New Constructions of Functional Adaptor Signatures: Broader Functions and Improved Efficiency*†

Nikhil Vanjani‡
Carnegie Mellon University
nikhilvanjani61@gmail.com

Garrett Greiner‡
University of Utah
garrett.greiner@utah.edu

Sri AravindaKrishnan Thyagarajan
University of Sydney
aravind.thyagarajan@sydney.edu.au

Pratik Soni
University of Utah
psoni@cs.utah.edu

Abstract

Functional adaptor signatures (FAS) are a novel cryptographic primitive introduced at CCS'24 that enable privacy-preserving, fine-grained data-payment exchanges between a seller and a buyer in a trustless and atomic manner. In this setup, the seller holds sensitive data x (e.g., patient records, climate data), and the buyer specifies a function f (e.g., aggregate, sum). FAS guarantees that the buyer learns $f(x)$ (and nothing beyond) if and only if the seller receives payment in blockchain-based tokens. Unlike generic smart contracts, FAS-powered solutions excel in privacy, efficiency, and compatibility with diverse blockchain systems. However, prior FAS constructions were limited to linear functions (where f was linear in x), restricting their applicability to more complex and prevalent applications including data analytics and ML model evaluations.

In this work, we extend the capabilities of FAS to support higher-degree functions ($deg \geq 2$), significantly broadening its range of applications. Our core contribution is a novel FAS design leveraging *homomorphic encryption*, which simultaneously achieves enhanced efficiency and compatibility for general functions. This approach diverges fundamentally from the restricted design in CCS'24 which relied on connections to functional encryption. We implement our homomorphic encryption-based FAS for functions arising in applications such as data analytics and machine learning inference. Remarkably, even for linear functions, our new design achieves an order-of-magnitude improvement in performance compared to CCS'24 constructions. Furthermore, our solutions seamlessly integrate with prominent blockchain systems, requiring only a basic signature verification script on standard transactions, thus ensuring practical deployability. As a conceptual contribution, we introduce the general paradigm of a blockchain-based *functional fair exchange* (FFE) protocol, rigorously define buyer and seller fairness, and show that FAS implies the general goal of FFE.

*A preliminary version of this submission appeared at IEEE S&P 2026, this is the full version.

†After the submission of the IEEE S&P 2026 camera-ready version, we identified an issue in the FHE-based instantiation. Section 2.2 describes the issue, its implications, and the changes made in this full version.

‡Equal Contribution.

Contents

1 Introduction

Modern data collaborations often require extracting value from sensitive datasets without exposing them directly. In many settings, a party may wish to obtain specific insights or perform computations on private data held by another party—but only if they can verify the result and fairly compensate the data holder. Ensuring such a fair exchange without relying on trust or revealing the data remains a core challenge, especially in decentralized environments.

Application 1: Statistics on database. A buyer wants advanced statistical insights-such as sum, average, variance, or second-order polynomial terms-derived from sensitive data held by a seller. These metrics arise in real-world use cases: computing salary variance in a company, measuring the covariance between age and income, or modeling treatment outcomes with respect to dosage. Computing such valuable information often involves complex functions ($deg > 1$) and must be exchanged without revealing the underlying data.

Application 2: Fine-Tuning ML Models. A buyer holds a pre-trained ML model—say, a *Stochastic Gradient Descent* (SGD)-based breast cancer classifier [Ama93]—and seeks to fine-tune it using a high-quality, sensitive dataset owned by a seller (e.g., a medical institution). The model must be refined using the seller's data, without exposing that data to the buyer. Similar needs arise in private inference, image classification, encrypted search, and collaborative training tasks.

These examples reflect a common objective: to atomically exchange the outcome of a computation on private data in return for payment, without disclosing the data itself, and without relying on trust. More formally, the seller holds sensitive input x (e.g., a database), and the buyer specifies a function f (e.g., a statistical query or model inference). The buyer learns $f(x)$ only if the seller receives payment. We refer to this capability as an atomic *functional sale of information*, which is increasingly relevant in data marketplaces, collaborative analytics, and privacy-preserving machine learning.

To enable such exchanges with privacy and minimal trust, Vanjani, Soni, and Thyagarajan [VST24a] introduced *functional adaptor signatures (FAS)*—a cryptographic mechanism that generalizes *adaptor signatures* [Poe17, AEE+20, EFH+21, DOY22, GSST24] to support selling functions of secrets. FAS enables atomic exchanges over blockchains *without requiring smart contracts*, relying only on standard signature verification.

The process, described in Figure 1, begins with the seller publicly posting an FAS advertisement ad, announcing the availability of some functional information about the secret data x. The buyer, after choosing f, locks the payment in a shared buyer-seller address (e.g., a 2-out-of-2 multisig address) on the blockchain. To proceed, the buyer generates a *functional pre-signature* $\tilde{\sigma}$ on a payment transaction m, which would release the payment to the seller. FAS ensures that the seller can *adapt* $\tilde{\sigma}$ into a valid buyer signature σ on m only if the seller possesses the secret data x. By posting m, the adapted signature σ, and their own signature (as required for the shared address), the seller receives the payment. Finally, FAS enables the buyer to *functional extract* $f(x)$ using $\tilde{\sigma}$ and σ, completing the fair exchange.

It is important to note that for this functional information exchange, we do *not* require *any* complex smart contracts. Specifically, the exchange is executed through a simple on-chain transaction verified using standard signature verification which is a necessary minimal check to authenticate transactions. This design aligns with previous efforts [Poe17, AEE+20, EFH+21, GMM+22, AMSKM21, DOY22, ATM+22, GSST24, TM21, TBM+20, TMMS22, MTV+22] to improve on-chain privacy, scalability, cost efficiency, and blockchain interoperability. Moreover, the FAS constructions proposed in [VST24a] are compatible with widely used signatures such as Schnorr, ensuring seamless integration with existing blockchain systems without modifications to transaction validation.

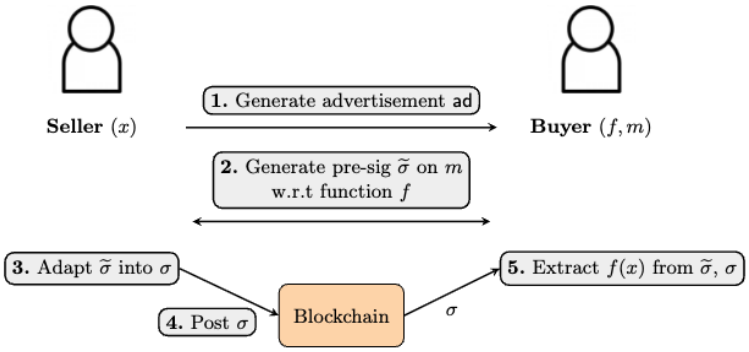

Figure 1: Fair exchange for learning a function f over seller's secret input x via FAS [VST24a].

Drawback of FAS constructions from [VST24a]. Unfortunately, all FAS constructions presented in [VST24a] can only support *linear* function evaluations: The function f chosen by the buyer has to be linear (i.e., $deg(f) = 1$) in the input data x. This confines applications to selling simple statistics such as sums, averages, or linear kernels, but *excludes* widely used degree-2 computations—such as variance, covariance, or fine-tuning machine learning models (i.e., where $deg(f) \geq 2$). Given the practical appeal of FAS in trustless information exchange, it is important to extend its capabilities to support higher-degree function evaluations, which are increasingly central in modern data analysis and workflows.

1.1 Our Contributions

In this work, we make a substantial leap in the capabilities of functional adaptor signatures by introducing the *first* constructions that support *non-linear* and expressive function classes. We present *three* new secure FAS constructions: one that significantly improves efficiency for linear functions, and two that extend FAS to handle degree-2 and higher computations for the first time. These constructions follow from a general framework that combines *homomorphic encryption* (HE) with Schnorr adaptor signatures in a technically novel and modular way, expanding the design space of FAS beyond prior work. To demonstrate deployability, we undertake a significant implementation effort that required around 10,000 LoC across several cryptography libraries including PBC [Lyn] and Zama's suite of *fully homomorphic encryption* (FHE) libraries [tfh, cona, conb]. Finally, we benchmark our constructions across *five* application settings – database analytics, fine-tuning SGD classifier, ML inference through second-order factorization machines, and high-degree multivariate polynomial evaluation. Specifically, we have the following contributions:

Contribution 1. We propose a novel construction paradigm that builds FAS for any function class F while relying on HE for the same class, Schnorr adaptor signatures, and a new *cut-and-choose* style sigma protocol that proves plaintext equality across HE ciphertexts and Elgamal PKE ciphertexts.[1] This framework, described in Section 5, represents our core conceptual contribution, offering a simpler and more intuitive alternative to the *functional encryption* (FE)-based framework proposed in [VST24a]. By building on extensively implemented HE schemes, our paradigm benefits from practical advancements in HE, addressing the limitations of FE constructions, which remain less efficient and less mature for comparable function classes.

Contribution 2. Instantiating the HE scheme in our template with existing *fully* homomorphic

[1]We require a natural randomness recovery property from our HE scheme, satisfied by several existing schemes: the randomness of a homomorphically evaluated ciphertext can be efficiently derived from the function and the randomness of the input ciphertexts. We formalize this via an algorithm REval.

encryption schemes [BGV11, GSW13] gives us a concrete scheme for FAS for *general functions* (see Section 8). To enable ML applications, we instead directly work with Zama's suite of FHE libraries that implement the Torus-FHE scheme [CGGI20]. However, this choice entailed significant implementation effort due to Zama's limited exposure to low-level FHE kernels. Specifically, for seller efficiency, our FAS construction requires accessing the random coins behind homomorphically evaluated ciphertexts, which unfortunately is inaccessible in the current implementation of TFHE-rs library [tfh]. We implement this feature (8000+ LoC), which may be of independent interest. All of our code is available at [fas].

As benchmark applications for general functions, we focus exclusively on ML-based applications. Firstly, we consider the setting where a buyer wants to learn high-degree polynomial evaluations of seller's secret input (Section 8.1). For a host of ML inference tasks that use transformers and neural network models, the underlying non-linear computations such as softmax and GeLU are often approximated through high-degree multivariate polynomials: degree 4 approximation of softmax [CZK+23], and degree 3 approximation of GeLU [ZYH+24, LHG+23, DLZ+23]. Given the prevalence of such high-degree polynomials in ML inference tasks, we believe polynomial evaluation is a representative application for FAS. Secondly, we demonstrate (in Section 8.1) the applicability of our FHE-based FAS scheme for fine-tuning a low-quality SGD classifier (Application #2 in Section 1).

Contribution 3. We explore the simpler case of degree-2 functions, which frequently arise in real-world applications, particularly in medical analytics and machine learning. For instance, analytics queries in medical settings can be encoded as weighted standard deviation functions, where the buyer specifies weights $w_1, \ldots, w_\ell$ for the seller's database $\vec{x}$, and computes the standard deviation of weighted samples $w_i \cdot \vec{x}[i]'s$. Similarly, inference using second-order factorization machines [Ren10]—a key method in machine learning to predict first- and second-order feature correlations—can also be cast as degree-2 functions.

To enable these applications, we optimize our construction by leveraging the Boneh-Goh-Nissim [BGN05] HE scheme, which supports degree-2 homomorphism. Additionally, we develop a *new* zero-knowledge proof of plaintext equality for Elgamal and BGN ciphertexts, generalizing the work of Chase et al. [COPZ22] to composite-order groups used in the BGN scheme. These optimizations target practical implementations, particularly for medical analytics (e.g., weighted standard deviation computations) and secure sales of second-order factorization machine inferences which are discussed in Section 8.2.

Contribution 4. To further demonstrate the power of our new HE-based template, we revisit the case of linear functions. In addition to rigid algebraic structural requirements between *functional keys* of the underlying functional encryption, the functional extraction procedure in [VST24a] involves an inefficient discrete-logarithm computation. In contrast, our construction outperforms the constructions in [VST24a] by up to 10x in total run time (as shown in Figure 2) for the same application benchmark when instantiated with a linearly homomorphic encryption scheme, where the decryption operation only involves group multiplication (and not the expensive discrete-logarithm computation).

Contribution 5. As an additional conceptual contribution, we formalize the notion of *fair functional exchange* as a cryptographic interactive protocol involving the seller, the buyer, and a wallet-based token system (Section 9). We rigorously define *buyer-fairness* and *seller-fairness* using the simulation-based framework. Our definitions are strictly more general in that they capture functional exchanges, compared to the ones in [BGKS25, TSZ+24] which only handle simple all-or-nothing payment-data exchanges. While our FAS constructions imply fair functional exchange (as we demonstrate), we deliberately decouple this abstraction from FAS to define a concrete and standalone target. This separation opens the door to future protocols that achieve fair functional exchange directly, rather than through FAS machinery.

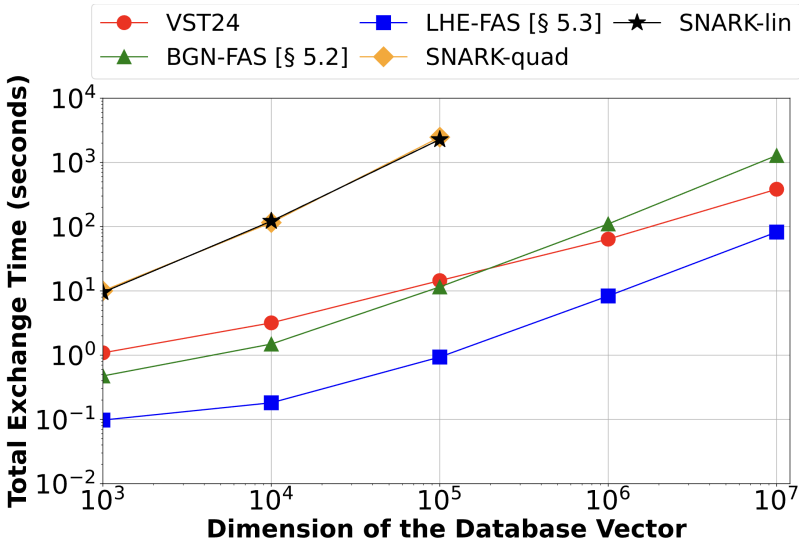

Figure 2: Runtime comparison for VST24 and our LHE-FAS (weighted sums), our BGN-FAS (weighted standard deviation), and SNARK-based strawmans over databases of varying dimensions with weights ≤ 1000.

2 Technical Overview

This section outlines our techniques, starting with the following setup: Let $\vec{x}$ be a ℓ-dimensional secret integer vector that the seller holds and the buyer wishes to purchase the output of an integer-valued function $f(\vec{x})$. Here we assume $\vec{x}$ satisfies a publicly known (NP) relation $R(X, \vec{x})$ for some public value X. Our goal is to design a protocol where the buyer learns $f(\vec{x})$ if and only if the seller receives payment, and neither party can cheat the other.

Strawman Construction of FAS. Adaptor signatures [Poe17] are a widely used alternative to *hash timelock contracts (HTLC)* for fair payments. For instance, Schnorr adaptor signatures [EFH+21] facilitate the sale of secret keys (e.g., discrete logarithms) of Elgamal public-key encryption (PKE) defined over the same group. By combining Schnorr adaptor signatures with generic cryptographic primitives, one can naturally construct FAS for *any* function. This approach is instructive and is captured in the following strawman protocol:

1. **Advertisement Phase:** The seller generates a fresh PKE key pair $(\mathsf{pk}, \mathsf{sk})$ and publishes a ciphertext $\mathsf{CT}_{\mathsf{ad}}$ encrypting $\vec{x}$ under pk. The seller also provides a non-interactive zero-knowledge (NIZK) proof certifying that $\mathsf{CT}_{\mathsf{ad}}$ encrypts a valid witness for X.
2. **Exchange Phase:** Upon receiving the buyer's function f, the seller computes the output $y = f(\vec{x})$ and encrypts it using Elgamal PKE with a fresh (public key, secret key) pair $(K = g^k, k)$, where g is the group generator of a prime-order group $\mathbb{G}$. Along with this ciphertext ct_y, the seller sends a NIZK proof π certifying that ct_y encrypts y derived from the witness encrypted in $\mathsf{CT}_{\mathsf{ad}}$. The buyer verifies π and, if valid, proceeds to purchase the Elgamal secret key for K using a Schnorr adaptor signature. The buyer then decrypts ct_y to obtain y.

The protocol ensures fairness through the semantic security of $\mathsf{CT}_{\mathsf{ad}}$ and the Elgamal PKE scheme, and the zero-knowledge property of NIZKs. The buyer learns y only after completing the Schnorr-based transaction, guaranteeing payment to the seller. The soundness of NIZKs and the extractability of adaptor signatures ensure that the buyer receives y, upholding fairness.

In theory, this strawman protocol can support the sale of arbitrary functions using general-purpose NIZKs. However, in practice, the above strawman protocol is constrained in the exchange

phase, by the inefficiency of the NIZK proof computation by the seller. To quantify this drawback, we implemented a significantly simpler variant of the exchange NIZK by using the Groth16 SNARK proof system in the Arkworks library [ac22].[2] In Figure 2, we plot the total time for the exchange phase for such strawman protocols for the sale of linear and degree-2 functions over varying database sizes (Similar comparison for high-degree functions can be found in Section 8.1.2). Here the weights refer to the specific weighted computation (like weighted standard deviation). Notice that computing the SNARK proof incurs significant overhead for the seller. While optimized NIZKs may be possible for specific functions, we foresee that this method incurs significant overhead on the seller for general functions.

Prior work of Vanjani, Soni, and Thyagarajan [VST24a], henceforth VST24, overcome this challenge by replacing the advertisement PKE with a public-key *functional encryption (FE) scheme* [BSW11], and then re-purposing the online phase to let the buyer buy a "functional" secret-key w.r.t. its function f instead of a regular PKE decryption key (in the strawman approach) through the Schnorr adaptors. However, their construction requires certain algebraic properties from the underlying FE scheme and adaptor signatures, and is efficient only for functions for which efficient FE schemes are known. This is why they restrict themselves to functions where f is *linear* in its input. Although intensely pursued, the state of progress for an efficient FE scheme for richer functions is underwhelming. Returning to Figure 2, we see that the total time for our HE-based constructions (LHE-FAS and BGN-FAS) for the same applications is at least *two orders of magnitude faster* than the SNARK strawman protocols. Moreover, our LHE-FAS scheme is about 10x better than VST24 for most database sizes. We defer the details of BGN-FAS and LHE-FAS to Section 6 and Section 7, respectively.

2.1 Our FAS Template: HE + Adaptors

Given the limitations of FE-based approaches, our key insight is to replace FE with (fully) homomorphic encryption ((F)HE), for which more practical constructions are available. Unlike FE, which allows adaptive decryption of $f(\vec{x})$ from an encryption of $\vec{x}$ without the master secret key, HE enables anyone to compute an encryption of $f(\vec{x})$ from the ciphertext of $\vec{x}$. However, HE lacks a mechanism for decryption without the secret key, necessitating a new protocol to build FAS from HE.

The high-level idea for our HE-based FAS construction is as follows: in the advertisement phase, the seller computes an HE encryption of $\vec{x}$. The buyer homomorphically evaluates this ciphertext with respect to a function f. To reveal the result, the seller "opens" the evaluated ciphertext, enabling the buyer to learn $f(\vec{x})$. Translating this idea into a practical FAS scheme requires addressing several technical challenges, which we detail after presenting the protocol's structure. We define the advertisement and exchange phases as follows:

- **Advertisement Phase:** The seller generates a fresh HE key pair $(\mathsf{pk}_{\mathsf{HE}}, \mathsf{sk}_{\mathsf{HE}})$ and publishes $\mathsf{CT}_{\mathsf{ad}} \leftarrow\!\$\ \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{HE}}, \vec{x})$ along with $\mathsf{pk}_{\mathsf{HE}}$. A NIZK proof is provided to certify that $\mathsf{CT}_{\mathsf{ad}}$ encrypts a valid witness for X and that $\mathsf{pk}_{\mathsf{HE}}$ is well-formed.[3] This phase is performed once and can be published off-chain, such as on the seller's website, for reuse across all sales.
- **Exchange Phase:** The buyer computes CT_y, an HE encryption of $y = f(\vec{x})$, by homomorphically evaluating $\mathsf{CT}_{\mathsf{ad}}$ with respect to f. This computation, performed offline, requires no seller interaction. Once the seller receives f, it computes $y = f(\vec{x})$ [4] and encrypts it using the Elgamal

[2] For completeness, we compute the Groth16 proof for the following simpler statement: $\exists \vec{x}$ such that some public hash value h equals $H(\vec{x})$ and the claimed value $y = f(\vec{x})$, where H is SNARK-friendly hash function such as Poseidon.

[3] Ensuring $\mathsf{pk}_{\mathsf{HE}}$ is well-formed becomes crucial for the soundness of proofs in the exchange phase.

[4] Proofs over homomorphically encrypted values [ACGSV24, GGW24] are orthogonal to our work, but potentially

PKE scheme with a fresh key pair $(K = g^k, k)$. Along with the Elgamal ciphertext ct_y, the seller provides a NIZK proof π that ct_y and CT_y (that was computed from the advertisement) encrypt the same plaintext, referred to as *Proof of Plaintext Equality (PoPE)*. The buyer verifies π, and if valid, subsequently purchases the Elgamal secret key k using the Schnorr adaptor signature. The buyer then decrypts the Elgamal ciphertext ct_y to retrieve y.

This protocol supports FAS for all functions f when the HE scheme is a fully homomorphic encryption scheme [GSW13, BGV11]. Its security (see Section 4 for definitions) follows from the properties of the underlying tools, including the semantic security of HE and Elgamal schemes, and the soundness of NIZKs.

2.2 Efficiency Improvements

While this template is secure, it lacks practicality in its current form. We identify three key challenges and opportunities to enhance efficiency for both parties:

- **Seller's Computation of CT_y and Randomness:** To compute the NIZK proof in the exchange phase, the seller must generate CT_y and its associated randomness r_y which serves as a witness in the NIZK proof. Doing so naively via HEval is computationally expensive, particularly for resource-constrained sellers, as in some applications. Moreover, this would not aid the seller with computing r_y.
- **Complexity of General-Purpose NIZKs:** Although the exchange-phase NIZK is independent of the function f's complexity (unlike the Strawman), general-purpose NIZKs add significant overhead when modeling the algebraic constraints of the HE scheme. This can also lead to non-black-box use of the HE scheme, introducing further inefficiencies.
- **Buyer's Cost of Decrypting Elgamal PKE:** The buyer's decryption process involves computing discrete logarithms (due to the decryption of standard Elgamal PKE), which are computationally expensive (as VST24 observed) and require strict bounds on the function's output, limiting general applicability.

We address these challenges to construct a practical scheme, as detailed below.

(a) Efficient Computation of CT_y and Randomness. Ensuring the seller and buyer agree on the same CT_y is critical for a valid statement in the PoPE NIZK. This issue can be mitigated using HE schemes that support a *deterministic* HEval procedure, a feature common to most existing HE schemes [GSW13, CGGI20, BGV11]. However, if both parties independently compute CT_y via HEval, it leads to redundant work, reducing the overall efficiency. The buyer's homomorphic computation remains essential because offloading it to the seller would require proving the correctness of f-sized computations in the exchange NIZK proof, resulting in significant performance degradation compared to the original template. To improve efficiency, we leverage a sibling algorithm REval, implicit in most HE schemes. REval computes the randomness r_y for CT_y as a function of the randomness from the input ciphertext $\mathsf{CT}_{\mathsf{ad}}$, mimicking HEval without operating on the ciphertexts. This makes REval significantly faster than HEval. Once the seller computes r_y, CT_y is obtained using $\mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{HE}}, y; r_y)$. Looking ahead, while such an REval algorithm is implicit in the TFHE and LHE schemes we use, for the case of degree-2 functions, we introduce a novel variant of the Boneh-Goh-Nissim (BGN) encryption scheme that supports deterministic HEval and efficient REval algorithms.

Modulo the computation of output y, can the seller's computation be made independent of the complexity of f? While this does not result in asymptotic improvements, this is highly relevant for practical efficiency. We show that this is indeed possible in HE schemes that support homomorphism

useful for improving seller efficiency in FAS. With efficiently verifiable FHE, the buyer could also prove that its ciphertext correctly encodes f(x), eliminating the need for the seller to compute the function output.

and, additionally, the bootstrapping functionality. In this work, we use the TFHE scheme [CGGI20], and rely on an insight regarding the underlying structure of its HEval algorithm: key switching, one of the steps involved in bootstrapping that transforms a ciphertext $\mathsf{ct}_{2,f}$ encrypting message $f(m)$ under a key pk_2 into a ciphertext $\mathsf{ct}_{1,f}$ under another key pk_1. This is accomplished by using a key switching key ksk, which in itself is a FHE ciphertext encrypting secret key sk_2 (corresponding to pk_2) under pk_1. Key switching here is roughly a linear computation as follows:

$$\mathsf{ct}_{1,f} := \langle \mathsf{ksk}, \mathsf{ct}_{2,f} \rangle,$$

where ksk is treated a vector of ciphertexts and $\mathsf{ct}_{2,f}$ is treated as a vector of known constants. Thus, the entire computation is simply a linear homomorphic evaluation on the ciphertext ksk. Crucially, after this transformation, the random coins $r_{1,f}$ underlying the new ciphertext $\mathsf{ct}_{1,f}$ only depend on the randomness r_{ksk} in the key switching key ksk, and not on the randomness of the original ciphertext $\mathsf{ct}_{2,f}$. Suppose r_{ksk} is the randomness underlying ciphertext ksk. Then $r_{1,f}$ can be simply recovered as $r_{1,f} := \langle r_{\mathsf{ksk}}, \mathsf{ct}_{2,f} \rangle$. This allows us to build an REval algorithm that takes a ciphertext $\mathsf{ct}_{2,f}$ computed using HEval, and key switches it into another one $\mathsf{ct}_{1,f}$ whose randomness $r_{1,f}$ is easy to compute as above. This computation is significantly cheaper than HEval itself, and is linear in the size of the key switching key and independent of the complexity of f. We refer to Section 8.1 for more details.

(b) Black-Box Efficient PoPE Protocol. The exchange phase NIZK in our protocol proves a simpler statement than in the strawman protocol. Specifically, the seller must demonstrate that an HE ciphertext and an Elgamal ciphertext encrypt the same plaintext value. This statement is independent of the function f, making it significantly more efficient for the seller, even with a general-purpose NIZK such as Groth16. However, this approach involves non-black-box use of the HE scheme. To address this, we propose a modular *cut-and-choose* based Σ-PoPE protocol[5] (Section 5.2) that makes black-box use of the HE scheme, and is of independent interest.

(c) Efficient Decryption. In the template construction, the buyer performs an Elgamal PKE decryption to obtain the result. Similar to VST24, this involves costly discrete-logarithm computations, adding a significant bottleneck, especially when combined with the buyer's HEval computation. We optimize this by slightly modifying the protocol. In the exchange phase, the seller encrypts the output y in a bitwise fashion. For an n-bit binary representation $y = y_1 || \ldots || y_n$, each bit y_i is encrypted using the same Elgamal public key K. The i-th ciphertext is given by $\mathsf{ct}_i = (g^{r_i}, K^{r_i} \cdot g^{y_i})$.

With this change, the buyer computes the result y by performing n parallel group multiplications, a far simpler operation than a discrete-logarithm computation. This modification necessitates minor updates to the PoPE protocol: the proof now addresses bitwise encryptions $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, rather than a single ciphertext ct_y. However, by leveraging the homomorphic properties of the Elgamal PKE scheme, the new PoPE statement can be efficiently reduced to the original statement with a single ciphertext ct_y. See Section 5 for details.

(d) Bug in the FHE Instantiation. After the submission of the IEEE S&P 2026 camera-ready version, we identified an issue in the FHE-based instantiation. The FHE instantiation with TFHE admits a bug in the security proof. In particular, a malicious buyer can learn some information about $f(x)$. Below we provide a detailed explanation of the bug and potential solutions.

Recall that the NIZK PoPE in the exchange phase requires that the seller knows the random coins of the homomorphically evaluated ciphertext. With the goal of producing a seller optimized protocol, we introduced efficient REval algorithms for each HE scheme that enable the seller to compute these random coins without having to do the expensive HE operations. The most natural way to compute REval is to start with the random coins of the original ciphertexts, and track

[5]This protocol can be made non-interactive using the Fiat-Shamir transform.

the changes made to the coins through the homomorphic evaluation of f, this is what we do for our degree 1 and 2 instantiations. However, for the TFHE scheme, this way of computing the randomness is as expensive as computing the homomorphic evaluation itself.

For this reason, we wanted to devise an alternative approach for REval, and we did that by having the buyer compute the homomorphically evaluated ciphertext, allowing the seller to perform a cheap key-switching operation that exploits the internal structure of the TFHE ciphertexts to easily compute the random coins (see Section 8.1.1). Although this is efficient, enabling the buyer to compute the homomorphically evaluated ciphertext in this way introduces a security bug. The issue is that the ciphertext the buyer produces is unauthenticated. A malicious buyer could manipulate one of the bits of the evaluated ciphertext by subtracting it from itself, thus effectively switching the underlying message bit to zero due to homomorphism. The seller would decrypt it to recover z and check if it matches $f(x)$. If it doesn't match, the seller aborts. But this abort reveals one bit of information about $f(x)$: the underlying bit must have been equal to one. Therefore, this approach to REval results in a security bug. We emphasize that this problem is limited only to the FHE instantiation, as this behavior is unique to the REval optimization for TFHE. Our construction for general functions (Section 5), as well as the quadratic (Section 6) and linear (Section 7) instantiations remain secure. This security bug could be resolved by having the seller compute the homomorphic evaluation. This increases the runtime of the seller in our instantiation. Therefore, our results detailed in Section 8.1.2 should be viewed as a lower bound for seller efficiency in a secure solution. With that being said, we are actively looking into solutions to this issue that still maintain seller efficiency.

A naive solution (that maintains seller efficiency) is to have the buyer include a proof that ensures the homomorphic evaluation was done properly. In the case where the proof does not verify, the seller can abort, and the buyer learns nothing. However, these techniques incur significant overheads as compared to basic FHE, effectively making the buyer's computation infeasible for practical use. Alternatively, one could hope to reduce this overhead by having the buyer utilize trusted hardware (like a Trusted Execution Environment (TEE)). We leave the study of the exact assumption needed from such hardware (whether a full-blown TEE or something weaker will suffice) for future work.

2.3 Improved Degree-2 FAS

We next optimize our general FAS construction for degree-2 functions—a simple yet powerful class of non-linear functions that are prevalent in real-world applications like medical analytics and machine learning.

Optimized Degree-2 FAS Construction. Our construction replaces the general HE scheme with the Boneh-Goh-Nissim (BGN) encryption scheme [BGN05], which supports homomorphic evaluation of degree-2 functions over composite-order bilinear groups. To integrate BGN into the FAS framework, we extend the BGN scheme with a deterministic HEval and REval procedure for efficient randomness tracking. Additionally, we design a novel PoPE protocol to prove plaintext equality across Elgamal and BGN ciphertexts. Key insights include:

- **New PoPE protocol.** We propose a new PoPE protocol to prove plaintext equality between an Elgamal ciphertext and a BGN commitment by generalizing the prior work of Chase et al. [COPZ22], who gave a PoPE proof across Pedersen commitments defined over two different prime-order groups. In contrast, our PoPE protocol works even when the second group's order is a product of large primes (i.e., composite order).
- **Variant of BGN.** By removing blinding terms in the BGN homomorphic operations, we enable deterministic randomness tracking using a new REval procedure. This modification allows the

seller to compute the randomness needed for the buyer's homomorphic evaluation, without compromising security.

The full construction, including the new PoPE protocol and REval formalization, are detailed in Section 6.

2.4 Defining Fair Functional Exchange

We define *Fair Functional Exchange* (FFE) as a generalization of the prior work on fair exchange protocols [BGKS25, TSZ+24], extending beyond the traditional all-or-nothing paradigm to the setting of functional sales. While FAS corresponds to a two-round instance of FFE,[6] we model FFE more generally as an interactive R-round protocol involving a seller, a buyer, and a wallet-based token system that abstracts the role of the blockchain.

FFE guarantees fairness for both parties. Buyer fairness is relatively straightforward: the buyer should not lose tokens without obtaining the correct output. Seller fairness is more delicate – it requires that a prematurely aborting buyer *learns nothing*, whereas a buyer who completes the protocol *learns only the intended output*. To capture this, we introduce a novel notion of *round-by-round zero-knowledge*, analogous to round-by-round soundness in interactive proofs [Hol19]. This guarantees that transcripts from aborted executions are simulatable without access to the output, while transcripts from non-aborting or complete executions can be simulated only given the output y. See Theorem 9.1 for formal details.

3 Preliminaries

Notation. The security parameter is denoted by λ. We denote uniform sampling of x from a set S by $x \leftarrow\!\$\ S$. If x is the output of a Probabilistic Polynomial Time (p.p.t.) algorithm $A(\cdot)$, we denote this by $x \leftarrow A(\cdot)$. If $A(\cdot)$ is a Deterministic Polynomial Time (DPT) algorithm, we instead use the notation $x := A(\cdot)$. We will denote negligible functions by ϵ, and $\epsilon(\lambda)$ denotes a function that is negligible in the security parameter λ. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible in λ if for every $k \in \mathbb{N}$, there exists a $\lambda_0 \in \mathbb{N}$ such that for every $\lambda \geq \lambda_0$, we have that $|f(\lambda)| \leq 1/\lambda^k$.

3.1 Hard Relations

Definition 3.1 (Hard Relation)**.** *We say a relation* R *for statement/witness pairs* (Y, y) *is hard if:*

(i) there exists a p.p.t. algorithm $\mathsf{GenR}(1^\lambda)$ *that generates a statement/witness pair* $(Y, y) \in \mathsf{R}$*,*

(ii) the relation R *is decidable in poly-time,*

(iii) for all p.p.t. adversaries $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*, we have that* $\Pr\left[\mathsf{G}_{\mathcal{A},\mathsf{R}}(1^\lambda) = 1\right] \leq \epsilon(\lambda)$*, where the game* $\mathsf{G}_{\mathcal{A},\mathsf{R}}$ *is defined in Figure 3.*

We denote L_R *as the associated language for* R*, defined as* $L_\mathsf{R} = \{Y \mid \exists y \text{ s.t. } (Y, y) \in \mathsf{R}\}$.

Definition 3.2 (Discrete Logarithm (DL))**.** *For a group* $\mathbb{G}$ *of order* p *with generator* $g \in \mathbb{G}$*, the* discrete logarithm *of an element* $Y \in \mathbb{G}$ *is the integer* $y \in \mathbb{Z}_p$ *such that* $Y = g^y$.

Definition 3.3 (Hard Relation R_DL)**.** *For a group* $\mathbb{G}$ *of order* p *with generator* $g \in \mathbb{G}$*, the* discrete log language *is defined as* $L_\mathsf{DL} := \{Y \mid \exists\ y \in \mathbb{Z}_p \text{ s.t. } Y = g^y\}$*, where the corresponding hard relation is* $\mathsf{R}_\mathsf{DL} = \{(Y, y) \mid Y = g^y\}$*. Note that the relation* R_DL *is hard when its corresponding instance of the* DL *problem is hard in the group* $\mathbb{G}$.

[6] Here, a "round" refers to a message from the seller followed by a response from the buyer.

Game $\mathsf{G}_{\mathcal{A},\mathsf{R}}(1^\lambda)$:

1 : $(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$
2 : $y' \leftarrow \mathcal{A}(Y)$
3 : if $((Y, y') \in \mathsf{R})$: return 1
4 : return 0

Figure 3: The hard relation game $\mathsf{G}_{\mathcal{A},\mathsf{R}}$

We also introduce another notion of a hard relation, called a $\mathcal{F}$-hard relation, with respect to a family of functions $\mathcal{F}$. It is a slight variation of the original hard relation definition, where it requires that given a statement Y of the relation, no efficient adversary $\mathcal{A}$ can find a value z and a function f such that $z = f(y^*)$, where (Y, y^*) is a valid instance of the relation. We formalize this in Definition 3.4.

Definition 3.4 ($\mathcal{F}$-Hard Relation)**.** *We say that a relation* R *of statement/witness pairs* (Y, y) *is* $\mathcal{F}$*-hard if:*

(i) there exists a p.p.t. algorithm $\mathsf{GenR}(1^\lambda)$ *that generates a statement/witness pair* $(Y, y) \in \mathsf{R}$*,*

(ii) for all p.p.t. adversaries $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*, we have that* $\Pr\left[\mathsf{G}'_{\mathcal{A},\mathsf{R},\mathcal{F}}(1^\lambda) = 1\right] \leq \epsilon(\lambda)$*, where the game* $\mathsf{G}'_{\mathcal{A},\mathsf{R},\mathcal{F}}$ *is defined in Figure 4.*

Game $\mathsf{G}'_{\mathcal{A},\mathsf{R},\mathcal{F}}(1^\lambda)$:

1 : $(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$
2 : $(f, z) \leftarrow \mathcal{A}(Y)$
3 : if $(f \in \mathcal{F}) \wedge \big(z \in \{f(y^*) \mid \exists y^* \text{ s.t } (Y, y^*) \in \mathsf{R}\}\big)$:
return 1
4 : return 0

Figure 4: The $\mathcal{F}$-hard relation game $\mathsf{G}'_{\mathcal{A},\mathsf{R},\mathcal{F}}$

3.2 Public Key Encryption

We will be using various public key encryption (PKE) schemes throughout the paper, so we formalize the definition of PKE in this section, as well as the necessary security properties.

Definition 3.5 (Public Key Encryption Scheme)**.** *A public key encryption scheme* $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a tuple of three algorithms* KGen*,* Enc*,* Dec*, defined as follows:*

(i) $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*: The key generation algorithm* KGen *takes as input the security parameter, and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *is called the public key, and* sk *is called the secret key.*

(ii) $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: The encryption algorithm takes as input a public key* pk *and a message* m*, and outputs a ciphertext* ct*, which is an encryption of* m *under the public key* pk*.*

(iii) $m \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: The decryption algorithm takes as input a secret key* sk *and a ciphertext* ct*, and outputs the plaintext* m*.*

Definition 3.6 (Correctness for PKE)**.** *A public key encryption scheme satisfies correctness if for all* $\lambda \in \mathbb{N}$*, all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*, and any message* m*:*

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1 \ .$$

We introduce an important security property, which is called semantic security.

