) fully homomorphic encryption (FHE)). *A set of algorithms* ($\mathsf{FHEKeyGen}, \mathsf{FHEEnc}, \mathsf{FHEDec}, \mathsf{FHEEval}$) *constitute a secure fully homomorphic public-key encryption scheme [38,9,8,39] if:*

Input-output specification $\mathsf{FHEKeyGen}(1^\lambda, \Lambda)$ *takes as input the security parameter and possibly system parameters* Λ *and outputs a secret key FHESK and a public key FHEPK.* $\mathsf{FHEEnc}(\mathit{FHEPK}, b)$ *takes as input the public key and a bit* $b \in \{0,1\}$ *and outputs a ciphertext c.* $\mathsf{FHEDec}(\mathit{FHESK}, c)$ *takes as input a ciphertext c and outputs the decrypted bit* $b \in \{0,1\}$. $\mathsf{FHEEval}(\mathit{FHEPK}, \Phi, c_1, \ldots, c_n)$ *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* $\mathsf{FHEDec}(\mathit{FHESK}, C) \neq \Phi(x)$ *where* $(\mathit{FHESK}, \mathit{FHEPK})$ *are output by* $\mathsf{FHEKeyGen}$, $c_1, \ldots, c_n$ *are ciphertexts where* $c_i \leftarrow \mathsf{FHEEnc}(\mathit{FHEPK}, x_i)$*, and* $c_\Phi = \mathsf{FHEEval}(\mathit{FHEPK}, \Phi, c_1, \ldots, c_n)$*, 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* $(\mathit{FHESK}, \mathit{FHEPK})$ *output by* $\mathsf{FHEKeyGen}(\lambda)$ *and for all input ciphertexts* $c_1, \ldots, c_n$ *generated by* $\mathsf{FHEEnc}(\mathit{FHEPK}, \cdot)$*,* c_Φ *generated by* $\mathsf{FHEEval}(\mathit{FHEPK}, \Phi, c_1, \ldots, c_n)$ *is at most* $s(\lambda)$ *bits long.*

An FHE scheme is, additionally, ***circuit-private*** *[38,47,7,30] for a circuit family* $\mathcal{C}$ *for any probabilistic polynomial-time algorithm* $\mathcal{A}$, $|p_{\mathcal{A},0} - p_{\mathcal{A},1}| = \nu(1^\lambda)$ *for a negligible* ν*, where for* $b \in \{0,1\}$*,* $p_{\mathcal{A},b}$ *is the probability that the following experiment outputs* 0*:*

```
FHECircHideExpt($1^\lambda$)
$(R, \Phi_0, \Phi_1, (x_1, r_1), \ldots, (x_n, r_n)) \leftarrow \mathcal{A}(1^\lambda)$
**if** $\Phi_0 \notin \mathcal{C} \vee \Phi_1 \notin \mathcal{C} \vee \Phi_0(x_1, \ldots, x_n) \neq \Phi_1(x_1, \ldots, x_n)$ : **reject**
$(FHEPK, FHESK) = \mathsf{FHEKeyGen}(1^\lambda; R)$
**for** $i \in \{1, \ldots, n\}$ :
    $c_i = \mathsf{FHEEnc}(FHEPK, x_i; r_i)$
$Z_0 \leftarrow \mathsf{FHEEval}(FHEPK, \Phi_0, c_1, \ldots, c_n)$
$Z_1 \leftarrow \mathsf{FHEEval}(FHEPK, \Phi_1, c_1, \ldots, c_n)$
**return** $\mathcal{A}(Z_b)$
```

Bibliographic note. Definitions of circuit-privacy in the literature come in different flavors; we chose to formulate the definition in a way that makes it easiest to prove Theorem 4 below. The strongest, malicious circuit-privacy [47,30], 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. [47] and by Döttling and Dujmović [30].

Similarly, we chose to formulate correctness as perfect correctness, rather than allowing a negligible probability (over the choice of 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 Appendix E.

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

HECSETUP(1^λ) Generate the FHE parameters Λ, if needed.

HECENC($1^\lambda, \Lambda, f, x$) First, 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 $d = FHESK$.

HECEVAL($hecpar, f, X, y$) 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 $Z = Z_1, \ldots, Z_\ell$.

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

HECDIRECT$(hecpar, X, z)$ 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 [29] 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 for helpful discussions. This research was supported by NSF grant 2154170, and by grants from Meta.

References

1. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer, Heidelberg, August 2000.
2. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013.
3. Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols Workshop*, volume 3957 of *Lecture Notes in Computer Science*, pages 20–42. Springer, 2004.
4. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.
5. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 626–643. Springer, Heidelberg, December 2012.
6. Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography.* `https://toc.cryptobook.us/`.

7. Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 62–89. Springer, Heidelberg, August 2016.
8. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.
9. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.
10. Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 423–443. Springer, Heidelberg, February 2020.
11. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
12. Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proc. 13th ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.
13. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In Ronald Cramer, editor, *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.
14. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In Roberto De Prisco and Moti Yung, editors, *Proceedings of the 5th International Conference on Security and Cryptography for Networks (SCN)*, volume 4116 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2006.
15. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2001.
16. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *LNCS*, pages 268–289, 2003.
17. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72, 2004.
18. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, Heidelberg, August 1997.
19. Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998.
20. Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Heidelberg, August 2020.
21. David Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, pages 199–203. Plenum Press, 1982.

22. David Chaum. Blind signature systems. In *CRYPTO '83*, pages 153–156. Plenum, 1983.
23. David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
24. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *CRYPTO '90*, volume 403 of *LNCS*, pages 319–327, 1990.
25. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.
26. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.
27. Ivan Damgård. On σ-protocols. Available at `http://www.daimi.au.dk/~ivan/Sigma.ps`, 2002.
28. Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain identity management. In Kenneth G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 552–576. Springer, Heidelberg, May 2021.
29. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.
30. Nico Döttling and Jesko Dujmovic. Maliciously circuit-private FHE from information-theoretic principles. Cryptology ePrint Archive, Report 2022/495, 2022. `https://eprint.iacr.org/2022/495`.
31. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.
32. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012.
33. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.
34. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.
35. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, volume 1294 of *LNCS*, pages 16–30, 1997.
36. Eiichiro Fujisaki and Tatsuaki Okamoto. Witness hiding protocols to confirm modular polynomial relations. In *The 1997 Symposium on Cryptograpy and Information Security*, Fukuoka, Japan, January 1997. The Institute of Electronics, Information and Communcation Engineers. SCSI97-33D.
37. Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). Cryptology ePrint Archive, Report 2021/1393, 2021. `https://eprint.iacr.org/2021/1393`.

38. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of STOC 2009*, pages 169–178, 2009.
39. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.
40. Matthew Green, Gabriel Kaptchuk, and Gijs Van Laer. Abuse resistant law enforcement access systems. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 553–583. Springer, Heidelberg, October 2021.
41. Joe Kilian and Erez Petrank. Identity escrow. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 169–185. Springer, Heidelberg, August 1998.
42. Anna Lysyanskaya. *Signature schemes and applications to cryptographic protocol design.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.
43. Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *LNCS*, 1999.
44. Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable sigma-protocols in the global random-oracle model. Cryptology ePrint Archive, Report 2022/290, 2022. https://eprint.iacr.org/2022/290.
45. Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 272–286. Springer, Heidelberg, June 2009.
46. C.Ãndrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 116–125. ACM press, November 2001.
47. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2014.
48. Rafail Ostrovsky and William E. Skeith III. Private searching on streaming data. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 223–240. Springer, Heidelberg, August 2005.
49. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.
50. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

A Deferred Preliminaries

A.1 IND-CPA security

An encryption scheme $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure if for any PPT adversary $\mathcal{A}$ participating in the experiment described in Figure A.1, the advantage

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

is some negligible function ν.

```
IND-CPA$^{\mathcal{A},b}_{\mathsf{Enc}}(1^\lambda)$
  1 : $(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
  2 : return $\mathcal{A}^{\mathsf{O}^b_{\mathsf{Enc}}(\mathsf{pk},\cdot,\cdot)}(1^\lambda,\mathsf{pk})$

$\mathsf{O}^b_{\mathsf{Enc}}(\mathsf{pk},m_0,m_1)$
  1 : return $\mathsf{Enc}(\mathsf{pk},m_b)$
```

Fig. A.1: IND-CPA game

Note that the standard definition for IND-CPA only allows an adversary to query the oracle once. It can be shown that our definition for IND-CPA is equivalent [6].

A.2 Black-Box (BB) vs. Straight-Line (SL) Simulation Extractability

It is straightforward to see that black-box simulation extractability is as strong or weaker than regular simulation extractability, because anything that an extractor can do without black-box access to $\mathcal{A}$ it can also do with it (it just won't use it). It is somewhat less obvious that in fact the resulting flavor of knowledge extraction is inferior, in the sense that a protocol that uses regular simulation extractability may have better security properties than one that uses the black-box flavor. The problem arises when the proof of security of such a protocol tries to use the extractor: this extractor needs $\mathsf{BB}(\mathcal{A})$, which requires resetting the adversary to a previous state and replaying its view. Sometimes (in some proofs of security) that also requires resetting the overall security experiment, which is something that a security reduction may not always be able to do. For that reason, regular (straight-line) extraction is preferred, when it can be achieved.

Let us now formalize BB simulation extractability; as before, 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. Φ is black-box simulation-extractable if there exists a polynomial-time extractor algorithm Ext such that for any PPT adversary $\mathcal{A}$ participating in the game defined in Figure A.2, the advantage function $\nu(\lambda)$ defined below is negligible. As 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 $\mathsf{O_S}$. In fact, the game here is identical to that in Figure 2.3, except now Ext also gets access to $\mathsf{BB}(\mathcal{A})$.

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

for some negligible function ν.

```
$\mathsf{NISimBBExtract}^{\mathcal{A}}(1^\lambda)$
1 : $\mathcal{Q}, \mathcal{Q}_\mathsf{S} \leftarrow [\,]$
2 : $(\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\tilde{\mathsf{O}}_\mathsf{S}(\cdot), \mathsf{O}_\mathsf{Sim}(\cdot)}(1^\lambda)$
3 : $\mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{BB}(\mathcal{A})}(\mathcal{Q}_\mathsf{S}, \mathbb{x}, \pi)$
4 : $\textbf{return } \mathsf{V}^{\mathsf{O}_\mathsf{S}}(\mathbb{x}, \pi) \wedge (\mathbb{x}, \pi) \notin \mathcal{Q} \wedge (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$

$\mathsf{O}_\mathsf{S}(m)$ $\underline{\tilde{\mathsf{O}}_\mathsf{S}(m)}$
1 : $\mathsf{state}, h, \tau_\mathsf{Ext} \leftarrow \mathsf{SimS}(\mathsf{state}, m)$
2 : $\mathcal{Q}_\mathsf{S}.\mathsf{add}((m, h, \tau_\mathsf{Ext}))$
3 : $\textbf{return } h\underline{, \tau_\mathsf{Ext}}$

$\mathsf{O}_\mathsf{Sim}(\mathbb{x})$
1 : $\mathsf{state}, \pi \leftarrow \mathsf{Sim}(\mathsf{state}, \mathbb{x})$
2 : $\mathcal{Q}.\mathsf{add}((\mathbb{x}, \pi))$
3 : $\textbf{return } \pi$
```

Fig. A.2: $\mathsf{NISimBBExtract}$ game: τ_Ext is only returned by $\tilde{\mathsf{O}}_\mathsf{S}(m)$

B Detailed Correctness definitions for f-Blueprint Scheme

Correctness of VerPK **and** $\mathsf{VerEscrow}$: An f-blueprint scheme satisfies correctness of VerPK and $\mathsf{VerEscrow}$, if the algorithm $\mathsf{VerCorrect}^{\mathcal{A}}_{\mathsf{Blu}}(\lambda, x, r_\mathsf{A}, y, r)$ in Figure B.1 always outputs 1.

```
$\mathsf{VerCorrect}_{\mathsf{Blu}}(\lambda, x, r_\mathsf{A}, y, r)$
1 : $cpar \leftarrow \mathsf{CSetup}(1^\lambda)$
2 : $\Lambda \leftarrow \mathsf{Setup}(1^\lambda, cpar)$
3 : $(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_\mathsf{A})$
4 : $C \leftarrow \mathsf{Commit}_{cpar}(y; r)$
5 : $Z \leftarrow \mathsf{Escrow}(\Lambda, \mathsf{pk}_\mathsf{A}, y, r)$
6 : $\textbf{return } [\mathsf{VerPK}(\Lambda, \mathsf{pk}_\mathsf{A}, C_\mathsf{A}) = \mathsf{accept} \wedge \mathsf{VerEscrow}(\Lambda, \mathsf{pk}_\mathsf{A}, C, Z) = \mathsf{accept}]$
```

Fig. B.1: Experiment $\mathsf{VerCorrect}_{\mathsf{Blu}}(\lambda, x, r_\mathsf{A}, y, r)$ with verification on honestly generated escrow.

Correctness of $\mathsf{Decrypt}$: An f-blueprint scheme satisfies correctness of $\mathsf{Decrypt}$, if the algorithm in Figure B.2 has the following output probability:

$$\Pr[\mathsf{DecCorrect}_{\mathsf{Blu}}(\lambda, x, r_\mathsf{A}, y, r) = 1] = 1 - \nu(\lambda)$$

for some negligible function ν and all inputs x, r_A, y, r.

```
$\mathsf{DecCorrect}_{\mathsf{Blu}}(\lambda, x, r_{\mathsf{A}}, y, r)$
1 : $cpar \leftarrow \mathsf{CSetup}(1^{\lambda})$
2 : $\Lambda \leftarrow \mathsf{Setup}(1^{\lambda}, cpar)$
3 : $(\mathsf{pk}_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}}) \leftarrow \mathsf{KeyGen}(\Lambda, x, r_{\mathsf{A}})$
4 : $C \leftarrow \mathsf{Commit}_{cpar}(y; r)$
5 : $Z \leftarrow \mathsf{Escrow}(\Lambda, \mathsf{pk}_{\mathsf{A}}, y, r)$
6 : $m \leftarrow \mathsf{Decrypt}(\Lambda, \mathsf{sk}_{\mathsf{A}}, C, Z)$
7 : $\mathbf{return}\ [m = f(x, y)]$
```

Fig. B.2: Experiment $\mathsf{DecCorrect}_{\mathsf{Blu}}(\lambda, x, r_{\mathsf{A}}, y, r)$ with decryption on honestly generated escrow.

C Deferred Proofs for ElGamal Instantiation

Lemma 11. *Under the decisional Diffie-Hellman assumption, our* HEC *scheme for f_k satisfies correctness with adversarial evaluation randomness.*

Proof. Let $\mathcal{A}$ be any fixed PPT adversary. We can construct the reduction $\mathcal{B}$ to the CPA security of ElGamal as described in Figure A.1. $\mathcal{B}$ creates two polynomials $P_j \leftarrow (\chi - s_0)\prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$, $j \in \{0, 1\}$. 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 Figure C.1. Observe that, regardless of b, $\mathcal{B}$ functions identical to $\mathcal{A}$'s challenger.

Let $\mathsf{extend}(\mathbf{x})$ denote the set $\{y | y \in domain_{f,y} \wedge \mathsf{lobits}_k(y) \in \mathbf{x}\}$. Elements y of $\mathsf{extend}(\mathbf{x})$ have the property that $P(\mathsf{lobits}_k(y)) = 0$, and decrypts to y if it is used in HECEVAL.

Here, we suppose that HECEVAL uses r_Z by partitioning it into r used for randomizing the encrypted polynomial evaluation and r_{Enc} to run Enc on y.

Consider the case where $\mathcal{A}$ succeeds in the HECCORRECT experiment. We know that, for y chosen by $\mathcal{A}$, it is not the case that $\mathsf{lobits}_k(y) \in \mathbf{x}$. Otherwise, $P_b(\mathsf{lobits}_k(y)) = 0$ and HECEVAL's output always decrypts to y no matter the r value chosen. As $s_b > 2^k$, $\mathsf{lobits}_k(y)$ cannot be s_b.

Therefore, y must be a value for which $P_b(\mathsf{lobits}_k(y)) \neq 0$. Additionally, since $\mathcal{A}$ succeeds, it must be that $\mathsf{Decrypt}$ returns something other than $\emptyset$. Therefore, we know that $\mathsf{lobits}_k(P_b(\mathsf{lobits}_k(y))r + y) \in \mathbf{x}$.

Let's denote $\overline{b}$ as $1-b$. Since $P_{\overline{b}}$ is generated independently of the adversary's view, we can imagine a different scenario where $s_{\overline{b}}$ and $P_{\overline{b}}$ are generated after all of $\mathcal{A}$'s responses. Here, $\mathsf{lobits}_k(P_{\overline{b}}(\mathsf{lobits}_k(y))r + y) \in \mathbf{x}$ occurs with the same probability as in our HEC correctness experiment. We claim that this probability is negligible.

```
IND-CPA$_{\mathcal{C},b}(1^\lambda)$
1 : $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
2 : return $\mathcal{C}^{\mathsf{O}_b(\mathsf{pk}_E,\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$

$\mathsf{O}_b(\mathsf{pk}_E, m_0, m_1)$
1 : return $\mathsf{Enc}(\mathsf{pk}_E, m_b)$

$\mathcal{B}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$
1 : $hecpar \leftarrow \mathrm{HECsetup}(\lambda)$
2 : $(f, \mathbf{x}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, hecpar)$
3 : if $f \in F, \mathbf{x} \in domain_{f,\mathbf{x}}$
4 :   $s_0 \leftarrow\$ Z_q \setminus \{0, 1, \ldots, 2^k - 1\}$
5 :   $s_1 \leftarrow\$ Z_q \setminus \{0, 1, \ldots, 2^k - 1\}$
6 :   $P_0 \leftarrow (\chi - s_0)\prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$
7 :   $P_1 \leftarrow (\chi - s_1)\prod_{i=1}^{|\mathbf{x}|}(\chi - x_i)$
8 :   for $i$ in $\{0, \ldots, |x| + 1\}$
9 :     $\mathbf{C}_i \leftarrow \mathsf{O}_b(g^{P_{0,i}}, g^{P_{1,i}})$
10 :  $X \leftarrow (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}|+1})$
11 :  $(y, r_Z) \leftarrow \mathcal{A}(\mathsf{state}_\mathcal{A}, X)$
12 :  if $y \in domain_{f,y}$
13 :    parse $r_Z = (r, r_{\mathsf{Enc}})$
14 :    $y_0 \leftarrow P_0(\mathsf{lobits}_k(y))r + y$
15 :    $y_1 \leftarrow P_1(\mathsf{lobits}_k(y))r + y$
16 :    for $b'$ in$\{1, 0\}$
17 :      if $(\mathsf{lobits}_k(y) \notin \mathbf{x} \cup \{s_{b'}\}) \wedge (\mathsf{lobits}_k(y_{b'}) \in \mathbf{x})$
18 :        return $b'$
19 :    return 0
20 :  return 0
21 : return 0
```