Definition 3.7 (Semantic Security for PKE)**.** *A public key encryption scheme is semantically secure if for all p.p.t. adversaries* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*, we have:*

$$\mathsf{Adv}^{\mathsf{PKE}}_{\mathcal{A}} = \left| \Pr[\mathcal{A} \text{ outs } 1, \text{ in } G_0] - \Pr[\mathcal{A} \text{ outs } 1, \text{ in } G_1] \right| \leq \epsilon(\lambda) \ ,$$

where games G_0 *and* G_1 *are defined as in Figure 5.*

Game $G_b(1^\lambda)$	
1 :	$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
2 :	$m_0, m_1 \leftarrow \mathcal{A}(\mathsf{pk})$
3 :	$\mathsf{ct}_b \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$
4 :	$b' \leftarrow \mathcal{A}(\mathsf{ct}_b)$
5 :	return $b' = 1$

Figure 5: Semantic Security Game G_b

3.2.1 ElGamal Encryption Scheme

An encryption scheme that will be useful to us later is the ElGamal encryption scheme. We provide the details of the exponential version of this encryption scheme in Definition 3.8.

Definition 3.8 (Exponential ElGamal Encryption)**.** *Let* $\mathbb{G}$ *be a cyclic group with prime order* p *where the* DL *problem is hard, and let* g *be a generator of* $\mathbb{G}$*. Then we denote the multiplicative ElGamal Encryption Scheme by* $\mathsf{ElGamal} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*, and is defined in Figure 6.*

$\mathsf{KGen}(1^\lambda)$:		$\mathsf{Enc}(\mathsf{pk}, m \in \mathbb{Z}_p)$:		$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:	
1 :	$\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$	1 :	$\beta \leftarrow\!\!\$\ \mathbb{Z}_p$	1 :	Parse ct as $(\mathsf{ct}_1, \mathsf{ct}_2)$
2 :	$\mathsf{pk} := g^{\mathsf{sk}}$	2 :	$\mathsf{ct}_1 := g^\beta$	2 :	$\mathsf{ct}' := \mathsf{ct}_2 \cdot (\mathsf{ct}_1)^{-\mathsf{sk}}$
3 :	return $(\mathsf{pk}, \mathsf{sk})$	3 :	$\mathsf{ct}_2 := g^m \cdot \mathsf{pk}^\beta$	3 :	$m := \mathsf{dlog}_g(\mathsf{ct}')$
		4 :	return $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$	4 :	return m

Figure 6: Exponential ElGamal Encryption Algorithm Definitions

Theorem 3.9 (ElGamal Semantic Security)**.** *The ElGamal* PKE *scheme is semantically secure.*

It will be useful later to encrypt messages $m \in \mathbb{Z}_p$ bitwise using ElGamal. That is, we decompose m into its bit representation, and encrypt each bit separately using exponential ElGamal. We will call this bitwise ElGamal encryption.

Game $G_b(1^\lambda)$	
1 :	$\mathsf{ck} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$
2 :	$m_0, m_1 \leftarrow \mathcal{A}(\mathsf{ck})$
3 :	$(c, d) \leftarrow \mathcal{C}.\mathsf{Com}(\mathsf{ck}, m_b)$
4 :	$b' \leftarrow \mathcal{A}(c, d)$
5 :	return $b' = 1$

Figure 7: Semantic Security Game G_b for Commitment Schemes

3.3 Commitment Schemes

We will be using various commitment schemes throughout the paper, so we formalize the definition of commitments in this section, as well as the necessary security properties.

Definition 3.10 (Commitment Scheme)**.** *A commitment scheme* $\mathcal{C} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is tuple of three algorithms* $\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}$*, defined as follows:*

(i) $\mathsf{ck} \leftarrow \mathsf{Gen}(1^\lambda)$*: The commitment key generation algorithm* Gen *takes as input the security parameter, and outputs the commitment key* ck*.*

(ii) $(c, d) \leftarrow \mathsf{Com}(\mathsf{ck}, m)$*: The commit algorithm takes as input the commitment key* ck *and a message* m*, and outputs a commitment* c *and an decommitment string* d*.*

(iii) $b \leftarrow \mathsf{Open}(c, d, m)$*: The opening algorithm takes as input a commitment* c*, a decommitment string* d*, and a message* m*, and outputs a bit* b*.*

We introduce two important security properties, which are called hiding and binding.

Definition 3.11 (Hiding)**.** *A commitment scheme is statistically hiding if for all p.p.t. adversaries* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*, we have:*

$$\mathsf{Adv}^{\mathcal{C}}_{\mathcal{A}} = \left| \Pr[\mathcal{A} \text{ outs } 1, \text{ in } G_0] - \Pr[\mathcal{A} \text{ outs } 1, \text{ in } G_1] \right| \leq \epsilon(\lambda) \ ,$$

where games G_0 *and* G_1 *are defined as in Figure 7. We say that the commitment scheme is perfectly hiding if the probability above is equal to* 0*.*

Definition 3.12 (Binding)**.** *A commitment scheme is statistically binding if for all p.p.t. adversaries* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*, we have that:*

$$\Pr[m_1 \neq m_2 \wedge \mathsf{Open}(c, d_1, m_1) = 1 \wedge \mathsf{Open}(c, d_2, m_2) = 1 : (m_1, m_2, c, d_1, d_2) \leftarrow \mathcal{A}] \leq \epsilon(\lambda) \ .$$

We say that the commitment scheme is perfectly binding if the probability above is equal to 0*.*

3.3.1 BGN Commitment Scheme

The BGN encryption scheme was first introduced by Boneh, Goh, and Nissim [BGN05]. In this paper, it will be more beneficial for us to consider them as commitments instead for a few reasons. The first is that we will not utilize the decryption algorithm in our protocols. The second is that we will want to utilize the property of binding, so it is simpler view the BGN encryptions as commitments instead. Therefore, we define it here as a commitment scheme.

Definition 3.13 (Bilinear Map)**.** *For two multiplicative cyclic groups $\mathbb{G}, \mathbb{T}$ of order n, we say that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ is a bilinear map if*

1. *For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have that $e(u^a, v^b) = e(u, v)^{ab}$.*
2. *If $g \in \mathbb{G}$ is a generator for $\mathbb{G}$, then $e(g, g) \in \mathbb{T}$ is a generator for $\mathbb{T}$.*

We say that a group $\mathbb{G}$ is a bilinear group if there exists a group $\mathbb{T}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{T}$.

Definition 3.14 (Bilinear Group Generator)**.** *A bilinear group generator $\mathcal{G}$ takes as input the security parameter 1^λ, and outputs a tuple $(q_1, q_2, \mathbb{G}, \mathbb{T}, e)$, where $\mathbb{G}$ and $\mathbb{T}$ are both groups of order $n = q_1 q_2$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ is a bilinear map. The generation algorithm is as follows:*

1. *Sample two λ-bit primes q_1, q_2 uniformly at random, let $n = q_1 q_2$.*
2. *Sample a bilinear group $\mathbb{G}$ of order n, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ is the bilinear map.*
3. **return** $(q_1, q_2, \mathbb{G}, \mathbb{T}, e)$

There are various ways to construct a bilinear group of order n, but the authors of BGN provide a particular construction [BGN05]. It is as follows:

1. Find the smallest $\ell \in \mathbb{Z}^+$ such that $p = \ell \cdot n - 1$ is prime, and $p \mod 3 = 2$.
2. Consider the super-singular elliptic curve $y^2 = x^3 + 1$ defined over $\mathbb{F}_p$. Then there exists an order n subgroup $\mathbb{G}$ of the points on the curve.
3. Let $\mathbb{T}$ be the order n subgroup of $\mathbb{F}^*_{p^2}$. Then we have that the modified Weil pairing on the curve yields the desired bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{T}$.

Definition 3.15 (Subgroup Decision Problem)**.** *Suppose that we have a bilinear group generator $\mathcal{G}$, and we are given a tuple $(q_1, q_2, \mathbb{G}, \mathbb{T}, e) \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$. The subgroup decision problem is as follows:given an element $x \in \mathbb{G}$, the goal is to determine if x is an element of a subgroup of $\mathbb{G}$ or not, without knowing the factorization of the order of the group $|\mathbb{G}| = n$.*

Definition 3.16 (Hardness of Subgroup Decision Problem)**.** *Suppose that we have a bilinear group generator $\mathcal{G}$, and we are given a tuple $(q_1, q_2, \mathbb{G}, \mathbb{T}, e) \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$. For any adversary $\mathcal{A}$, we define its advantage $\mathsf{SDP\text{-}Adv}_{\mathcal{A}}(1^\lambda)$ in solving the subgroup decision problem as:*

$$\mathsf{SDP\text{-}Adv}_{\mathcal{A}}(1^\lambda) = \Bigg| \Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{T}, e, x) = 1 : \begin{array}{c}(q_1, q_2, \mathbb{G}, \mathbb{T}, e) \leftarrow\!\!\$\ \mathcal{G}(1^\lambda), \\ n = q_1 q_2,\ \ x \leftarrow\!\!\$\ \mathbb{G}\end{array}\right] - \Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{T}, e, x^{q_2}) = 1 : \begin{array}{c}(q_1, q_2, \mathbb{G}, \mathbb{T}, e) \leftarrow\!\!\$\ \mathcal{G}(1^\lambda), \\ n = q_1 q_2,\ \ x \leftarrow\!\!\$\ \mathbb{G}\end{array}\right] \Bigg| .$$

We say that the Subgroup Decision Problem is hard for a bilinear group generator $\mathcal{G}$ if for all adversaries $\mathcal{A}$, there exists a negligible function ϵ such that for all $\lambda \in \mathbb{N}$, its advantage $\mathsf{SDP\text{-}Adv}_{\mathcal{A}}(1^\lambda) \leq \epsilon(\lambda)$.

The BGN encryption scheme was propsed by Boneh, Goh, and Nissim [BGN05]. It uses bilinear groups to enable degree 2 homomorphism. Its security is guaranteed by the hardness of the subgroup decision problem. As mentioned before, we will instead use BGN commitments. Throughout the paper, we will denote this commitment scheme by BGN, and it is defined by three main algorithms $\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}$, defined as in Figure 8.

Note that later in implementation, we will need an additional algorithm Com^*, which we describe in Figure 9. This algorithm enables generation of ciphertexts that exist in the target group.

$\mathsf{Gen}(1^\lambda)$:	$\mathsf{Com}(\mathsf{ck}, m)$:	$\mathsf{Open}(\mathsf{ck}, \mathsf{ct}, d, m)$:
1 : $(q_1, q_2, \mathbb{G}, \mathbb{T}, e) \leftarrow\!\$\ \mathcal{G}(1^\lambda)$	1 : Parse ck as $(n, \mathbb{G}, \mathbb{T}, e, g, h, i)$	1 : Parse ck as $(n, \mathbb{G}, \mathbb{T}, e, g, h, i)$
2 : $n = q_1 q_2$	2 : $r \leftarrow\!\$\ \mathbb{Z}_n$	2 : Parse d as r
3 : sample generators $g, u \leftarrow\!\$\ \mathbb{G}$	3 : $\mathsf{ct} = g^m h^r \in \mathbb{G}$	3 : return $\mathsf{ct} = g^m h^r$
4 : $h = u^{q_2}$	4 : $d = r$	
5 : $i = e(h, h)$	5 : return (ct, d)	
6 : $\mathsf{ck} = (n, \mathbb{G}, \mathbb{T}, e, g, h, i)$		
7 : return ck		

Figure 8: BGN Commitment Scheme Algorithm Descriptions

$\mathsf{Com}^*(\mathsf{ck}, m)$:
1 : Parse ck as $(n, \mathbb{G}, \mathbb{T}, e, g, h, i)$
2 : $r \leftarrow\!\$\ \mathbb{Z}_n$
3 : $\mathsf{ct} = g^m h^r \in \mathbb{G}$
4 : $\mathsf{ct}^* = e(g, \mathsf{ct}) \in \mathbb{T}$
5 : $d = r$
6 : return (ct^*, d)

Figure 9: Extra Algorithm Com^*

Theorem 3.17 (BGN Commitment Hiding)**.** *If the subgroup decision problem is hard for the bilinear group generator $\mathcal{G}$ used in* BGN.Gen*, then the* BGN *commitment scheme is computationally hiding.*

Proof. The proof immediately follows from the semantic security proof of the BGN encryption scheme (see BGN paper [BGN05]). □

Theorem 3.18 (BGN Commitment Binding)**.** *If the discrete log problem is hard for the base group $\mathbb{G}$ sampled from the bilinear group generator $\mathcal{G}$ used in* BGN.Gen*, then the* BGN *commitment scheme is computationally binding.*

Idea. The prover can't change the powers in the commitment since they don't know the discrete log of g with respect to h. □

Given that we will want to perform homomorphic evaluations of degree 2 functions f on BGN commitments, we need to examine the homomorphic properties of the BGN commitment scheme.

Homomorphic Properties. There are two operations that we can perform on BGN commitments: Add, and Multiply. To add the values committed in two commitments $\mathsf{ct}_1, \mathsf{ct}_2$, do $\mathsf{BGN.Add}(\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{ct}_1 \cdot \mathsf{ct}_2 \in \mathbb{G}$. To multiply the values committed in two commitments $\mathsf{ct}_3, \mathsf{ct}_4$, $\mathsf{BGN.Multiply}(\mathsf{ct}_3, \mathsf{ct}_4) = e(\mathsf{ct}_3, \mathsf{ct}_4) \in \mathbb{T}$.

Now we will describe the general process for a homomorphic evaluation. Suppose we have a function $f : \mathbb{Z}^\ell \to \mathbb{Z}$, and for some $w = (w_1, w_2, \ldots, w_\ell) \in \mathbb{Z}^\ell$, we have corresponding BGN commitments $(ct_1, ct_2, \ldots, ct_\ell)$. To homomorphically compute $f(w)$, we can perform Add and Multiply operations on the ℓ ciphertexts encrypting values $(w_1, w_2, \ldots, w_\ell)$ as follows:

1. Perform all additions in the base group.
2. Perform multiplications (generating ciphertexts in the target group)
3. Map all elements into the target group, and then add all components to obain the final result.

Remark 3.19. *Note that in the original* BGN *construction term, during* Add *and* Multiply*, blinding terms are multiplied to generate fresh encryptions after the homomorphic operation is done. However, in our case, we want the homomorphic operations to be deterministic, as the seller needs to know what randomness the buyer will have after performing the homomorphic operations. This is essential for the success of the proof of equality that is used in our* BGN-FAS *implementation, as we will see later. Further, removing this blinding term will not affect security in our application.*

3.4 Digital Signatures

Definition 3.20 (Digital Signature)**.** *A digital signature scheme is defined as* $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$*, where the three algorithms* $\mathsf{KGen}, \mathsf{Sign}$*, and* Vrfy *are defined as follows:*

(i) $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$*: The key generation algorithm* KGen *takes as input the security parameter, and outputs a signing key and verification key pair* $(\mathsf{sk}, \mathsf{vk})$*.*

(ii) $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*: The signing algorithm* Sign *takes as input the signing key* sk *and a message* m*, and outputs a signature* σ*.*

(iii) $b \in \{0,1\} := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$*: The verification algorithm* Vrfy *takes as input the verification key* vk*, a message* m*, and a signature* σ*, and outputs a bit* b*, which is* 1 *if* σ *is valid, and* 0 *otherwise.*

In order for a digital signature scheme to be secure, it must satisfy two properties: correctness *(Definition 3.21) and* strong unforgeability *(Definition 3.22).*

Definition 3.21 (Correctness)**.** *A digital signature scheme* DS *satisfies* correctness *if for every* $\lambda \in \mathbb{N}$*, and message* $m \in \{0,1\}^*$*, we have that:*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1 \ : \ \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] = 1 \ .$$

Definition 3.22 (Strong Unforgeability)**.** *A digital signature scheme* DS *satisfies* strong unforgeability *if for every p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\mathsf{strongForge}_{\mathcal{A},\mathsf{DS}}(1^\lambda) = 1\right] \leq \epsilon(\lambda) \ ,$$

where $\mathsf{strongForge}_{\mathcal{A},\mathsf{DS}}$ *is defined in Figure 10.*

3.4.1 Implementation: Schnorr Digital Signature Scheme

One of the most common signature schemes is the Schnorr Signature Scheme. It utilizes the hardness of the discrete log problem to create a secure signature scheme. We denote the Schnorr Signature Schme by DS_{sch}, and is defined in Definition 3.23.

Definition 3.23 (Schnorr Digital Signature Scheme, DS_{sch})**.** *Let* $\mathbb{G}$ *be a cyclic group with prime order* p *where the* DL *problem is hard, and let* g *be a generator of* $\mathbb{G}$*. Further, for the desired message space* $\mathcal{M}$*, let* $H : \mathbb{G} \times \mathcal{M} \to \mathbb{Z}_p$ *be a secure hash function. The* $\mathsf{KGen}, \mathsf{Sign}$*, and* Vrfy *algorithms for* DS_{sch} *are defined in Figure 11.*

Experiment $\mathsf{strongForge}_{\mathcal{A},\mathsf{DS}}(1^\lambda)$	Oracle $\mathcal{O}_s(m)$
$(\mathsf{sk},\mathsf{vk}) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$Q := \emptyset$	$Q := Q \cup \{(m,\sigma)\}$
$(m^*,\sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_s}(\mathsf{vk})$	return σ
return $((m^*,\sigma^*) \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$	

Figure 10: Strong Unforgeability experiment for Digital Signature Schemes

$\mathsf{KGen}(1^\lambda)$	$\mathsf{Sign}(\mathsf{sk}, m)$	$\mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$
$\mathsf{sk} \leftarrow\!\$\ \mathbb{Z}_p$	$r \leftarrow\!\$\ \mathbb{Z}_p$	Parse σ as (R, s)
$\mathsf{vk} := g^{sk}$	$R := g^r$	$h := H(R\|\|m)$
return $(\mathsf{sk},\mathsf{vk})$	$h := H(R\|\|m)$	return $g^s \stackrel{?}{=} R \cdot vk^h$
	$s := h \cdot \mathsf{sk} + r$	
	return $\sigma := (R, s)$	

Figure 11: Definition for Schnorr Digital Signature Scheme DS_{sch}

3.5 Adaptor Signatures

Definition 3.24 (Adaptor Signature)**.** *An adaptor signature scheme* $\mathsf{AS}_{\mathsf{DS},\mathsf{R}} := (\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext})$ *is defined with respect to a digital signature scheme* DS *(as defined in Definition 3.20) and a hard relation* R*. The adaptor signature scheme inherits the algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *of the underlying digital signature scheme* DS*. The four additional algorithms needed for adpator signatures are defined as:*

(i) $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$*: The pre-signature algorithm* pSign *takes as input the signing key* sk*, a message* m*, and a statement* Y *of the relation* R*, and outputs a pre-signature* $\widetilde{\sigma}$ *of* m *with respect to* Y*.*

(ii) $b \in \{0,1\} \leftarrow \mathsf{pVrfy}(\mathsf{vk}, m, Y, \widetilde{\sigma})$*: The pre-signature verification algorithm* pVrfy *takes as input the verification key* vk*, a message* m*, a statement* Y *of the relation* R*, and a pre-signature* $\widetilde{\sigma}$*, and outputs a bit* b*.*

(iii) $\sigma := \mathsf{Adapt}(\mathsf{vk}, m, y, \widetilde{\sigma})$*: The adapt algorithm* Adapt *takes as input the verification key* vk*, a message* m*, a witness* y *for statement* Y*, and a pre-signature* $\widetilde{\sigma}$*, and uses* y *to adapt* $\widetilde{\sigma}$ *into a valid signature* σ*. It then outputs the signature* σ*.*

(iv) $y := \mathsf{Ext}(\mathsf{vk}, Y, \widetilde{\sigma}, \sigma)$*: The extract algorithm* Ext *takes as input the verification key* vk*, a statement* Y*, a pre-signature* $\widetilde{\sigma}$*, and a signature* σ*, and uses* Y *to extract witness* y *such that* $(Y, y) \in \mathsf{R}$*. It then outputs the extracted witness* y*.*

In order for an adaptor signature scheme to be secure, it must satisfy four properties: correctness *(Definition 3.25),* pre-signature adaptability *(Definition 3.26),* unforgeability *(Definition 3.27), and* witness extractability *(Definition 3.28).*

Definition 3.25 (Correctness)**.** *An adaptor signature scheme* $\mathsf{AS}_{\mathsf{DS},\mathsf{R}}$ *satisfies* correctness *if for every* $\lambda \in \mathbb{N}$*,* $m \in \{0,1\}^*$*, and* $(Y, y) \in \mathsf{R}$*:*

$$\Pr\left[\begin{array}{c|c} \mathsf{pVrfy}(\mathsf{vk}, m, Y, \widetilde{\sigma}) = 1 & (\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda) \\ \wedge\ \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1 & \widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y) \\ \wedge\ (Y, y') \in \mathsf{R} & \sigma := \mathsf{Adapt}(\mathsf{vk}, m, y, \widetilde{\sigma}) \\ & y' := \mathsf{Ext}(\mathsf{vk}, Y, \widetilde{\sigma}, \sigma) \end{array}\right] = 1 \ .$$

Definition 3.26 (Pre-Signature Adaptability)**.** *An adaptor signature scheme* $\mathsf{AS}_{\mathsf{DS},\mathsf{R}}$ *satisfies* pre-signature adaptability *if for every* $\lambda \in \mathbb{N}$*,* $m \in \{0,1\}^*$*, and* $(Y, y) \in \mathsf{R}$*:*

$$\Pr\left[\begin{array}{c|c} \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1 & (\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda), \\ & \mathsf{pVrfy}(\mathsf{vk}, m, Y, \widetilde{\sigma}) = 1, \\ & \sigma := \mathsf{Adapt}(\mathsf{vk}, m, y, \widetilde{\sigma}) \end{array}\right] = 1 \ .$$

Definition 3.27 (Unforgeability)**.** *An adaptor signature scheme* $\mathsf{AS}_{\mathsf{DS},\mathsf{R}}$ *satisfies* unforgeability *if for every p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\mathsf{aUnforge}_{\mathcal{A},\mathsf{AS}_{\mathsf{DS},\mathsf{R}}}(1^\lambda) = 1\right] \leq \epsilon(\lambda) \ ,$$

where $\mathsf{aUnforge}_{\mathcal{A},\mathsf{AS}_{\mathsf{DS},\mathsf{R}}}$ *is defined in Figure 12.*

Experiment $\mathsf{aUnforge}_{\mathcal{A},\mathsf{AS}_{\mathsf{DS},\mathsf{R}}}(1^\lambda)$	Oracle $\mathcal{O}_s(m)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$Q := \emptyset$	$Q := Q \cup \{m\}$
$(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{ps}}(\mathsf{vk})$	return σ
$\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m^*, Y^*)$	Oracle $\mathcal{O}_{ps}(m, Y)$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{ps}}(\mathsf{vk}, \widetilde{\sigma})$	$\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$
return $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$	$Q := Q \cup \{m\}$
	return $\widetilde{\sigma}$

Figure 12: Unforgeability experiment for Adaptor Signature Schemes

Definition 3.28 (Witness Extractability)**.** *An adaptor signature scheme* AS *satisfies* witness extractability *if for every p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\mathsf{witExt}_{\mathcal{A},\mathsf{AS}_{\mathsf{DS},\mathsf{R}}}(1^\lambda) = 1\right] \leq \epsilon(\lambda) \ ,$$

where $\mathsf{witExt}_{\mathcal{A},\mathsf{AS}_{\mathsf{DS},\mathsf{R}}}$ *is defined in Figure 13.*

3.5.1 Implementation: Schnorr Adaptor Signature Scheme

One of the most common forms of Adaptor Signatures is the Schnorr Adaptor Signature. As with the Schnorr Signature scheme, the Schnorr Adaptor Signature Scheme utilizes the hardness of the discrete log problem to create a secure signature scheme. We denote the Schnorr Adaptor Signature Scheme by AS_{sch}, and is defined in Definition 3.29.

Definition 3.29 (Schnorr Adaptor Signature Scheme (AS_{sch}))**.** *The Schnorr Adaptor Signature Scheme, denoted* AS_{sch}*, is defined with respect to the Schnorr Digital Signature Scheme* DS_{sch} *(as in Definition 3.23) and the hard relation* R_{DL} *(as defined in Definition 3.3). As before,* $\mathbb{G}$ *is a cyclic group with prime order* p *where the* DL *problem is hard, and* g *is a generator of* $\mathbb{G}$*. The hash function* H *is defined as before. The* $\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}$ *and* Ext *algorithms for* AS_{sch} *are defined in Figure 14.*

Experiment $\mathsf{witExt}_{\mathcal{A},\mathsf{AS}_{\mathsf{DS},\mathsf{R}}}(1^\lambda)$	Oracle $\mathcal{O}_s(m)$
1 : $(\mathsf{sk},\mathsf{vk}) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)$	1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $Q := \emptyset$	2 : $Q := Q \cup \{m\}$
3 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_s,\mathcal{O}_{ps}}(\mathsf{vk})$	3 : return σ
4 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m^*, Y^*)$	Oracle $\mathcal{O}_{ps}(m, Y)$
5 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s,\mathcal{O}_{ps}}(\mathsf{vk}, \widetilde{\sigma})$	1 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$
6 : $y' := \mathsf{Ext}(\mathsf{vk}, Y^*, \widetilde{\sigma}, \sigma^*)$	2 : $Q := Q \cup \{m\}$
7 : return $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (Y^*, y') \notin \mathsf{R}$	3 : return $\widetilde{\sigma}$

Figure 13: Witness Extractability experiment for Adaptor Signature Schemes

$\mathsf{pSign}(\mathsf{sk}, m, Y)$	$\mathsf{pVrfy}(\mathsf{vk}, m, Y, \widetilde{\sigma})$	$\mathsf{Adapt}(\mathsf{vk}, m, y, \widetilde{\sigma})$
1 : $r \leftarrow\!\$\ \mathbb{Z}_p$	1 : Parse $\widetilde{\sigma}$ as $(R_{sign}, \widetilde{s})$	1 : Parse $\widetilde{\sigma}$ as $(R_{sign}, \widetilde{s})$
2 : $R_{pre} := g^r$	2 : $h' := H(R_{sign} \| m)$	2 : $s = \widetilde{s} + y$
3 : $R_{sign} := R_{pre} \cdot Y$	3 : $R_{pre} := R_{sign} \cdot Y^{-1}$	3 : return $\sigma := (R_{sign}, s)$
4 : $h := H(R_{sign} \| m)$	4 : return $g^{\widetilde{s}} \stackrel{?}{=} R_{pre} \cdot \mathsf{vk}^{h'}$	$\mathsf{Ext}(\mathsf{vk}, Y, \widetilde{\sigma}, \sigma)$
5 : $\widetilde{s} := h \cdot \mathsf{sk} + r$		1 : Parse $\widetilde{\sigma}$ as $(R_{sign}, \widetilde{s})$
6 : return $\widetilde{\sigma} := (R_{sign}, \widetilde{s})$		2 : Parse σ as (R_{sign}, s)
		3 : return $y = s - \widetilde{s}$

Figure 14: Definition for Schnorr Adaptor Signature Scheme AS_{sch}

3.6 Non-Interactive Zero Knowledge (NIZK) Arguments

Definition 3.30 (Non-Interactive Zero Knowledge (NIZK) Argument)**.** *A* NIZK *argument for a language* L_{R}*, where* R *is a relation between statement/witness pairs, in the common reference string (CRS) model is defined by the three algorithms:*

(i) $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes the security parameter* λ *as input, and outputs a common reference string* crs *which is then released publicly. The* crs *will be used to prove statements non-interactively.*

(ii) $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*: The* Prove *algorithm takes as input the common reference string* crs*, a statement* stmt*, and a witness* wit*, and outputs a corresponding argument* π*.*

(iii) $b \in \{0,1\} \leftarrow \mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \pi)$*: The verification algorithm takes as input the common reference string* crs*, a statement* stmt*, and an argument* π*, and outputs a bit* b *indicating the validity of the argument.*

In order for a NIZK *argument system to be secure, it must satisfy three properties:* completeness *(Definition 3.31),* adaptive soundness *(Definition 3.32), and* zero-knowledge *(Definition 3.33).*

Definition 3.31 (Completeness)**.** *A* NIZK *argument for a language* L_{R} *satisfies* completeness *if for all statement/witness pairs* $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{R}$*, we have that if* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*, then:*

$$\Pr[\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1] = 1 \ .$$

Definition 3.32 (Adaptive Soundness)**.** *A* NIZK *argument for a language* L_{R} *satisfies* adaptive soundness *if for all p.p.t. provers* $\mathcal{P}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*, we have that if* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{stmt}, \pi) \leftarrow \mathcal{P}(\mathsf{crs})$*, then:*

$$\Pr[\mathsf{stmt} \notin L_{\mathsf{R}} \wedge \mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1] \leq \epsilon(\lambda) \ .$$

Definition 3.33 (Zero Knowledge)**.** *A* NIZK *argument for a language* L_{R} *satisfies* zero knowledge *if there exists a p.p.t. simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Prove}^*)$ *and there exists a negligible function* ϵ *such that for all p.p.t. distinguishers* $\mathcal{D}$*,* $\lambda \in \mathbb{N}$*, and statement/witness pairs* $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{R}$*, we have that if* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*,* $(\mathsf{crs}^*, \mathsf{td}) \leftarrow \mathsf{Setup}^*(1^\lambda)$*, and* $\pi^* \leftarrow \mathsf{Prove}^*(\mathsf{crs}^*, \mathsf{stmt}, \mathsf{td})$*, then:*

$$\left| \Pr[\mathcal{D}(\mathsf{crs}, \pi) = 1] - \Pr[\mathcal{D}(\mathsf{crs}^*, \pi^*) = 1] \right| \leq \epsilon(\lambda) \ ,$$

where td *is a trapdoor that the simulator* Sim *uses to generate the argument* π^* *for the statement* stmt *without knowing the corresponding witness* wit*.*

Note that NIZK arguments are known from several assumptions [FLS90, CHK03, GOS, PS19, CCH+19]. Later in the paper (Section 6), for the zero knowledge proof of equality, we will need to prove in zero knowledge that a given ElGamal encryption ct encrypts a bit $b \in \{0, 1\}$. Boneh and Shoup provide a zero knowledge protocol for this (Example 20.3 [BS17]), and it relies on the Chaum-Pedersen protocol that proves a given triple is a Diffie-Hellman triple. For completeness, we include the Chaum-Pedersen protocol here (considering the version in Boneh-Shoup Section 19.5.2 [BS17]).

Definition 3.34 (Chaum-Pedersen Protocol)**.** *Let* $\mathbb{G}$ *be a cyclic group of prime order* p *with some generator* $g \in \mathbb{G}$*. For some* $\alpha, \beta, \gamma \in \mathbb{Z}_p$*, we say that* $(g^\alpha, g^\beta, g^\gamma)$ *is a DH-triple if* $\gamma = \alpha\beta$*. Similarly, we can then say a triple* $(u, v, w) \in \mathbb{G}^3$ *is a DH-triple if and only if there exists a* $\beta \in \mathbb{Z}_p$ *such that* $v = g^\beta$*, and* $w = u^\beta$*. The relation for the Chaum-Pedersen Protocol is then:*

$$\mathsf{R}_{\mathsf{cp}} = \{(\beta, (u, v, w)) \in \mathbb{Z}_p \times \mathbb{G}^3 \mid v = g^\beta \wedge w = u^\beta\} \ .$$

The three rounds of the sigma protocol Π_{cp} *are as follows:*

1. *The prover samples* $\beta_t \leftarrow\!\!\$\ \mathbb{Z}_p$*, and sets* $v_t \leftarrow g^{\beta_t}$ *and* $w_t \leftarrow u^{\beta_t}$*, sending* (v_t, w_t) *to the verifier.*
2. *The verifier samples a challenge* $c \leftarrow\!\!\$\ \mathcal{C}$*, and sends* c *to the prover.*
3. *The prover calculates* $\beta_z = \beta_t + \beta \cdot c$*, and sends* β_z *to the verifier.*

The verifier then does two checks to verify that (u, v, w) *is a DH-triple:*

(1) $g^{\beta_z} = v_t \cdot v^c$

(2) $u^{\beta_z} = u_t \cdot u^c$

Boneh and Shoup also provide a proof for the special soundness of the Π_{cp} protocol. The idea is that they can construct an extractor so that when given two accepting conversations $((v_t, w_t), c, \beta_z)$ and $((v_t, w_t), c', \beta'_z)$ where $c \neq c'$, a witness β can be extracted, where:

$$\beta = \Delta\beta / \Delta c,\ \Delta\beta = \beta_z - \beta'_z, \text{ and } \Delta c = c - c' \ .$$

For completeness, we include a sketch of the protocol to prove that an ElGamal encryption ct encrypts a bit. As stated before, it utilizes the Chaum-Pedersen protocol.

Definition 3.35 (Encrypted ElGamal Bits Relation)**.** *Let $\mathbb{G}$ be a cyclic group of prime order p with some generator $g \in \mathbb{G}$. Suppose we have a bit ElGamal encryption $(v, e) = (g^\beta, g^b u^\beta)$. We can show that (v, e) is a valid ElGamal encryption of a bit by proving the following relation:*

$$\mathsf{R}_{\mathsf{egb}} = \left\{ \big((b, \beta), (u, v, e)\big) \;\middle|\; \begin{array}{c} v = g^\beta \wedge e = u^\beta g^b \\ \wedge\ b \in \{0, 1\} \end{array} \right\} .$$

The protocol for the above relation relies on the observation that either (u, v, e) or $(u, v, e/g)$ is a DH-triple. So we can simply apply the Chaum-Pedersen protcol to both tuples (u, v, e) and $(u, v, e/g)$, and show that one of them is a valid DH-triple. This approach is enabled by the OR-proof construction found in Boneh-Shoup (Section 19.7.2 [BS17]). Throughout the paper, we will denote this sigma protocol as Π_{egb}.

3.7 Homomorphic Encryption

Definition 3.36 (Homomorphic Property of Encryption Schemes)**.** *An encryption scheme is said to be* homomorphic *with respect to some operation $*$ if for every message m_1, m_2:*

$$\mathsf{Enc}(m_1) * \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 * m_2) \ .$$

Definition 3.37 (Homomorphic Encryption Scheme)**.** *A Homomorphic Encryption Scheme is denoted* $\mathsf{HE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$*, where the algorithms are defined as follows (note that here we assume the method uses asymmetric keys):*

(i) $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$*: The key generation algorithm* $\mathsf{KGen}(1^\lambda)$ *takes as input the security parameter, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*

(ii) $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: The encryption algorithm* Enc *takes as input the public key* pk*, and a message m, and outputs the encryption* ct *of m under the key* pk*.*

(iii) $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: The decryption algorithm* Dec *takes as input the secret key* sk*, and a ciphertext* ct*, and outputs the plaintext m' from the decryption of* ct*.*

(iv) $\widehat{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, f)$*: The evaluation algorithm* Eval *takes as input the public key* pk*, a ciphertext* ct*, and a function f, and outputs the result of* $\widehat{\mathsf{ct}} = f(\mathsf{ct})$*. Note that the function f should preserve the structure of the encryption, and* $\widehat{\mathsf{ct}}$ *can be decrypted so that the value $f(m)$ is extracted, where m is the corresponding plaintext to* ct*.*

Remark. Different HE schemes have different levels of homomorphism, and it is important to identify the differences between them. We further classify HE schemes into three main groups, as defined in Definition 3.38. Note that for each classification, the encryption schemes still have the same structure as detailed in Definition 3.37, but the way that the Eval algorithm is implemented depends on which classification the scheme falls into.

Definition 3.38 (Classification of Homomorphic Encryption Schemes)**.** *We classify homomorphic encryption schemes into three main groups:*

(i) Partially Homomorphic Encryption (PHE)*: An encryption scheme that is homomorphic with respect to* one operation*, and can perform the operation an* unlimited *number of times.*

(ii) Somewhat Homomorphic Encryption (SWHE)*: An encryption scheme that is homomorphic with respect to* more than one operation*, but can only perform the operations a* limited *number of times.*

(iii) Fully Homomorphic Encryption (FHE)*: An encryption scheme that is homomorphic with respect to an* unlimited number of operations*, and can perform the operations an* unlimited *number of times.*

An example of SWHE would be the BGN encryption scheme [BGN05]. It is homomorphic with respect to addition and multiplication, but can only do one multiplication.

4 Definition of Functional Adaptor Signatures (FAS)

In this section, we formalize FAS's syntax and security properties. FAS builds upon the cryptographic primitives of digital signatures and adaptor signatures, which we define in Section 3.4 and Section 3.5, respectively.