Fig. C.1: Reduction from HEC correctness to IND-CPA

Suppose $\mathcal{A}$ chose values $y, r, \mathbf{x}$. Consider the event E:

$$\begin{aligned} E &= [\mathsf{lobits}_k(P_b(\mathsf{lobits}_k(y))r + y) \in \mathbf{x}] \\ &= [P_b(\mathsf{lobits}_k(y))r + y \in \mathsf{extend}(\mathbf{x})] \end{aligned}$$

There are at most $2^{l_y - k}|\mathbf{x}|$ values for $P_{\overline{b}}(\mathsf{lobits}_k(y))$ such that the equation corresponding to E holds. Fixing $P_{\overline{b}}(\mathsf{lobits}_k(y))$, along with $\mathbf{x}$ also uniquely identifies a polynomial $P_{\overline{b}}$ since $y \notin \mathsf{extend}(\mathbf{x})$. This is because doing so gives us $|\mathbf{x}| + 1$ points on the polynomial, and we know a degree n polynomial can be uniquely identified by n unique points on the polynomial. But we know that there are $q - 2^k$ unique polynomials $P_{\overline{b}}$ can be due to how we generate the root $s_{\overline{b}}$, and each occurs with the same probability. Therefore, the probability that the event E happens is at most: $\frac{2^{l_y - k}|\mathbf{x}|}{q - 2^k}$. We know that the numerator is a product of two values that are polynomial to the security parameter λ, and the denominator is at least 2^k, which is super polynomial in λ. Therefore, the above probability is negligible.

Suppose that $\mathcal{A}$ succeeds with probability $p(\lambda)$. With $b = 0$, $\mathcal{B}$ outputs 1 with probability equal to some negligible $\nu(\lambda)$ as argued above. With $b = 1$, $\mathcal{B}$ outputs 1 with probability $p(\lambda)$. By CPA security of ElGamal, we know that $|p(\lambda) - \nu(\lambda)|$ is also negligible, so p is a negligible function. Therefore, $\mathcal{A}$ succeeds with negligible probability, as required.

```
$\text{IND-CPA}_{\mathcal{B},b}(1^\lambda)$
1 :  $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
2 :  return $\mathcal{B}^{\mathsf{O}_b(\mathsf{pk}_E,\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$

$\mathcal{B}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$
1 :  $hecpar \leftarrow \text{HECsetup}(\lambda)$
2 :  $(f, \mathbf{x}_0, \mathbf{x}', \mathsf{state}) \leftarrow \mathcal{A}(hecpar)$
3 :  $s_0 \leftarrow\$ \mathbb{Z}_q \setminus \{0, 1, \ldots, 2^k - 1\}$
4 :  $s_1 \leftarrow\$ \mathbb{Z}_q \setminus \{0, 1, \ldots, 2^k - 1\}$
5 :  $P = (\chi - s_0)\prod_{i=1}^{|\mathbf{x}|}(\chi - \mathbf{x}_i)$
6 :  $P' = (\chi - s_1)\prod_{i=1}^{|\mathbf{x}'|}(\chi - \mathbf{x}'_i)$
7 :  for $i$ in $\{0, |\mathbf{x}| + 1\}$
8 :      $\mathbf{C}_i = \mathsf{O}_b(g^{P_i}, g^{P'_i})$
9 :  $X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, C_{|\mathbf{x}|+1})$
10 : return $\mathcal{A}(hecpar, X, \mathsf{state})$

$\mathsf{O}_b(\mathsf{pk}_E, m_0, m_1)$
1 :  return $\mathsf{Enc}(\mathsf{pk}_E, m_b)$
```

Fig. C.2: Reduction from security of $\mathbf{x}$ to IND-CPA

Lemma 12. *Our ElGamal construction satisfies* ***Security of*** $\mathbf{x}$

Proof. Let $\mathcal{A}$ be any probabilistic polynomial algorithm. We shall construct a reduction $\mathcal{B}$ to CPA security of ElGamal as described in figure C.2. Since ElGamal is CPA secure, we get that:

$$\begin{aligned} \Pr\left[\text{IND-CPA}_{\mathcal{B},0}(1^\lambda) = 0\right] - \Pr\left[\text{IND-CPA}_{\mathcal{B},1}(1^\lambda) = 0\right]| &= \nu(\lambda) \\ \Pr\left[\mathsf{SecX}_0^{\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{SecX}_1^{\mathcal{A}}(\lambda) = 0\right]| &= \nu(\lambda) \end{aligned}$$

for some negligible function ν, as required.

Lemma 13. *Our ElGamal construction achieves* ***Security of*** x ***and*** y ***from third parties***

Proof. Let $\mathcal{A}$ be a fixed p.p.t. algorithm. We shall construct a reduction $\mathcal{B}$ as described in figure C.3. Note that $\mathcal{B}$ acts identically as $\mathcal{A}$'s challenger in the SecXY game in Figure 4.1. In particular, X is computed identically as how $\mathcal{A}$'s challenger would, and we claim that Z is distributed identically as if $\mathcal{A}$'s challenger would have computed it.

The distribution of Z in our reduction $\mathcal{B}$ is:

$$\begin{aligned} &\left\{(\alpha,\beta,r) \leftarrow\$\ \mathbb{Z}_q^3 : (g^{\alpha r}, g^{P_b(\mathsf{lobits}_k(y_b))r} h^{\alpha r}) \oplus (g^\beta, g^y h^\beta)\right\} \\ =&\left\{(\alpha,\beta,r) \leftarrow\$\ \mathbb{Z}_q^3 : (g^{\alpha r+\beta}, g^{P_b(\mathsf{lobits}_k(y_b))r+y} h^{\alpha r+\beta})\right\} \\ =&\left\{(\alpha,\beta,r) \leftarrow\$\ \mathbb{Z}_q^3 : (g^{\beta}, g^{P_b(\mathsf{lobits}_k(y_b))r+y} h^{\beta})\right\} \end{aligned}$$

since β is sampled uniformly over $\mathbb{Z}_q$ and so is the distribution of $\beta + \alpha r$.

Let r_i denote the randomness used to encrypt $\mathbf{C}_i$ for X_b. The distribution of Z in SecXY can be expressed as:

$$\left\{(\beta,r) \leftarrow\$\ \mathbb{Z}_q^2 : (g^{(\sum_{i=0}^{|\mathbf{x}|+1} r_i \mathsf{lobits}_k(y)^i)^r+\beta}, g^{(\sum_{i=0}^{|\mathbf{x}|+1} P_i \mathsf{lobits}_k(y)^i)^r+y} h^{(\sum_{i=0}^{|\mathbf{x}|+1} r_i \mathsf{lobits}_k(y)^i)^r+\beta})\right\}$$

And since β is sampled uniformly from $\mathbb{Z}_q$, then so is the distribution of $\beta + (\sum_{i=0}^{|\mathbf{x}|+1} r_i \mathsf{lobits}_k(y)^i)r$. So we get:

$$\begin{aligned} &\left\{(\beta,r) \leftarrow\$\ \mathbb{Z}_q^2 : (g^\beta, g^{(\sum_{i=0}^{|\mathbf{x}|+1}(P_b)_i \mathsf{lobits}_k(y)^i)r+y} h^\beta)\right\} \\ &\left\{(\beta,r) \leftarrow\$\ \mathbb{Z}_q^2 : (g^\beta, g^{P_b(\mathsf{lobits}_k(y))r+y} h^\beta)\right\} \end{aligned}$$

which is identical to our distribution of Z in $\mathcal{B}$.

Since $\mathcal{B}$ is identical to $\mathcal{A}$'s challenger, we get:

$$\begin{aligned} \left|\Pr\left[\text{IND-CPA}_{\mathcal{B},0}(1^\lambda) = 0\right] - \Pr\left[\text{IND-CPA}_{\mathcal{B},1}(1^\lambda) = 0\right]\right| &= \nu(\lambda) \\ \left|\Pr\left[\mathsf{SecXY}_0^{\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{SecXY}_1^{\mathcal{A}}(\lambda) = 0\right]\right| &= \nu(\lambda) \end{aligned}$$

as required.

```
IND-CPA$_{\mathcal{C},b}(1^\lambda)$
    $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
    **return** $\mathcal{C}^{\mathsf{O}_b(\mathsf{pk}_E,\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$

$\mathsf{O}_b(\mathsf{pk}_E, m_0, m_1)$
    **return** $\mathsf{Enc}(\mathsf{pk}_E, m_b)$

$\mathcal{C}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, \mathsf{pk}_E)$
    $hecpar \leftarrow \text{HECsetup}(1^\lambda)$
    $(f, \mathbf{x}, \mathbf{x}_1, \mathsf{state}) \leftarrow \mathcal{A}(hecpar)$
    **if** $f \in F, \mathbf{x}_0, \mathbf{x}_1 \in domain_{f,\mathbf{x}}$
        $s_0 \leftarrow\!\!\$\ \mathbb{Z}_q \setminus \{0, 1, \ldots, 2^k - 1\}$
        $s_1 \leftarrow\!\!\$\ \mathbb{Z}_q \setminus \{0, 1, \ldots, 2^k - 1\}$
        $P_0 \leftarrow (\chi - s_0)\prod_{i=1}^{|\mathbf{x}|}(\chi - \mathbf{x}_{0,i})$
        $P_1 \leftarrow (\chi - s_1)\prod_{i=1}^{|\mathbf{x}|}(\chi - \mathbf{x}_{1,i})$
        **for** $i$ **in**$\{0, |\mathbf{x}_0| + 1\}$
            $\mathbf{C}_i \leftarrow \mathsf{O}_b(g^{P_{0,i}}, g^{P_{1,i}})$
        $X \leftarrow (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}_0|+1})$
        $(y_0, y_1, \mathsf{state}) \leftarrow \mathcal{A}(X, \mathsf{state})$
        **if** $y_0, y_1 \in domain_{f,y}$
            $eval \leftarrow \mathsf{O}_b(\mathsf{pk}_E, g^{P_0(\mathsf{lobits}_k(y_0))}, g^{P_1(\mathsf{lobits}_k(y_1))})$
            $enc \leftarrow \mathsf{O}_b(\mathsf{pk}_E, y_0, y_1)$
            $r \leftarrow\!\!\$\ \mathbb{Z}_q$
            $Z \leftarrow eval^r \oplus enc$
            **return** $\mathcal{A}(Z, \mathsf{state})$
        **return** $\mathcal{A}(\perp, \mathsf{state})$
    **return** $\mathcal{A}(\perp, \mathsf{state})$
```

Fig. C.3: Reduction from security of XY to IND-CPA

Lemma 14. *Our ElGamal construction achieves* ***Security of*** *DirectZ.*

Proof. Here, we can show that Z_0 and Z_1 in the experiment in Figure 4.1 are identically distributed, regardless of the value of X and $\mathbf{x}$. From the proof in