Definition 4.1 (FAS [VST24a])**.** *A FAS scheme* FAS *is defined w.r.t. a signature scheme* DS *, a hard relation* R*, and a family of functions* $\mathcal{F}$*, and consists of algorithms* Setup*,* AdGen*,* AdVerify*,* FPreSign*,* FPreVerify*,* Adapt*,* FExt *defined below:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: on input the security parameter* 1^λ*,* Setup *outputs the public parameters* pp*.*
- $(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, Y, y)$*: on inputs public parameters* pp*, a statement* Y *for the language* L_R*, and a witness* y*,* AdGen *outputs a public advertisement* ad *and a secret state* st*.*
- $b \leftarrow \mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y)$*: on inputs public parameters* pp*, an advertisement* ad*, and a statement* Y *for the language* L_R*,* AdVerify *outputs a bit* $b \in \{0,1\}$ *checking validity of* ad*.*
- $(\widetilde{\sigma}_1, \widetilde{\sigma}_2) \leftarrow \mathsf{FPreSign}$*, defined as* $\mathsf{FPreSign} = (\mathsf{FPreSign}_1, \mathsf{FPreSign}_2)$ *models a two-round protocol where the seller runs* $\mathsf{FPreSign}_1$*, and the buyer runs* $\mathsf{FPreSign}_2$*. We denote* $\widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)$*, and call* $\widetilde{\sigma}$ *a functional pre-signature. Formally, these algorithms are defined as follows:*
  - $\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, y, f)$*: on input public parameters* pp*, an advertisement* ad*, a secret state* st*, a statement/witness pair* (Y, y) *for the language* L_R*, and a function* f *from family* $\mathcal{F}$*,* $\mathsf{FPreSign}_1$ *outputs* $\widetilde{\sigma}_1$*.*
  - $\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$*: on input public parameters* pp*, an advertisement* ad*, a signing key* sk*, a message* m*, a statement* Y *for the language* L_R*, a function* f *from family* $\mathcal{F}$*, and* $\widetilde{\sigma}_1$*,* $\mathsf{FPreSign}_2$ *outputs* $\widetilde{\sigma}_2$*.*
- $(b_1, b_2) \leftarrow \mathsf{FPreVerify}$*, is defined as* $\mathsf{FPreVerify} = (\mathsf{FPreVerify}_1, \mathsf{FPreVerify}_2)$*, where the buyer runs* $\mathsf{FPreVerify}_1$*, and the seller runs* $\mathsf{FPreVerify}_2$*. Formally, these algorithms are defined as follows:*
  - $b_1 \leftarrow \mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1)$*: on input public parameters* pp*, an advertisement* ad*, a statement* Y *for the language* L_R*, a function* f *from family* $\mathcal{F}$*, and* $\widetilde{\sigma}_1$*,* $\mathsf{FPreVerify}_1$ *outputs a bit* $b_1 \in \{0,1\}$ *checking* $\widetilde{\sigma}_1$*'s validity.*
  - $b_2 \leftarrow \mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2))$*: on input public parameters* pp*, an advertisement* ad*, a verification key* vk*, a message* m*, a statement* Y *for the language* L_R*, a function* f *from family* $\mathcal{F}$*, and a functional pre-signature* $\widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)$*,* $\mathsf{FPreVerify}_2$ *outputs a bit* $b_2 \in \{0,1\}$ *checking* $\widetilde{\sigma}_2$*'s validity*
- $\sigma := \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2))$*: on inputs an advertisement* ad*, a secret state* st*, a verification key* vk*, a message* m*, a statement/witness pair* (Y, y) *for the language* L_R*, a function* f *from family* $\mathcal{F}$*, and a functional pre-signature* $\widetilde{\sigma}$*,* Adapt *outputs a signature* σ*.*
- $z := \mathsf{FExt}(\mathsf{ad}, \mathsf{vk}, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2), \sigma)$*: on inputs an advertisement* ad*, a verification key* vk*, a statement* Y *for the language* L_R*, a function* f *from family* $\mathcal{F}$*, a functional pre-signature* $\widetilde{\sigma}$*, and a signature* σ*,* FExt *outputs a value* z*.*

4.1 Security Properties of FAS

We now provide formal definitions for the security properties of FAS, inspired by the security definitions from [VST24a]. First, we introduce correctness (Definition 4.2), which ensures that in an honest execution of FAS, FExt must return $z = f(y)$.

Definition 4.2 (Correctness)**.** *A functional adaptor signature scheme* FAS *satisfies correctness if for every* $\lambda \in \mathbb{N}$*, every message* $m \in \{0,1\}^*$*, every statement/witness pair* $(Y, y) \in \mathsf{R}$*, and every function* $f \in \mathcal{F}$*, we have that:*

$$\Pr\left[\begin{array}{c} \mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1 \\ \wedge\ \mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1) = 1 \\ \wedge\ \mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma}) = 1 \\ \wedge \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1 \wedge z = f(y) \end{array}\right] = 1 \ ,$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, Y, y)$*,* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$*,* $\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, y, f)$*,* $\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$*,* $\sigma := \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2))$*, and* $z := \mathsf{FExt}(\mathsf{ad}, \mathsf{vk}, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2), \sigma)$*.*

The remaining security properties of FAS are grouped into two categories: those against a corupt seller, and those against a corrupt buyer.

4.1.1 Security Against Corrupt Sellers

For corrupt sellers, FAS must satisfy the properties of: *advertisement soundness*, *advertisement binding*, *pre-signature validity*, *unforgeability*, and *witness extractability*.

Advertisement Soundness. This property requires that given public parameters pp, it should be infeasible for a p.p.t. malicious seller to produce a statement $Y \in L_\mathsf{R}$ and an advertisement ad such that ad is accepted by $\mathsf{AdVerify}$. The formal description is provided in Definition 4.3.

Definition 4.3 (Advertisement Soundness)**.** *A functional adaptor signature scheme* FAS *satisfies advertisement soundness if there exists a negligible function* ϵ *such that for every p.p.t. malicious seller* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, for all NP languages* L_R*, if* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $(Y, \mathsf{ad}) \leftarrow \mathcal{A}(\mathsf{pp})$*, then we have that:*

$$\Pr[(Y \notin L_\mathsf{R}) \wedge (\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1)] \leq \epsilon(\lambda) \ .$$

Advertisement Binding. This property ensures that if a p.p.t. adversary produces a statement $Y \in L_\mathsf{R}$ and a corresponding advertisment ad such that (ad, Y) passes $\mathsf{AdVerify}$, then ad is bound to a specific witness y. The formal description is provided in Definition 4.4.

Definition 4.4 (Advertisement Binding)**.** *A functional adaptor signature scheme* FAS *satisfies advertisement binding if there exists a negligible function* ϵ *such that for every p.p.t. malicious seller* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, if* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(Y, \mathsf{ad}) \leftarrow \mathcal{A}^*(\mathsf{pp})$*, such that* $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$*, then both the following hold:*

$$\Pr\left[\begin{array}{c} \exists (y, r), (y', r')\ s.t. \\ y \neq y' \wedge \\ \mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot) \wedge \\ \mathsf{AdGen}(\mathsf{pp}, Y, y'; r') = (\mathsf{ad}, \cdot) \end{array}\right] \leq \epsilon(\lambda) \ , \ \Pr\left[\begin{array}{c} \nexists (y, r)\ s.t. \\ \mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot) \end{array}\right] \leq \epsilon(\lambda) \ .$$

Informally, both above equations ensure that every (Y, ad) *(even maliciously chosen) has exactly one tuple* (y, r) *that results in* ad *via* AdGen*. Such a tuple can be computed (possibly inefficiently) by brute-force search algorithm which we refer to as the extractor* $\mathcal{E}$*.*

Experiment $\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}(1^\lambda)$	Oracle $\mathcal{O}_{\mathsf{s}}(m)$
$Q := \emptyset, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$	$Q := Q \cup \{m\}$
$(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$	return σ
$(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{s}}, \mathcal{O}_{\mathsf{fps}_2}}(\mathsf{pp}, \mathsf{vk}, Y^*)$	**Oracle** $\mathcal{O}_{\mathsf{fps}_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$
if $\big(\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}^*, Y^*) = 0 \vee (f^* \notin \mathcal{F})$ $\vee\, \mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}^*, Y^*, f^*, \widetilde{\sigma}_1^*) = 0\big)$: **return** $\perp$	if $(\mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1) = 0)$: **return** $\perp$
$\widetilde{\sigma}_2^* \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}^*, \mathsf{sk}, m^*, Y^*, f^*, \widetilde{\sigma}_1^*)$	$\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{s}}, \mathcal{O}_{\mathsf{fps}_2}}(\widetilde{\sigma}_2^*)$	$Q := Q \cup \{m\}$
return $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$	return $\widetilde{\sigma}_2$

Figure 15: Unforgeability experiment for Functional Adaptor Signature Schemes

Pre-Signature Validity. This property ensures that given a partial pre-signature $\widetilde{\sigma}_1$ such that it passes $\mathsf{FPreVerify}_1$, one can always use $\widetilde{\sigma}_1$ to produce $\widetilde{\sigma}_2$ via $\mathsf{FPreSign}_2$ such that it passes $\mathsf{FPreVerify}_2$. The formal description is provided in Definition 4.5.

Definition 4.5 (Pre-Signature Validity)**.** *A functional adaptor signature scheme* FAS *satisfies pre-signature validity if for every* $\lambda \in \mathbb{N}$*, any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, any statement/witness pair* $(Y, y) \in \mathsf{R}$*, any message* $m \in \{0,1\}^*$*, any family of functions* $\mathcal{F}$*, every function* $f \in \mathcal{F}$*, any* ad *such that* $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$*, any* $\widetilde{\sigma}_1$ *such that* $\mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1) = 1$*, and any* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$*, we have that:*

$$\Pr[\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 1] = 1 \ ,$$

where $\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$*.*

Unforgeability. This property ensures that adversary who doesn't know the witness cannot forge honest buyers' signatures, even after seeing multiple valid pre-signatures. The formal description is provided in Definition 4.6.

Definition 4.6 (Unforgeability)**.** *A functional adaptor signature scheme* FAS *satisfies unforgeability if for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\Big[\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\Big] \leq \epsilon(\lambda) \ ,$$

where $\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}$ *is defined in Figure 15.*

Witness Extractability. This property ensures that the adversary cannot publish a buyer's signature such that when the buyer tries to extract using FExt, it gets $z' \neq f(y)$. The formal description is provided in Definiton 4.7.

Definition 4.7 (Witness Extractability)**.** *A functional adaptor signature scheme* FAS *satisfies witness extractability if for every p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\Big[\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\Big] \leq \epsilon(\lambda) \ ,$$

where $\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}$ *is defined in Figure 16.*

Experiment $\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}(1^\lambda)$

$Q := \emptyset, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$

$(\mathsf{ad}^*, m^*, Y^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\mathsf{pp}, \mathsf{vk})$

if $\big(\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}^*, Y^*) = 0 \vee (f^* \notin \mathcal{F})$
$\vee\, \mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}^*, Y^*, f^*, \widetilde{\sigma}_1^*) = 0\big)$:
return $\perp$

$\widetilde{\sigma}_2^* \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}^*, \mathsf{sk}, m^*, Y^*, f^*, \widetilde{\sigma}_1^*)$

$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\widetilde{\sigma}_2^*)$

$z := \mathsf{FExt}(\mathsf{ad}^*, \mathsf{vk}, Y^*, f^*, (\widetilde{\sigma}_1^*, \widetilde{\sigma}_2^*), \sigma^*)$

return $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$
$\wedge \big(z \notin \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}\big)$

Oracle $\mathcal{O}_s(m)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$

$Q := Q \cup \{m\}$

return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

if $(\mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1) = 0)$:
return $\perp$

$\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$

$Q := Q \cup \{m\}$

return $\widetilde{\sigma}_2$

Figure 16: Witness Extractability experiment for Functional Adaptor Signature Schemes

4.1.2 Security Against Corrupt Buyers

For corrupt buyers, FAS must satisfy the properties of: *pre-signature adaptability*, *zero-knowledge* (non-abort), and *zero-knowledge* (abort).

Pre-Signature Adaptability. This property ensures that if the adversary outputs a valid functional pre-signature, then adapting it using a valid witness y should result in a valid signature under the buyer's key. The formal description is provided in Definition 4.8.

Definition 4.8 (Pre-Signature Adaptability)**.** *A functional adaptor signature scheme* FAS *satisfies pre-signature adaptability if for every* $\lambda \in \mathbb{N}$*, any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, any statement/witness pair* $(Y, y) \in \mathsf{R}$*, any* $(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, \mathsf{ad}, Y)$*, any message* $m \in \{0,1\}^*$*, any family of functions* $\mathcal{F}$*, any function* $f \in \mathcal{F}$*, any* $\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1)$*, any key pair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$*, and any pre-signature* $\widetilde{\sigma}_2$ *with* $\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 1$*, we have that:*

$$\Pr[\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma}))] = 1 \ .$$

Zero-Knowledge. Next, we define zero-knowledge of FAS. This property is defined with respect to a malicious buyer and is defined using a simulator that simulates view for the buyer using only the information about the witness y that the malicious buyer gets to learn anyway. We consider two cases here: (i) malicious buyer does not abort. In this case, the buyer must learn only $f(y)$ about y and the simulator will have access to this information only. (ii) the malicious buyer aborts the protocol before providing $\widetilde{\sigma_2}$ to the honest seller. In this case, the buyer must learn nothing about y and the simulator will also have no information about y. We provide the formal description for the non-abort case in Definition 4.9, and the abort case in Definition 4.10.

Definition 4.9 (Zero Knowledge (non-abort))**.** *A* FAS *scheme satisfies zero knowledge (non-abort) if there exists a negligible function* ϵ *such that for every stateful p.p.t. malicious buyer* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, for all* $(Y, y) \in \mathsf{R}$*, there exists a stateful p.p.t. simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{FPreSign}_1^*, \mathsf{Adapt}^*)$

Experiment $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$	Experiment $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{pp} \leftarrow \mathsf{Setup}^*(1^\lambda)$
$(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, Y, y)$	$(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}^*(\mathsf{pp}, Y)$
$(m, f, \mathsf{vk}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ad}, Y)$	$(m, f, \mathsf{vk}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ad}, Y)$
while $\mathcal{A}$ is not done making queries:	while $\mathcal{A}$ is not done making queries:
$\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, y, f)$	$\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}^*_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, f(y), f)$
$\widetilde{\sigma}_2 \leftarrow \mathcal{A}(\widetilde{\sigma}_1)$	$\widetilde{\sigma}_2 \leftarrow \mathcal{A}(\widetilde{\sigma}_1)$
if $(\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 0)$: **return** $\perp$	if $(\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 0)$: **return** $\perp$
$\sigma := \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, (\widetilde{\sigma}_1, \widetilde{\sigma}_2))$	$\sigma := \mathsf{Adapt}^*(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, f(y), f, (\widetilde{\sigma}_1, \widetilde{\sigma}_2))$
$(m, f) \leftarrow \mathcal{A}(\sigma)$	$(m, f) \leftarrow \mathcal{A}(\sigma)$
return view of $\mathcal{A}$	return view of $\mathcal{A}$

Figure 17: Zero Knowledge (non-abort) experiments for FAS

such that the following holds for any p.p.t. distinguisher $\mathcal{D}$:

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{V}_{\mathcal{A}}\right) = 1 : \mathsf{V}_{\mathcal{A}} \leftarrow \mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right] - \Pr\left[\mathcal{D}\left(\mathsf{V}_{\mathcal{A}}\right) = 1 : \mathsf{V}_{\mathcal{A}} \leftarrow \mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right] \right| \leq \epsilon(\lambda) \ ,$$

where $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ are defined as in Figure 17 and $\mathsf{V}_{\mathcal{A}}$ is defined as the view of $\mathcal{A}$ in the two games respectively.

Definition 4.10 (Zero Knowledge (abort))**.** *A* FAS *scheme satisfies zero knowledge (abort) if there exists a negligible function ϵ such that for every stateful p.p.t. malicious buyer $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, for all $(Y, y) \in \mathsf{R}$, there exists a stateful p.p.t. simulator $\mathsf{Sim}^{\mathsf{a}} = (\mathsf{Setup}^{*,\mathsf{a}}, \mathsf{AdGen}^{*,\mathsf{a}}, \mathsf{FPreSign}^{*,\mathsf{a}}_1)$ such that the following holds for any p.p.t. distinguisher $\mathcal{D}$:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{V}_{\mathcal{A}}\right) = 1 : \mathsf{V}_{\mathcal{A}} \leftarrow \mathsf{fZKReal}^{\mathsf{a}}{}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right] - \Pr\left[\mathcal{D}\left(\mathsf{V}_{\mathcal{A}}\right) = 1 : \mathsf{V}_{\mathcal{A}} \leftarrow \mathsf{fZKIdeal}^{\mathsf{a},\mathsf{Sim}^{\mathsf{a}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right] \right| \leq \epsilon(\lambda) \ ,$$

where $\mathsf{fZKReal}^{\mathsf{a}}_{\mathcal{A},\mathsf{FAS}}$ (defined as in Figure 18) is real-world game where the adversary $\mathcal{A}$ aborts after receiving $\widetilde{\sigma_1}$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ (defined as in Figure 18) is ideal-world game where the simulator simulates the adversary $\mathcal{A}$'s view up to receiving $\widetilde{\sigma_1}$, and $\mathsf{V}_{\mathcal{A}}$ is defined as the view of $\mathcal{A}$ in the two games respectively.

Remark 4.11. *Crucially, notice that $\mathsf{FPreSign}^*_1$ in $\mathsf{fZKIdeal}$ game (Figure 17) gets $f(x)$ as input, whereas $\mathsf{FPreSign}^{*,\mathsf{a}}_1$ in $\mathsf{fZKIdeal}^{\mathsf{a}}$ game (Figure 18) does not.*

5 FAS for General Functions from FHE

In this section, we build a FAS construction for general functions from FHE. Our FAS construction is defined w.r.t. the Schnorr Signature scheme, any hard relation R with statement/witness pairs

Experiment $\mathsf{fZKReal}^{\mathsf{a}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$	Experiment $\mathsf{fZKIdeal}^{\mathsf{a},\mathsf{Sim}^{\mathsf{a}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{pp} \leftarrow \mathsf{Setup}^{*,\mathsf{a}}(1^\lambda)$
$(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, Y, y)$	$(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}^{*,\mathsf{a}}(\mathsf{pp}, Y)$
$(m, f, \mathsf{vk}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ad}, Y)$	$(m, f, \mathsf{vk}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ad}, Y)$
while $\mathcal{A}$ is not done making queries:	while $\mathcal{A}$ is not done making queries:
$\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, y, f)$	$\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1^{*,\mathsf{a}}(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, f)$
$(m, f) \leftarrow \mathcal{A}(\widetilde{\sigma}_1)$	$(m, f) \leftarrow \mathcal{A}(\widetilde{\sigma}_1)$
return view of $\mathcal{A}$	return view of $\mathcal{A}$

Figure 18: Zero Knowledge (abort) experiments for FAS

(Y, y) where witness y is a vector of integers, and the function family $\mathcal{F}_{\mathsf{circuits}}$ of unbounded depth polynomial-size circuits. The construction uses the following building blocks and is detailed in Figure 20: (a) semantically-secure homomorphic encryption scheme FHE, (b) the Elgamal public-key encryption scheme EG, (c) two NIZK argument systems $\mathsf{NIZK}_{\mathsf{ad}}$ and $\mathsf{NIZK}_{\mathsf{eq}}$, and (d) the Schnorr Adaptor signature AS. Refer to the preliminaries (Section 3) for definitions of the various cryptographic primitives used as building blocks.

$\mathsf{NIZK}_{\mathsf{ad}}$ is used in the AdGen algorithm, which is run only once. So, we do not optimize it and rely on a general-purpose NIZK. In contrast, $\mathsf{NIZK}_{\mathsf{eq}}$ is invoked in the $\mathsf{FPreSign}_1$ algorithm, which is run by the seller *once* every functional sale, so it must be efficient. We present an optimized instantiation, with the remainder of this section organized as follows: Section 5.1 covers the REval algorithm we augment homomorphic encryption schemes with, Section 5.2 presents the efficient $\mathsf{NIZK}_{\mathsf{eq}}$ instantiation, and Section 5.3 details the full construction.

5.1 REval for Homomorphic Encryption Schemes

Homomorphic Encryption (HE). Informally, a homomorphic encryption scheme additionally supports homomorphism over the ciphertext space. Linear homomorphism requires that for any two messages m_0 and m_1, we have that $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m_0) * \mathsf{Enc}(\mathsf{pk}, m_1)) = m_0 + m_1$, where $+$ is the group operation of the message space and $*$ is the group operator of the ciphertext space. *Full homomorphism* generalizes this property to any function f, formalized via a public evaluation algorithm HEval that satisfies $\mathsf{Dec}(\mathsf{sk}, \mathsf{HEval}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m), f)) = f(m)$ for all m and f.

Augmenting HE syntax with REval algorithm. We extend the syntax of HE schemes with (a possibly *private*) algorithm REval for efficiently recovering the randomness corresponding to a homomorphically evaluated ciphertext. Suppose $\mathsf{ct} := \mathsf{Enc}(\mathsf{pk}, m; r)$ and $\mathsf{ct}_f := \mathsf{HEval}(\mathsf{pk}, \mathsf{ct}, f)$. Then, we define $r_f := \mathsf{REval}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, m, r, f, \mathsf{ct}_f)$. Correctness requires $\mathsf{Enc}(\mathsf{pk}, f(m); r_f) = \mathsf{ct}_f$. Since REval is not exposed to the adversary, it does not impact the semantic security of HE. For some schemes in this work (e.g., BGN and LHE), REval can recover r_f without $\mathsf{sk}, \mathsf{ct}_f$ as inputs, thus providing a more efficient alternative to compute ct_f than HEval (i.e., directly as $\mathsf{ct}_f = \mathsf{Enc}(pk, f(m); r_f)$).[7]

Remark 5.1. *While we explicitly include* sk *as input to* REval*, in most HE schemes—including FHE schemes like BGV [BGV11], GSW [GSW13], and TFHE [CGGI20], as well as low-degree schemes like BGN [BGN05]—*r_f *can be computed from* pk*,* f*, and the input randomness* r *by*

[7] For soundness, we require that all HE ciphertexts ct, even those computed through HEval, satisfy binding w.r.t. well-formed pk - that is, $\forall (m, r), (m', r')$, if $\mathsf{ct} = \mathsf{Enc}(pk, m; r) = \mathsf{Enc}(pk, m'; r')$ then $m = m'$.

mimicking HEval. *In fact, in our BGN-based instantiation for quadratic functions, the HE public-key is generated via a trusted setup, and the seller does not hold* sk *to compute* REval. *In contrast, in our TFHE-FAS, the seller selects* pk *and retains* sk. *However, due to complex bootstrapping operations,* REval *is as expensive as* HEval. *To address this, we leverage knowledge of* sk *to compute* r_f *more efficiently using techniques like key-switching. See Section 8.1 for details.*

5.2 $\mathsf{NIZK}_{\mathsf{eq}}$ Instantiation

Our FAS construction in Figure 20 relies on a NIZK proof system $\mathsf{NIZK}_{\mathsf{eq}}$ to prove that two ciphertexts, $\mathsf{ct}_{\mathsf{FHE}}$ and $\mathsf{ct}_{\mathsf{EG}}$, encrypt the same message [8]. Here, FHE and EG are encryption schemes with message space $\mathbb{Z}_p$, where p is a constant (e.g., $p = 256$ in our implementation). Since this proof is computed once per functional sale, we propose a new cut-and-choose [CD00] based proof that is optimized for this relation. Unlike other SNARK-based approaches, our focus is on reducing prover time, rather than minimizing proof size. Since our FHE-based approach aims to minimize seller computation, we prefer proof systems with faster provers. Our results in Figure 26 justify the choice of the cut-and-choose approach.

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

1 : Sample $\mathsf{hk} \leftarrow \mathsf{H.Gen}(1^\lambda)$

2 : **ret** $\mathsf{crs} := \mathsf{hk}$

$\mathsf{Prove}(\mathsf{crs}, \mathsf{stmt} = (\mathsf{ct}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{EG}}), \mathsf{wit} = (z, r, \overline{r}))$:

1 : For all $i \in [2\lambda]$: Sample messages $s_i \xleftarrow{\$} \mathbb{Z}_p$, sample random coins r_i and $\overline{r_i}$, compute $\mathsf{ct}_{\mathsf{FHE},i} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, s_i; r_i)$, and compute $\mathsf{ct}_{\mathsf{EG},i} := \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, s_i; \overline{r_i})$.

2 : Compute $str := \mathsf{H.Hash}(\mathsf{hk}, \mathsf{ct}_{\mathsf{FHE}} || \mathsf{ct}_{\mathsf{EG}} || \mathsf{ct}_{\mathsf{FHE},1} || \ldots || \mathsf{ct}_{\mathsf{FHE},2\lambda} || \mathsf{ct}_{\mathsf{EG},1} || \ldots || \mathsf{ct}_{\mathsf{EG},2\lambda}) \in \{0,1\}^\lambda$.

3 : Define set $I = \{2 \cdot i + str[i] : i \in [\lambda]\}$ of size λ.

4 : **ret** $\pi := (\{\mathsf{ct}_{\mathsf{FHE},i}, \mathsf{ct}_{\mathsf{EG},i}\}_{i\in[2\lambda]}, I, \{s_i, r_i, \overline{r_i}\}_{i\in I}, \{\widehat{s_i} := z + s_i, \widehat{r_i} := r + r_i, \widehat{\overline{r_i}} := \overline{r} + \overline{r_i}\}_{i\notin I})$.

$\mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi)$:

1 : Parse $\mathsf{stmt} = (\mathsf{ct}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{EG}})$.

2 : Parse $\pi = (\{\mathsf{ct}_{\mathsf{FHE},i}, \mathsf{ct}_{\mathsf{EG},i}\}_{i\in[2\lambda]}, I, \{s_i, r_i, \overline{r_i}\}_{i\in I}, \{\widehat{s_i}, \widehat{r_i}, \widehat{\overline{r_i}}\}_{i\notin I})$.

3 : Compute $str := \mathsf{H.Hash}(\mathsf{hk}, \mathsf{ct}_{\mathsf{FHE}} || \mathsf{ct}_{\mathsf{EG}} || \mathsf{ct}_{\mathsf{FHE},1} || \ldots || \mathsf{ct}_{\mathsf{FHE},2\lambda} || \mathsf{ct}_{\mathsf{EG},1} || \ldots || \mathsf{ct}_{\mathsf{EG},2\lambda}) \in \{0,1\}^\lambda$.

4 : **ret** 1 iff $I = \{2 \cdot i + str[i] : i \in [\lambda]\}$ and, for all $i \in I$:
$\mathsf{ct}_{\mathsf{FHE},i} = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, s_i; r_i)$,
$\mathsf{ct}_{\mathsf{EG},i} = \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, s_i; \overline{s_i})$,
and for all $i \notin I$:
$\mathsf{ct}_{\mathsf{FHE}} + \mathsf{ct}_{\mathsf{FHE},i} = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, \widehat{s_i}; \widehat{r_i})$,
$\mathsf{ct}_{\mathsf{EG}} \cdot \mathsf{ct}_{\mathsf{EG},i} = \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, \widehat{s_i}; \widehat{\overline{r_i}})$.

Figure 19: Cut-and-choose based $\mathsf{NIZK}_{\mathsf{eq}}$ for proving plaintext equality across HE and ElGamal ciphertexts.

More formally, NIZK language $L_{\mathsf{eq}}^{\mathsf{pk}_{\mathsf{FHE}}}$ is parameterized by FHE's public-key $\mathsf{pk}_{\mathsf{FHE}}$, and contains statements of the form $(\mathsf{ct}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{EG}})$ if there exists witness $(z, r, \overline{r})$ such that $\mathsf{ct}_{\mathsf{FHE}} = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, z; r)$, $\mathsf{ct}_{\mathsf{EG}} = \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, z; \overline{r})$, and $\mathsf{pk}_{\mathsf{EG}} \in \mathbb{G}$.

Suppose $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is a hash function. Then $\mathsf{NIZK}_{\mathsf{eq}}$ is as in Figure 19. We note that the prover here is extremely fast since it only has to perform 2λ FHE encryptions, 2λ El Gamal encryptions, 1 hash computation, and 3λ additions. Furthermore, the verifier is extremely fast since it only has to perform 1 hash computation, 2λ FHE encryptions, 2λ El Gamal encryptions, λ FHE ciphertext additions, λ El Gamal ciphertext additions, and 4λ equality checks. This is evidenced by our implementation numbers, as highlighted in Figure 26.[9] The formal security proof for the

[8] We also need standard range proof for proving well-formedness of El Gamal encryption of bits. See Definition 3.35 and its subsequent discussion for more details.

[9] For reasonable security level, setting $\lambda = 40$ in $\mathsf{NIZK}_{\mathsf{eq}}$ suffices.

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

1 : $\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda)$
2 : $\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
3 : **ret** $\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$

$\mathsf{AdGen}(\mathsf{pp}, Y, \vec{y})$:

1 : $(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{FHE}}) := \mathsf{FHE.KGen}(1^\lambda; r_0)$
2 : $\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, \vec{y}; r_1)$
3 : $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$
4 : $\mathsf{wit}_{\mathsf{ad}} := (\vec{y}, r_0, r_1)$
5 : $\pi_{\mathsf{ad}} \leftarrow \mathsf{Prove}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \mathsf{wit}_{\mathsf{ad}})$
6 : $\mathsf{ad} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
7 : $\mathsf{st} := (\mathsf{sk}_{\mathsf{FHE}}, r_0, r_1)$
8 : **ret** $(\mathsf{ad}, \mathsf{st})$

$\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y)$:

1 : Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
2 : $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$
3 : **ret** $\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$

$\mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, \vec{y}, f)$:

1 : Parse st as $(\mathsf{sk}_{\mathsf{FHE}}, r_0, r_1)$
2 : Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
3 : $r_2 := \mathsf{FHE.REval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \vec{y}, r_1, f, \bot)$[11]
4 : $(\mathsf{sk}_{\mathsf{EG}}, \mathsf{pk}_{\mathsf{EG}}) \leftarrow \mathsf{EG.KGen}(1^\lambda)$
5 : $z := f(\vec{y})$
6 : $(b_0, \ldots, b_k) :=$ bit-decompose z
7 : $\forall j \in \{0, \ldots, k\}$: $\mathsf{ct}_j := \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, b_j; r_{3,j})$
8 : $\vec{r_3} := (r_{3,0}, \ldots, r_{3,k})$
9 : $\vec{\mathsf{ct}} := (\mathsf{ct}_0, \ldots, \mathsf{ct}_k)$
10 : $\widehat{\mathsf{ct}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, z; r_2)$
11 : $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
12 : $\mathsf{wit}_{\mathsf{eq}} := (\mathsf{sk}_{\mathsf{FHE}}, z, r_2, \vec{r_3})$
13 : $\pi_{\mathsf{eq}} \leftarrow \mathsf{Prove}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \mathsf{wit}_{\mathsf{eq}})$
14 : Update $\mathsf{st} = (\mathsf{sk}_{\mathsf{FHE}}, r_0, r_1, \mathsf{sk}_{\mathsf{EG}})$
15 : **ret** $\widetilde{\sigma}_1 := (\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \pi_{\mathsf{eq}})$

$\mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1)$:

1 : Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
2 : Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \pi_{\mathsf{eq}})$
3 : $\widehat{\mathsf{ct}} = \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
4 : $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
5 : **ret** $\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}})$

$\mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$:

1 : Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \pi_{\mathsf{eq}})$
2 : $\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$
3 : **ret** $\widetilde{\sigma}_2$

$\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma})$:

1 : Parse $\widetilde{\sigma}$ as $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$
2 : Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \pi_{\mathsf{eq}})$
3 : **ret** $\mathsf{AS.pVrfy}(\mathsf{vk}, m, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2)$

$\mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, \vec{y}, f, \widetilde{\sigma})$:

1 : Parse st as $(\mathsf{sk}_{\mathsf{FHE}}, r_0, r_1, \mathsf{sk}_{\mathsf{EG}})$
2 : Parse $\widetilde{\sigma}$ as $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$
3 : $\sigma := \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{sk}_{\mathsf{EG}}, \widetilde{\sigma}_2)$
4 : **ret** σ

$\mathsf{FExt}(\mathsf{ad}, \mathsf{vk}, Y, f, \widetilde{\sigma}, \sigma)$:

1 : Parse $\widetilde{\sigma}$ as $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$
2 : Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \vec{\mathsf{ct}}, \pi_{\mathsf{eq}})$
3 : Parse $\vec{\mathsf{ct}}$ as $(\mathsf{ct}_0, \ldots, \mathsf{ct}_k)$
4 : $\mathsf{sk}'_{\mathsf{EG}} := \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2, \sigma)$
5 : $\forall i \in \{0, \ldots, k\}$: $b_i := \mathsf{EG.Dec}(\mathsf{sk}'_{\mathsf{EG}}, \mathsf{ct}_i)$
6 : **ret** $z := \sum_{i=0}^{k} 2^i \cdot b_i$

Figure 20: Description of algorithms in our HE-based FAS construction.

protocol in Figure 19 is provided in Section B.

5.3 FAS Construction

The proposed FAS construction is described in Figure 20. Note that we have two NIZK argument systems in the scheme, which we define as $\mathsf{NIZK}_{\mathsf{ad}} = (\mathsf{Setup}_{\mathsf{ad}}, \mathsf{Prove}_{\mathsf{ad}}, \mathsf{Verify}_{\mathsf{ad}})$ and $\mathsf{NIZK}_{\mathsf{eq}} = (\mathsf{Setup}_{\mathsf{eq}}, \mathsf{Prove}_{\mathsf{eq}}, \mathsf{Verify}_{\mathsf{eq}})$. The argument system $\mathsf{NIZK}_{\mathsf{ad}}$ is used for proving that the advertisement ad is correctly generated, and $\mathsf{NIZK}_{\mathsf{eq}}$ is used for proving that the two ciphertexts, ct and $\widehat{\mathsf{ct}}$, both encrypt the same value $z = f(y)$. The language L_{eq} is defined in Section 5.2 and L_{ad} is defined as follows: L_{ad} contains statements of the form $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$ if there exists a witness (y, r_1, r_0) such that $(Y, y) \in R$ and $\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1)$ and $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) := \mathsf{FHE.KGen}(1^\lambda; r_0)$. We assume that functions f output $(k+1)$-bit integers for some integer k. The security of our construction is

[11] In our FHE based instantiation of FAS (Section 8.1), the buyer supplies $(f, \widehat{\mathsf{ct}})$ after $\mathsf{AdVerify}$ at the beginning of exchange phase, and $\mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, \vec{y}, (f, \widehat{\mathsf{ct}}))$ algorithm computes r_2 as $r_2 := \mathsf{FHE.REval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \bot, r_1, f, \widehat{\mathsf{ct}})$. This significantly boosts efficiency of REval, as the seller does not need to recompute $\widehat{\mathsf{ct}}$ in REval internally. **Note this introduces a security bug - see Section 2.2 for more details.**

captured in Theorem 5.2.

Theorem 5.2 (Security of FAS Construction)**.** *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy zero-knowledge and adaptive soundness, the encryption schemes* FHE *and* EG *satisfy semantic security, the underlying* AS *scheme satisfies witness extractability, and relation* R *is* $\mathcal{F}$*-hard, then the proposed* FAS *construction in Figure 20 satisfies the properties of advertisement soundness, advertisement binding, pre-signature validity, unforgeability, witness extractability, pre-signature adaptability, zero-knowledge (non-abort), and zero-knowledge (abort) w.r.t.* $\mathcal{F}$*.*

Proof. In Section A, we provide formal proofs for each security property listed in Theorem 5.2. □

6 FAS for Degree-2 Functions

In this section, we provide a concrete construction of the FAS template presented in Figure 20 that enables the buyer to learn degree 2 functions over the integers. This construction utilizes the Boneh-Goh-Nissim (BGN) [BGN05] homomorphic encryption scheme to enable the degree 2 homomorphism needed for this task. As part of our construction, we develop a new sigma protocol (inspired by [COPZ22]) to prove that the values encrypted in an ElGamal ciphertext and a BGN ciphertext are equal.

The structure of this section is as follows: Section 6.1 defines the basic building blocks such as the function class and the NIZK languages, Section 6.2 introduces the construction for the sigma protocol used in the proof of equality $\mathsf{NIZK}_{\mathsf{eq}}$, and Section 6.3 provides the full construction for the BGN-FAS scheme. Note that formal security proofs for BGN-FAS are deferred to Section D.

6.1 Building Blocks

Before we can define the BGN-FAS scheme, it is important to first define the function class we consider for this scenario. We will be using a specific function class for this implementation, which is precisely the degree 2 functions over the integers whose output is sufficiently "small". By sufficiently "small", we mean that its output is bounded. We will call this class of functions $\mathcal{F}_{small}$, and it is defined as in Definition 6.1. We also define the REval algoritm in Figure 21 for this class of degree 2 functions we want to evaluate.

Definition 6.1 (Function Class $\mathcal{F}_{small}$)**.** *We say a function* $f \in \mathcal{F}_{small}$ *if* $f : \mathbb{Z}^\ell \to \mathbb{Z}$*, and the output of* f *is "small," that is, the output is bounded. Further,* f *contains no constant terms. So the functions* $f \in \mathcal{F}_{small}$ *can be represented as (for input* $\vec{x} = (x_1, x_2, \ldots, x_\ell) \in \mathbb{Z}^\ell$*):*

$$f(\vec{x}) = \left(\sum_i^\ell \sum_j^\ell \alpha_{(i,j)} \cdot x_i \cdot x_j \right) + \sum_i^\ell \beta_{(i)} \cdot x_i \ ,$$

where we have that $\alpha_{(i,j)} \neq 0$ *for some* (i, j)*, so that there is at least one degree 2 term. Further, the* $\alpha_{(i,j)}$ *and* $\beta_{(i)}$ *are scalars in* $\mathbb{Z}$*.*

Recall that in the original FAS template, there were two NIZK proofs, called $\mathsf{NIZK}_{\mathsf{ad}}$ and $\mathsf{NIZK}_{\mathsf{eq}}$. The first NIZK proof, $\mathsf{NIZK}_{\mathsf{ad}}$, ensures that the advertisement ad was generated correctly. The second NIZK proof, $\mathsf{NIZK}_{\mathsf{eq}}$, ensures that the homomorphically evaluated BGN commitment and the ElGamal ciphertext encrypt the same value. The languages for the two proofs of equality, which we denote as L_{ad} and L_{eq} respectively, are defined slightly differently for this degree 2 implementation. We define $L_{\mathsf{ad}}^{\mathsf{pk}_{\mathsf{BGN}}}$ as:

$\mathsf{BGN.REval}(\mathsf{pk} = \mathsf{ck}, \mathsf{sk} = \perp, \mathsf{ct}, \vec{x}, \vec{r}, f, \mathsf{ct}_f)$:

1 : Parse $\mathsf{ck} = (n, \mathbb{G}, \mathbb{T}, e, g, h, i)$

2 : Parse f as $\alpha_{(i,j)}$ and $\beta_{(i)}$

3 : $r_h = \left(\sum_{i=0}^{\ell} \sum_{j=0}^{\ell} \alpha_{(i,j)} \cdot (x_i \cdot r_j + x_j \cdot r_i)\right) + \sum_{i=0}^{\ell} \beta_{(i)} \cdot r_i$

4 : $r_i = \left(\sum_{i=0}^{\ell} \sum_{j=0}^{\ell} \alpha_{(i,j)} \cdot r_i \cdot r_j\right)$

5 : **ret** $r_f = (r_h, r_i)$

Figure 21: BGN.REval for degree 2 functions $f \in \mathcal{F}_{\text{small}}$

$$L_{\mathsf{ad}}^{\mathsf{pk}_{\mathsf{BGN}}} = \left\{ (\mathsf{ct}_{\mathsf{ad}}, Y) \;\middle|\; \begin{array}{c} \exists (y, r_1) \text{ such that: } (Y, y) \in \mathsf{R}, \\ \mathsf{ct}_{\mathsf{ad}} := \mathsf{BGN.Com}(\mathsf{pk}_{\mathsf{BGN}}, y; r_1) \end{array} \right\}.$$

We define the language $L_{\mathsf{eq}}^{\mathsf{pk}_{\mathsf{BGN}}, k}$ as follows:

$$L_{\mathsf{eq}}^{\mathsf{pk}_{\mathsf{BGN}}, k} = \left\{ \left(\mathsf{pk}_{\mathsf{EG}}, \widehat{\mathsf{ct}}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)\right) \;\middle|\; \begin{array}{c} \exists (z, r_2, r_3, (\beta_0, b_0), \ldots, (\beta_k, b_k)) \text{ such that:} \\ \forall j \in \{0, \ldots, k\} : \mathsf{ct}_j = \mathsf{ElGamal.Enc}(\mathsf{pk}_{\mathsf{EG}}, b_j; \beta_j), \\ \mathsf{ct}_0 \cdots \mathsf{ct}_k = \mathsf{ElGamal.Enc}(\mathsf{pk}_{\mathsf{EG}}, z; r_3), \\ \widehat{\mathsf{ct}} = \mathsf{BGN.Com}^*(\mathsf{pk}_{\mathsf{BGN}}, z; r_2) \end{array} \right\}.$$

Note that the function $\mathsf{BGN.Com}^*(\mathsf{pk}_{\mathsf{BGN}}, x; r)$ is a special function that outputs a commitment in the target group, rather than the base group. This is important, since evaluating a degree 2 function homomorphically will result in a commitment in the target group, but if you use the basic BGN.Com function, you will get back a commitment in the base group. So we need to use the $\mathsf{BGN.Com}^*$ function to ensure we get a commitment in the target group.