Theorem 13, we get that the distribution of HECeval is as follows:

$$\left\{(\beta, r) \leftarrow\!\$\ \mathbb{Z}_q^2 : (g^\beta, g^{(\sum_{i=1}^{|\mathbf{x}|+1} P_i \mathsf{lobits}_k(y)^i) r + y} h^\beta)\right\}$$
$$\left\{(\beta, r) \leftarrow\!\$\ \mathbb{Z}_q^2 : (g^\beta, g^{P(\mathsf{lobits}_k(y)) r + y} h^\beta)\right\}$$

Similarly, we get that the distribution of HECdirect when $\mathsf{lobits}_k(y) \in X$ is as follows:

$$\left\{(\beta) \leftarrow\!\$\ \mathbb{Z}_q^2 : (g^\beta, g^{f(\mathbf{x}, y)} h^\beta)\right\}$$

And otherwise, it is an encryption of a random value, distributed as:

$$\left\{(\alpha, \beta) \leftarrow\!\$\ \mathbb{Z}_q^2 : (g^\beta, g^\alpha h^\beta)\right\}$$

Now suppose we fix a y value. If $\mathsf{lobits}_k(y) \in X$, then $f(\mathbf{x}, y) = y$, and $P(\mathsf{lobits}_k(y)) = 0$, so we get that both distributions are uniform over all ElGamal encryptions of g^y.

Otherwise, if $\mathsf{lobits}_k(y) \notin X$, then $f(\mathbf{x}, y) = \emptyset$. In DirectZ$_0$, we know that $P(\mathsf{lobits}_k(y)) \neq 0$ since $\mathsf{lobits}_k(y)$ cannot be one of the $|\mathbf{x}| + 1$ roots of $\mathbf{x}$. This is because we explicitly restrict the root to $\mathbb{Z}_q \setminus \{0..2^k - 1\}$ which excludes the domain of lobits_k. We know that r is chosen uniform randomly, so $P(\mathsf{lobits}_k(y))r + y$ is also uniformly distributed over $\mathbb{Z}_q$ and the two distributions are identical.

D Technical Details of the NIZK proof systems for ElGamal Instantiation of Generic *f*-blueprint Scheme

Recall that in our Generic f-Blueprint scheme from HEC in Figure 5.1, we need a BB-extractible NIZK system for the following relation for Ψ_1:

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

In our instantiation found in Figure 6.1, in HECEnc we first compute an ElGamal key pair using KGen. Then, we sample some $s \in \mathbb{Z}_q \setminus \{0, 1, \ldots, 2^k - 1\}$. Finally, we encrypt coefficients of the polynomial $P = (\chi - s) \prod_{i=1}^{|\mathbf{x}|} (\chi - x_i)$.

Recall also that we assume the commitment C_{A} contains commitments to coefficients of $P' = \prod_{i=1}^{|\mathbf{x}|} x_i$. More formally, let P'_i denote the i^{th} coefficient of P', Then we have commitments $C_{\mathsf{A}} = \mathsf{Commit}_{cpar}(P'_0, P'_1, \ldots, P'_{|x|+1}; r_{\mathsf{A}})$.

Let Ψ be a NIZK proof system for R_{eqrep}, we can construct Ψ_1 with P_{Ψ_1} as described in Figure D.1. We assume r_X can be split into r_E used in generating pk_E and r_s used in generating the additional root s to the polynomial, and r_i for $i \in 0, \ldots, |\mathbf{x}|$ used in encrypting $\mathbf{C}_i$. In Ψ_1, we have a commitment to P', but have to prove that we have encrypted $(\chi - s)P'$. We can express $P_i = P'_{i-1} - sP'_i$, which gives us $P'_{i-1} = P_i + sP'_i$ that we can use to compute encryptions to coefficients of P'. We use this to compute an encryption of P' from the encryption of P and the root s and publicize it (effectively including it in the proof), along with proof that it was computed from encrypted coefficients of P. This does not give away information about s, otherwise we would violate the DDH assumption. Observe that, other than proving $s \geq 2^k$, which can be done using Bulletproofs, all other components of our proof system involves proof of equivalent discrete logarithm representation.

```
$\mathsf{P}^{\mathsf{S}}_{\Psi_1}((hecpar, f, cpar, X, C_{\mathsf{A}}), (\mathbf{x}, d, r_X, r_{\mathsf{A}}))$
1 :  parse $X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}|+1})$
2 :  parse $r_X = (r_E, r_s)$
3 :  parse $cpar = (g_0, \ldots, g_{|\mathbf{x}|}, H)$
4 :  $e \xleftarrow{r_E} \mathbb{Z}_q$
5 :  $(\mathsf{pk}_E, \mathsf{sk}_E) \leftarrow \mathsf{KGen}(hecpar; r_E) = ((g, h = g^e), e)$
6 :  $s \xleftarrow{r_s} \mathbb{Z}_q \setminus \left\{0, \ldots, 2^k - 1\right\}$
7 :  $P' = \prod_{i=1}^{|\mathbf{x}|}(\chi - \mathbf{x}_i)$
8 :  $P = (\chi - s)P'$
9 :  $\mathbf{C}'_{|\mathbf{x}|} = \mathbf{C}_{|\mathbf{x}|+1}$
10 : $r'_{|\mathbf{x}|} = r_{|\mathbf{x}|+1}$
11 : for $i \in \{|\mathbf{x}| - 1, \ldots, 0\}$
12 :     $\mathbf{C}'_i = \mathbf{C}_{i+1} \oplus (\mathbf{C}'_{i+1})^s$
13 :     $r'_i = r_{i+1} + s \cdot r_{i+1}$
14 : publish $(\mathbf{C}'_0, \ldots, \mathbf{C}'_{|X|})$
15 : $\pi \leftarrow \mathsf{PoK}_{\Psi}\Big\{\mathbb{w} = (P'_0, \ldots, P'_{|\mathbf{x}|}, r_0, \ldots, r_{|\mathbf{x}|+1}, r'_0, \ldots, r'_{|\mathbf{x}|}, s, r_{\mathsf{A}}, e) :$
16 :     $h = g^e \wedge s \geq 2^k \wedge C_{\mathsf{A}} = H^{r_{\mathsf{A}}} \prod_{i=0}^{|\mathbf{x}|} g_i^{P'_i} \wedge$
17 :     $\mathbf{C}'_i = (g^{r'_i}, g^{P'_i} h^{r'_i})$ for $i \in \{0, \ldots, |\mathbf{x}|\} \wedge$
18 :     $\mathbf{C}_0 = (\mathbf{C}'_0)^s \wedge \mathbf{C}_i = \mathbf{C}'_{i-1} \oplus (\mathbf{C}'_i)^{-s}$ for $i \in \{1, \ldots, |\mathbf{x}|\}$
19 : $\Big\}$
20 : return $\pi$, any published values
```

Fig. D.1: Instantiation of Ψ_1 for ElGamal

Recall that Ψ_2 is used to prove the following relation:

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

In our instantiation found in Figure 6.1, in HEC<span style="font-variant:small-caps">eval</span> we first evaluate our private input y on the encrypted polynomial $\mathbf{C}$ to obtain $eval$. Then, we encrypt g^y to enc, and sample $r \leftarrow\!\!\$\ \mathbb{Z}_q$. Finally, we output $eval^r \oplus enc$. Our proof system Ψ_2, detailed in Figure D.2, also goes through its steps in this order, and again utilizes Ψ for R_{eqrep}. The verifier $\mathsf{V}^{\mathsf{S}}_{\Psi_2}$ works by first verifying the proof π, then verify:

$$\hat{Z} = (t^{-U} X \prod_{i=0}^{|x|+1} G_i^{AR}, t^{-U'} YZ \prod_{i=0}^{|x|+1} H_i^{AR})$$

And that (U, O) and (U', O') are openings to $\mathcal{X} \prod_{i=0}^{|x|+1} \mathcal{R}_i$ and $\mathcal{Y}\mathcal{Z} \prod_{i=0}^{|x|+1} \mathcal{R}'_i$ respectively, both under base (g, h).

Let's walk through P_{Ψ_2}. We deal with proving $a = \mathsf{lobits}_k(y)$ by first proving $a < 2^k$, then $a \equiv y \bmod 2^k$, which is captured by $1 = g^{y-a}(g^{2^k})^{-m}$. Next, the idea is that we are computing Pedersen commitments A_i^R to $a^i r$, that can later be used to compute commitments to components of $\mathbf{C}_i^{a^i r}$ composed with some added noise $(t^{\rho_i}, t^{\rho'_i})$. Additionally, we compute Pedersen commitments X, Y, and Z where (X, YZ) correspond to $\mathsf{Enc}(\mathsf{pk}_E, y; r_{\mathsf{Enc}})$ composed with some noise $(t^{\gamma_X}, t^{\gamma_Y + \gamma_Z})$. After line 24, we reveal the cumulated noise's exponents U and U', which allows $\hat{Z}$ to be revealed. This still perfectly hides our witness, since a Pedersen commitment is hiding even if part of the opening is known. To prove that the U and U' we reveal are indeed the cumulated noise, we generate Pedersen commitments of the old noise captured in U and U', and reveal the cumulative new noise O and O', and we can verify that (U, O) and (U', O') are openings to the composition of the new Pedersen commitments.

When we refer to *published commitments correctly computed* in PoK, we are indicating that whenever P runs $\mathsf{publish}$ on a commited value of the form $A \leftarrow \mathsf{Commit}_{B,C}(D, E)$, we include the line $A = \mathsf{Commit}_{B,C}(D; E)$ in the PoK statement on Line 38. In most of our calls to Commit, the arguments are all either public or members of the witness, so we can directly including them in the PoK statement. There is a slight nuance involving proving that G_i^{AR} and H_i^{AR} are correctly computed. Here, we know that $A_i^R = \mathsf{Commit}_{A_i,h}(r'; \beta_i)$ can be expressed as $A_i^R = \mathsf{Commit}_{g,h}(a^i r; \beta_i + \sum_{j=0}^{i} \alpha_j)$, which can be proven by induction on i. We can instead construct a sigma protocol for equality of representation of two Pedersen commitments, which also reduces to an instance of R_{eqrep} and therefore can be included in our proof.