6.2 Building the Proof of Plaintext Equality (PoPE)

The purpose of $\mathsf{NIZK}_{\mathsf{eq}}$ is to prove that a bitwise ElGamal encryption ct and a BGN commitment $\widehat{\mathsf{ct}}$ both encrypt the same secret value z. Although general purpose NIZKs can always be used in these situations, these NIZKs are often inefficient in practice. Building an efficient NIZK for a particular application, called a special purpose NIZK, is a common practice. Special purpose NIZKs utilize information specific to the application in order to create a more efficient proof than one generated from a general purpose NIZK.

Previously, a special purpose NIZK was developed by [COPZ22] to prove that two Pedersen commitments in different prime order groups commit to the same secret value. Notice that ElGamal encryptions and BGN commitments have a similar structure to that of a Pedersen commitment. Therefore, we would like to be able to modify the Proof of Plaintext Equality (PoPE) developed in [COPZ22] to work for our situation. However, there are two main issues that prevent us from directly using the existing PoPE:

- The existing PoPE works for commitments over different prime order groups. Although the ElGamal encryption exists in a prime order group, the BGN commitment exists in a composite order group.

- The existing PoPE is only sound if the prover does not know the discrete log between the group elements used in the commitments. In this case, the prover knows the discrete log of the group elements used in the ElGamal encryption.

With some modifications, we show that the PoPE developed in [COPZ22] can be extended to our situation. It turns out that the properties of the ElGamal encryption and the composite group order enable us to port the protocol to our situation while preserving security properties such as special soundness and honest verifier zero knowledge.

Syntax for Protocol. Consider three primes p, q_1, q_2, where $p < q_1, q_2$. Let $n = q_1 \cdot q_2$.

- Let $\mathbb{G}_p$ be a group of prime order p, with two generators $g_p, h_p \in \mathbb{G}_p$
- Let $\mathbb{G}_n$ be a group of order n, with two generators $g, u \in \mathbb{G}_n$. Let $H \subset \mathbb{G}_n$ be the order q_1 subgroup generated by $h = u^{q_2}$. Let $i \in H$ be another generator of H. Note it is important that $\log_g(h)$ is not known to the prover.
- Let $b_g = \log_2(p)$. Choose constants b_x, b_c, b_f such that $b_x + b_c + b_f < b_g$, and $0 < b_x < \log_2(p) - 2$ (note that we also need $b_x, b_c, b_f > 0$). You can think of these constants as follows:

  b_g : the bitlength of p.

  b_x : the bitlength of the witness.

  b_c : the bitlength of the verifier's challenge c.

  b_f : controls the probability of aborting.
- We define a relation R_{eq} for statements of the form $\mathsf{stmt} = (X_{p,0}, \ldots, X_{p,k}, X_n)$, and witnesses of the form $\mathsf{wit} = (x, r_p, r_{n1}, r_{n2}, (\beta_0, b_0), \ldots, (\beta_k, b_k))$. We will say that $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{R}_{\mathsf{eq}}$ if:

$$\begin{gathered} X_{p,j} = \left(g_p^{\beta_j},\ g_p^{b_j} \cdot h_p^{\beta_i}\right), \forall j \in \{0 \ldots, k\} \\ \wedge\ X_{p,0} \cdots X_{p,k} = \left(g_p^{r_p},\ g_p^{x} \cdot h_p^{r_p}\right) \\ \wedge\ X_n = g^x \cdot h^{r_{n1}} \cdot i^{r_{n2}}\ . \end{gathered}$$

Note that here, $X_{p,j} \in \mathbb{G}_p \times \mathbb{G}_p$ represents an ElGamal encryption, and $X_n \in \mathbb{G}_n$ represents a BGN commitment. Further, we have that $x \in \mathbb{Z}$, $r_p \in \mathbb{Z}_p$, and $r_{n1}, r_{n2} \in \mathbb{Z}_n$. We also need that $0 \leq x < 2^{b_x}$.

Protocol. The sigma protocol we developed is detailed in Figure 22. The protocol has a few moving parts, split between the ElGamal bit proofs and the main proof of equality. The ElGamal bit proof is from Boneh-Shoup (Example 20.3) [BS17], and it utilizes the Chaum-Pedersen protocol. For the proof of equality itself, we do something very similar to Okamoto's Identification Protocol. We run this style protocol for the ElGamal encryption and the BGN commitment simultaneously. There is one minor difference, which is the inclusion of an additional check that can lead to the prover aborting. This check is put in place to ensure that the prover does not leak any information, as explained in [COPZ22].

Now, we must show that the protocol Π_{eq} satisfies (1) special soundness and (2) honest verifier zero knowledge (HVZK). These security properties are captured in Theorem 6.2 and Theorem 6.3. The proofs for these two theorems are deferred to Section C.

Theorem 6.2. *If $b_x + b_c + b_f < b_g$, then the protocol Π_{eq} is computational special sound for the relation R_{eq} with error $\epsilon = 2^{-b_c} + \epsilon_{DL_n} + \epsilon_{\mathsf{egb}}$, where ϵ_{DL_n} is the advantages of solving the discrete log problem in $\mathbb{G}_n$, and ϵ_{egb} is the knowledge error of the extractor for the Encrypted Bits Protocol Π_{egb} (Definition 3.35).*

Round 1

1 : $\forall j \in \{1, \ldots, k\}$, the prover runs Round 1 of the Encrypted Bits Protocol Π_{egb} for the ElGamal ciphertext $X_{p,j}$.

2 : The prover samples $t_p \leftarrow\!\!\$\ \mathbb{Z}_p$, t_{n1}, $t_{n2} \leftarrow\!\!\$\ \mathbb{Z}_n$, and $y \leftarrow\!\!\$\ \{0, \ldots, 2^{b_x+b_c+b_f} - 1\}$.

3 : The prover constructs $Y_p = (g_p^{t_p}, g_p^y \cdot h_p^{t_p})$ and $Y_n = g^y \cdot h^{t_{n1}} \cdot i^{t_{n2}}$,

4 : and sends $\big(Y_p, Y_n\big)$ to the verifier.

Round 2

1 : $\forall j \in \{0, \ldots, k\}$, the verifier runs Round 2 of the Encrypted Bits Protocol Π_{egb} for the ElGamal ciphertext $X_{p,j}$.

2 : The verifier samples a challenge $c \leftarrow\!\!\$\ \{0, \ldots, 2^{b_c} - 1\}$ and sends c to the prover.

Round 3

1 : $\forall j \in \{0, \ldots, k\}$, the prover runs Round 3 of the Encrypted Bits Protocol Π_{egb} for the ElGamal ciphertext $X_{p,j}$.

2 : Prover checks that $c \in \{0, \ldots, 2^{b_c} - 1\}$ to ensure that the challenge is well formed.

3 : The prover computes (z, s_p, s_{n1}, s_{n2}) as follows:
$z := y + c \cdot x$
$s_p := t_p + c \cdot r_p \pmod p$
$s_{n1} := t_{n1} + c \cdot r_{n1} \pmod n$
$s_{n2} := t_{n2} + c \cdot r_{n2} \pmod n$

4 : If $z \notin \{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_f} - 1\}$, then the prover aborts the protocol

5 : Else, sends (z, s_p, s_{n1}, s_{n2}) to the verifier

Verification

1 : $\forall j \in \{0, \ldots, k\}$, the verifier runs the verification for the Encrypted Bits Protocol Π_{egb} for the ElGamal ciphertext $X_{p,j}$.

2 : Verifier computes $X_p := X_{p,0} \cdots X_{p,k}$

3 : The verifier does three checks to verify the equality of the values committed in X_p and X_n :
$(g_p^{s_p}, g_p^z \cdot h_p^{s_p}) = Y_p \cdot (X_p)^c$
$g^z \cdot h^{s_{n1}} \cdot i^{s_{n2}} = Y_n \cdot (X_n)^c$
$z \in \{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_f} - 1\}$

Figure 22: Overview of PoPE protocol

Proof. We provide the formal proof in Section C. □

Theorem 6.3. *The protocol Π_{eq} for the relation R_{eq} is computational honest-verifier zero-knowledge.*

Proof. We provide the formal proof in Section C. □

Note that although this is an interactive sigma protocol, we can use the Fiat-Shamir Transformation to make the protocol non-interactive. In practice, we would use the non-interactive version, rather than the interactive protocol.

Remark 6.4. *Note that in implementation, since we use the secp256k1 curve for the ElGamal encrptions, we have that $b_g = 256$. With this in mind, we selected the following values for the remaining parameters: $b_x = 180$, $b_c = 64$, and $b_f = 8$. With these parameter selections, in order to achieve 128-bit security, we need to run two repetitions of the proof of equality.*

Lemma 6.5 (Soundness of $\mathsf{NIZK}_{\mathsf{eq}}$)**.** *The $\mathsf{NIZK}_{\mathsf{eq}}$ argument (i.e. the protocol Π_{eq}) for the language L_{eq} (see Section 6.1) satisfies adaptive soundness for the parameters chosen in Remark 6.4.*

Proof. By the way we setup the constants b_x, b_c, b_f, b_g, we have that $b_x + b_c + b_f < b_g$. Further, we know that the value encrypted in the ElGamal encryption is sufficiently small ($< 2^{b_x}$). The verifier also knows this is true, as when they combine the bitwise encryption into the single ElGamal encryption of $f(y)$, they have a pseudo range proof, as if there are less than b_x bit encryptions, then the combined encryption must be $< 2^{b_x}$. Therefore, by Theorem 6.2, we know that the $\mathsf{NIZK}_{\mathsf{eq}}$ argument for the language L_{eq} satisfies special soundness. Therefore, we also know that the $\mathsf{NIZK}_{\mathsf{eq}}$ argument satisfies adaptive soundness, as desired.

□

Lemma 6.6 (Zero Knowledge of $\mathsf{NIZK}_{\mathsf{eq}}$)**.** *The $\mathsf{NIZK}_{\mathsf{eq}}$ argument (i.e. the protocol Π_{eq}) for the language L_{eq} (see Section 6.1) satisfies computational honest verifier zero knowledge for the parameters chosen in Remark 6.4.*

Proof. By the way we setup the constants b_x, b_c, b_f, b_g, we have that $b_x + b_c + b_f < b_g$. Further, we know that the value encrypted in the ElGamal encryption is sufficiently small ($< 2^{b_x}$). The verifier also knows this is true, as when they combine the bitwise encryption into the single ElGamal encryption of $f(y)$, they have a pseudo range proof, as if there are less than b_x bit encryptions, then the combined encryption must be $< 2^{b_x}$. Therefore, by Theorem 6.3, we know that the $\mathsf{NIZK}_{\mathsf{eq}}$ satisfies computational honest verifier zero knowledge, as desired.

□

6.3 BGN-FAS Construction for Degree 2 Functions

We provide a specific implementation for the proposed FAS scheme that enables degree 2 homomorphism. It utilizes the BGN commitment scheme and bitwise ElGamal encryption. We will call this scheme BGN-FAS. Our construction for BGN-FAS follows the template in Figure 20. Note that a major difference is that the seller does not have access to the secret key for BGN, as this would compromise the soundness of the PoPE protocol. Security proofs for BGN-FAS can be found in Appendix D.

Replacing FHE with BGN Commitments. Since we are now moving to the degree 2 setting, we can replace FHE with a simpler degree 2 homormorphic encryption scheme. For this, we choose the BGN encryption scheme [BGN05]. We can further reduce to simply using BGN Commitments, as there is no need for decryption in our scheme, and the seller will not have access to the BGN secret key (important for soundness of PoPE).

Given two encryptions $\mathsf{ct}_1 = g^{m_1} h^{r_1}$ and $\mathsf{ct}_2 = g^{m_2} h^{r_2}$, we can perform homomorphic addition by $\mathsf{ct}_1\mathsf{ct}_2 = g^{m_1+m_2} h^{r_1+r_2}$. We can enable homomorphic multiplication through the use of pairings, that is, $e(\mathsf{ct}_1, \mathsf{ct}_2) = \hat{g}^{m_1 m_2} \hat{h}^{m_1 r_2 + m_2 r_1 + q r_1 r_2}$, where q is one of the prime factors of n, the order of the composite group for BGN. Note that the presence of the prime factor q in the exponent for the random term is problematic for REval, since the seller will not have access to the secret key (which in this case is the prime factors of n). Therefore, we need to introduce a new group element, which we denote by i, which is computed from $i = e(h, h)$. The ciphertext $\hat{g}^{m_1 m_2} \hat{h}^{m_1 r_2 + m_2 r_1 + q r_1 r_2}$

that was output from the pairing operation can instead be expressed as $\hat{g}^{m_1 m_2} \hat{h}^{m_1 r_2 + m_2 r_1} i^{r_1 r_2}$. The seller can use this equivalent representation for the new PoPE that we implement.

New PoPE for ElGamal Ciphertexts and BGN Commitments. The proof of equality protocol Π_{eq} presented above in Section 6.2. will be used for the NIZK argument $\mathsf{NIZK}_{\mathsf{eq}}$. Our PoPE extends the previous work of Chase et al [COPZ22], who proposed a sigma protocol to prove plaintext equality between commitments across different prime order groups. We generalize this protocol to work for ElGamal ciphertexts over a prime order group, and a BGN commitment over a composite order group. Since the $\mathsf{NIZK}_{\mathsf{eq}}$ protocol will need to be run each time a transaction occurs, we want an efficient NIZK, which is why we developed and implemented this a special purpose NIZK for BGN-FAS.

Efficiency. We note that the BGN-FAS construction is considerably more efficient than a SNARK-strawman approach. BGN-FAS's runtime is over an order of magnitude faster for large database sizes, as can be seen in Figure 2. Notice that the main bottleneck of this scheme is the homomorphic evaluation for the buyer, as can be seen in Figure 28.

7 FAS for Linear Functions (Revisited)

In this section, we describe how to make our FAS construction for general functions even more efficient when restricted to linear functions only. Our efficient instantiation follows the template in Figure 20 with two modifications as follows.

Replacing FHE with El Gamal Encryption. Since we are restricted to the linear functions setting, it sufficies to replace the FHE scheme with a linearly homomorphic encryption (LHE) scheme. For this, we choose the El Gamal Encryption scheme with messages encoded in the exponent. This is linearly homomorphic since for any two messages $m_0, m_1 \in \mathbb{Z}_p$, note that $\mathsf{ct}_0 = (g^{r_0}, \mathsf{pk}^{r_0} \cdot g^{m_0})$ and $\mathsf{ct}_1 = (g^{r_1}, \mathsf{pk}^{r_1} \cdot g^{m_1})$. As a consequence $\mathsf{ct} = ((g^{r_0}) \cdot (g^{r_1}), (\mathsf{pk}^{r_0} \cdot g^{m_0}) \cdot (\mathsf{pk}^{r_1} \cdot g^{m_1}))$ is an encryption of $m_0 + m_1$ under the randomness $r_0 + r_1$. We define the REval algorithm in Figure 23 for the class of linear functions we will evaluate. Furthermore, for sake of simplicity of $\mathsf{NIZK}_{\mathsf{eq}}$ as discussed in the next paragraph, we use the prime-order cyclic group based on the secp256k1 elliptic-curve for this El Gamal Encryption scheme.

$\mathsf{LHE.REval}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, \vec{x}, \vec{r}, f, \mathsf{ct}_f)$:

1 : Parse f as $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_\ell)$
2 : $r_f = \langle \vec{r}, \vec{\alpha} \rangle$
3 : **ret** r_f

Figure 23: Description of REval for our LHE scheme.

$\mathsf{NIZK}_{\mathsf{eq}}$: Replacing cut-and-choose style proof with Schnorr-style proof. Recall that $\mathsf{NIZK}_{\mathsf{eq}}$ is a proof of plaintext equality. In particular, the language L_{eq} is about statements containing $\mathsf{ct}_{\mathsf{FHE}}$ which is an FHE evaluated ciphertexted and $\mathsf{ct}_{\mathsf{EG}}$ which is an El Gamal ciphertext. Since FHE and El Gamal encryption schemes were based on different mathematical structure, lattices and prime-order cyclic groups respectively, we needed to rely on cut-and-choose style proof. But observe that now we have replaced FHE with El Gamal encryption scheme as well. As a consequence the language L_{eq} is even more simplified and only concerns a single prime-order cyclic group. This allows us to use Schnorr-style proofs for the language. The main upside here is that the soundness error after 1 repetition is already negligible and we do not need to do any repetitions. This helps

make the NIZK prover and verifier even more efficient in both computation and communication. In particular, communication just involves 7 group elements of group $\mathbb{G}$ and 6 scalars in $\mathbb{Z}_p$.

Efficiency. We note that the overall resulting FAS construction is far more efficient than that of Vanjani et al [VST24a]. Specifically, the seller does not have to do any expensive computation. Further, the buyer has to perform only 2 multi-scalar multiplication (MSM) operations that are involved in the homomorphic evaluation. In contrast, [VST24a] required seller to perform 2 MSMs and buyer to perform 1 MSM and 1 discrete log computation. Our implementation numbers reflect the same and we refer to Section 8.3 for more details.

8 Implementation and Evaluation

In this section, we present the details of our implementations and evaluations. We implemented all FAS constructions – referred to as FHE FAS, LHE FAS, and BGN FAS – and the SNARK-based strawman to evaluate real-world tradeoffs. All implementations are available at [fas]. We comprehensively benchmark each scheme across various parameters, showing practical performance in real-world scenarios.

Communication Overhead. We note that across all our implementations, only a single Schnorr signature (64 bytes) is put on-chain.

8.1 FHE FAS

Our FHE FAS scheme is implemented partly in Python, and partly in Rust. We use the Secp256k1 elliptic curve that is used by Bitcoin [sec] and other cryptocurrencies. We use the curve's python implementation from hanabi1224 [han]. For Schnorr signatures, we use a modified version of BIP-340 reference implementation [bip]. We implement the Schnorr adaptor signatures [AEE+21] on top of it. These parts are borrowed from the open-source FAS implementation of [VST24b]. Further, we implement ElGamal encryption and $\mathsf{NIZK}_{\mathsf{eq}}$ parts related to ElGamal on top of the secp256k1 elliptic curve. For the FHE scheme, we use the TFHE scheme [CGGI20] implemented by Zama: this involves a suite of three libraries: TFHE-rs [tfh] is a Rust library that implements FHE, Concrete [cona] is a Python library that is a layer on top of TFHE-rs to convert high-level circuits into FHE friendly circuits. Further, ConcreteML [conb] is a python library that is a layer on top of Concrete that provides APIs for ML applications. Further, we implement the $\mathsf{NIZK}_{\mathsf{eq}}$ parts related to FHE in Rust. We do not benchmark the one-time $\mathsf{NIZK}_{\mathsf{ad}}$ (from the setup phase of the seller), as this is needed only for AdGen and AdVerify.

For high-degree polynomials and fine-tuning models settings, we use the Concrete and ConcreteML libraries, respectively. Before presenting performance evaluations, we present our core optimization of an efficient REval algorithm for the TFHE scheme [tfh, CGGI20].

8.1.1 Core Optimization: efficient REval

Recall that if the seller's advertisement contains $\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r)$, and the buyer wants to buy function f, then, the $\mathsf{NIZK}_{\mathsf{eq}}$'s witness contains r_f, which the seller must compute using REval before proceeding to compute π_{eq}. As noted in Theorem 5.1, we want REval to be more efficient than HEval, and we achieve it by using the secret key $\mathsf{sk}_{\mathsf{FHE}}$ and the structure of HEval.

We first recall the TFHE [tfh, CGGI20] scheme in Figure 24 that internally relies on three encryption schemes: an LWE-based encryption scheme denoted LWE, a generalized LWE-based encryption scheme denoted GLWE, and a generalized GSW encryption scheme denoted GGSW.

$\mathsf{TFHE.KGen}(1^\lambda)$:

1 : Compute $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{LWE.KGen}(1^\lambda)$
2 : Compute $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{GLWE.KGen}(1^\lambda)$
3 : Compute $\mathsf{ksk} := \mathsf{LWE.Enc}(\mathsf{pk}_1, \mathsf{sk}_2; r_{\mathsf{ksk}})$
4 : Compute $\mathsf{bsk} \leftarrow \mathsf{GGSW.Enc}(\mathsf{sk}_2, \mathsf{sk}_1)$
5 : **ret** $\mathsf{pk}_{\mathsf{FHE}} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ksk}, \mathsf{bsk})$,
$\mathsf{sk}_{\mathsf{FHE}} := (\mathsf{sk}_1, \mathsf{sk}_2, r_{\mathsf{ksk}}, \mathsf{pk}_{\mathsf{FHE}})$

$\mathsf{TFHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, m)$:

1 : Parse $\mathsf{pk}_{\mathsf{FHE}} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ksk}, \mathsf{bsk})$
2 : Compute $\mathsf{ct}_2 := \mathsf{GLWE.Enc}(\mathsf{pk}_2, m; r)$
3 : Compute $\mathsf{ct}_1 := \mathsf{KeySwitch}(\mathsf{ksk}, \mathsf{ct}_2)$
4 : **ret** $\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2)$

$\mathsf{TFHE.Dec}(\mathsf{sk}_{\mathsf{FHE}}, ct)$:

1 : Parse $\mathsf{sk}_{\mathsf{FHE}} = (\mathsf{sk}_1, \mathsf{sk}_2, r_{\mathsf{ksk}}, \mathsf{pk}_{\mathsf{FHE}})$
2 : Parse $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$
3 : **ret** $\mathsf{GLWE.Dec}(\mathsf{sk}_2, \mathsf{ct}_2)$

Figure 24: TFHE encryption scheme from [tfh, CGGI20].

The homomorphic evaluation HEval proceeds as follows: For linear functions f, it is essentially free by mimicking linear operations on the ciphertexts. For non-linear f, HEval uses a combination of programmable bootstrapping followed by key-switching in two steps, as follows: (1) ct_1 is transformed to a GLWE ciphertext $\mathsf{ct}_{2,f}$ under pk_2 encrypting message $f(m)$ by using the bootstrapping key bsk – a GGSW encryption of sk_1 under pk_2. (2) $\mathsf{ct}_{2,f}$ is transformed back to a LWE ciphertext $\mathsf{ct}_{1,f}$ under key pk_1 encrypting message $f(m)$ using the key-switching key ksk – an LWE encryption of the key sk_2 under pk_1. Since $\mathsf{ct}_{1,f}$ is encrypted under pk_1, it can be used again for further evaluations. See Figure 25 for details on HEval's non-linear component.

$\mathsf{TFHE.HEval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}, f)$:

1 : Parse $\mathsf{pk}_{\mathsf{FHE}} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ksk}, \mathsf{bsk})$
2 : Parse $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$
3 : Compute $\mathsf{ct}_{2,f} := \mathsf{Bootstrap}(\mathsf{bsk}, \mathsf{ct}_1, f)$
4 : Compute $\mathsf{ct}_{1,f} := \mathsf{KeySwitch}(\mathsf{ksk}, \mathsf{ct}_{2,f})$
5 : **ret** $\mathsf{ct}_f := (\mathsf{ct}_{1,f}, \mathsf{ct}_{2,f})$

$\mathsf{TFHE.REval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}, m, r, f, \mathsf{ct}_f)$:

1 : Parse $\mathsf{sk}_{\mathsf{FHE}} = (\mathsf{sk}_1, \mathsf{sk}_2, r_{\mathsf{ksk}}, \mathsf{pk}_{\mathsf{FHE}})$
2 : If $\mathsf{ct}_f = \perp : \mathsf{ct}_f := \mathsf{TFHE.HEval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}, f)$
3 : Parse $\mathsf{ct}_f = (\mathsf{ct}_{1,f}, \mathsf{ct}_{2,f})$
4 : Compute $r_{1,f} := \langle r_{\mathsf{ksk}}, \mathsf{ct}_{2,f} \rangle$
5 : **ret** $r_f := r_{1,f}$

Figure 25: TFHE.HEval and TFHE.REval for non-linear f in [tfh, CGGI20].

Defining REval algorithm to compute r_f. Suppose that $\mathsf{ct}_f = (\mathsf{ct}_{1,f}, \mathsf{ct}_{2,f})$ is the final evaluated ciphertext encrypting $f(m)$ where both $\mathsf{ct}_{i,f}$'s encrypt $f(m)$ with randomness, say $r_{i,f}$'s, respectively. Unfortunately, as decryption involves rounding operations which are lossy, we cannot exactly compute $r_{1,f}$ (or $r_{2,f}$) by decrypting $\mathsf{ct}_{1,f}$ (or $\mathsf{ct}_{2,f}$) using $\mathsf{sk}_{\mathsf{FHE}}$. One option is to track how the original randomness evolves through KeySwitch and Bootstrap, but this is as costly as HEval itself—especially since Bootstrap is complex. Moreover, if f is a composition of k functions, tracking randomness takes $O(k)$ time, common in systems like ConcreteML.

Can we make REval cheaper than HEval and independent of f? We show this is possible by explicitly tracking $r_{1,f}$'s evolution through HEval, which motivates making ct_1 and $\mathsf{ct}_{1,f}$ explicit in our algorithms – despite them being the result of deterministic computations.

The randomness r_f from REval is used to construct the $\mathsf{NIZK}_{\mathsf{eq}}$ proof that ct_f and $\mathsf{ct}_{\mathsf{EG}}$ encrypt the same message. We reduce this proof to showing that $\mathsf{ct}_{1,f} := \mathsf{KeySwitch}(\mathsf{ksk}, \mathsf{ct}_{2,f})$ and $\mathsf{ct}_{\mathsf{EG}}$ encrypt the same message – making the proof more tractable.

Why this transformation? The key insight is that the *randomness in* $\mathsf{ct}_{1,f}$ *depends only on the*

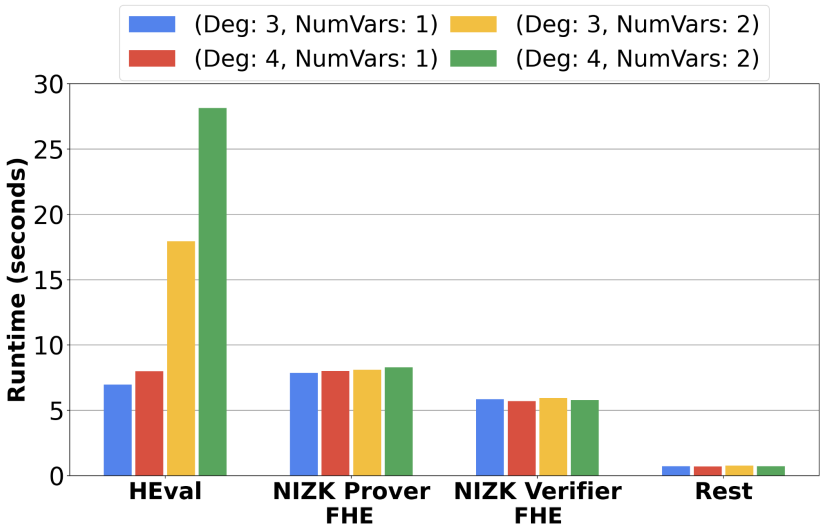

Figure 26: Runtime of FHE-FAS algorithms for polynomial evaluations. We plot times for HEval, NIZK Prover and Verifier. "Rest" includes $\mathsf{FPreSign}_2$, $\mathsf{FPreVerify}_2$, Adapt, and Vf.

randomness in ksk*, not on* $r_{2,f}$ *from* $\mathsf{ct}_{2,f}$*!* This greatly simplifies REval: we only need to track how randomness r_{ksk} in ksk yields $r_{1,f}$, avoiding the need to trace $r_{2,f}$. This shift makes randomness tracking far more efficient.

The reason for this simplification lies in the structure of the KeySwitch operation. Recall that ksk is an LWE ciphertext: it encrypts the secret key sk_2 under the public key pk_1 with randomness r_{ksk}, that is, $\mathsf{ksk} := \mathsf{LWE.Enc}(\mathsf{pk}_1, \mathsf{sk}_2; r_{\mathsf{ksk}})$. The KeySwitch operation computes $\mathsf{ct}_{1,f}$ as an inner product of ksk and $\mathsf{ct}_{2,f}$: that is, $\mathsf{ct}_{1,f} := \langle \mathsf{ksk}, \mathsf{ct}_{2,f} \rangle$, where ksk is treated a vector of ciphertexts and $\mathsf{ct}_{2,f}$ is treated as a vector of known constants. Thus, the entire computation is simply a linear homomorphic evaluation on the ciphertext.

To track the randomness in $\mathsf{ct}_{1,f}$, the REval algorithm computes the inner product of the known constants $\mathsf{ct}_{2,f}$ with the randomness r_{ksk} from ksk as described in Figure 25. We emphasize that this is remarkably more efficient than HEval, and is independent of the description of non-linear functions f.

8.1.2 Applications

We demonstrate the practicality of our implementations via two main applications related to data analytics and ML fine-tuning.

High-degree polynomial evaluations. Suppose a seller holds a proprietary ML model, and the buyer wants to obtain an inference on some data using the seller's model. Currently, all business scenarios involve some trust assumption between the buyer and seller, such as pay first and get service later, or vice versa. In trustless scenarios, an FAS-based approach is warranted. Popular ML models involve non-linear functions (e.g., exp, softmax, ReLU, GeLU) that are not crypto-friendly, so private inference relies on polynomial approximations—e.g., degree-4 softmax [CZK+23], and degree-3 GeLU [ZYH+24, LHG+23, DLZ+23]. We support degree-3 and degree-4 polynomial evaluations and report performance in Figure 26.

Fine-tuning Machine Learning models. In a dual setting, a seller holds a proprietary dataset, and a buyer with a pre-trained model wants to fine-tune it. Due to privacy laws like HIPAA, raw data (e.g., medical records) can't be sold, but function evaluations useful for training may be allowed.

Our FAS protocol supports this setting. We demonstrate this by training an SGD classifier on the Iris dataset using a log-loss function, with results shown in Figure 27. As dataset size grows, homomorphic evaluation dominates the cost, while other FAS components remain constant.

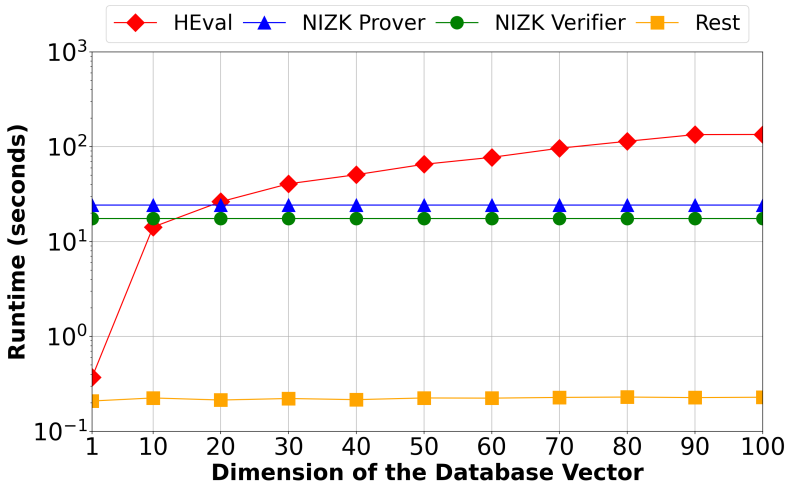

Figure 27: Runtime of FHE-FAS algorithms for ML model fine-tuning model task. See Figure 26's caption for descriptions of reported algorithms. NIZK times are estimates based on Figure 26.

Comparison. We compare the performance of our FHE-based FAS scheme with SNARK-based FAS scheme. Recall that our design goal for FAS is to keep the seller lightweight (critical for ML fine-tuning and scalability, as stated above). Accordingly, our FHE-FAS scheme places only a lightweight PoPE on the seller, independent of dataset size, while shifting FHE evaluation to the buyer. By contrast, the SNARK-based approach requires a seller-side prover whose cost grows linearly with dataset size. In the following evaluation, we compute $\sum_i a_i \cdot x_i^3$ where $i \in [N]$, $(x_1, \ldots, x_N)$ is the seller's dataset and $(a_1, \ldots, a_N)$ describes the buyer's function. All variables and co-efficients are random integers between $0 - 10$. Seller and buyer times are reported below in Table 1.

N	Scheme	Seller	Buyer	Breakdown (Seller)	Breakdown (Buyer)	Total
10^3	SNARK	5.111s	0.007s	SNARK Prover: 5.111s	SNARK Verify: 0.007s	5.118s
10^3	FHE	8.224s	70.862s	FHE Decrypt: 0.150s NIZK Prover: 8.074s	FHE Eval: 65.029s NIZK Verify: 5.833s	79.086s
10^4	SNARK	41.355s	0.030s	SNARK Prover: 41.355s	SNARK Verify: 0.030s	41.385s
10^4	FHE	8.224s	754.453s	FHE Decrypt: 0.150s NIZK Prover: 8.074s	FHE Eval: 748.620s NIZK Verify: 5.833s	762.677s

Table 1: Comparison of Buyer and Seller efficiency of SNARK and FHE + Cut-and-choose NIZK approaches for different N values.

These results confirm our hypothesis: SNARKs favor the seller on small datasets ($N = 10^3$), while for larger ones ($N \geq 10^4$) FHE is superior since seller time stays $\sim 8s$. At $N = 10^4$, the FHE seller is $\sim 5\times$ faster though the overall protocol is $\sim 18\times$ slower. As N increases, the gap between our seller and SNARK seller becomes even more pronounced: e.g., at $N = 10^5$, we expect $50\times$ faster seller but an overall $20\times$ slower protocol.[12]

[12]Our FHE-based FAS implementation uses FHE in a black-box manner, and does not optimize it. The main bottleneck is FHE evaluation and recent line of works [JCT+25, ACJ24, KKC+23, SS24] try to improve its performance.

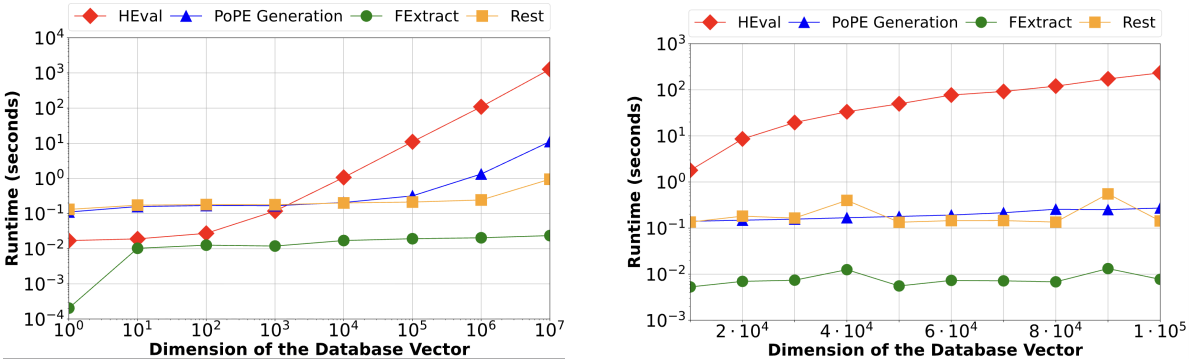

Figure 28: Left: runtime of BGN-FAS algorithms for the WSD setting with AdGen encrypts both x_i and x_i^2. **Right:** runtime of BGN-FAS algorithms for second-order FM prediction for sparse input features with 0.1% values being non-zero. Times reported for HEval, FExt, and the PoPE prover time (also includes REval time). Rest covers the remaining algorithms: $\mathsf{FPreSign}_2$, $\mathsf{FPreVerify}_2$, Adapt, and Vf.

8.2 BGN FAS

We implemented the BGN FAS scheme in Go, building on BGN's Go library [SS20], which uses Stanford's PBC library [Lyn]. The BGN group was a 2048-bit composite order group with 1024-bit prime factors. We used the secp256k1 curve for our ElGamal implementation [Dec]. We used the BLAKE-256 hash function for Schnorr Adaptor Signatures. We implemented BGN FAS for two different functions: weighted standard deviation and second order factorization machines [Ren10].