Apart from the one range proof we need to use at the start to ensure a is within range, the rest of the statements within the proof are instances of eqrep. Therefore, this construction gives us a $f(J, \cdot)$-BB-PSL simulation extractable NIZK proof system by Theorem 1.

```
$\mathsf{P}^{\mathsf{S}}_{\Psi_2}((\hat{Z}, hecpar, f, X, C, cpar), (y, r, r_{\hat{Z}}))$
parse $r_{\hat{Z}} = (r', r_{\mathsf{Enc}})$
parse $X = (\mathsf{pk}_E, \mathbf{C}_0, \ldots, \mathbf{C}_{|\mathbf{x}|+1})$
parse $\mathsf{pk}_E = (g, h)$
$t \leftarrow\!\!\$\ \mathbb{G}$
$(\alpha_0, \ldots, \alpha_{|x|+1}, \beta_0, \ldots, \beta_{|x|+1}$
$\rho_0, \ldots, \rho_{|x|+1}, \rho'_0, \ldots, \rho'_{|x|+1}$
$\delta_0, \ldots, \delta_{|x|+1}, \delta'_0, \ldots, \delta'_{|x|+1}$
$\gamma_X, \gamma_Y, \gamma_Z, \delta_X, \delta_Y, \delta_Z) \leftarrow\!\!\$\ \mathbb{Z}_q^{4|x|+10}$
$a \leftarrow \mathsf{lobits}_k(y)$
$m \leftarrow \lfloor \frac{y}{2^k} \rfloor$
publish $R \leftarrow \mathsf{Commit}_{g,h}(r')$
publish $A_0 \leftarrow \mathsf{Commit}_{g,h}(1; \alpha_0)$
publish $A_0^R = \mathsf{Commit}_{g,h}(r'; \beta_0)$
for $i \in \{1, \ldots, |x| + 1\}$
    publish $A_i = \mathsf{Commit}_{A_{i-1},h}(a; \alpha_i)$
    publish $A_i^R = \mathsf{Commit}_{A_i,h}(r'; \beta_i)$
for $i \in \{1, \ldots, |x| + 1\}$
    parse $\mathbf{C}_i = G_i, H_i$
    publish $G_i^{AR} \leftarrow \mathsf{Commit}_{G_i,t}(a^i r'; \rho_i)$
    publish $H_i^{AR} \leftarrow \mathsf{Commit}_{H_i,t}(a^i r'; \rho'_i)$
publish $X \leftarrow \mathsf{Commit}_{g,t}(r_{\mathsf{Enc}}; \gamma_X)$
publish $Y \leftarrow \mathsf{Commit}_{g,t}(y; \gamma_Y)$
publish $Z \leftarrow \mathsf{Commit}_{h,t}(r_{\mathsf{Enc}}; \gamma_Z)$

$\mathsf{P}_{\Psi_2}$ continued
publish $U \leftarrow \gamma_X + \sum_{i=0}^{|x|+1} \rho_i$
publish $U' \leftarrow \gamma_Y + \gamma_Z + \sum_{i=0}^{|x|+1} \rho'_i$
for $i \in \{1, \ldots, |x|\}$
    publish $\mathcal{R}_i \leftarrow \mathsf{Commit}_{t,g}(\rho_i, \delta_i)$
    publish $\mathcal{R}'_i \leftarrow \mathsf{Commit}_{t,g}(\rho'_i, \delta'_i)$
publish $\mathcal{X} \leftarrow \mathsf{Commit}_{t,g}(\gamma_X, \delta_X)$
publish $\mathcal{Y} \leftarrow \mathsf{Commit}_{t,g}(\gamma_Y, \delta_Y)$
publish $\mathcal{Z} \leftarrow \mathsf{Commit}_{t,g}(\gamma_Z, \delta_Z)$
publish $O = \delta_X \prod_{i=0}^{|x|+1}$
publish $O' = \delta_Y \delta_Z \prod_{i=0}^{|x|+1}$
$\pi \leftarrow \mathsf{PoK}_{\Psi} \Big\{ \mathrm{w} = (y, r_{\mathsf{A}}, r',$
$r_{\mathsf{Enc}}, a, m,$ and all greek letters) :
    $a < 2^k \wedge 1 = g^{y-a}(g^{2^k})^{-m} \wedge$
    $C = \mathsf{Commit}_{cpar}(y; r_{\mathsf{A}}) \wedge$
    published commitments correctly computed
$\Big\}$
return $\pi$, any published values
```

Fig. D.2: Instantiation of Ψ_2 for ElGamal

E Achieving strongly correct FHE

The correctness requirement as stated in Definition 6 is stronger than standard correctness. Namely, we need to ensure that, for the specific circuit Φ used in our construction, the probability that the adversary can find random coins R for FHEKeyGen, and inputs $x_1, \ldots, x_n$, and randomness $r_1, \ldots, r_n$ for FHEEnc such that $\mathsf{FHEDec}(\mathit{FHESK}, C) \neq \Phi(x)$ where $\mathit{FHEPK}, \mathit{FHESK}) = \mathsf{FHEKeyGen}(1^\lambda; R)$, $c_i = \mathsf{FHEEnc}(\mathit{FHEPK}, x_i; r_i)$ and $C = \mathsf{FHEval}(\mathit{FHEPK}, \Phi, c_1, \ldots, c_n)$, is 0, i.e., the scheme is perfectly correct. Known constructions of FHE either have perfect correctness or can be adapted to achieve it by bounding the amount of random noise, and therefore stating correctness this way is justifiable.

If we allow a negligible probability that an adversary can make a decryption error happen, then we get another correctness notion, let us refer to it as *strong correctness*. It is easy to see that strong correctness is good enough for our construction of HEC from CP-FHE: Strong correctness ensures that even in the event when the adversary controls the choice of the keys and input ciphertexts, as long as he is following the protocol, he cannot (with more than negligible probability) create a situation in which decryption is incorrect; and therefore an incorrect decryption event cannot leak information about the inputs.

Regular (not perfect or strong) correctness requires only that a decryption error occurs negligibly often when keys are ciphertexts are selected honestly. In the common-random-string model, strong correctness can be achieved from regular correctness by standard techniques, which we describe here. Here is a standard technique first seen in Naor's commitment scheme. Let the circuit Φ be fixed. Suppose that for a random public key and a random set of n ciphertexts, the probability of incorrect decryption is $p < 1/10$. By the Chernoff bound, repeated encryption and taking the majority when decrypting, we can transform this cryptosystem into one whose probability (over the choice of randomness for the public key and the set of input ciphertexts) of incorrect decryption is $2^{-(n+2)(\lambda+1)}$.

Let the common random string consist of $n+1$ random strings $\rho_0, \rho_1, \ldots, \rho_n$. ρ_0 has length $\ell_{\mathsf{FHEKeyGen}}$, where $\ell_{\mathsf{FHEKeyGen}}$ is an upper bound on the number of random bits needed for $\mathsf{FHEKeyGen}$. For $1 \leq i \leq n$, ρ_i has length ℓ_{FHEEnc}, where ℓ_{FHEEnc} is an upper bound on the number of random bits needed for FHEEnc.

In order to generate a public key, sample a λ-bit random seed s_0, and let $R = G(s_0) \oplus \rho_0$; output $(\mathit{FHEPK}, \mathit{FHESK}) = \mathsf{FHEKeyGen}(1^\lambda; R)$ where G is a PRG with appropriate lengths. In order to generate a the i^{th} ciphertext, sample a λ-bit random seed s_i, and let $r_i = G'(s_i) \oplus \rho_i$; output $c_i = \mathsf{FHEEnc}(\mathit{FHEPK}, x_i; r_i)$ where G' is a PRG with appropriate lengths. Note that there are only $2^{(n+1)(\lambda+1)}$ ways of setting these random strings, and for each of them the probability that truly random $\rho_i's$ are chosen so as to lead to a decryption error is $2^{-(n+2)(\lambda+1)}$ (since that's the probability of a decryption error). Thus, by the union bound, the probability that there exists a way to set the randomness for key generation and ciphertexts so as to lead to a decryption error is $2^{(n+1)(\lambda+1)}2^{-(n+2)(\lambda+1)} = 2^{-\lambda-1}$, and the resulting FHE is strongly correct.