Weighted Standard Deviation (WSD). Suppose a seller has a database $X = \{x_1, x_2, \ldots, x_\ell\}$ of length ℓ. The buyer then specifies weights $\vec{w} = (w_1, \ldots, w_\ell)$ to compute a weighted standard deviation as follows:

$$\mathsf{WSD}(X) = \sum_{i \in [\ell]} (\ell \cdot w_i \cdot x_i - \sum_{i \in [\ell]} w_i \cdot x_i)^2 \ .$$

In this case, we made a slight modification to the scheme in order to boost performance. The seller provides BGN commitments to both x_i and x_i^2. This reduces the number of pairings computed during homomorphic evaluation from $O(\ell)$ pairings to $O(1)$, decreasing runtime for the buyer drastically. Performance overheads are shown in Figure 28 (for $x_i, w_i \in [0, 1000]$). This design choice shifts more work to the seller, but it is part of advertisement generation, a one-time computation.

Second Order Factorization Machines (FM). Suppose a seller with a trained FM model (parameters: bias w_0, weights $w_1, \ldots, w_\ell$, and latent vectors $v_1, \ldots, v_\ell$ of dimension k) releases BGN commitments to these parameters as an advertisement. A buyer can then perform inference on a feature vector $\vec{x} = \{x_1, x_2, \ldots, x_\ell\}$ by sending $\vec{x}$ as their function. The computation is:

$$\mathsf{FM}(X) = w_0 + \sum_{i=1}^{\ell} w_i \cdot x_i + \sum_{i=1}^{\ell} \sum_{j=i+1}^{\ell} \langle v_i, v_j \rangle \cdot x_i \cdot x_j \ .$$

While inner product computations suggest many pairing operations, sparsity in FM inputs helps reduce them: if x_i or x_j is zero, $\langle v_i, v_j \rangle$ need not be computed. Thus, pairing cost is largely governed by input sparsity. FMs' strength lies in high-dimensional, sparse data, common in recommendation tasks where $x_i \in \{0, 1\}$. See Figure 28 for performance with $w_i, v_i \in [0, 1000]$.

Such advances in FHE will directly improve our FAS performance as well.

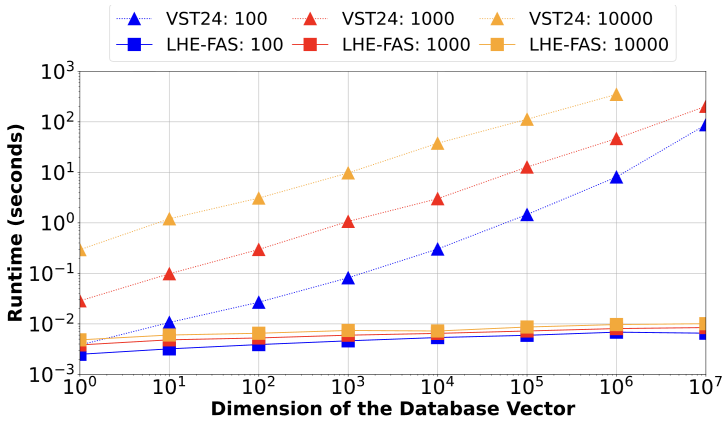

Figure 29: Runtime of FExt for VST24 and LHE-FAS for varying witness bounds. For example, "VST24: 100" means VST24 with a witness bound of 100.

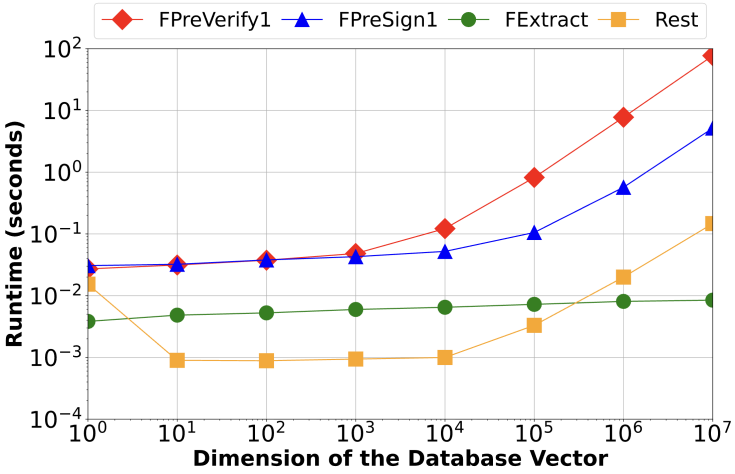

Figure 30: Runtime of LHE-FAS algorithms for different database sizes. We report times for $\mathsf{FPreVerify}_1$, $\mathsf{FPreSign}_1$, FExt, with "Rest" including total combined time for $\mathsf{FPreSign}_2$, $\mathsf{FPreVerify}_2$, Adapt, and Vf.

Comparison. We implemented a SNARK strawman approach as a baseline for comparison. We also compare the BGN-FAS to LHE-FAS and the prior work done in VST24 [VST24a]. Note that BGN was coded in Go, while VST24 was originally coded in Python. For a fairer comparison, we re-implemented VST24 in Go.

8.3 LHE FAS

We implemented LHE FAS in Go, and the $\mathsf{NIZK}_{\mathsf{eq}}$ is implemented using the Schnorr PoK protocol. We benchmark running times by varying the length of the witness vector y from 1 to 10^6. Figure 30 indicates that the seller is very efficient with sub-5 second runtime under all scenarios. For the buyer, FExt is very fast, as in Figure 29. Even so, our buyer exhibits some tradeoffs when compared with VST24. When each witness and function entry is bounded by 10^2, our buyer matches VST24 at vector length 10^6. At bounds of 10^3 and 10^4, our buyer is 2×–100× and 12×–100× faster, respectively.

These tradeoffs stem from our buyer performing 2 MSMs, while the buyer in [VST24a] performs 1 MSM and 1 discrete log. For vector length ℓ and bound B, MSM costs $O(\ell \log B)$, and discrete

log costs $O(\sqrt{\ell} \cdot B)$. Moreover, our 2 MSMs can be parallelized, unlike the sequential MSM and discrete log in [VST24a]. With parallelism, our buyer outperforms VST24 in all cases.

9 Functional Fair Exchange (FFE)

We define Functional Fair Exchange (FFE) in an advertisement-based setting, and show that any FAS scheme implies an FFE protocol. Our definition is inspired by Della Monica et al. [DMVVZ25], and models FFE as a three-party protocol: Alice (seller) advertises an asset, Bob (buyer) seeks a function evaluation in exchange for payment, and T is a trusted third party (TTP).

Notations. For PPT algorithms $\mathsf{A}, \mathsf{B}, \mathsf{C}$, we write $[x, y, z] \leftarrow \mathsf{A}(a), \mathsf{B}(b), \mathsf{C}(c)$ to denote their interaction on common input d, private inputs a, b, c, and independent randomness, where x, y, z are their respective outputs. Inspired by [BK14, DMVVZ25], we define tokens with two properties: (i) exclusive ownership – only one user can hold a token at a time, and (ii) transferable and verifiable – tokens can be freely transferred, and recipients can instantly verify their validity. We assume that each user A owns a wallet W_A. $\mathcal{T}$ denotes the set of all possible tokens with the same value. $\mathsf{tk}[n] := \{\mathsf{tk}_1, \ldots, \mathsf{tk}_n\}$ denotes a set of n tokens such that $\mathsf{tk}_i \in \mathcal{T}$ for all $i \in [n]$. $\mathsf{tk}[n] \in \mathsf{W}_\mathsf{A}$ denotes that the tokens $\mathsf{tk}[n]$ are by W_A. Let $\mathsf{owner}(\mathsf{tk}[n], \mathsf{W})$ be a predicate that returns 1 if all tokens $\{\mathsf{tk}_i\}_{i \in [n]}$ are in wallet W, and 0 otherwise.

Definition 9.1. *A FFE protocol Π_{FFE} for function class $\mathcal{F}$ is a three-party protocol run by a p.p.t. seller A (with wallet W_A), a p.p.t. buyer B (with wallet W_B) and a trusted-third party T satisfying Completeness, Advertisement Binding, and Buyer- and Seller- Fairness with following algorithms:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: outputs random public parameters $\mathsf{pp} \in \{0,1\}^*$ on input a security parameter λ.*
- $(x_{\mathsf{prv}}, x_{\mathsf{pub}}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}})$*: Seller A runs this randomized algorithm that takes as input public parameter pp, an asset $x \in \{0,1\}^*$, and $\mathsf{tk}_{\mathsf{ask}} \in \mathbb{N}^+$, and outputs an advertisement x_{pub} to buyer B and stores internal state x_{prv}.*
- $b = \mathsf{AdVerify}(\mathsf{pp}, x_{\mathsf{pub}})$*: Buyer B runs this deterministic algorithm that takes as input the public parameter pp, and advertisement x_{pub} and outputs a bit $b \in \{0,1\}$. If $b = 0$, buyer B aborts the protocol.*
- $[\alpha, \beta] \leftarrow [\mathsf{A}(x_{\mathsf{prv}}), \mathsf{B}(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}, x_{\mathsf{pub}}, f)$*: the exchange phase is a three party protocol where B wants to learn $f(x)$ from A in exchange for tokens $\mathsf{tk}[n]$ and the trusted third-party T assists them in this protocol. Alice's private inputs are x_{prv} and Bob's private input are $\mathsf{tk}[n] \in \mathsf{W}_\mathsf{B}$. Without loss of generality, we say the exchange phase is R rounds where, for all $i \in [R]$: the first message is from the seller A and second message is from the buyer B. After R rounds, there is just a single message from the seller A that terminates the protocol. At the end of the protocol, A's output is $\alpha \in \{0,1\}$ defined as the predicate evaluation $\alpha := \mathsf{owner}(\mathsf{tk}[n], \mathsf{W}_\mathsf{A})$ and B's output is β.*

Completeness. *Given an honest seller A with wallet W_A and an honest buyer B with wallet W_B such that $\mathsf{tk}[n] \in \mathsf{W}_\mathsf{B}$, for any asset $x \in \{0,1\}^*$, any $\mathsf{tk}_{\mathsf{ask}} \in \mathbb{N}^+$, and any $f \in \mathcal{F}$: suppose $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(x_{\mathsf{prv}}, x_{\mathsf{pub}}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}})$, $[\alpha, \beta] \leftarrow [\mathsf{A}(x_{\mathsf{prv}}), \mathsf{B}(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}, x_{\mathsf{pub}}, f)$, then $\Pr[\mathsf{AdVerify}(\mathsf{pp}, x_{\mathsf{pub}}) = 1 \wedge \alpha = 1 \wedge \beta = f(x)] = 1$.*

Advertisement Binding. *There exists a negligible function negl such that for all p.p.t. and stateful malicious seller A^*, for any $\lambda \in \mathbb{N}$ if $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $x_{\mathsf{pub}} \leftarrow \mathsf{A}^*(\mathsf{pp})$, such that $\mathsf{AdVerify}(\mathsf{pp}, x_{\mathsf{pub}}) = 1$, then both the following hold: Informally, both equations ensure that any (even malicious) x_{pub} has a unique tuple $(x, \mathsf{tk}_{\mathsf{ask}}, r)$ mapping to it via AdGen. This tuple can be found via brute-force by*

$\mathsf{SFReal}_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)$:

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(x_{\mathsf{prv}}, x_{\mathsf{pub}}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}})$
3 : $[\alpha, \beta] \leftarrow [\mathsf{A}(x_{\mathsf{prv}}), \mathsf{B}^*(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}, x_{\mathsf{pub}}, f)$
4 : **ret** $\mathsf{V}_{\mathsf{B}^*}$

$\mathsf{SFIdeal}^{\mathsf{Sim}}_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)$:

1 : $\mathsf{pp}^* \leftarrow \mathsf{Sim}_0(1^\lambda)$
2 : $x^*_{\mathsf{pub}} \leftarrow \mathsf{Sim}_1(\mathsf{pp}^*, \mathsf{tk}_{\mathsf{ask}})$
3 : $[\alpha, \beta] \leftarrow [\mathsf{Sim}_2(f(x)), \mathsf{B}^*(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}^*, x^*_{\mathsf{pub}}, f)$
4 : **ret** $\mathsf{V}_{\mathsf{B}^*}$

$\mathsf{SFReal}^i_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)$:

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(x_{\mathsf{prv}}, x_{\mathsf{pub}}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}})$
3 : $[\alpha, \beta] \leftarrow [\mathsf{A}(x_{\mathsf{prv}}), \mathsf{B}^*(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}, x_{\mathsf{pub}}, f)$
4 : **ret** $\mathsf{V}_{\mathsf{B}^*}$

$\mathsf{SFIdeal}^{i,\mathsf{Sim}^i}_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)$:

1 : $\mathsf{pp}^* \leftarrow \mathsf{Sim}^i_0(1^\lambda)$
2 : $x^*_{\mathsf{pub}} \leftarrow \mathsf{Sim}^i_1(\mathsf{pp}^*, \mathsf{tk}_{\mathsf{ask}})$
3 : $[\alpha, \beta] \leftarrow [\mathsf{Sim}^i_2, \mathsf{B}^*(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}^*, x^*_{\mathsf{pub}}, f)$
4 : **ret** $\mathsf{V}_{\mathsf{B}^*}$

Figure 31: SFReal, SFIdeal, SFReal^i and $\mathsf{SFIdeal}^i$ games for Seller Fairness.

an extractor $\mathcal{E}$ which we used below.

$$\Pr\left[\begin{array}{c}\exists(x, \mathsf{tk}_{\mathsf{ask}}, r), (x', \mathsf{tk}'_{\mathsf{ask}}, r') \ s.t. \\ x \neq x' \ \wedge \\ \mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}}; r) = (\cdot, x_{\mathsf{pub}}) \ \wedge \\ \mathsf{AdGen}(\mathsf{pp}, x', \mathsf{tk}'_{\mathsf{ask}}; r') = (\cdot, x_{\mathsf{pub}})\end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

$$\Pr\left[\begin{array}{c}\nexists(x, \mathsf{tk}_{\mathsf{ask}}, r) \ s.t. \\ \mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}}; r) = (\cdot, x_{\mathsf{pub}})\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

Buyer Fairness. *There exists an extractor algorithm (possibly inefficient) $\mathcal{E}$ and there exists a negligible function* negl *such that for any honest buyer* B *with wallet* W_{B} *and* $\mathsf{tk}[n] \in \mathsf{W}_{\mathsf{B}}$*, for all p.p.t. and stateful malicious seller* A^**, for any $\lambda \in \mathbb{N}$, for any $f \in \mathcal{F}$, it holds that: if* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $x_{\mathsf{pub}} \leftarrow \mathsf{A}^*(\mathsf{pp})$ *such that* $\mathsf{AdVerify}(\mathsf{pp}, x_{\mathsf{pub}}) = 1$*,* $[\alpha, \beta] \leftarrow [\mathsf{A}^*, \mathsf{B}(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}, x_{\mathsf{pub}}, f)$ *then the following is* $\leq \mathsf{negl}(\lambda)$*:*

$$\Pr\left[\begin{array}{c}\exists i \in [n] \ s.t. \ \mathsf{tk}_i \notin \mathsf{W}_{\mathsf{B}} \\ \wedge \ \beta \neq f(x^*)\end{array} : x^* := \mathcal{E}(\mathsf{pp}, x_{\mathsf{pub}})\right] .$$

Seller Fairness. *Suppose the exchange phase contains R rounds with seller and buyer speaking once in each round with the interaction concluding with a single message from the seller. There exists a negligible function* negl *such that for any honest seller* A*, asset $x \in \{0,1\}^*$ and $\mathsf{tk}_{\mathsf{ask}} \in \mathbb{N}^+$, any p.p.t. malicious buyer* B^* *with wallet* W_{B} *such that* $\mathsf{tk}[n] \in \mathsf{W}_{\mathsf{B}}$ *and $f \in \mathcal{F}$, seller fairness requires:*

- *there exists a p.p.t. stateful simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that for any p.p.t. distinguisher $\mathcal{D}$, the difference in* $\Pr[\mathcal{D}(\mathsf{V}_{\mathsf{B}^*}) = 1 : \mathsf{V}_{\mathsf{B}^*} \leftarrow \mathsf{SFReal}_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)]$ *and* $\Pr[\mathcal{D}(\mathsf{V}_{\mathsf{B}^*}) = 1 : \mathsf{V}_{\mathsf{B}^*} \leftarrow \mathsf{SFIdeal}^{\mathsf{Sim}}_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)]$ *is* $\leq \mathsf{negl}(\lambda)$*, where* SFReal *and* SFIdeal *are games defined as in Figure 31 and* $\mathsf{V}_{\mathsf{B}^*}$ *is* B^* *view that includes all messages and its internal state.*
- *for all $i \in [R]$, there exists a p.p.t. stateful simulator* $\mathsf{Sim}^i = (\mathsf{Sim}^i_0, \mathsf{Sim}^i_1, \mathsf{Sim}^i_2)$ *such that for any p.p.t. distinguisher $\mathcal{D}$, the difference in* $\Pr[\mathcal{D}(\mathsf{V}_{\mathsf{B}^*}) = 1 : \mathsf{V}_{\mathsf{B}^*} \leftarrow \mathsf{SFReal}^i_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)]$ *and* $\Pr[\mathcal{D}(\mathsf{V}_{\mathsf{B}^*}) = 1 : \mathsf{V}_{\mathsf{B}^*} \leftarrow \mathsf{SFIdeal}^{i,\mathsf{Sim}^i}_{\mathsf{B}^*,\mathsf{T}}(1^\lambda, x, \mathsf{tk}_{\mathsf{ask}}, f)]$ *is* $\leq \mathsf{negl}(\lambda)$*, where* SFReal^i *game (defined in Figure 31) is real-world game where the buyer* B^* *aborts after receiving i^{th} message of the exchange phase and* $\mathsf{SFIdeal}^i$ *game (defined in Figure 31) is ideal-world game where the simulator*

simulates the buyer B^**'s view upto* i *rounds of the exchange phase and* $\mathsf{V}_{\mathsf{B}^*}$ *is defined as the view of* B^* *in the two games respectively.*

Crucially, Sim_2 in $\mathsf{SFIdeal}$ game gets $f(x)$ as input, whereas for all $i \in [R]$, Sim_2^i in $\mathsf{SFIdeal}^i$ does not.

Remark 9.2. *In this MPC-like setting, the malicious buyer* B *has no private input and specifies* f*, after which the honest seller already knows* $f(x)$*. If* B *aborts, it learns nothing; otherwise, it learns* $f(x)$*. This differs from standard MPC, where the malicious party sees the output first and decides whether the honest party learns it—hence, requiring a different ideal functionality.*

9.1 FFE protocol from FAS

In this section, we formalize the FFE protocol obtained from FAS that was informally described in Section 1 and in [VST24a]. Let FAS be defined w.r.t. a digital signature scheme DS, a hard relation R, and a family of functions $\mathcal{F}$. Then, FFE for $\mathcal{F}$ is as follows, where the role of T is played by a blockchain and B owns a key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{DS.KGen}(1^\lambda)$ where vk is public.

- $\mathsf{Setup}(1^\lambda)$: **ret** $\mathsf{pp} \leftarrow \mathsf{FAS.Setup}(1^\lambda)$.
- $\mathsf{AdGen}(\mathsf{pp}, x, \mathsf{tk}_{\mathsf{ask}})$: A computes X ensuring $(X, x) \in R$, and computes $(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{FAS.AdGen}(\mathsf{pp}, X, x)$ to send $x_{\mathsf{pub}} := (X, \mathsf{ad}, \mathsf{tk}_{\mathsf{ask}})$ to B and keep $x_{\mathsf{prv}} := (x, \mathsf{st})$ locally.
- $\mathsf{AdVerify}(\mathsf{pp}, x_{\mathsf{pub}})$ is run by B and it computes $\mathsf{FAS.AdVerify}(\mathsf{pp}, X, \mathsf{ad})$. If the output is 0, B aborts, else it continues to the exchange phase described next.
- $[\alpha, \beta] \leftarrow [\mathsf{A}(x_{\mathsf{prv}}), \mathsf{B}(\mathsf{tk}[n]), \mathsf{T}](\mathsf{pp}, x_{\mathsf{pub}}, f, \mathsf{vk})$:
  1. A computes and sends $\widetilde{\sigma_1}$ to B where $\widetilde{\sigma_1} \leftarrow\!\!\$\ \mathsf{FAS.FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, X, x, f)$.
  2. B aborts if $\mathsf{FAS.FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, X, f, \widetilde{\sigma_1}) = 0$. Else, B computes $\widetilde{\sigma_2} \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, X, f, \widetilde{\sigma_1})$ where $m :=$ "transfer $\mathsf{tk}[n]$ from W_B to W_A" and $n := \mathsf{tk}_{\mathsf{ask}}$. B sends $\widetilde{\sigma_2}$ to A.
  3. A aborts if $\mathsf{FAS.FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, X, f, \widetilde{\sigma} := (\widetilde{\sigma_1}, \widetilde{\sigma_2})) = 0$ or if the number of tokens n in the message m satisfy $n < \mathsf{tk}_{\mathsf{ask}}$. Else, A computes $\sigma \leftarrow \mathsf{FAS.Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, X, x, f, \widetilde{\sigma} := (\widetilde{\sigma_1}, \widetilde{\sigma_2}))$ and sends the signed transaction (m, σ) to the blockchain T.
     - T rejects the signed transaction (m, σ) if $\mathsf{DS.Vf}(\mathsf{vk}, m, \sigma) = 0$, else it accepts and transfers tokens $\mathsf{tk}[n]$ from W_B to W_A.
     - When B sees signed transaction (m, σ) posted to the blockchain T, it computes $\beta := \mathsf{FAS.FExt}(\mathsf{ad}, \mathsf{vk}, X, f, \widetilde{\sigma} := (\widetilde{\sigma_1}, \widetilde{\sigma_2}), \sigma)$.

     The protocol ends at this point and A's output is $\alpha := \mathsf{owner}(\mathsf{tk}[n], \mathsf{W}_\mathsf{A})$ and B's output is β.

Correctness. For honest A and B, the correctness of the FFE protocol follows from that of FAS.

Theorem 9.3. *The FFE protocol in Section 9.1 satisfies:*

- *advertisement binding when* FAS *satisfies advertisement binding.*
- *buyer fairness when* FAS *satisfies witness extractability and* DS *is unforgeable.*
- *seller fairness when* FAS *satisfies zero-knowledge (non-abort) and zero-knowledge (abort).*

9.2 Proof of Theorem 9.3

In this section, we discuss buyer and seller fairness. We note that advertisement binding of FFE follows from advertisement binding of FAS naturally.

Buyer Fairness. The FFE extractor algorithm $\mathcal{E}$ is essentially the same as the FAS extractor algorithm $\mathcal{E}$ associated with advertisement binding. Suppose its output is x^*. Then, it suffices to analyze all the cases when honest buyer's output $\beta \neq f(x^*)$. In each of the cases, we will argue that if $\exists i \in [n] s.t. \mathsf{tk}_i \notin \mathsf{W}_\mathsf{B}$, then, we can break security of one of the underlying primitives. There are three cases here:

1. Case 1: B aborted because $\mathsf{AdVerify}$ failed and in this case $\beta = \perp$ canonically.
2. Case 2: B aborted because in the exchange phase $\mathsf{FAS.FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, X, f, \widetilde{\sigma_1}) = 0$. In this case, $\beta = \perp$ canonically.
3. Case 3: B sent $\widetilde{\sigma_2}$ to A in step 2 of exchange phase and A posted σ on the blockchain T to get the tokens $\mathsf{tk}[n]$ and then B computed $\beta := \mathsf{FAS.FExt}(\mathsf{ad}, \mathsf{vk}, X, f, \widetilde{\sigma} := (\widetilde{\sigma_1}, \widetilde{\sigma_2}), \sigma)$, but it is the case that $\beta \neq f(x^*)$.

For cases 1 and 2, if $\exists i \in [n] s.t.\ \mathsf{tk}_i \notin \mathsf{W_B}$, then, we can design a simple reduction that breaks unforgeability of the digital signature scheme DS with respect to which FAS is defined. For case 3, if $\exists i \in [n] s.t. \mathsf{tk}_i \notin \mathsf{W_B}$, then, we can design a simple reduction that breaks the witness extractability of the FAS scheme.

Seller Fairness. Observe that the FAS-based FFE protocol has $R = 1$ number of rounds in the exchange phase. So, we need to show malicious buyer B^*'s view $\mathsf{V}_{\mathsf{B}^*}$ (i) in SFReal and $\mathsf{SFIdeal}$ games is indistinguishable and (ii) in SFReal^1 and $\mathsf{SFIdeal}^2$ games is indistinguishable. We prove the former using the zero-knowledge (non-abort) property of FAS and the latter by using the zero-knowledge (abort) property of FAS.

1. We describe how to construct the simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ by using the FAS zero-knowledge (non-abort) simulator $\mathsf{FAS.Sim} = (\mathsf{FAS.Setup}^*, \mathsf{FAS.AdGen}^*, \mathsf{FAS.FPreSign}_1^*, \mathsf{FAS.Adapt}^*)$. $\mathsf{Sim}_0(1^\lambda)$ simply outputs $\mathsf{pp}^* \leftarrow \mathsf{FAS.Setup}^*(1^\lambda)$. $\mathsf{Sim}_1(\mathsf{pp}^*, \mathsf{tk_{ask}})$ samples $X \in_R$, runs $(\mathsf{ad}^*, \mathsf{st}^*) \leftarrow \mathsf{FAS.AdGen}^*(\mathsf{pp}^*, X)$ and outputs $x^*_{\mathsf{pub}} := (X, \mathsf{ad}^*, \mathsf{tk_{ask}})$. $\mathsf{Sim}_2(f(x))$ next computes $\widetilde{\sigma_1^*} \leftarrow \mathsf{FAS.FPreSign}_1^*(\mathsf{pp}^*, \mathsf{ad}^*, \mathsf{st}^*, X, f(x), f)$ and sends $\widetilde{\sigma_1^*}$ to B^* in step 1 of exchange phase. In step 3 of exchange phase, it computes $\sigma := \mathsf{FAS.Adapt}^*(\mathsf{ad}^*, \mathsf{st}^*, \mathsf{vk}, m, X, f(x), f, (\widetilde{\sigma_1^*}, \widetilde{\sigma_2}))$ and posts σ to the blockchain T. From the description of the FFE simulator, we see that if FAS satisfies zero-knowledge (non-abort), then the FFE protocol satisfies seller fairness' first requirement.
2. We describe how to construct the simulator $\mathsf{Sim}^{\mathsf{a}} = (\mathsf{Sim}_0^{\mathsf{a}}, \mathsf{Sim}_1^{\mathsf{a}}, \mathsf{Sim}_2^{\mathsf{a}})$ by using the FAS zero-knowledge (abort) simulator $\mathsf{FAS.Sim} = (\mathsf{FAS.Setup}^{*,\mathsf{a}}, \mathsf{FAS.AdGen}^{*,\mathsf{a}}, \mathsf{FAS.FPreSign}_1^{*,\mathsf{a}})$. $\mathsf{Sim}_0^{\mathsf{a}}, \mathsf{Sim}_1^{\mathsf{a}}$ use $\mathsf{FAS.Setup}^{*,\mathsf{a}}, \mathsf{FAS.AdGen}^{*,\mathsf{a}}$ like in the previous case. $\mathsf{Sim}_2^{\mathsf{a}}$ computes $\widetilde{\sigma_1^*} \leftarrow \mathsf{FAS.FPreSign}_1^{*,\mathsf{a}}(\mathsf{pp}^*, \mathsf{ad}^*, \mathsf{st}^*, X, f)$ and sends $\widetilde{\sigma_1^*}$ to B^* in step 1 of the exchange. From the description of the FFE simulator, we see that if FAS satisfies zero-knowledge (abort), then the FFE protocol satisfies seller fairness' second requirement.

10 Conclusion

In this work, we present new constructions for Functional Adaptor Signatures (FAS) using homomorphic encryption, supporting higher-degree functions (e.g., BGN-FAS for quadratic, FHE-FAS for general functions) and achieving concrete efficiency gains—even our LHE-FAS outperforms VST24 by an order of magnitude. We implement and benchmark all variants across five applications, demonstrating their expressiveness and practicality. We also formalize *fair functional exchange* (FFE), a simulation-based notion capturing fairness beyond all-or-nothing exchanges. Our work opens future directions, including designing FAS without NIZKs to reduce latency, and extending FAS beyond two-party settings to enable fair multiparty exchanges and decentralized coordination.

Acknowledgements

We thank Elaine Shi and the anonymous reviewers for their valuable feedback. Nikhil Vanjani was supported by the following grants awarded to Elaine Shi: the National Science Foundation (award numbers CNS-2044679 and CNS-2212746), the Office of Naval Research (award number

N000142212064), and the Algorand Foundation. Garrett Greiner and Pratik Soni were supported by the State of Utah's Governor's Office of Economic Opportunity (IAA #230632319) through a seed grant from the Center for Financial Technology (University of Utah). Garrett Greiner, Pratik Soni and Sri AravindaKrishnan Thyagarajan were also supported by gifts from the Stellar Development Foundation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

[ac22] arkworks contributors. `arkworks` zksnark ecosystem, 2022. https://arkworks.rs.

[ACGSV24] Diego F Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez. Heliopolis: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 302–334. Springer, 2024.

[ACJ24] Rashmi Agrawal, Anantha Chandrakasan, and Ajay Joshi. Heap: A fully homomorphic encryption accelerator with parallelized bootstrapping. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 756–769. IEEE, 2024.

[AEE+20] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. Cryptology ePrint Archive, 2020.

[AEE+21] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security*, 2021.

[Ama93] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[AMSKM21] Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure Multi-Hop payments without Two-Phase commits. In *30th USENIX Security Symposium*, 2021.

[ATM+22] Lukas Aumayr, Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, Pedro Moreno-Sanchez, and Matteo Maffei. Sleepy channels: Bi-directional payment channels without watchtowers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, 2022.

[BGKS25] Ruben Baecker, Paul Gerhart, Jonathan Katz, and Dominique Schröder. Fair exchange for decentralized autonomous organizations via threshold adaptor signatures. Cryptology ePrint Archive, Paper 2025/388, 2025.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography*, 2005.

[BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Paper 2011/277, 2011.

[bip] Bip 340: Schnorr signatures for secp256k1. https://github.com/bitcoin/bips/blob/master/bip-0340/reference.py.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Annual Cryptology Conference*, 2014.

[BS17] Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2017.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N Rothblum, Ron D Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. 2000.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, 2003 Proceedings 22*, 2003.

[cona] https://github.com/zama-ai/concrete.

[conb] https://github.com/zama-ai/concrete-ml.

[COPZ22] Melissa Chase, Michele Orrù, Trevor Perrin, and Greg Zaverucha. Proofs of discrete logarithm equality across groups. Cryptology ePrint Archive, Paper 2022/1593, 2022.

[CZK+23] Dake Chen, Yuke Zhang, Souvik Kundu, Chenghao Li, and Peter A. Beerel. Rna-vit: Reduced-dimension approximate normalized attention vision transformers for latency efficient private inference. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9, 2023.

[Dec] Decred. dcrd. https://github.com/decred/dcrd.

[DLZ+23] Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Chen. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533*, 2023.

[DMVVZ25] Pierpaolo Della Monica, Ivan Visconti, Andrea Vitaletti, and Marco Zecchini. Public channel-based fair exchange protocols with advertising. *arXiv preprint arXiv:2503.10411*, 2025.

[DOY22] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. Cryptology ePrint Archive, Paper 2022/1687, 2022.

[EFH+21] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In *IACR International Conference on Public-Key Cryptography*, pages 451–480. Springer, 2021.

[fas] https://github.com/nikhilvanjani/fas-from-he-impl.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, 1990.

[GGW24] Sanjam Garg, Aarushi Goel, and Mingyuan Wang. How to prove statements obliviously? In *Annual International Cryptology Conference*, pages 449–487. Springer, 2024.

[GMM+22] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, 2022.

[GOS] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006.*

[GSST24] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri AravindaKrishnan Thyagarajan. Foundations of adaptor signatures. In *Advances in Cryptology – EUROCRYPT 2024*, 2024.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. Cryptology ePrint Archive, 2013.

[han] Secp256k1 python. https://github.com/hanabi1224/Programming-Language-Benchmarks/blob/main/bench/algorithm/secp256k1/1.py.

[Hol19] Justin Holmgren. On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Paper 2019/1261, 2019.

[JCT+25] Siddharth Jayashankar, Edward Chen, Tom Tang, Wenting Zheng, and Dimitrios Skarlatos. Cinnamon: A framework for scale-out encrypted ai. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 133–150, 2025.

[KKC+23] Jongmin Kim, Sangpyo Kim, Jaewan Choi, Jaiyoung Park, Donghwan Kim, and Jung Ho Ahn. Sharp: A short-word hierarchical accelerator for robust and practical fully homomorphic encryption. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–15, 2023.

[LHG+23] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive*, 2023.

[Lyn] Ben Lynn. Pbc library. https://crypto.stanford.edu/pbc/.

[MTV+22] Varun Madathil, Sri AravindaKrishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. *Cryptology ePrint Archive*, 2022.

[Poe17] Andrew Poelstra. Scriptless scripts. *Presentation Slides*, 2017.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. In *Annual International Cryptology Conference*, 2019.

[Ren10] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[sec] Bitcoin wiki: Secp256k1. https://en.bitcoin.it/wiki/Secp256k1.

[SS20] Sacha Servan-Schreiber. Bgn, 2020. https://github.com/sachaservan/bgn.

[SS24] Nikola Samardzic and Daniel Sanchez. Bitpacker: Enabling high arithmetic efficiency in fully homomorphic encryption accelerators. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 137–150, 2024.

[TBM+20] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[tfh] https://github.com/zama-ai/tfhe-rs.

[TM21] Sri Aravinda Krishnan Thyagarajan and Giulio Malavolta. Lockable signatures for blockchains: Scriptless scripts for all signatures. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 937–954. IEEE, 2021.

[TMMS22] Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, and Pedro Moreno-Sanchez. Universal atomic swaps: Secure exchange of coins across all blockchains. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1299–1316. IEEE, 2022.

[TSZ+24] Ertem Nusret Tas, István András Seres, Yinuo Zhang, Márk Melczer, Mahimna Kelkar, Joseph Bonneau, and Valeria Nikolaenko. Atomic and fair data exchange via blockchain. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, 2024.

[VST24a] Nikhil Vanjani, Pratik Soni, and Sri AravindaKrishnan Thyagarajan. Functional adaptor signatures: Beyond all-or-nothing blockchain-based payments. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024.

[VST24b] Nikhil Vanjani, Pratik Soni, and Sri AravindaKrishnan Thyagarajan. Functional adaptor signatures: Implementation, 2024. https://github.com/nikhilvanjani/fas-impl.

[ZYH+24] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. *Cryptology ePrint Archive*, 2024.

A Security Proofs for FHE-FAS

In this section, we provide formal security proofs for the FAS construction presented in Figure 20. The structure of this section is as follows. Section A.1 contains the proof for correctness (Definition 4.2). Section A.2 contains the proofs for advertisement soundness (Definition 4.3) and advertisement binding (Definition 4.4). Section A.3 contains the proofs for pre-signature adaptability (Definition 4.8) and pre-signature validity (Definition 4.5). Section A.4 contains the proof for unforgeability (Definition 4.6). Section A.5 contains the proof for witness extractability (Definition 4.7). Section A.6 contains the proof for non-abort case of zero knowledge (Definition 4.9). Section A.7 contains the proof for the abort case of zero knowledge (Definition 4.10). These proofs, when combined, prove Theorem 5.2.

A.1 Correctness

First, we will show correctness for the FAS construction presented in Figure 20.

Theorem A.1 (Correctness of FAS Construction)**.** *If the NIZK schemes* $\mathsf{NIZK}_{\mathsf{ad}}$ *and* $\mathsf{NIZK}_{\mathsf{eq}}$ *satisfy completeness, the adpator signature scheme* AS *satisfies correctness, and the encryption schemes* FHE *and* $\mathsf{ElGamal}$ *satisfy correctness, then the* FAS *construction in Figure 20 satisfies the property of correctness as defined in Definition 4.2.*

Proof. For every $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}^*$, every statement/witness pair $(Y, y) \in \mathsf{R}$, and every function $f \in \mathcal{F}$, assume that everything is run honestly. That is, suppose that $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, Y, y)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, \mathsf{st}, Y, y, f)$, $\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$, $\sigma := \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2))$, and $z := \mathsf{FExt}(\mathsf{ad}, \mathsf{vk}, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2), \sigma)$. Then, notice that:

1. $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$ due to the completeness of $\mathsf{NIZK}_{\mathsf{ad}}$,
2. $\mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1) = 1$ due to completeness of $\mathsf{NIZK}_{\mathsf{eq}}$ and corretness of FHE,
3. $\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma}) = 1$ due to the correctness of the adaptor signature AS,
4. $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$ due to the correctness of AS, and
5. $z = f(y)$ due to the correctness of the AS and the $\mathsf{ElGamal}$ encryption scheme EG.

Therefore, we have that FAS satisfies the property of correctness, as desired. □

A.2 Advertisement Soundness and Binding

Theorem A.2 (Advertisement Soundness for FAS Construction)**.** *If the* NIZK *argument for the language* L_{ad} *satisfies adaptive soundness, then the proposed* FAS *construction in Figure 20 satisfies the property of advertisement soundness as defined in Definition 4.3.*

Proof. We will show that if a p.p.t. adversary $\mathcal{A}$ breaks the advertisement soundness of our FAS construction with non-negligible advantage, then there exists a p.p.t. reduction $\mathcal{B}$ that is able to break the adaptive soundness of the underlying NIZK scheme. Let $\mathcal{C}$ be the challenger for the adaptive soundness of the NIZK argument for the language L_{ad}. The reduction $\mathcal{B}$ is as follows:

1. The challenger $\mathcal{C}$ sends the $\mathsf{crs}_{\mathsf{ad}}$ to $\mathcal{B}$.
2. $\mathcal{B}$ generates the public parameters pp and sends it to $\mathcal{A}$.
3. $\mathcal{A}$ generates an advertisement ad for statement Y and sends the tuple (ad, Y) to $\mathcal{B}$.
4. $\mathcal{B}$ parses ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$. It then sends the statement $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$ and proof π_{ad} to the challenger $\mathcal{C}$.

Note that when $\mathcal{A}$ breaks the advertisement soundness of the FAS scheme, we have that $Y \notin L_{\mathsf{R}}$ and $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$. Further, the fact that $Y \notin L_{\mathsf{R}}$ implies that $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y) \notin L_{\mathsf{ad}}$. Also, we know that since $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$, then it must be the case that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1$. Therefore, by definition, $\mathcal{B}$ is able to use $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ to break the adaptive soundness of the NIZK argument for the language L_{ad}. Further, since $\mathcal{A}$ breaks the advertisement soundness of FAS with non-negligible probability, then $\mathcal{B}$ breaks the adaptive soundness of NIZK with the same non-negligible probability. This is a contradiction to our origianl assumption, which concludes the proof.

□

Theorem A.3 (Advertisement Binding for FAS Construction)**.** *If the* NIZK *argument* $\mathsf{NIZK}_{\mathsf{ad}}$ *for the language* L_{ad} *satisfies adaptive soundness, and the* FHE *scheme satisfies binding (see Footnote 7), then the proposed* FAS *construction in Figure 20 satisfies advertisement binding as defined in Definition 4.4.*

Proof. We want to show that there exists a negligible function ϵ such that for every p.p.t. malicious seller $\mathcal{A}$, and for all $\lambda \in \mathbb{N}$, if $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(Y, \mathsf{ad}) \leftarrow \mathcal{A}^*(\mathsf{pp})$, such that $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$, then both the following hold:

$$\Pr\left[\begin{array}{c} \nexists(y, r)\ s.t. \\ \mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot) \end{array}\right] \leq \epsilon(\lambda)\ ,\ \Pr\left[\begin{array}{c} \exists(y, r), (y', r')\ s.t. \\ y \neq y' \wedge \\ \mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot) \wedge \\ \mathsf{AdGen}(\mathsf{pp}, Y, y'; r') = (\mathsf{ad}, \cdot) \end{array}\right] \leq \epsilon(\lambda)\ .$$

First, we will prove the former condition. Recall that ad is parsed as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$. Since we have that $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$, we know that for $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$, we have that $\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1$. By the the adaptive soundness of $\mathsf{NIZK}_{\mathsf{ad}}$, we know that there exists a negligible function ϵ such that:

$$\Pr[\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}] \leq \epsilon(\lambda)\ .$$

Recall that $\mathsf{stmt}_{\mathsf{ad}} \in L_{\mathsf{ad}}$ implies that there exists a witness (y, r_1, r_0) such that $\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1)$, $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) \leftarrow \mathsf{FHE.KGen}(1^\lambda; r_0)$, and $(Y, y) \in \mathsf{R}$. Note that $r := (r_0, r_1)$

are the random coins used during AdGen. Therefore, $\mathsf{stmt}_{\mathsf{ad}} \in L_{\mathsf{ad}}$ implies that $\exists(y, r)$ such that $\mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot)$. Combining this observation with the equation above, we obtain that:

$$\Pr\left[\begin{array}{c} \nexists(y,r)\ s.t. \\ \mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot) \end{array}\right] \leq \epsilon(\lambda) \ ,$$

as desired.

Now, we will prove the latter condition. Suppose there exists $(y, r), (y', r')$ such that $y \neq y'$, and $\mathsf{AdGen}(\mathsf{pp}, Y, y; r) = \mathsf{AdGen}(\mathsf{pp}, Y, y'; r') = (\mathsf{ad}, \cdot)$. Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$. Also, parse the random coins as $r = (r_0, r_1)$ and $r' = (r'_0, r'_1)$.

Since the advertisements generated with (y, r) and (y, r') are the same, then it must be that $\mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1) = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_1) = \mathsf{ct}_{\mathsf{ad}}$. However, since the FHE scheme satisfies binding, it must be the case that $y = y'$, a contradiction to our original assumption. Therefore, we have that no such (y, r) and (y', r') can exist, giving that:

$$\Pr\left[\begin{array}{c} \exists(y,r),(y',r')\ s.t. \\ y \neq y' \ \wedge \\ \mathsf{AdGen}(\mathsf{pp}, Y, y; r) = (\mathsf{ad}, \cdot) \ \wedge \\ \mathsf{AdGen}(\mathsf{pp}, Y, y'; r') = (\mathsf{ad}, \cdot) \end{array}\right] = 0 \leq \epsilon(\lambda) \ ,$$

as desired. This concludes the proof □

A.3 Pre-Signature Adaptability and Validity

Theorem A.4 (Pre-Signature Adaptability for FAS Construction)**.** *If the underlying adaptor signature scheme* AS *satisfies pre-signature adaptability (Definition 3.26), then the proposed* FAS *construction in Figure 20 satisfies the property of pre-signature adaptability as defined in Definition 4.8.*

Proof. For every $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, any statement/witness pair $(Y, y) \in \mathsf{R}$, any $(\mathsf{ad}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, \mathsf{ad}, Y)$, any message $m \in \{0,1\}^*$, any family of functions $\mathcal{F}$, any function $f \in \mathcal{F}$, any $\widetilde{\sigma}_1 \leftarrow \mathsf{FPreSign}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1)$, any key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$, and any pre-signature $\widetilde{\sigma}_2$ with $\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 1$, suppose that we compute $\sigma := \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma})$. We want to show that:

$$\Pr[\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma}))] = 1 \ .$$

Note that since $\mathsf{FPreSign}_2$ passed, we know that $\mathsf{AS.pVrfy}(\mathsf{vk}, m, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2) = 1$, where $\mathsf{pk}_{\mathsf{EG}}$ is obtained from $\widetilde{\sigma}_1$. Let $\mathsf{sk}_{\mathsf{EG}}$ be the corresponding secret key to $\mathsf{pk}_{\mathsf{EG}}$. By the pre-signature adaptability property of the underlying adaptor signature scheme, we have that:

$$\Pr[\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{sk}_{\mathsf{EG}}, \widetilde{\sigma}_2))] = 1 \ .$$

Notice that we have that $\mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma}) = \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{sk}_{\mathsf{EG}}, \widetilde{\sigma}_2)$. Therefore, we have that:

$$\Pr[\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{ad}, \mathsf{st}, \mathsf{vk}, m, Y, y, f, \widetilde{\sigma}))] = 1 \ ,$$

as desired. This concludes the proof. □

Theorem A.5 (Pre-Signature Validity for FAS Construction)**.** *If the underlying adaptor signature scheme* AS *satisfies correctness, then the proposed* FAS *construction in Figure 20 satisfies the property of pre-signature validity as defined in Definition 4.5.*

Proof. For every $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, any statement/witness pair $(Y, y) \in \mathsf{R}$, any message $m \in \{0,1\}^*$, any family of functions $\mathcal{F}$, every function $f \in \mathcal{F}$, any ad with $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{ad}, Y) = 1$, any $\widetilde{\sigma}_1$ such that $\mathsf{FPreVerify}_1(\mathsf{pp}, \mathsf{ad}, Y, f, \widetilde{\sigma}_1) = 1$, and any $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$, suppose that we compute $\widetilde{\sigma}_2 \leftarrow \mathsf{FPreSign}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{sk}, m, Y, f, \widetilde{\sigma}_1)$. We want to show that:

$$\Pr[\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 1] = 1 \ .$$

Note that $\mathsf{FPreSign}_2$ simply performs $\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$, where $\mathsf{pk}_{\mathsf{EG}}$ is a public key for the $\mathsf{ElGamal}$ encryption scheme. Recall that the public keys of $\mathsf{ElGamal}$ are elements of a group $\mathbb{G}$ where the discrete log problem is hard, so we have that $\mathsf{pk}_{\mathsf{EG}} \in L_{\mathsf{DL}}$ (see Definition 3.3). Therefore, $\mathsf{pk}_{\mathsf{EG}}$ is a compatible statement with Schnorr adaptor signatures. Therefore, $\widetilde{\sigma}_2$ is well-formed.

By correctness of the adaptor signature scheme, we have that since $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$ and $\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$, then it must be that:

$$\Pr[\mathsf{AS.pVrfy}(\mathsf{vk}, m, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2) = 1] = 1 \ .$$

Note that $\mathsf{FPreVerify}_2$ simply checks if $\mathsf{AS.pVrfy}(\mathsf{vk}, m, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2) = 1$. So we have that:

$$\Pr[\mathsf{FPreVerify}_2(\mathsf{pp}, \mathsf{ad}, \mathsf{vk}, m, Y, f, \widetilde{\sigma} := (\widetilde{\sigma}_1, \widetilde{\sigma}_2)) = 1] = \Pr[\mathsf{AS.pVrfy}(\mathsf{vk}, m, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2) = 1] = 1 \ ,$$

as desired. This concludes the proof.

□

A.4 Unforgeability

Theorem A.6 (Unforgeability for FAS Construction)**.** *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy adaptive soundness, the underlying* AS *scheme satisfies witness extractability, and the relation* R *is* $\mathcal{F}$*-hard, then the proposed* FAS *construction in Figure 20 satisfies the property of unforgeability as defined in Definition 4.6.*

Overview. In order to prove that the FAS construction is unforgeable, we will reduce it to the $\mathcal{F}$-hardness of the relation R. Consider the sequence of games G_0, G_1, G_2, G_3 in Figure 32.

- Game G_0 is the original game $\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose. In this game, the adversary $\mathcal{A}$ is tasked with creating a valid forgery σ^* for a message m^* of their choice. The adversary has access to the signature and functional pre-signature oracles, which are denoted $\mathcal{O}_{\mathsf{s}}$ and $\mathcal{O}_{\mathsf{fps}_2}$, respectively.

- Game G_1 is identical to Game G_0, except that after line 15, the game additionally checks if $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$ is in the language L_{ad}. If it is not in the language, then we set the flag $\mathsf{bad}_1 = \mathsf{true}$. The winning condition of G_1 is also different from G_0, as in order for $\mathcal{A}$ to win in G_1, it must also satisfy $\neg\mathsf{bad}_1$. That is, the flag bad_1 must be false.

- Game G_2 is identical to Game G_1, except that after line 16, the game additionally checks if $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$ is in the language L_{eq}. If it is not in the language, then we set the flag $\mathsf{bad}_2 = \mathsf{true}$. The winning condition of G_2 is also different from G_1, as in order for $\mathcal{A}$ to win in G_2, it must also satisfy $\neg\mathsf{bad}_2$. That is, the flag bad_2 must be false.

- Game G_3 is identical to Game G_2, except that after line 17, the game additionally checks if the pair $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \in \mathsf{R}_{\mathsf{EG}}$. If this is not a valid key pair, then we set the flag $\mathsf{bad}_3 = \mathsf{true}$. The winning condition of G_3 is also different from G_2, as in order for $\mathcal{A}$ to win in G_3, it must also satisfy $\neg\mathsf{bad}_3$. That is, the flag bad_3 must be false.

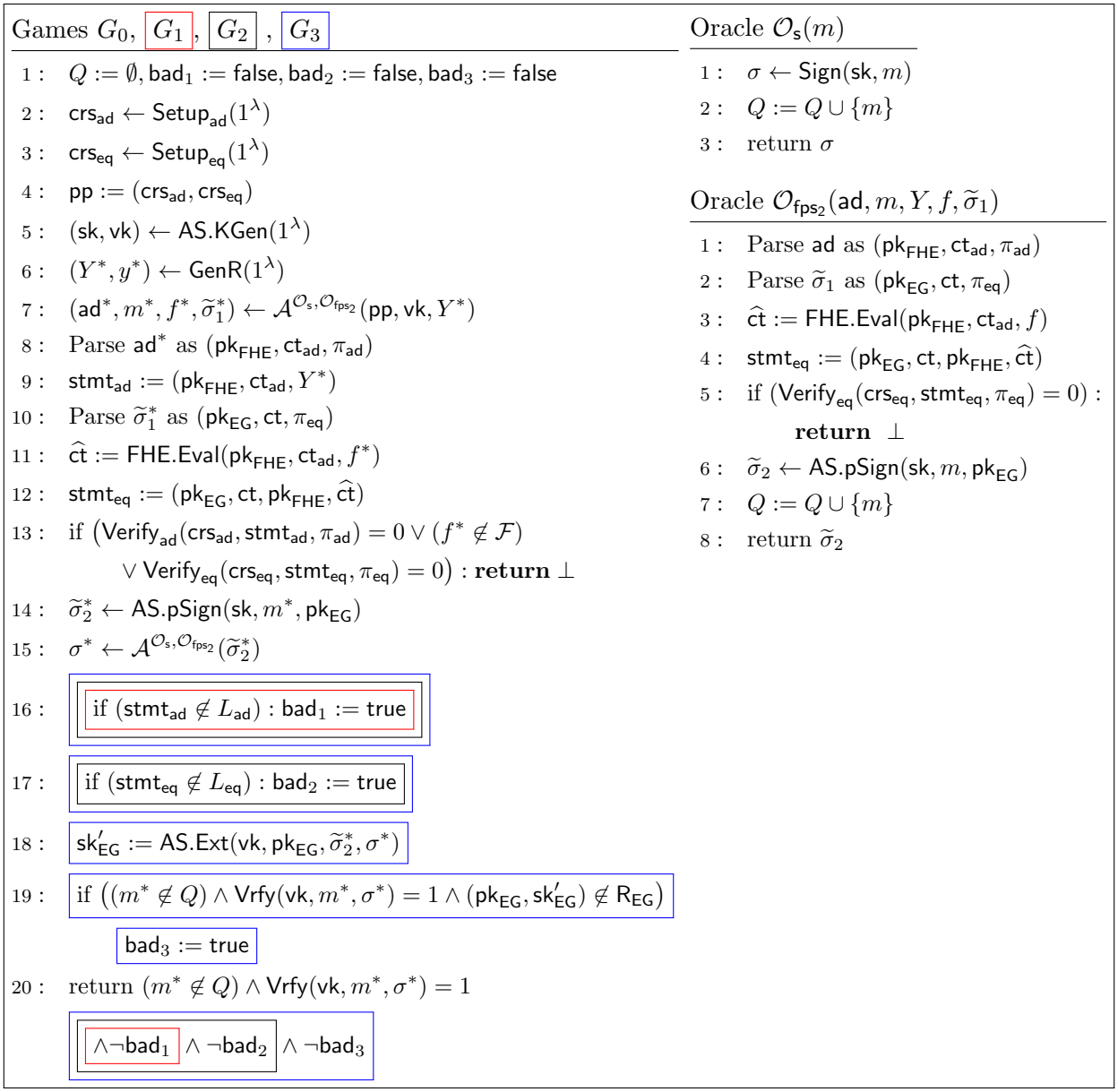

Figure 32: Unforgeability Proof: Games G_0, G_1, G_2

First, we will argue in Claims A.7, A.8, and A.9 that the game hops from G_0 to G_3 are close. After, we will argue in Claim A.10 that if there exists a stateful p.p.t. adversary $\mathcal{A}$ that wins G_3 with non-negligible probability, then $\mathcal{A}$ could be used to break the $\mathcal{F}$-hardness of the relation R. The formal proof for Theorem A.6 can be found in Claim A.11. □

Claim A.7. *If the* NIZK *argument for language* L_{ad} *satisfies adaptive soundness, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*:*

$$\big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| \leq \epsilon(\lambda) \ .$$

Proof. Assume towards a contradiction that $\big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| > \epsilon(\lambda)$. Notice that since

Reduction $\mathcal{P}$ for the proof of Claim A.7

$Q = \emptyset, \mathsf{bad}_1 := \mathsf{false}$
$\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathcal{C}$
$\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
$\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$
$(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
$(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{\mathsf{fps}_2}}(\mathsf{pp}, \mathsf{vk}, Y^*)$
Parse ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
$\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$
if $(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 0)$: **return** $\perp$
return $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ to $\mathcal{C}$

Oracle $\mathcal{O}_s(m)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$Q := Q \cup \{m\}$
return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0)$: **return** $\perp$
$\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$
$Q := Q \cup \{m\}$
return $\widetilde{\sigma}_2$

Figure 33: Reduction for prover $\mathcal{P}$ playing the adaptive soundness game with challenger $\mathcal{C}$

Games G_0 and G_1 are identical except for the check in G_1 that determines the flag bad_1, we have:

$$\begin{aligned} \Pr[\mathsf{bad}_1 = \mathsf{true}, \text{ in } G_1] &\geq \big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| \\ &> \epsilon(\lambda) \ . \end{aligned} \tag{1}$$

We want to show that if this is the case, then there exists some p.p.t. prover $\mathcal{P}$ that uses $\mathcal{A}$ to break the adaptive soundness of the NIZK argument for the language L_{ad}. Recall that in order to win this game, the prover $\mathcal{P}$ needs to find a pair $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ such that it satisfies two conditions: $(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) \wedge (\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1)$. So we have that $\Pr[\mathcal{P} \text{ wins}]$ is given by:

$$\Pr[\mathcal{P} \text{ wins}] = \Pr\big[(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) \wedge (\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1)\big] \ . \tag{2}$$

The reduction $\mathcal{P}$ in Figure 33 will simultaneously play the role of challenger to adversary $\mathcal{A}$ as in Game G_1, and the role of the prover in the adaptive soundness game for the NIZK argument for the language L_{ad}. Let $\mathcal{C}$ be the corresponding challenger for that adaptive soundness game. The procedure for the reduction is as follows:

1. $\mathcal{C}$ honestly samples a $\mathsf{crs}_{\mathsf{ad}}$, and sends it to $\mathcal{P}$.

2. $\mathcal{P}$ generates the public parameters pp (which includes the $\mathsf{crs}_{\mathsf{ad}}$ from $\mathcal{C}$), and also samples $(\mathsf{sk}, \mathsf{vk})$ for the AS scheme. It also generates an instance (Y^*, y^*) of the relation R. $\mathcal{P}$ then sends $(\mathsf{pp}, \mathsf{vk}, Y^*)$ to $\mathcal{A}$.

3. $\mathcal{A}$ performs queries to the signature and functional pre-signature oracles, $\mathcal{O}_s$ and $\mathcal{O}_{fps_2}$, which $\mathcal{P}$ answers.

4. After making queries, $\mathcal{A}$ then sends a response $(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*)$ to $\mathcal{P}$.

5. $\mathcal{P}$ then parses ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$, and constructs $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$. Then, $\mathcal{P}$ sends the response $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ to $\mathcal{C}$.

Notice that during Game G_1, when $(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}})$, we set the flag $\mathsf{bad}_1 := \mathsf{true}$ in line 16. But it must also have been the case that $\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1$, as otherwise, G_1 would have aborted in line 13 (and therefore it would not have reached line 16 to set the flag bad_1 to true). Therefore, we have that:

$$\Pr[\mathsf{bad}_1 = \mathsf{true},\ \text{in } G_1] = \Pr\left[(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) \wedge (\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1)\right] . \tag{3}$$

Notice that combining equations 1, 2 and 3 gives that:

$$\Pr[\mathcal{P} \text{ wins}] = \Pr[\mathsf{bad}_1 = \mathsf{true},\ \text{in } G_1] > \epsilon(\lambda) , \tag{4}$$

where ϵ is a negligible function. Therefore, we have constructed a prover $\mathcal{P}$ that wins the adaptive soundness game of the NIZK argument for the language L_{ad} with non-negligible probability, a contradiction to our initial assumption of the claim. This concludes the proof. □

Claim A.8. *If the* NIZK *argument for language* L_{eq} *satisfies adaptive soundness, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*:*

$$\left|\Pr[G_1 = 1] - \Pr[G_2 = 1]\right| \leq \epsilon(\lambda) .$$

Proof. Assume towards a contradiction that $\left|\Pr[G_1 = 1] - \Pr[G_2 = 1]\right| > \epsilon(\lambda)$. Notice that since Games G_1 and G_2 are identical except for the check that determines the flag bad_2, we have:

$$\begin{aligned} \Pr[\mathsf{bad}_2 = \mathsf{true}, \text{ in } G_2] &\geq \left|\Pr[G_1 = 1] - \Pr[G_2 = 1]\right| \\ &> \epsilon(\lambda) . \end{aligned} \tag{5}$$

We want to show that if this is the case, then there exists some p.p.t. prover $\mathcal{P}^*$ that uses $\mathcal{A}$ to break the adaptive soundness of the NIZK argument for the language L_{eq}. Recall that in order to win this game, the prover $\mathcal{P}^*$ needs to find a pair $(\mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}})$ such that it satisfies two conditions: $(\mathsf{stmt}_{\mathsf{eq}} \notin L_{\mathsf{eq}}) \wedge (\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1)$. So we have that $\Pr[\mathcal{P}^* \text{ wins}]$ is given by:

$$\Pr[\mathcal{P}^* \text{ wins}] = \Pr\left[(\mathsf{stmt}_{\mathsf{eq}} \notin L_{\mathsf{eq}}) \wedge (\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1)\right] . \tag{6}$$

Notice that the winning condition of $\mathcal{P}^*$ is similar to that of the winning condition for $\mathcal{P}$ in Claim A.7. Further, the hop from G_1 to G_2 is similar to that of the hop from G_0 to G_1. Therefore, the rest of this proof follows from the proof of Claim A.7. This concludes the proof. □

Claim A.9. *If the underlying adaptor signature scheme* AS *satisfies witness extractability, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*:*

$$\left|\Pr[G_2 = 1] - \Pr[G_3 = 1]\right| \leq \epsilon(\lambda) .$$

Proof. Assume towards a contradiction that $\left|\Pr[G_2 = 1] - \Pr[G_3 = 1]\right| > \epsilon(\lambda)$. Notice that since Games G_2 and G_3 are identical except for the check that determines the flag bad_3, we have:

$$\begin{aligned} \Pr[\mathsf{bad}_3 = \mathsf{true}, \text{ in } G_3] &\geq \left|\Pr[G_2 = 1] - \Pr[G_3 = 1]\right| \\ &> \epsilon(\lambda) . \end{aligned} \tag{7}$$

We want to show that if this is the case, then we can create some reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the witness extractability of the underlying AS scheme. Recall that in order to win the witness extractability game, $\mathcal{B}$ needs to find $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ such that all of the following conditions

Reduction $\mathcal{B}$ for the proof of Claim A.9

$Q = \emptyset, \mathsf{bad}_1 := \mathsf{false}, \mathsf{bad}_2 := \mathsf{false}, \mathsf{bad}_3 := \mathsf{false}$
$\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda),\ \mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
$\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$
$\mathsf{vk} \leftarrow \mathcal{C}$
$(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
$(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\mathsf{pp}, \mathsf{vk}, Y^*)$
Parse ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
$\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$
Parse $\widetilde{\sigma}_1^*$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f^*)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $\big(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 0 \vee (f^* \notin \mathcal{F})$
$\quad \vee \mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0\big)$: **return** $\perp$
$\widetilde{\sigma}_2^* \leftarrow \mathcal{C}(m^*, \mathsf{pk}_{\mathsf{EG}})$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\widetilde{\sigma}_2^*)$
return $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ to $\mathcal{C}$

Oracle $\mathcal{O}_s(m)$

$\sigma \leftarrow \mathcal{C}(m)$
$Q := Q \cup \{m\}$
return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0)$:
$\quad$ **return** $\perp$
$\widetilde{\sigma}_2 \leftarrow \mathcal{C}(m, \mathsf{pk}_{\mathsf{EG}})$
$Q := Q \cup \{m\}$
return $\widetilde{\sigma}_2$

Figure 34: Reduction for $\mathcal{B}$ playing the witness extractability game for AS with challenger $\mathcal{C}$

are met: $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)) \notin \mathsf{R}_{\mathsf{EG}}$. We will denote $\mathsf{sk}'_{\mathsf{EG}} = \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)$, so we get that:

$$\Pr[\mathcal{B} \text{ wins}] = \Pr\left[(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}\right] . \tag{8}$$

The reduction $\mathcal{B}$ in Figure 34 will simultaneously play the role of challenger to adversary $\mathcal{A}$ as in Game G_3, and the role of the adversary in the witness extractability game for the underlying AS scheme. Let $\mathcal{C}$ be the corresponding challenger for the witness extractability game for the underlying AS scheme. The procedure for the reduction is as follows:

1. $\mathcal{C}$ honestly samples the signing and verification keys $(\mathsf{sk}, \mathsf{vk})$, and sends vk to $\mathcal{B}$.
2. $\mathcal{B}$ generates the public parameters pp. It also generates an instance (Y^*, y^*) of the relation R. $\mathcal{B}$ then sends $(\mathsf{pp}, \mathsf{vk}, Y^*)$ to $\mathcal{A}$.
3. $\mathcal{A}$ performs queries to the signature and functional pre-signature oracles, $\mathcal{O}_s$ and $\mathcal{O}_{fps_2}$. Note that $\mathcal{B}$ answers these queries by forwarding the queries to its challenger, $\mathcal{C}$. After making queries, $\mathcal{A}$ then sends a response $(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*)$ to $\mathcal{B}$.
4. $\mathcal{B}$ then uses $\mathcal{C}$ to generate a honest pre-signature $\widetilde{\sigma}_2^*$ for $\mathcal{A}$'s chosen message m^* and statement $\mathsf{pk}_{\mathsf{EG}}$. $\mathcal{B}$ then forwards this pre-signature $\widetilde{\sigma}_2^*$ to $\mathcal{A}$.
5. After making more queries, $\mathcal{A}$ then sends a response σ^* to $\mathcal{B}$.
6. Then, $\mathcal{B}$ sends $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ to $\mathcal{C}$.

Now, we need to show that when the flag $\mathsf{bad}_3 = \mathsf{true}$ in G_3, the winning conditions for $\mathcal{B}$ are also met. We know that when the flag $\mathsf{bad}_3 = \mathsf{true}$, we have that the three following conditions are met: $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}$. So we have that:

$$\Pr[\mathsf{bad}_3 = \mathsf{true}, \text{ in } G_3] = \Pr\left[(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}\right] . \tag{9}$$

Notice that combining equations 7, 8 and 9 gives that:

$$\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathsf{bad}_3 = \mathsf{true}, \text{ in } G_3] > \epsilon(\lambda) , \tag{10}$$

where ϵ is a negligible function. Therefore, we have constructed a reduction $\mathcal{B}$ that wins the witness extractability game of the underlying AS scheme with non-negligible probability, a contradiction to our initial assumption of the claim. This concludes the proof.

□

Claim A.10. *If the relation* R *is* $\mathcal{F}$*-hard, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*, we have that:*

$$\Pr[G_3 = 1] \leq \epsilon(\lambda) .$$

Proof. Assume towards a contradiction that $\Pr[G_3 = 1] > \epsilon(\lambda)$, that is, there exists some p.p.t. adversary $\mathcal{A}$ that wins Game G_3 with non-negligible probability. We want to show that if this is the case, then we can create some reduction $\mathcal{B}'$ that uses $\mathcal{A}$ to break the $\mathcal{F}$-hardness of the relation R. Recall that in order to win this game, given some statement $Y^* \in L_{\mathsf{R}}$, $\mathcal{B}'$ needs to produce a pair (f^*, z) such that $(f^* \in \mathcal{F}) \wedge \big(z \in \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}\big)$. So we get that:

$$\Pr[\mathcal{B}' \text{ wins}] = \Pr\left[(f^* \in \mathcal{F}) \wedge \big(z \in \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}\big)\right] . \tag{11}$$

The reduction $\mathcal{B}'$ in Figure 35 will simultaneously play the role of challenger to adversary $\mathcal{A}$ as in Game G_3, and the role of adversary in the $\mathcal{F}$-hardness game for the relation R. Let $\mathcal{C}$ be the corresponding challenger for the $\mathcal{F}$-hardness game for the relation R. The procedure for the reduction is as follows:

1. $\mathcal{C}$ honestly samples an instance (Y^*, y^*) of the relation R. It then sends Y^* to $\mathcal{B}'$.
2. $\mathcal{B}'$ generates the public parameters pp. It also samples the signing and verification keys $(\mathsf{sk}, \mathsf{vk})$. $\mathcal{B}'$ then sends $(\mathsf{pp}, \mathsf{vk}, Y^*)$ to $\mathcal{A}$.
3. $\mathcal{A}$ performs queries to the signature and functional pre-signature oracles, $\mathcal{O}_s$ and $\mathcal{O}_{fps_2}$. Note that $\mathcal{B}'$ answers these queries. After making queries, $\mathcal{A}$ then sends a response $(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*)$ to $\mathcal{B}'$.
4. $\mathcal{B}'$ then generates an honest pre-signature $\widetilde{\sigma}_2^*$ for $\mathcal{A}$'s chosen message m^* and statement $\mathsf{pk}_{\mathsf{EG}}$. $\mathcal{B}'$ then sends this pre-signature $\widetilde{\sigma}_2^*$ to $\mathcal{A}$.
5. After making more queries, $\mathcal{A}$ then sends a response σ^* to $\mathcal{B}'$.
6. Then, $\mathcal{B}'$ extracts the key $\mathsf{sk}'_{\mathsf{EG}}$ from the signatures, and uses it to decrypt the ciphertext ct to obtain value z. $\mathcal{B}'$ then sends (f^*, z) to $\mathcal{C}$.

Now, we need to show that when $\mathcal{A}$ wins Game G_3, the winning conditions for $\mathcal{B}'$ are also met. When $\mathcal{A}$ wins Game G_3, we know that the following occur:

Reduction $\mathcal{B}'$ for the proof of Claim A.10

$Q = \emptyset, \mathsf{bad}_1 := \mathsf{false}, \mathsf{bad}_2 := \mathsf{false}, \mathsf{bad}_3 := \mathsf{false}$
$\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda)$
$\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
$\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$
$Y^* \leftarrow \mathcal{C}$
$(\mathsf{ad}^*, m^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\mathsf{pp}, \mathsf{vk}, Y^*)$
Parse ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
$\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$
Parse $\widetilde{\sigma}_1^*$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f^*)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $\big(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 0 \vee (f^* \notin \mathcal{F})$
$\quad \vee \mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0\big)$: **return** $\perp$
$\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m^*, \mathsf{pk}_{\mathsf{EG}})$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\widetilde{\sigma}_2^*)$
$\mathsf{sk}'_{\mathsf{EG}} := \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)$
$z := \mathsf{EG.Dec}(\mathsf{sk}'_{\mathsf{EG}}, \mathsf{ct})$
return (f^*, z) to $\mathcal{C}$

Oracle $\mathcal{O}_s(m)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$Q := Q \cup \{m\}$
return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0)$:
$\quad$ **return** $\perp$
$\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$
$Q := Q \cup \{m\}$
return $\widetilde{\sigma}_2$

Figure 35: Reduction for $\mathcal{B}'$ which breaks the $\mathcal{F}$-hardness of the relation R.

(1) $\big(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1\big) \wedge \big(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1\big) \wedge \big(f^* \in \mathcal{F}\big)$

(2) $\big(\mathsf{bad}_1 = \mathsf{false}\big) \implies \mathsf{stmt}_{\mathsf{ad}} \in L_{\mathsf{ad}}$

(3) $\big(\mathsf{bad}_2 = \mathsf{false}\big) \implies \mathsf{stmt}_{\mathsf{eq}} \in L_{\mathsf{eq}}$

(4) $\big(\mathsf{bad}_3 = \mathsf{false}\big) \implies (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}_{\mathsf{EG}}) \in \mathsf{R}_{\mathsf{EG}}$

From (1), we know that one of $\mathcal{B}$'s winning conditions, $f^* \in \mathcal{F}$, is satisfied when $\mathcal{A}$ wins G_3. Therefore, it remains to show that $\mathcal{A}$ winning G_3 implies $\mathcal{B}$'s last winning condition, which is that $\big(z \in \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}\big)$

Notice that when $\mathcal{A}$ wins G_3, we have that conditions (2), (3) are satisfied, so we know that the statements $\mathsf{stmt}_{\mathsf{ad}}$ and $\mathsf{stmt}_{\mathsf{eq}}$ are valid statements for their corresponding languages. This fact is vital for the correctness of the analysis below.

When condition (1) is satisfied, we have that $\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1$, which means that $\exists (y, r_1, r_0)$ such that $(Y^*, y) \in \mathsf{R}$ and $\mathsf{ct}_{\mathsf{ad}} = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1)$. We also know from condition (1) that $\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1$, which means that $\exists (\mathsf{sk}_{\mathsf{FHE}}, z, r_2, r_0)$ such that $\mathsf{ct} = \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, z; r_2)$, and $\mathsf{FHE.Dec}(\mathsf{sk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}}) = z$, where $\widehat{\mathsf{ct}}$ is the functional evaluation of $\mathsf{ct}_{\mathsf{ad}}$ (which we showed above was correctly generated). Since $\widehat{\mathsf{ct}}$ was correctly generated, we know that $z = f^*(y)$. Since condition (4) is satisfied, we have that $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \in \mathsf{R}_{\mathsf{EG}}$, meaning that it is a valid key pair. Therefore, know that $z = \mathsf{EG.Dec}(\mathsf{sk}'_{\mathsf{EG}}, \mathsf{ct})$ is a correct decryption, which outputs $z = f^*(y)$. This gives that $z \in \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}$, which is the second winning condition

for $\mathcal{B}'$. Therefore, we have shown that when $\mathcal{A}$ wins Game G_3, $\mathcal{B}'$ also wins its game. Since we assumed that $\Pr[G_3 = 1] > \epsilon(\lambda)$, we have that:

$$\Pr[\mathcal{B} \text{ wins}] \geq \Pr[G_3 = 1] > \epsilon(\lambda) \ , \tag{12}$$

where ϵ is a negligible function. Therefore, we have constructed a reduction $\mathcal{B}'$ that wins the $\mathcal{F}$-hardness game of the relation R with non-negligible probability, a contradiction to our initial assumption of the claim. This concludes the proof.

□

Claim A.11. *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy adaptive soundness, the underlying* AS *scheme satisfies witness extractability, and the relation* R *is* $\mathcal{F}$*-hard, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\right] \leq \epsilon(\lambda) \ .$$

Proof. Since Game G_0 (as defined in Figure 32) is the original game $\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose, we have that $\Pr[G_0 = 1] = \Pr[\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1]$. By triangle inequality,

$$\begin{aligned}\Pr[G_0 = 1] \leq \ & \big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| \\ & + \big|\Pr[G_1 = 1] - \Pr[G_2 = 1]\big| \\ & + \big|\Pr[G_2 = 1] - \Pr[G_3 = 1]\big| + \Pr[G_3 = 1] \ .\end{aligned}$$

For every stateful p.p.t. adversary $\mathcal{A}$, we have:

- Since the NIZK argument for the language L_{ad} satisfies adaptive soundness, Claim A.7 guarantees that there exists some negligible function ϵ_1 such that for every $\lambda \in \mathbb{N}$, we have that $\big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| \leq \epsilon_1(\lambda)$.
- Since the NIZK argument for the language L_{eq} satisfies adaptive soundness, Claim A.8 guarantees that there exists some negligible function ϵ_2 such that for every $\lambda \in \mathbb{N}$, we have that $\big|\Pr[G_1 = 1] - \Pr[G_2 = 1]\big| \leq \epsilon_2(\lambda)$.
- Since the underlying adaptor signature scheme AS satisfies witness extractability, Claim A.9 guarantees that there exists some negligible function ϵ_3 such that for every $\lambda \in \mathbb{N}$, we have that $\big|\Pr[G_2 = 1] - \Pr[G_3 = 1]\big| \leq \epsilon_3(\lambda)$.
- Since the relation R is $\mathcal{F}$-hard, Claim A.10 guarantees that there exists some negligible function ϵ_4 such that for every $\lambda \in \mathbb{N}$, we have that $\Pr[G_3 = 1] \leq \epsilon_4(\lambda)$.

Combining these results with the expression obtained above for $\Pr[G_0 = 1]$, we have that:

$$\Pr[G_0 = 1] \leq \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda) + \epsilon_4(\lambda) = \epsilon(\lambda) \ . \tag{13}$$

Notice that since $\epsilon_1(\lambda), \epsilon_2(\lambda), \epsilon_3(\lambda)$, and $\epsilon_4(\lambda)$ are all negligible functions, we have that the function $\epsilon(\lambda) = \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda) + \epsilon_4(\lambda)$, for all $\lambda \in \mathbb{N}$, is also a negligible function. Note that since we can find an $\epsilon_1(\lambda), \epsilon_2(\lambda), \epsilon_3(\lambda)$, and $\epsilon_4(\lambda)$ for any stateful p.p.t. adversary $\mathcal{A}$, we can then find a negligible function $\epsilon(\lambda)$ for any stateful p.p.t. adversary $\mathcal{A}$ as well.

Therefore, we have shown that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ, such that for every $\lambda \in \mathbb{N}$, we have that $\Pr[G_0 = 1] \leq \epsilon(\lambda)$. This means that for any p.p.t. adversary $\mathcal{A}$, we have that $\Pr\left[\mathsf{fUnforge}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\right] \leq \epsilon(\lambda)$ as well. This concludes the proof of unforgeability for the proposed FAS construction.

□

A.5 Witness Extractability

Theorem A.12 (Witness Extractability for FAS Construction)**.** *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy adaptive soundness, and the underlying* AS *scheme satisfies witness extractability, then the proposed* FAS *construction in Figure 20 satisfies the property of witness extractability as defined in Definition 4.7.*

Overview. In order to prove that the FAS construction satisfies witness extractability, we will reduce it to the witness extractability of the underlying AS scheme. Consider the sequence of games G_0, G_1, G_2 in Figure 36.

- Game G_0 is the original game $\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose. In this game, the adversary $\mathcal{A}$ is tasked with creating a signature σ^* for a message m^* of their choice, in such a way that the the function evaluation of the witness is unable to be extracted using the FExt algorithm. The adversary has access to the signature and functional pre-signature oracles, which are denoted $\mathcal{O}_s$ and $\mathcal{O}_{fps_2}$, respectively.
- Game G_1 is identical to Game G_0, except that after line 16, the game additionally checks if $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$ is in the language L_{ad}. If it is not in the language, then we set the flag $\mathsf{bad}_1 = \mathsf{true}$. The winning condition of G_1 is also different from G_0, as in order for $\mathcal{A}$ to win in G_1, it must also satisfy $\neg\mathsf{bad}_1$. That is, the flag bad_1 must be false.
- Game G_2 is identical to Game G_1, except that after line 17, the game additionally checks if $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$ is in the language L_{eq}. If it is not in the language, then we set the flag $\mathsf{bad}_2 = \mathsf{true}$. The winning condition of G_2 is also different from G_1, as in order for $\mathcal{A}$ to win in G_2, it must also satisfy $\neg\mathsf{bad}_2$. That is, the flag bad_2 must be false.

First, we will argue in Claims A.13, and A.14 that the game hops from G_0 to G_2 are close. After, we will argue in Claim A.15 that if there exists a stateful p.p.t. adversary $\mathcal{A}$ that wins G_2 with non-negligible probability, then $\mathcal{A}$ could be used to break the witness extractability of the underlying AS scheme. The formal proof for Theorem A.12 can be found in Claim A.16. □

Claim A.13. *If the* NIZK *argument for language* L_{ad} *satisfies adaptive soundness, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*:*

$$\big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| \leq \epsilon(\lambda) \ .$$

Proof. Assume towards a contradiction that $\big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| > \epsilon(\lambda)$. Notice that since Games G_0 and G_1 are identical except for the check in G_1 that determines the flag bad_1, we have:

$$\begin{aligned} \Pr[\mathsf{bad}_1 = \mathsf{true}, \text{ in } G_1] &\geq \big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\big| \\ &> \epsilon(\lambda) \ . \end{aligned} \tag{14}$$

We want to show that if this is the case, then there exists some p.p.t. prover $\mathcal{P}$ that uses $\mathcal{A}$ to break the adaptive soundness of the NIZK argument for the language L_{ad}. Recall that in order to win this game, the prover $\mathcal{P}$ needs to find a pair $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ such that it satisfies two conditions: $(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) \wedge (\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1)$. So we have that $\Pr[\mathcal{P} \text{ wins}]$ is given by:

$$\Pr[\mathcal{P} \text{ wins}] = \Pr\big[(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) \wedge (\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1)\big] \ . \tag{15}$$

The reduction $\mathcal{P}$ in Figure 37 will simultaneously play the role of challenger to adversary $\mathcal{A}$ as in Game G_1, and the role of the prover in the adaptive soundness game for the NIZK argument for the language L_{ad}. Let $\mathcal{C}$ be the corresponding challenger for that adaptive soundness game. The procedure for the reduction is as follows:

Games G_0, $\boxed{G_1}$, $\boxed{G_2}$

1 : $Q = \emptyset, \mathsf{bad}_1 := \mathsf{false}, \mathsf{bad}_2 := \mathsf{false}$
2 : $\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda)$
3 : $\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
4 : $\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$
5 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$
6 : $(\mathsf{ad}^*, m^*, Y^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\mathsf{pp}, \mathsf{vk})$
7 : Parse ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
8 : $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$
9 : Parse $\widetilde{\sigma}_1^*$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
10 : $\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f^*)$
11 : $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
12 : if $\big(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 0 \vee (f^* \notin \mathcal{F})$
$\quad \vee \mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0\big)$: **return** $\perp$
13 : $\widetilde{\sigma}_2^* \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m^*, \mathsf{pk}_{\mathsf{EG}})$
14 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\widetilde{\sigma}_2^*)$
15 : $\mathsf{sk}'_{\mathsf{EG}} := \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)$
16 : $z := \mathsf{EG.Dec}(\mathsf{sk}'_{\mathsf{EG}}, \mathsf{ct})$
17 : $\boxed{\text{if } (\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) : \mathsf{bad}_1 := \mathsf{true}}$
18 : $\boxed{\text{if } (\mathsf{stmt}_{\mathsf{eq}} \notin L_{\mathsf{eq}}) : \mathsf{bad}_2 := \mathsf{true}}$
19 : return $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$
$\quad \wedge \big(z \notin \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}\big)$ $\boxed{\wedge \neg \mathsf{bad}_1} \wedge \neg \mathsf{bad}_2$

Oracle $\mathcal{O}_s(m)$

1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $Q := Q \cup \{m\}$
3 : return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

1 : Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
2 : Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
3 : $\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
4 : $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
5 : if $(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0)$:
$\quad$ **return** $\perp$
6 : $\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$
7 : $Q := Q \cup \{m\}$
8 : return $\widetilde{\sigma}_2$

Figure 36: Witness Extractability Proof: Games G_0, G_1, G_2

1. $\mathcal{C}$ honestly samples a $\mathsf{crs}_{\mathsf{ad}}$, and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ generates the public parameters pp (which includes the $\mathsf{crs}_{\mathsf{ad}}$ from $\mathcal{C}$), and also samples $(\mathsf{sk}, \mathsf{vk})$ for the AS scheme. $\mathcal{P}$ then sends $(\mathsf{pp}, \mathsf{vk})$ to $\mathcal{A}$.
3. $\mathcal{A}$ performs queries to the signature and functional pre-signature oracles, $\mathcal{O}_s$ and $\mathcal{O}_{fps_2}$. Note that $\mathcal{P}$ answers these queries.
4. After making queries, $\mathcal{A}$ then sends a response $(\mathsf{ad}^*, m^*, Y^*, f^*, \widetilde{\sigma}_1^*)$ to $\mathcal{P}$.
5. $\mathcal{P}$ then parses ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$, and constructs $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$. Then, $\mathcal{P}$ sends the response $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ to $\mathcal{C}$.

Notice that during an instance of Game G_1, when $(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}})$, we set the flag $\mathsf{bad}_1 = \mathsf{true}$ in line 17. But it must also have been the case that $\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1$, as otherwise, G_1 would have aborted in line 12 (and therefore it would not have reached line 17 to set the flag bad_1 to true). Therefore, we have that:

$$\Pr[\mathsf{bad}_1 = \mathsf{true}, \text{ in } G_1] = \Pr\big[(\mathsf{stmt}_{\mathsf{ad}} \notin L_{\mathsf{ad}}) \wedge (\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1)\big] \ . \tag{16}$$

Reduction $\mathcal{P}$ for the proof of Claim A.13

$Q = \emptyset, \mathsf{bad}_1 := \mathsf{false}$
$\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathcal{C}$
$\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
$\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$
$(\mathsf{ad}^*, m^*, Y^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\mathsf{pp}, \mathsf{vk})$
Parse ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
$\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$
if $(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 0)$: **return** $\perp$
return $(\mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$ to $\mathcal{C}$

Oracle $\mathcal{O}_s(m)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$Q := Q \cup \{m\}$
return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0)$: **return** $\perp$
$\widetilde{\sigma}_2 \leftarrow \mathsf{AS.pSign}(\mathsf{sk}, m, \mathsf{pk}_{\mathsf{EG}})$
$Q := Q \cup \{m\}$
return $\widetilde{\sigma}_2$

Figure 37: Reduction for prover $\mathcal{P}$ playing the adaptive soundness game with challenger $\mathcal{C}$

Notice that combining equations 14, 15 and 16 gives that:

$$\Pr[\mathcal{P} \text{ wins}] = \Pr[\mathsf{bad}_1 = \mathsf{true}, \text{ in } G_1] > \epsilon(\lambda) \ , \tag{17}$$

where ϵ is a negligible function. Therefore, we have constructed a prover $\mathcal{P}$ that wins the adaptive soundness game of the NIZK argument for the language L_{ad} with non-negligible probability, a contradiction to our initial assumption of the claim. This concludes the proof. □

Claim A.14. *If the* NIZK *argument for language L_{eq} satisfies adaptive soundness, then for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ, such that for every $\lambda \in \mathbb{N}$:*

$$\big|\Pr[G_1 = 1] - \Pr[G_2 = 1]\big| \leq \epsilon(\lambda) \ .$$

Proof. Assume towards a contradiction that $\big|\Pr[G_1 = 1] - \Pr[G_2 = 1]\big| > \epsilon(\lambda)$. Notice that since Games G_1 and G_2 are identical except for the check that determines the flag bad_2, we have:

$$\begin{aligned} \Pr[\mathsf{bad}_2 = \mathsf{true}, \text{ in } G_2] &\geq \big|\Pr[G_1 = 1] - \Pr[G_2 = 1]\big| \\ &> \epsilon(\lambda) \ . \end{aligned} \tag{18}$$

We want to show that if this is the case, then there exists some p.p.t. prover $\mathcal{P}^*$ that uses $\mathcal{A}$ to break the adaptive soundness of the NIZK argument for the language L_{eq}. Recall that in order to win this game, the prover $\mathcal{P}^*$ needs to find a pair $(\mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}})$ such that it satisfies two conditions: $(\mathsf{stmt}_{\mathsf{eq}} \notin L_{\mathsf{eq}}) \wedge (\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1)$. So we have that $\Pr[\mathcal{P}^* \text{ wins}]$ is given by:

$$\Pr[\mathcal{P}^* \text{ wins}] = \Pr\big[(\mathsf{stmt}_{\mathsf{eq}} \notin L_{\mathsf{eq}}) \wedge (\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1)\big] \ . \tag{19}$$

Notice that the winning condition of $\mathcal{P}^*$ is similar to that of the winning condition for $\mathcal{P}$ in Claim A.13. Further, the hop from G_1 to G_2 is similar to that of the hop from G_0 to G_1. Therefore, the rest of this proof follows from the proof of Claim A.13. This concludes the proof. □

Reduction $\mathcal{B}$ for the proof of Claim A.15

$Q = \emptyset, \mathsf{bad}_1 := \mathsf{false}, \mathsf{bad}_2 := \mathsf{false}$
$\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda)$
$\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$
$\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$
$\mathsf{vk} \leftarrow \mathcal{C}$
$(\mathsf{ad}^*, m^*, Y^*, f^*, \widetilde{\sigma}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\mathsf{pp}, \mathsf{vk})$
Parse ad^* as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
$\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y^*)$
Parse $\widetilde{\sigma}_1^*$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f^*)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $\big(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 0 \vee (f^* \notin \mathcal{F})$
$\quad \vee \mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0\big)$: **return** $\perp$
$\widetilde{\sigma}_2^* \leftarrow \mathcal{C}(m^*, \mathsf{pk}_{\mathsf{EG}})$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_s, \mathcal{O}_{fps_2}}(\widetilde{\sigma}_2^*)$
return $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ to $\mathcal{C}$

Oracle $\mathcal{O}_s(m)$

$\sigma \leftarrow \mathcal{C}(m)$
$Q := Q \cup \{m\}$
return σ

Oracle $\mathcal{O}_{fps_2}(\mathsf{ad}, m, Y, f, \widetilde{\sigma}_1)$

Parse ad as $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$
Parse $\widetilde{\sigma}_1$ as $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$
$\widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, f)$
$\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$
if $(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 0)$:
$\quad$ **return** $\perp$
$\widetilde{\sigma}_2 \leftarrow \mathcal{C}(m, \mathsf{pk}_{\mathsf{EG}})$
$Q := Q \cup \{m\}$
return $\widetilde{\sigma}_2$

Figure 38: Reduction for $\mathcal{B}$ playing the witness extractability game for AS with challenger $\mathcal{C}$

Claim A.15. *If the underlying adaptor signature scheme* AS *satisfies witness extractability, then for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ, such that for every $\lambda \in \mathbb{N}$:*

$$\Pr[G_2 = 1] \leq \epsilon(\lambda) \ .$$

Proof. Assume towards a contradiction that $\Pr[G_2 = 1] > \epsilon(\lambda)$ that is, there exists some p.p.t. adversary $\mathcal{A}$ that wins Game G_2 with non-negligible probability. We want to show that if this is the case, then we can construct a reduction $\mathcal{B}$ that uses the adversary $\mathcal{A}$ to break the witness extractability of the underlying AS scheme. Recall that in order for $\mathcal{B}$ to win its game, it needs to find $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ such that the following conditions are satisfied: $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$ $\wedge(\mathsf{pk}_{\mathsf{EG}}, \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)) \notin \mathsf{R}_{\mathsf{EG}}$. Let $\mathsf{sk}'_{\mathsf{EG}} = \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)$, so we get that:

$$\Pr[\mathcal{B} \text{ wins}] = \Pr\big[(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}\big] \ . \tag{20}$$

The reduction $\mathcal{B}$ in Figure 38 will simultaneously play the role of challenger to adversary $\mathcal{A}$ as in Game G_2, and the role of the adversary in the witness extractability game for the underlying AS scheme. Let $\mathcal{C}$ be the corresponding challenger for the witness extractability game for the underlying AS scheme. The procedure for the reduction is as follows:

1. $\mathcal{C}$ honestly samples the signing and verification keys $(\mathsf{sk}, \mathsf{vk})$, and sends vk to $\mathcal{B}$.
2. $\mathcal{B}$ generates the public parameters pp, and then sends $(\mathsf{pp}, \mathsf{vk})$ to $\mathcal{A}$.
3. $\mathcal{A}$ performs queries to the signature and functional pre-signature oracles, $\mathcal{O}_s$ and $\mathcal{O}_{fps_2}$. Note that $\mathcal{B}$ answers these queries by forwarding the queries to its challenger, $\mathcal{C}$. After making queries, $\mathcal{A}$ then sends a response $(\mathsf{ad}^*, m^*, Y^*, f^*, \widetilde{\sigma}_1^*)$ to $\mathcal{B}$.

4. $\mathcal{B}$ then uses $\mathcal{C}$ to generate a honest pre-signature $\widetilde{\sigma}_2^*$ for $\mathcal{A}$'s chosen message m^* and statement $\mathsf{pk}_{\mathsf{EG}}$. $\mathcal{B}$ then forwards this pre-signature $\widetilde{\sigma}_2^*$ to $\mathcal{A}$.

5. After making more queries, $\mathcal{A}$ then sends a response σ^* to $\mathcal{B}$.

6. Then, $\mathcal{B}$ sends $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ to $\mathcal{C}$.

Now, we need to show that when $\mathcal{A}$ wins Game G_2, the winning conditions for $\mathcal{B}$ are also met. When $\mathcal{A}$ wins Game G_2, we know that the following occur:

(1) $\big(\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1\big) \wedge \big(\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1\big) \wedge \big(f^* \in \mathcal{F}\big)$

(2) $\big(\mathsf{bad}_1 = \mathsf{false}\big) \wedge \big(\mathsf{bad}_2 = \mathsf{false}\big)$

(3) $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge\ z \notin \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}$

From (3), we know that two of $\mathcal{B}$'s winning conditions, $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$, are satisfied when $\mathcal{A}$ wins G_2. Therefore, it remains to show that $\mathcal{A}$ winning G_2 implies $\mathcal{B}$'s last winning condition: $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{AS.Ext}(\mathsf{vk}, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2^*, \sigma^*)) \notin \mathsf{R}_{\mathsf{EG}}$. In this case, that is, $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}$.

Suppose towards a contradiction that $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \in \mathsf{R}_{\mathsf{EG}}$. Notice that when $\mathcal{A}$ wins G_2, we have that condition (2) is satisfied, so we know that the statements $\mathsf{stmt}_{\mathsf{ad}}$ and $\mathsf{stmt}_{\mathsf{eq}}$ are valid statements for their corresponding languages. This fact is vital for the correctness of the analysis below.

When condition (1) is satisfied, we have that $\mathsf{Verify}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1$, which means that $\exists(y, r_1, r_0)$ such that $(Y^*, y) \in \mathsf{R}$ and $\mathsf{ct}_{\mathsf{ad}} = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1)$. We also know from condition (1) that $\mathsf{Verify}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \pi^*_{\mathsf{eq}}) = 1$, which means that $\exists(\mathsf{sk}_{\mathsf{FHE}}, z, r_2, r_0)$ such that $\mathsf{ct} = \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, z; r_2)$, and $\mathsf{FHE.Dec}(\mathsf{sk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}}) = z$, where $\widehat{\mathsf{ct}}$ is the functional evaluation of $\mathsf{ct}_{\mathsf{ad}}$ (which we showed above was correctly generated). Since $\widehat{\mathsf{ct}}$ was correctly generated, we know that $z = f^*(y)$, where $f^* \in \mathcal{F}$. Since $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \in \mathsf{R}_{\mathsf{EG}}$, we know that $z = \mathsf{EG.Dec}(\mathsf{sk}'_{\mathsf{EG}}, \mathsf{ct})$ is a correct decryption, which outputs $z = f^*(y)$. This gives that $z \in \{f^*(y) \mid \exists y \text{ s.t } (Y^*, y) \in \mathsf{R}\}$, which contradicts one of the win conditions of G_2 (see condition (3)). Therefore, when $\mathcal{A}$ wins G_2, it must be the case that $(\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}$.

This means that when $\mathcal{A}$ wins G_2, it generates $(m^*, \mathsf{pk}_{\mathsf{EG}}, \sigma^*)$ that satisfy the three conditions $(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}$. We then obtain the following result.

$$\Pr\big[(m^* \notin Q) \wedge \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}'_{\mathsf{EG}}) \notin \mathsf{R}_{\mathsf{EG}}\big] \geq \Pr[G_2 = 1] \ . \tag{21}$$

Notice that combining equations 20 and 21, along with the assumption that $\Pr[G_2 = 1] > \epsilon(\lambda)$:

$$\Pr[\mathcal{B} \text{ wins}] \geq \Pr[G_2 = 1] > \epsilon(\lambda) \ , \tag{22}$$

where ϵ is a negligible function. Therefore, we have constructed an adversary $\mathcal{B}$ that wins the witness extractability game of the underlying AS scheme with non-negligible probability, a contradiction to our initial assumption of the claim. This concludes the proof. □

Claim A.16. *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy adaptive soundness, and the underlying* AS *scheme satisfies witness extractability, then for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for every* $\lambda \in \mathbb{N}$*:*

$$\Pr\Big[\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\Big] \leq \epsilon(\lambda) \ .$$

Proof. Since Game G_0 (as defined in Figure 36) is the original game $\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose, we have that $\Pr[G_0 = 1] = \Pr\left[\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\right]$. Notice that the triangle inequality gives that:

$$\begin{aligned}\Pr[G_0 = 1] &\leq \left|\Pr[G_0 = 1] - \Pr[G_1 = 1]\right| + \Pr[G_1 = 1] \\ &\leq \left|\Pr[G_0 = 1] - \Pr[G_1 = 1]\right| + \left|\Pr[G_1 = 1] - \Pr[G_2 = 1]\right| + \Pr[G_2 = 1] \ .\end{aligned}$$

For every stateful p.p.t. adversary $\mathcal{A}$, we have the following:

- Since the NIZK argument for the language L_{ad} satisfies adaptive soundness, Claim A.13 guarantees that there exists some negligible function ϵ_1 such that for every $\lambda \in \mathbb{N}$, we have that $\left|\Pr[G_0 = 1] - \Pr[G_1 = 1]\right| \leq \epsilon_1(\lambda)$.
- Since the NIZK argument for the language L_{eq} satisfies adaptive soundness, Claim A.14 guarantees that there exists some negligible function ϵ_2 such that for every $\lambda \in \mathbb{N}$, we have that $\left|\Pr[G_1 = 1] - \Pr[G_2 = 1]\right| \leq \epsilon_2(\lambda)$.
- Since the underlying adaptor signature scheme AS satisfies witness extractability, Claim A.15 guarantees that there exists some negligible function ϵ_3 such that for every $\lambda \in \mathbb{N}$, we have that $\Pr[G_2 = 1] \leq \epsilon_3(\lambda)$.

Combining these results with the expression obtained above for $\Pr[G_0 = 1]$, we have that:

$$\Pr[G_0 = 1] \leq \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda) = \epsilon(\lambda) \ . \tag{23}$$

Notice that since $\epsilon_1(\lambda), \epsilon_2(\lambda)$, and $\epsilon_3(\lambda)$ are all negligible functions, we have that the function $\epsilon(\lambda) = \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda)$, for all $\lambda \in \mathbb{N}$, is also a negligible function. Note that since we can find an $\epsilon_1(\lambda), \epsilon_2(\lambda)$, and $\epsilon_3(\lambda)$ for any stateful p.p.t. adversary $\mathcal{A}$, we can then find a negligible function $\epsilon(\lambda)$ for any stateful p.p.t. adversary $\mathcal{A}$ as well.

Therefore, we have shown that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ, such that for every $\lambda \in \mathbb{N}$, we have that $\Pr[G_0 = 1] \leq \epsilon(\lambda)$. This means that for any stateful p.p.t. adversary $\mathcal{A}$, we have that $\Pr\left[\mathsf{fWitExt}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1\right] \leq \epsilon(\lambda)$ as well. This concludes the proof of witness extractability for the proposed FAS construction.

□

A.6 Zero-Knowledge (non-abort)

Theorem A.17 (Zero Knowledge (non-abort) for FAS Construction)**.** *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy zero knowledge, and the* FHE *encryption scheme satisfies semantic security, then the proposed* FAS *construction in Figure 20 satisfies zero knowledge (non-abort) as defined in Definition 4.9.*

Overview. In order to prove the zero knowledge (non-abort) of FAS, we need to show that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a stateful p.p.t. simulator Sim and a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and all $(Y, y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right]\right| \leq \epsilon(\lambda) \ .$$

First, we need to construct the stateful p.p.t. simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{FPreSign}_1^*, \mathsf{Adapt}^*)$. Let $\mathsf{Sim}_{\mathsf{ad}} = (\mathsf{Setup}^*_{\mathsf{ad}}, \mathsf{Prove}^*_{\mathsf{ad}})$ be the simulator (as defined in Definition 3.33) for the NIZK argument

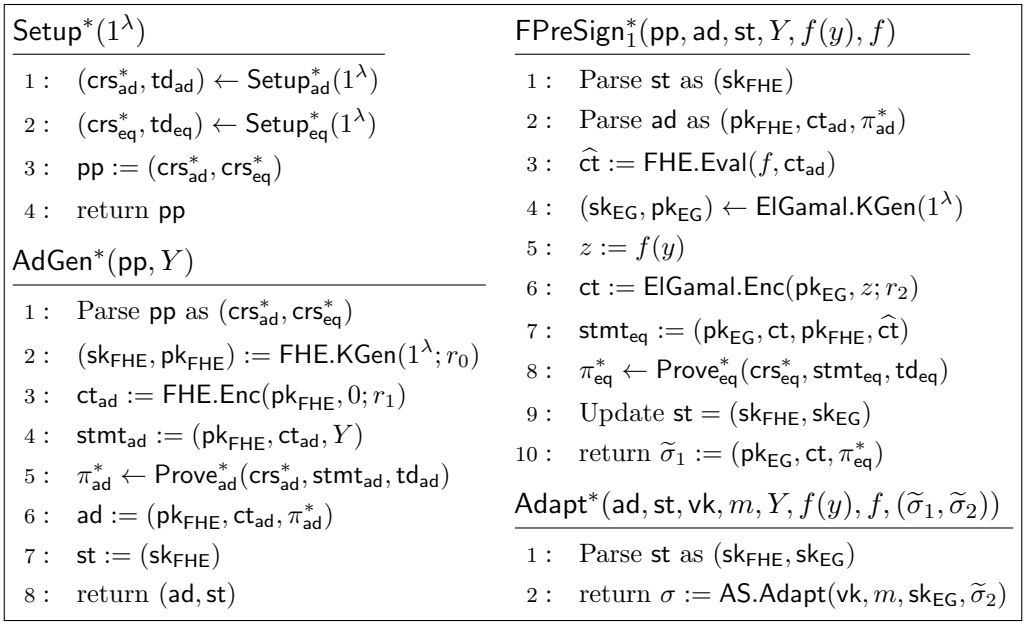

Figure 39: Simulator Sim for zero knowledge (non-abort) of FAS proof

for the language L_{ad}. Similarly, let $\mathsf{Sim}_{\mathsf{eq}} = (\mathsf{Setup}^*_{\mathsf{eq}}, \mathsf{Prove}^*_{\mathsf{eq}})$ be the simulator for the NIZK argument for the language L_{eq}. We then define our simulator Sim as in Figure 39.

Next, consider the sequence of games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ as described in Figure 40.

- The first game is $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose. In this game, the adversary $\mathcal{A}$ is interacting with a challenger who has access to the witness y. The adversary is able to run as many interactions as it wants, and we then we examine the view of $\mathcal{A}$.
- The game $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ is identical to the first game, except that it uses the simulator $\mathsf{Sim}_{\mathsf{ad}}$'s methods $(\mathsf{Setup}^*_{\mathsf{ad}}, \mathsf{Prove}^*_{\mathsf{ad}})$ to generate the proof of correctness for $\mathsf{ct}_{\mathsf{ad}}$. This involves generating the pair $(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}})$ during setup, and using the trapdoor $\mathsf{td}_{\mathsf{ad}}$ instead of $\mathsf{wit}_{\mathsf{ad}}$ during the proof. Since we no longer need $\mathsf{wit}_{\mathsf{ad}}$, we set it to $\perp$, to indicate we no longer need to use y for the proof.
- The game $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ is identical to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$, except that it uses the simulator $\mathsf{Sim}_{\mathsf{eq}}$'s methods $(\mathsf{Setup}^*_{\mathsf{eq}}, \mathsf{Prove}^*_{\mathsf{eq}})$ to generate the proof of equality for ct and $\widehat{\mathsf{ct}}$. This involves generating the pair $(\mathsf{crs}^*_{\mathsf{eq}}, \mathsf{td}_{\mathsf{eq}})$ during setup, and using the trapdoor $\mathsf{td}_{\mathsf{eq}}$ instead of $\mathsf{wit}_{\mathsf{eq}}$ during the proof. Since we no longer need $\mathsf{wit}_{\mathsf{eq}}$, we set it to $\perp$ to indicate this.
- The last game is $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose. Note that this game is identical to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ except that $\mathsf{ct}_{\mathsf{ad}}$ now encrypts 0, rather than y. In this game, the adversary $\mathcal{A}$ is interacting with a challenger who has access to only the function $f(y)$ of the witness y. The adversary is able to run as many interactions as it wants, and we then we examine the view of $\mathcal{A}$.

First, we will argue in Claim A.19 that the hop from $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$ to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ is close due to the zero knowledge property of the NIZK argument for the language L_{ad}. Similarly, we argue in Claim

Games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$, $\boxed{\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)}$, $\boxed{\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)}$, $\boxed{\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)}$

1 : $\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda)$ $\boxed{(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}}) \leftarrow \mathsf{Setup}^*_{\mathsf{ad}}(1^\lambda)}$

2 : $\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$ $\boxed{(\mathsf{crs}^*_{\mathsf{eq}}, \mathsf{td}_{\mathsf{eq}}) \leftarrow \mathsf{Setup}^*_{\mathsf{eq}}(1^\lambda)}$

3 : $\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$ $\boxed{\mathsf{pp} := (\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})}$ $\boxed{\mathsf{pp} := (\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{crs}^*_{\mathsf{eq}})}$

4 : $(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{FHE}}) := \mathsf{FHE.KGen}(1^\lambda; r_0)$

5 : $\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1)$ $\boxed{\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, 0; r_1)}$

6 : $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$

7 : $\mathsf{wit}_{\mathsf{ad}} := (y, r_1, r_0)$ $\boxed{\mathsf{wit}_{\mathsf{ad}} := \bot}$

8 : $\pi_{\mathsf{ad}} \leftarrow \mathsf{Prove}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \mathsf{wit}_{\mathsf{ad}})$ $\boxed{\pi_{\mathsf{ad}} \leftarrow \mathsf{Prove}^*_{\mathsf{ad}}(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}})}$

9 : $\mathsf{ad} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$

10 : $\mathsf{st} := (\mathsf{sk}_{\mathsf{FHE}})$

11 : $(m, f, \mathsf{vk}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ad}, Y)$

12 : while $\mathcal{A}$ is not done making queries:

13 : $\quad \widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(f, \mathsf{ct}_{\mathsf{ad}})$

14 : $\quad (\mathsf{sk}_{\mathsf{EG}}, \mathsf{pk}_{\mathsf{EG}}) \leftarrow \mathsf{ElGamal.KGen}(1^\lambda)$

15 : $\quad z := f(y)$

16 : $\quad \mathsf{ct} := \mathsf{ElGamal.Enc}(\mathsf{pk}_{\mathsf{EG}}, z; r_2)$

17 : $\quad \mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$

18 : $\quad \mathsf{wit}_{\mathsf{eq}} := (\mathsf{sk}_{\mathsf{FHE}}, z, r_2, r_0)$ $\boxed{\mathsf{wit}_{\mathsf{eq}} := \bot}$

19 : $\quad \pi_{\mathsf{eq}} \leftarrow \mathsf{Prove}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \mathsf{wit}_{\mathsf{eq}})$ $\boxed{\pi_{\mathsf{eq}} \leftarrow \mathsf{Prove}^*_{\mathsf{eq}}(\mathsf{crs}^*_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \mathsf{td}_{\mathsf{eq}})}$

20 : $\quad$ Update $\mathsf{st} := (\mathsf{sk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{EG}})$

21 : $\quad \widetilde{\sigma}_1 := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$

22 : $\quad \widetilde{\sigma}_2 \leftarrow \mathcal{A}(\widetilde{\sigma}_1)$

23 : $\quad$ if $(\mathsf{AS.pVrfy}(\mathsf{vk}, m, \mathsf{pk}_{\mathsf{EG}}, \widetilde{\sigma}_2) = 0)$:

$\quad\quad$ **return** $\bot$

24 : $\quad \sigma := \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{sk}_{\mathsf{EG}}, \widetilde{\sigma}_2)$

25 : $\quad (m, f) \leftarrow \mathcal{A}(\sigma)$

26 : return view of $\mathcal{A}$

Figure 40: Zero Knowledge (non-abort) Proof: Games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$, and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$.

A.20 that the hop from $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ is close due to the zero knowledge property of the NIZK argument for the language L_{eq}. Finally, we prove in Claim A.21 that the hop from $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ to $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ is close due to the semantic security of the FHE encryption scheme. The formal proof for Theorem A.17 can be found in Claim A.22.

□

Remark A.18. *To simply notation throughout the next few proofs, let E_{ad} be the event that:*

$$\begin{aligned}\mathsf{crs}_{\mathsf{eq}} &\leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda) \ , \\ \pi_{\mathsf{eq}} &\leftarrow \mathsf{Prove}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \mathsf{wit}_{\mathsf{eq}}) \ ,\end{aligned}$$

and let E^*_{ad} be the event that:

$$\begin{aligned}(\mathsf{crs}^*{}_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}}) &\leftarrow \mathsf{Setup}^*_{\mathsf{ad}}(1^\lambda) \ , \\ \pi^*_{\mathsf{ad}} &\leftarrow \mathsf{Prove}^*_{\mathsf{ad}}(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}}) \ .\end{aligned}$$

Claim A.19. *If the* NIZK *argument for the language L_{ad} satisfies zero knowledge, then for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, for all $(Y, y) \in \mathsf{R}$:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) \ .$$

Proof. Assume towards a contradiction that:

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| > \epsilon(\lambda) \ . \tag{24}$$

We want to show that if this is the case, then there exists a p.p.t. distinguisher $\mathcal{D}^*$ that uses $\mathcal{D}$ to break the zero knowledge of the NIZK argument for the language L_{ad}.

Recall that since the NIZK argument for the language L_{ad} satisfies zero knowledge, for any p.p.t. distinguisher $\mathcal{D}^*$, we have that for all statement/witness pairs $(\mathsf{stmt}_{\mathsf{ad}}, \mathsf{wit}_{\mathsf{ad}})$:

$$\begin{aligned}P_1 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1 : E_{\mathsf{ad}}\right], \\ P_2 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}^*_{\mathsf{ad}}, \pi^*_{\mathsf{ad}}) = 1 : E^*_{\mathsf{ad}}\right], \\ |P_1 - P_2| &\leq \epsilon(\lambda) \ .\end{aligned} \tag{25}$$

Suppose that for some statement $\mathsf{stmt}_{\mathsf{ad}} = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$, the inputs to $\mathcal{D}^*$ are some common reference string $\mathsf{crs}_{\mathsf{ad}}$ and a proof π_{ad}. The distinguisher $\mathcal{D}^*$ can use this information to generate the public parameters $\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$, as well as the advertisement $\mathsf{ad} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$. The distinguisher $\mathcal{D}^*$ then passes $(\mathsf{pp}, \mathsf{ad}, Y)$ to the adversary $\mathcal{A}$. Note that since the two games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ are the same during the while loop, the only differences in $\mathcal{A}$'s view are from the inputs received before the while loop. Therefore, it is sufficient to have $\mathcal{D}^*$ only simulate the inputs before the while loop.

After $\mathcal{D}^*$ has sent $(\mathsf{pp}, \mathsf{ad}, Y)$ to the adversary $\mathcal{A}$, it then runs the distinguisher $\mathcal{D}$ on $\mathcal{A}$. Then, $\mathcal{D}^*$ returns whatever $\mathcal{D}$ returns. We then have that:

$$\begin{aligned}P_1 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}_{\mathsf{ad}}, \pi_{\mathsf{ad}}) = 1 : E_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}(\mathsf{pp}, \mathsf{ad}, Y) = 1 : E_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \ .\end{aligned}$$

Similarly, we also get that:

$$\begin{aligned}P_2 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}^*_{\mathsf{ad}}, \pi^*_{\mathsf{ad}}) = 1 : E^*_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}(\mathsf{pp}, \mathsf{ad}, Y) = 1 : E^*_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \ .\end{aligned}$$

We then have that:

$$|P_1 - P_2| = \left| \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| . \tag{26}$$

Combining equations 24 and 26, we obtain that $|P_1 - P_2| > \epsilon(\lambda)$, a contradiction to equation 25. This concludes the proof.

□

Claim A.20. *If the* NIZK *argument for the language* L_{eq} *satisfies zero knowledge, then for every stateful adversary* $\mathcal{A}$*, there exists a negligible function* ϵ*, such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, for all* $(Y, y) \in \mathsf{R}$*:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) .$$

Proof. Assume towards a contradiction that:

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| > \epsilon(\lambda) . \tag{27}$$

We want to show that if this is the case, then there exists a p.p.t. distinguisher $\mathcal{D}^*$ that uses $\mathcal{D}$ to break the zero knowledge of the NIZK argument for the language L_{eq}.

Recall that since the NIZK argument for the language L_{eq} satisfies zero knowledge, for any p.p.t. distinguisher $\mathcal{D}^*$, we have that for all statement/witness pairs $(\mathsf{stmt}_{\mathsf{eq}}, \mathsf{wit}_{\mathsf{eq}})$:

$$\begin{aligned} P_1 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1 : E_{\mathsf{ad}}\right], \\ P_2 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}^*_{\mathsf{eq}}, \pi^*_{\mathsf{eq}}) = 1 : E^*_{\mathsf{ad}}\right], \\ |P_1 - P_2| &\leq \epsilon(\lambda) . \end{aligned} \tag{28}$$

Suppose that for some statement $\mathsf{stmt}_{\mathsf{eq}} = (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$, the inputs to $\mathcal{D}^*$ are some common reference string $\mathsf{crs}_{\mathsf{eq}}$ and a proof π_{eq}. The distinguisher $\mathcal{D}^*$ can use this information to generate the information $\widetilde{\sigma}_1 = (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$. The distinguisher can then pass the information $\widetilde{\sigma}_1 = (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$ to the adversary $\mathcal{A}$. Note that since the two games $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ are only different during the while loop, the only differences in $\mathcal{A}$'s view are from the inputs received during the while loop. The rest of the simulation proceeds as normal in each of the games.

After $\mathcal{D}^*$ has sent $\widetilde{\sigma}_1 = (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$ to the adversary $\mathcal{A}$, it then runs the distinguisher $\mathcal{D}$. Then, $\mathcal{D}^*$ returns whatever $\mathcal{D}$ returns. We then have that:

$$\begin{aligned} P_1 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}_{\mathsf{eq}}, \pi_{\mathsf{eq}}) = 1 : E_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}}) = 1 : E_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] . \end{aligned}$$

Similarly, we also get that:

$$\begin{aligned} P_2 &= \Pr\left[\mathcal{D}^*(\mathsf{crs}^*_{\mathsf{eq}}, \pi^*_{\mathsf{eq}}) = 1 : E^*_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}(\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}}) = 1 : E^*_{\mathsf{ad}}\right] \\ &= \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] . \end{aligned}$$

We then have that:

$$|P_1 - P_2| = \left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| . \quad (29)$$

Combining equations 27 and 29, we obtain that $|P_1 - P_2| > \epsilon(\lambda)$, a contradiction to equation 28. This concludes the proof.

□

Claim A.21. *If the* FHE *scheme satisfies semantic security, then for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) .$$

Proof. Assume towards a contradiction that for any stateful p.p.t. adversary $\mathcal{A}$, there exists a p.p.t. distinguisher $\mathcal{D}$ such that for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$:

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| > \epsilon(\lambda) . \quad (30)$$

Notice that the games $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ are exactly the same, except for the way in which $\mathsf{ct}_{\mathsf{ad}}$ is generated. In $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{ct}_{\mathsf{ad}}$ is a FHE encryption of the witness y, while in $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{ct}_{\mathsf{ad}}$ is a FHE encryption of 0. We want to show that if the p.p.t. distinguisher $\mathcal{D}$ exists (as we assumed above), then we can break the semantic security of the FHE scheme.

We will show that if such a distinguisher $\mathcal{D}$ exists, then we can create a reduction $\mathcal{B}$ that breaks the semantic security of the FHE encryption scheme. Let $\mathcal{C}$ be the challenger for the semantic security game for the FHE encryption scheme. $\mathcal{B}$ plays the role of the adversary in this semantic security game. $\mathcal{B}$ also serves as the challenger to adversary $\mathcal{A}$ playing the games $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$. The procedure for the reduction is as follows:

1. $\mathcal{C}$ honestly samples the FHE public parameters pp', and sends pp' to $\mathcal{B}$.
2. $\mathcal{B}$ completes the rest of the setup necessary to generate the public parameters pp for the game $\mathcal{A}$ is playing. In this case, this involves performing the setup for the NIZK arguments.
3. $\mathcal{B}$ then sends the two values $(0, y)$ to $\mathcal{C}$. $\mathcal{C}$ then responds with the FHE encryption ($\mathsf{ct}_{\mathsf{ad}}$) of one of them, along with the $\mathsf{pk}_{\mathsf{FHE}}$ used to generate the encryption. Note that if $\mathcal{C}$ chooses to encrypt 0, then $\mathcal{A}$'s view will be as in the game $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$. Similarly, if $\mathcal{C}$ chooses to encrypt y, then $\mathcal{A}$'s view will be as in the game $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$.
4. $\mathcal{B}$ then generates a proof π_{ad} for $\mathsf{ct}_{\mathsf{ad}}$, and creates the advertisement $\mathsf{ad} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$. Afterwards, $\mathcal{B}$ sends the information $(\mathsf{pp}, \mathsf{ad}, Y)$ to the adversary $\mathcal{A}$.
5. The adversary $\mathcal{A}$ is then able to query $\mathcal{B}$ as much as it wants.
6. After $\mathcal{A}$ is done making queries, $\mathcal{B}$ uses the distinguisher $\mathcal{D}$ on $\mathcal{A}$. Then, $\mathcal{B}$ simply forwards the output of $\mathcal{D}$ to the challenger $\mathcal{C}$.

Since the only difference between the two games $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ is the way $\mathsf{ct}_{\mathsf{ad}}$ is encrypted, if $\mathcal{D}$ can distinguish between the games with some probability p, then that means that $\mathcal{D}$ can tell which value $\mathsf{ct}_{\mathsf{ad}}$ encrypts with the same probability p. Since $\mathcal{B}$ outputs the result that $\mathcal{D}$ outputs, we can then express the probability that $\mathcal{B}$ wins the semantic security game as follows:

$$\Pr[\mathcal{B} \text{ wins}] \geq \left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| . \quad (31)$$

Combining equations 30 and 31, we have that $\Pr[\mathcal{B} \text{ wins}] > \epsilon(\lambda)$, where ϵ is a negligible function. Therefore, we have constructed an adversary $\mathcal{B}$ that breaks the semantic security of the FHE encryption scheme, which is a contradiction to the assumptions made in the claim. This concludes the proof.

□

Claim A.22. *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy zero knowledge, and the* FHE *encryption scheme satisfies semantic security, then for all stateful p.p.t. adversaries* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, and for all* $(Y, y) \in \mathsf{R}$ *, we have that:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) ,$$

where Sim *is the simulator defined in Figure 39, and the two games* $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ *and* $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ *are defined as in Figure 40.*

Proof. Using the triangle inequality multiple times gives that:

$$\begin{aligned}
\Pr[\mathcal{D}(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y))=1] \leq & \left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right] \\
\leq & \left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right] \\
\leq & \left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right] \ .
\end{aligned}$$

Since the NIZK argument for the language L_{ad} satisfies zero knowledge, then Claim A.19 guarantees that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ_1 such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y,y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \leq \epsilon_1(\lambda) \ . \tag{32}$$

Similarly, since the NIZK argument for the language L_{eq} satisfies zero knowledge, then Claim A.20 guarantees that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ_2 such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y,y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \leq \epsilon_2(\lambda) \ . \tag{33}$$

Lastly, since the FHE encryption scheme satisfies semantic security, Claim A.21 guarantees that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function ϵ_3 such that for all p.p.t.

distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right]\right| \leq \epsilon_3(\lambda) \ . \tag{34}$$

Combining equations 32, 33, and 34 with the inequality above that was derived from the triangle inequality, we obtain that:

$$\begin{aligned}\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \leq \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \\ + \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda) \ .\end{aligned}$$

Notice that since $\epsilon_1(\lambda), \epsilon_2(\lambda)$, and $\epsilon_3(\lambda)$ are all negligible functions, we have that the function $\epsilon(\lambda) = \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda)$, for all $\lambda \in \mathbb{N}$, is also a negligible function. Note that since for any stateful p.p.t. adversary $\mathcal{A}$, for any p.p.t. distinguisher $\mathcal{D}$, and for all $(Y, y) \in \mathsf{R}$, we can always find the functions $\epsilon_1(\lambda), \epsilon_2(\lambda)$, and $\epsilon_3(\lambda)$, we can then also always find the negligible function $\epsilon(\lambda)$ for the same constraints. The inequality then becomes:

$$\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \leq \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] + \epsilon(\lambda) \ ,$$

which implies that:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right]\right| \leq \epsilon(\lambda) \ . \tag{35}$$

Therefore, we have shown that for all stateful p.p.t. adversaries $\mathcal{A}$, there exists a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$, we have that equation 35 holds, which is what we wanted to show. This concludes the proof of zero knowledge (non-abort) for the FAS scheme.

□

A.7 Zero-Knowledge (abort)

Theorem A.23 (Zero Knowledge (abort) for FAS Construction)**.** *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy zero knowledge, the* FHE *encryption scheme satisfies semantic security, and the* ElGamal *encryption scheme satisfies semantic security, then the proposed* FAS *construction in Figure 20 satisfies zero knowledge (abort) as defined in Definition 4.10.*

Overview. In order to prove the zero knowledge (abort) of FAS, we need to show that for every stateful p.p.t. adversary $\mathcal{A}$ that aborts before providing $\widetilde{\sigma}_2$ to the honest seller, there exists a stateful p.p.t. simulator Sim and a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and all $(Y, y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right]\right| \leq \epsilon(\lambda) \ .$$

First, we need to construct the stateful p.p.t. simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{FPreSign}^*_1)$. Let $\mathsf{Sim}_{\mathsf{ad}} = (\mathsf{Setup}^*_{\mathsf{ad}}, \mathsf{Prove}^*_{\mathsf{ad}})$ be the simulator (as defined in Definition 3.33) for the NIZK argument for the language L_{ad}. Similarly, let $\mathsf{Sim}_{\mathsf{eq}} = (\mathsf{Setup}^*_{\mathsf{eq}}, \mathsf{Prove}^*_{\mathsf{eq}})$ be the simulator for the NIZK argument for the language L_{eq}. We then define our simulator Sim as in Figure 41.

Next, consider the sequence of games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ as described in Figure 42.

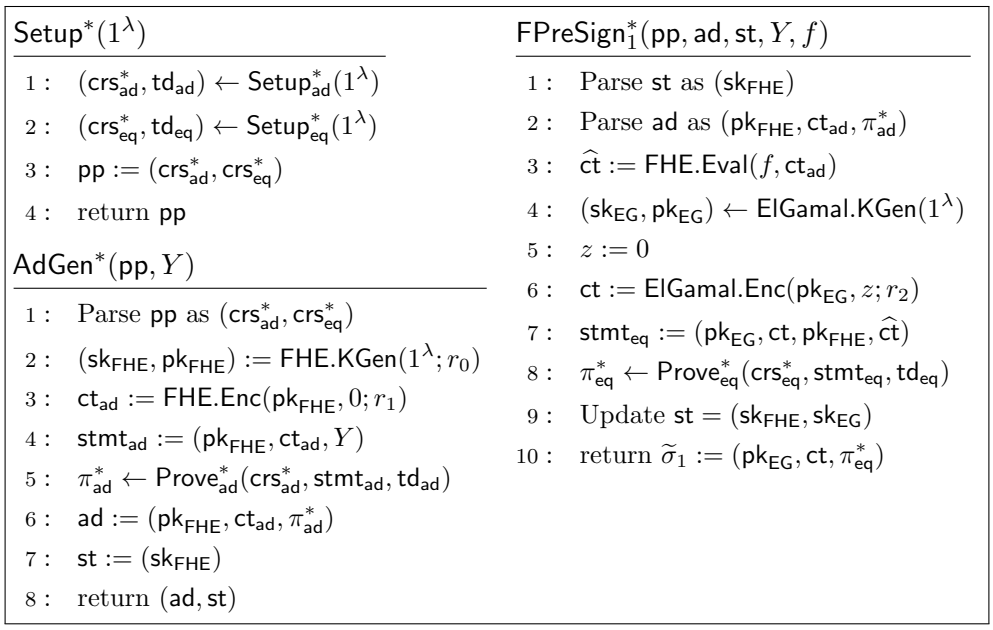

Figure 41: Simulator Sim for zero knowledge (abort) of FAS proof

- The first game is $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose. In this game, the adversary $\mathcal{A}$ is interacting with a challenger who has access to the witness y. The adversary is able to run as many interactions as it wants, and we then we examine the view of $\mathcal{A}$.
- The game $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ is identical to the first game, except that it uses the simulator $\mathsf{Sim}_{\mathsf{ad}}$'s methods $(\mathsf{Setup}^*_{\mathsf{ad}}, \mathsf{Prove}^*_{\mathsf{ad}})$ to generate the proof of correctness for $\mathsf{ct}_{\mathsf{ad}}$. This involves generating the pair $(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}})$ during setup, and using the trapdoor $\mathsf{td}_{\mathsf{ad}}$ instead of $\mathsf{wit}_{\mathsf{ad}}$ during the proof. Since we no longer need $\mathsf{wit}_{\mathsf{ad}}$, we set it to $\perp$, to indicate we no longer need to use y for the proof.
- The game $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$ is identical to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$, except that it uses the simulator $\mathsf{Sim}_{\mathsf{eq}}$'s methods $(\mathsf{Setup}^*_{\mathsf{eq}}, \mathsf{Prove}^*_{\mathsf{eq}})$ to generate the proof of equality for ct and $\widehat{\mathsf{ct}}$. This involves generating the pair $(\mathsf{crs}^*_{\mathsf{eq}}, \mathsf{td}_{\mathsf{eq}})$ during setup, and using the trapdoor $\mathsf{td}_{\mathsf{eq}}$ instead of $\mathsf{wit}_{\mathsf{eq}}$ during the proof. Since we no longer need $\mathsf{wit}_{\mathsf{eq}}$, we set it to $\perp$ to indicate this.
- The last game is $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ for the FAS scheme we propose. Note that this game is identical to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$ except that $\mathsf{ct}_{\mathsf{ad}}$ and ct now both encrypt 0, rather than y and $f(y)$ respectively. In this game, the adversary $\mathcal{A}$ is interacting with a challenger who has access to no information of the witness y. The adversary is able to run as many interactions as it wants, and we then we examine the view of $\mathcal{A}$.

First, we will argue in Claim A.24 that the hop from $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$ to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ is close due to the zero knowledge property of the NIZK argument for the language L_{ad}. Similarly, we argue in Claim A.25 that the hop from $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$ to $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$ is close due to the zero knowledge property of the NIZK argument for the language L_{eq}. Finally, we prove in Claim A.26 that the hop from $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$ to $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ is close due to the semantic security of the FHE encryption scheme and the semantic security of the $\mathsf{ElGamal}$ encryption scheme. The formal proof for Theorem A.23 can be found in Claim A.27.

Games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$, $\boxed{\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)}$, $\boxed{\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)}$, $\boxed{\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)}$

1 : $\mathsf{crs}_{\mathsf{ad}} \leftarrow \mathsf{Setup}_{\mathsf{ad}}(1^\lambda)$ $\boxed{(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}}) \leftarrow \mathsf{Setup}^*_{\mathsf{ad}}(1^\lambda)}$

2 : $\mathsf{crs}_{\mathsf{eq}} \leftarrow \mathsf{Setup}_{\mathsf{eq}}(1^\lambda)$ $\boxed{(\mathsf{crs}^*_{\mathsf{eq}}, \mathsf{td}_{\mathsf{eq}}) \leftarrow \mathsf{Setup}^*_{\mathsf{eq}}(1^\lambda)}$

3 : $\mathsf{pp} := (\mathsf{crs}_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})$ $\boxed{\mathsf{pp} := (\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{crs}_{\mathsf{eq}})}$ $\boxed{\mathsf{pp} := (\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{crs}^*_{\mathsf{eq}})}$

4 : $(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{FHE}}) := \mathsf{FHE.KGen}(1^\lambda; r_0)$

5 : $\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, y; r_1)$ $\boxed{\mathsf{ct}_{\mathsf{ad}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, 0; r_1)}$

6 : $\mathsf{stmt}_{\mathsf{ad}} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, Y)$

7 : $\mathsf{wit}_{\mathsf{ad}} := (y, r_1, r_0)$ $\boxed{\mathsf{wit}_{\mathsf{ad}} := \bot}$

8 : $\pi_{\mathsf{ad}} \leftarrow \mathsf{Prove}_{\mathsf{ad}}(\mathsf{crs}_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \mathsf{wit}_{\mathsf{ad}})$ $\boxed{\pi_{\mathsf{ad}} \leftarrow \mathsf{Prove}^*_{\mathsf{ad}}(\mathsf{crs}^*_{\mathsf{ad}}, \mathsf{stmt}_{\mathsf{ad}}, \mathsf{td}_{\mathsf{ad}})}$

9 : $\mathsf{ad} := (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{ad}}, \pi_{\mathsf{ad}})$

10 : $\mathsf{st} := (\mathsf{sk}_{\mathsf{FHE}})$

11 : $(m, f, \mathsf{vk}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ad}, Y)$

12 : while $\mathcal{A}$ is not done making queries:

13 : $\quad \widehat{\mathsf{ct}} := \mathsf{FHE.Eval}(f, \mathsf{ct}_{\mathsf{ad}})$

14 : $\quad (\mathsf{sk}_{\mathsf{EG}}, \mathsf{pk}_{\mathsf{EG}}) \leftarrow \mathsf{ElGamal.KGen}(1^\lambda)$

15 : $\quad z := f(y)$ $\boxed{z := 0}$

16 : $\quad \mathsf{ct} := \mathsf{ElGamal.Enc}(\mathsf{pk}_{\mathsf{EG}}, z; r_2)$

17 : $\quad \mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \mathsf{pk}_{\mathsf{FHE}}, \widehat{\mathsf{ct}})$

18 : $\quad \mathsf{wit}_{\mathsf{eq}} := (\mathsf{sk}_{\mathsf{FHE}}, z, r_2, r_0)$ $\boxed{\mathsf{wit}_{\mathsf{eq}} := \bot}$

19 : $\quad \pi_{\mathsf{eq}} \leftarrow \mathsf{Prove}_{\mathsf{eq}}(\mathsf{crs}_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \mathsf{wit}_{\mathsf{eq}})$ $\boxed{\pi_{\mathsf{eq}} \leftarrow \mathsf{Prove}^*_{\mathsf{eq}}(\mathsf{crs}^*_{\mathsf{eq}}, \mathsf{stmt}_{\mathsf{eq}}, \mathsf{td}_{\mathsf{eq}})}$

20 : $\quad$ Update $\mathsf{st} := (\mathsf{sk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{EG}})$

21 : $\quad \widetilde{\sigma}_1 := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$

22 : $\quad (m, f) \leftarrow \mathcal{A}(\widetilde{\sigma}_1)$

23 : return view of $\mathcal{A}$

Figure 42: Zero Knowledge (abort) Proof: Games $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad,eq}}}_{\mathcal{A},\mathsf{FAS}}$, and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$

□

Claim A.24. *If the* NIZK *argument for the language L_{ad} satisfies zero knowledge, then for every stateful p.p.t. adversary $\mathcal{A}$ that aborts before providing $\widetilde{\sigma}_2$ to the honest seller, there exists a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, for all $(Y, y) \in \mathsf{R}$:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) \ .$$

Proof. Similar to proof of Claim A.19. □

Claim A.25. *If the* NIZK *argument for the language* L_{eq} *satisfies zero knowledge, then for every stateful adversary* $\mathcal{A}$ *that aborts before providing* $\widetilde{\sigma}_2$ *to the honest seller, there exists a negligible function* ϵ*, such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, for all* $(Y, y) \in \mathsf{R}$*:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) \ .$$

Proof. Similar to proof of Claim A.20. □

Claim A.26. *If the* FHE *scheme satisfies semantic security and the* ElGamal *encryption scheme satisfies semantic security, then for every stateful p.p.t. adversary* $\mathcal{A}$ *that aborts before providing* $\widetilde{\sigma}_2$ *to the honest seller, there exists a negligible function* ϵ*, such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, and for all* $(Y, y) \in \mathsf{R}$*:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) \ .$$

Proof. We prove by introducing an intermediate hybrid $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ that is same as the hybrid $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ except that only the ciphertext $\mathsf{ct}_{\mathsf{ad}}$ is changed to encrypt 0. Observe that we can reduce the indistinguishability of hybrids $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ and $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ to the semantic security of the FHE encryption scheme. This proof is similar to the proof of Claim A.21. Therefore, all that remains to show is that hybrids $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ are indistinguishable. Observe that $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ is same as $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ except that the ElGamal ciphertext ct encrypts 0. We reduce the indistinguishability of the two hybrids to the semantic security of the ElGamal encryption scheme next. Note that if the adversary makes Q number of queries (m_i, f_i), then, we need to change all the Q ElGamal ciphertexts ct_i encrypting $f_i(y)$ to encryptions of 0 and this will require Q number of hybrids, where in i^{th} hybrid we switch ct_i. For sake of simplicity, we present the proof below for $Q = 1$, that is, a single query. Note that the proof for Q queries follows similarly via a hybrid argument.

Assume towards a contradiction that for any stateful p.p.t. adversary $\mathcal{A}$, there exists a p.p.t. distinguisher $\mathcal{D}$ such that for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$:

$$\left| \Pr\left[\mathcal{D}\left(\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| > \epsilon(\lambda) \ . \tag{36}$$

Notice that the games $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ are exactly the same, except for the way in which ElGamal ciphertext ct is generated. In $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$, ct is an ElGamal encryption of $f(y)$, while in $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$, ct is an ElGamal encryption of 0. We want to show that if the p.p.t. distinguisher $\mathcal{D}$ exists (as we assumed above), then we can break the semantic security of the ElGamal encryption scheme.

We will show that if such a distinguisher $\mathcal{D}$ exists, then we can create a reduction $\mathcal{B}$ that breaks the semantic security of the ElGamal encryption scheme. Let $\mathcal{C}$ be the challenger for the semantic security game for the ElGamal encryption scheme. $\mathcal{B}$ plays the role of the adversary in this semantic security game. $\mathcal{B}$ also serves as the challenger to adversary $\mathcal{A}$ playing the games $\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$. The procedure for the reduction is as follows:

1. $\mathcal{C}$ honestly samples the ElGamal key pair $(\mathsf{sk}_{\mathsf{EG}}, \mathsf{pk}_{\mathsf{EG}})$, and sends $\mathsf{pk}_{\mathsf{EG}}$ to $\mathcal{B}$.

2. $\mathcal{B}$ completes the setup necessary to generate the public parameters pp for the game $\mathcal{A}$ is playing. In this case, this involves performing the setup for the NIZK arguments. Next, $\mathcal{B}$ samples the FHE key pair and computes $\mathsf{ct}_{\mathsf{ad}}$ as FHE encryption of 0. Next, $\mathcal{B}$ computes π_{ad} using the $\mathsf{NIZK}_{\mathsf{ad}}$ simulator, computes ad and sends $\mathsf{pp}, \mathsf{ad}, Y$ to the adversary ad. $\mathcal{B}$ obtains (m, f, vk) from $\mathcal{A}$. $\mathcal{B}$ computes $\widehat{\mathsf{ct}}$ as the FHE evaluation of f on $\mathsf{ct}_a dvt$.

3. $\mathcal{B}$ sends the challenge messages $(f(y), 0)$ to the challenger $\mathcal{C}$ and obtains a ciphertext ct from it. Based on this, it sets $\mathsf{stmt}_{\mathsf{eq}} := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ctpk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{FHE}})$. Next, $\mathcal{B}$ computes π_{eq} using the $\mathsf{NIZK}_{\mathsf{eq}}$ simulator and sends $\widetilde{\sigma}_1 := (\mathsf{pk}_{\mathsf{EG}}, \mathsf{ct}, \pi_{\mathsf{eq}})$ to the adversary $\mathcal{A}$.

4. At this point, the interaction with $\mathcal{A}$ ends since it does not return any $\widetilde{\sigma}_2$. So, $\mathcal{B}$ uses the distinguisher $\mathcal{D}$ on $\mathcal{A}$. Then, $\mathcal{B}$ simply forwards the output of $\mathcal{D}$ to the challenger $\mathcal{C}$.

Since the only difference between the two games $\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ is the way ct is encrypted, if $\mathcal{D}$ can distinguish between the games with some probability p, then that means that $\mathcal{D}$ can tell which value ct encrypts with the same probability p. Since $\mathcal{B}$ outputs the result that $\mathcal{D}$ outputs, we can then express the probability that $\mathcal{B}$ wins the semantic security game as follows:

$$\Pr[\mathcal{B} \text{ wins}] \geq \left| \Pr\left[\mathcal{D}\left(\overline{\mathsf{Hyb}}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| . \quad (37)$$

Combining equations 36 and 37, we have that $\Pr[\mathcal{B} \text{ wins}] > \epsilon(\lambda)$, where ϵ is a negligible function. Therefore, we have constructed an adversary $\mathcal{B}$ that breaks the semantic security of the $\mathsf{ElGamal}$ encryption scheme, which is a contradiction to the assumptions made in the claim. This concludes the proof.

□

Claim A.27. *If the* NIZK *arguments for the languages* L_{ad} *and* L_{eq} *both satisfy zero knowledge, the* FHE *encryption scheme satisfies semantic security, and the* $\mathsf{ElGamal}$ *encryption scheme satisfies semantic security, then for all stateful p.p.t. adversaries* $\mathcal{A}$*, there exists a negligible function* ϵ *such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, and for all* $(Y, y) \in \mathsf{R}$ *, we have that:*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \right| \leq \epsilon(\lambda) ,$$

where Sim *is the simulator defined in Figure 41, and the two games* $\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ *and* $\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)$ *are defined as in Figure 42.*

Proof. Using the triangle inequality multiple times gives that:

$$\begin{aligned}
\Pr[\mathcal{D}(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y))=1] \leq & \left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right] \\
\leq & \left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right] \\
\leq & \left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right. \\
& \left.-\Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \\
& +\Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]
\end{aligned}$$

Since the NIZK argument for the language L_{ad} satisfies zero knowledge, then Claim A.24 guarantees that for every stateful p.p.t. adversary $\mathcal{A}$ that aborts before providing $\widetilde{\sigma}_2$ to the honest seller, there exists a negligible function ϵ_1 such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y,y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \leq \epsilon_1(\lambda)\ . \tag{38}$$

Similarly, since the NIZK argument for the language L_{eq} satisfies zero knowledge, then Claim A.25 guarantees that for every stateful p.p.t. adversary $\mathcal{A}$ that aborts before providing $\widetilde{\sigma}_2$ to the honest seller, there exists a negligible function ϵ_2 such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y,y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]-\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda,Y,y)\right)=1\right]\right| \leq \epsilon_2(\lambda)\ . \tag{39}$$

Lastly, since the FHE and ElGamal encryption schemes satisfy semantic security, Claim A.26 guarantees that for every stateful p.p.t. adversary $\mathcal{A}$ that aborts before providing $\widetilde{\sigma}_2$ to the honest seller, there exists a negligible function ϵ_3 such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{Hyb}^{\mathsf{Sim}_{\mathsf{ad},\mathsf{eq}}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right]\right| \leq \epsilon_3(\lambda) \ . \tag{40}$$

Combining the equations 38, 39, and 40 with the inequality above that was derived from the triangle inequality, we obtain that:

$$\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \leq \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] + \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda) \ .$$

Notice that since $\epsilon_1(\lambda), \epsilon_2(\lambda)$, and $\epsilon_3(\lambda)$ are all negligible functions, we have that the function $\epsilon(\lambda) = \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda)$, for all $\lambda \in \mathbb{N}$, is also a negligible function. Note that since for any stateful p.p.t. adversary $\mathcal{A}$, for any p.p.t. distinguisher $\mathcal{D}$, and for all $(Y, y) \in \mathsf{R}$, we can always find the functions $\epsilon_1(\lambda), \epsilon_2(\lambda)$, and $\epsilon_3(\lambda)$, we can then also always find the negligible function $\epsilon(\lambda)$ for the same constraints. The inequality then becomes:

$$\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] \leq \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] + \epsilon(\lambda) \ ,$$

which implies that:

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{fZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{fZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, Y, y)\right) = 1\right]\right| \leq \epsilon(\lambda) \ . \tag{41}$$

Therefore, we have shown that for all stateful p.p.t. adversaries $\mathcal{A}$, there exists a negligible function ϵ, such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, and for all $(Y, y) \in \mathsf{R}$, we have that equation 41 holds, which is what we wanted to show. This conlcudes the proof of zero knowledge (abort) for the FAS scheme.

□

B Security Proofs for Cut-and-choose based $\mathsf{NIZK}_{\mathsf{eq}}$

Claim B.1 (Completeness)**.** *If encryption schemes* FHE *and* EG *are linearly homomorphic, then* $\mathsf{NIZK}_{\mathsf{eq}}$ *in Figure 19 satisfies completeness.*

Proof. Follows in a straightforward way. □

Claim B.2 (Adaptive Soundness)**.** $\mathsf{NIZK}_{\mathsf{eq}}$ *in Figure 19 satisfies adaptive soundness (Theorem 3.32) in the random oracle model.*

Proof. We prove soundness via a sequence of two hybrid experiments Hyb_0 and Hyb_1. Hyb_0 corresponds to the original soundness game against an adversarial p.p.t. prover $\mathcal{P}^*$. Hyb_1 is same as Hyb_0 except the hash function H is replaced by a random function RF. Since we are in the random oracle model, Hyb_0 and Hyb_1 are identically distributed. Thus, it suffices to show that in Hyb_1, for all $\mathcal{P}^*$, there exists a negligible function ε such that for all $\lambda \in \mathbb{N}$, we have

$$\Pr\left[\mathsf{stmt} \notin L_{\mathsf{R}} \wedge \mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1 \ : \ \begin{matrix} \mathsf{crs} \leftarrow \mathsf{Setup}(\lambda), \\ (\mathsf{stmt}, \pi) \leftarrow \mathcal{P}(\mathsf{crs}) \end{matrix}\right] \leq \epsilon(\lambda) \ .$$

We will show that the above holds for the negligible function $\epsilon(\lambda) = 1/2^\lambda$.

Suppose $\mathcal{P}^*$ on input crs returns (stmt, π), where $\mathsf{stmt} = (\mathsf{ct}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{EG}})$ and $\pi = (\{\mathsf{ct}_{\mathsf{FHE},i}, \mathsf{ct}_{\mathsf{EG},i}\}_{i\in[2\lambda]}, I, \{\widetilde{s_i}, \widetilde{r_i}, \widetilde{\overline{r_i}}\}_{i\in I}, \{\widehat{s_i}, \widehat{r_i}, \widehat{\overline{r_i}}\}_{i\notin I})$. Without loss of generality, suppose that $\mathsf{ct}_{\mathsf{FHE}} = \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, z; r)$ and $\mathsf{ct}_{\mathsf{EG}} = \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, \overline{z}; \overline{r})$ for some $z, \overline{z}, r$ and $\overline{r}$. Then, $\mathsf{stmt} \notin L_{\mathsf{eq}}^{\mathsf{pk}_{\mathsf{FHE}},\mathsf{pk}_{\mathsf{EG}}}$ implies that $z \neq \overline{z}$. Further, $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1$ implies all of the following:

- $I = \{2 \cdot i + str[i] : i \in [\lambda]\}$, where

  $str := \mathsf{RF}(\mathsf{ct}_{\mathsf{FHE}}||\mathsf{ct}_{\mathsf{EG}}||\mathsf{ct}_{\mathsf{FHE},1}||\ldots||\mathsf{ct}_{\mathsf{FHE},2\lambda}||\mathsf{ct}_{\mathsf{EG},1}||\ldots||\mathsf{ct}_{\mathsf{EG},2\lambda}) \in \{0,1\}^\lambda$.
- For all $i \in I$: $\widetilde{s_i} = s_i = \overline{s_i}$, $\widetilde{r_i} = r_i$ and, $\widetilde{\overline{r_i}} = \overline{r_i}$.
- For all $i \notin I$: $\widehat{s_i} = z + s_i = \overline{z} + \overline{s_i}$, $\widetilde{r_i} = r + r_i$ and, $\widetilde{\overline{r_i}} = \overline{r} + \overline{r_i}$. This implies that $\overline{s_i} = s_i + (z - \overline{z})$.

Observe that the cheating prover above can freely choose the variables $z, \overline{z}, r, \overline{r}, \{s_i, r_i, \overline{r_i}\}_{i\in[2\lambda]}$ and its task is reduced to guessing $str \in \{0,1\}^\lambda$ which determines the set I. It can then set $\{\overline{s_i} := s_i + (z - \overline{z})\}_{i\notin I}$. This fully determines the values of the input to the random function RF and thus its output $str' \in \{0,1\}^\lambda$ and the corresponding set I'. The prover wins if $I = I'$. This holds iff $str = str'$. Since RF is a random function, the probability of this event is $1/2^\lambda$. Thus, $\epsilon(\lambda) = 1/2^\lambda$. □

Claim B.3 (Zero-Knowledge)**.** $\mathsf{NIZK}_{\mathsf{eq}}$ *in Figure 19 satisfies zero-knowledge (Theorem 3.33) in the programmable random oracle model.*

Proof. We start by describing the p.p.t. simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Prove}^*)$. Let PRO be the programmable random oracle. Then, the simulator is as in Figure 43.

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

1 : Sample $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{PRO.Gen}(1^\lambda)$

2 : **ret** $\mathsf{crs}^* := \mathsf{hk}$ and td

$\mathsf{Prove}^*(\mathsf{crs}^*, \mathsf{stmt} = (\mathsf{ct}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{EG}}))$:

1 : Compute $str \xleftarrow{\$} \{0,1\}^\lambda$.

2 : Define set $I = \{2 \cdot i + str[i] : i \in [\lambda]\}$ of size λ.

3 : For all $i \in I$: Sample random messages $s_i \xleftarrow{\$} \mathbb{Z}_p$, random coins r_i and $\overline{r_i}$ and compute $\mathsf{ct}_{\mathsf{FHE},i} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, s_i; r_i)$ and $\mathsf{ct}_{\mathsf{EG},i} := \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, s_i; \overline{r_i})$.

4 : For all $i \notin I$: Sample random messages $\widehat{s_i} \xleftarrow{\$} \mathbb{Z}_p$, random coins $\widehat{r_i}$ and $\widehat{\overline{r_i}}$ and compute $\widehat{\mathsf{ct}_{\mathsf{FHE},i}} := \mathsf{FHE.Enc}(\mathsf{pk}_{\mathsf{FHE}}, \widehat{s_i}; \widehat{r_i})$, $\widehat{\mathsf{ct}_{\mathsf{EG},i}} := \mathsf{EG.Enc}(\mathsf{pk}_{\mathsf{EG}}, \widehat{s_i}; \widehat{\overline{r_i}})$, $\mathsf{ct}_{\mathsf{FHE},i} := \widehat{\mathsf{ct}_{\mathsf{FHE},i}} - \mathsf{ct}_{\mathsf{FHE}}$ and, $\mathsf{ct}_{\mathsf{EG},i} := \widehat{\mathsf{ct}_{\mathsf{EG},i}}/\mathsf{ct}_{\mathsf{EG}}$.

5 : Program PRO using td such that $\mathsf{PRO.Hash}(\mathsf{hk}, \mathsf{ct}_{\mathsf{FHE}}||\mathsf{ct}_{\mathsf{EG}}||\mathsf{ct}_{\mathsf{FHE},1}||\ldots||\mathsf{ct}_{\mathsf{FHE},2\lambda}||\mathsf{ct}_{\mathsf{EG},1}||\ldots||\mathsf{ct}_{\mathsf{EG},2\lambda}) := str$.

6 : **ret** $\pi := (\{\mathsf{ct}_{\mathsf{FHE},i}, \mathsf{ct}_{\mathsf{EG},i}\}_{i\in[2\lambda]}, I, \{s_i, r_i, \overline{r_i}\}_{i\in I}, \{\widehat{s_i}, \widehat{r_i}, \widehat{\overline{r_i}}\}_{i\notin I})$.

Figure 43: Simulator used in proof of Claim B.3

Having described the simulator Sim, we now show that there exists a negligible function ϵ such that for all p.p.t. distinguishers $\mathcal{D}$, $\lambda \in \mathbb{N}$, and statement/witness pairs $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{R}_{\mathsf{eq}}^{\mathsf{pk}_{\mathsf{FHE}},\mathsf{pk}_{\mathsf{EG}}}$, we have that:

$$\left|\Pr[\mathcal{D}(\mathsf{crs},\pi)=1]-\Pr[\mathcal{D}(\mathsf{crs}^*,\pi^*)=1]\right|\leq\epsilon(\lambda)\ ,$$

where $\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda)$, $\pi\leftarrow\mathsf{Prove}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})$,$(\mathsf{crs}^*,\mathsf{td})\leftarrow\mathsf{Setup}^*(\lambda)$,$\pi^*\leftarrow\mathsf{Prove}^*(\mathsf{crs}^*,\mathsf{stmt},\mathsf{td})$.

Let experiment $\mathsf{NIZKReal}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt},\mathsf{wit})$ return (crs,π), where $\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda)$ and $\pi\leftarrow\mathsf{Prove}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})$ and experiement $\mathsf{NIZKIdeal}^{\mathsf{Sim}}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt})$ return (crs^*,π^*), where $(\mathsf{crs}^*,\mathsf{td})\leftarrow\mathsf{Setup}^*(1^\lambda)$ and $\pi^*\leftarrow\mathsf{Prove}^*(\mathsf{crs}^*,\mathsf{td},\mathsf{stmt})$. Then, it suffices to show

$$\left|\Pr\left[\mathcal{D}(\mathsf{NIZKReal}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt},\mathsf{wit}))=1\right]-\Pr\left[\mathcal{D}(\mathsf{NIZKIdeal}^{\mathsf{Sim}}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt}))=1\right]\right|\leq\epsilon(\lambda).$$

We show this through a sequence of hybrid experiments $\mathsf{Hyb}_0,\ldots,\mathsf{Hyb}_3$ that are as follows:

Experiment Hyb_0: This is same as $\mathsf{NIZKReal}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt},\mathsf{wit})$.

Experiment Hyb_1: Same as Hyb_0 except that the hash function is switch from H to the programmable random oracle PRO.

Experiment Hyb_2: Same as Hyb_1 except that str is sampled uniformly at random and the random oracle is programmed using the trapdoor td such that $\mathsf{PRO.Hash}(\mathsf{hk},\mathsf{ct}_{\mathsf{FHE}}||\mathsf{ct}_{\mathsf{EG}}||\mathsf{ct}_{\mathsf{FHE},1}||\ldots||\mathsf{ct}_{\mathsf{FHE},2\lambda}||\mathsf{ct}_{\mathsf{EG},1}||\ldots||\mathsf{ct}_{\mathsf{EG},2\lambda}):=str$.

Experiment Hyb_3: This is same as $\mathsf{NIZKIdeal}^{\mathsf{Sim}}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt})$. In other words, this is same as Hyb_2 except that for all $i\notin I$, $\mathsf{ct}_{\mathsf{FHE},i}$ and $\mathsf{ct}_{\mathsf{EG},i}$ are computed as in $\mathsf{NIZKIdeal}^{\mathsf{Sim}}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt})$.

Analysis. Observe that Hyb_0 and Hyb_1 are identically distributed since we are in the random oracle model. Further, due to programmability of the random oracle, Hyb_1 and Hyb_2 are identically distributed. Lastly, due to the correctness and linear homomorphism of FHE and EG encryption schemes, Hyb_2 and Hyb_3 are identically distributed. By a hybrid argument, it follows that Hyb_0 and Hyb_3 are identically distributed. Thus, for the distinguisher $\mathcal{D}$, we have

$$\left|\Pr\left[\mathcal{D}(\mathsf{NIZKReal}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt},\mathsf{wit}))=1\right]-\Pr\left[\mathcal{D}(\mathsf{NIZKIdeal}^{\mathsf{Sim}}_{\mathsf{NIZK}_{\mathsf{eq}}}(1^\lambda,\mathsf{stmt}))=1\right]\right|=0\ .$$

This completes the proof. □

C Security Proofs for Proof of Plaintext Equality (PoPE)

Here, we include the formal security proofs for the soundness and zero-knowledge properties of the PoPE proposed in Section 6.2.

Proof of Soundness (see Theorem 6.2). Recall that a sigma protocol is computationally special sound [COPZ22] if there exists an (efficient) extractor algorithm Ext such that for any p.p.t. adversary that outputs a statement stmt and two non-aborting transcripts $t=(a,c,z)$ and $t'=(a,c',z')$ such that $c\neq c'$, running $\mathsf{Ext}(\mathsf{stmt},t,t')$ returns some valid witness wit such that $(\mathsf{stmt},\mathsf{wit})\in\mathsf{R}_{\mathsf{eq}}$, only failing with probability ϵ, where ϵ is called the knowledge error of the protocol.

So we need to show that there exists an extractor Ext that on input $\mathsf{stmt}=\left(X_{p,0},\ldots,X_{p,k},X_n\right)\in\mathbb{G}_p^k\times\mathbb{G}_n$, and two accepting conversations $\left((Y_p,Y_n),c,(z,s_p,s_{n1},s_{n2})\right)$ and $\left((Y_p,Y_n),c',(z',s'_p,s'_{n1},s'_{n2})\right)$, recovers a valid witness $\mathsf{wit}=(x,r_p,r_{n1},r_{n2},(\beta_0,b_0),\ldots,(\beta_k,b_k))$ such that $(\mathsf{stmt},\mathsf{wit})\in\mathsf{R}_{\mathsf{eq}}$, that

is, $X_p = X_{p,0} \cdots X_{p,k} = (g_p^{r_p}, g_p^x \cdot h_p^{r_p})$, $X_{p,j} = \left(g_p^{\beta_j},\ g_p^{b_j} \cdot h_p^{\beta_i}\right)$ for all $j \in \{0, \ldots, k\}$, and also $X_n = g^x \cdot h^{r_{n1}} \cdot i^{r_{n2}}$. We define the extractor Ext as follows:

---

(1) Divide the two equations $(g_p^{s_p}, g_p^z \cdot h_p^{s_p}) = Y_p \cdot (X_p)^c$ and $(g_p^{s'_p}, g_p^{z'} \cdot h_p^{s'_p}) = Y_p \cdot (X_p)^{c'}$ from the two accepting conversations to obtain the expression:

$$\left(g_p^{(s_p - s'_p)}, g_p^{(z-z')} \cdot h_p^{(s_p - s'_p)}\right) = X_p^{(c-c')} \ .$$

(2) Find the multiplicative inverse $(c-c')^{-1} \in \mathbb{Z}_p$ of $(c-c')$, and raise both sides to $(c-c')^{-1} \in \mathbb{Z}_p$, obtaining:

$$\left(g_p^{(s_p - s'_p)(c-c')^{-1}}, g_p^{(z-z')(c-c')^{-1}} \cdot h_p^{(s_p - s'_p)(c-c')^{-1}}\right) = X_p \ .$$

(3) Set $x_p = (z - z')(c - c')^{-1}$ and $r_p = (s_p - s'_p)(c - c')^{-1}$. Note that $x_p, r_p \in \mathbb{Z}_p$.

(4) If $(z - c \cdot x_p, s_p - c \cdot r_p) \neq (z' - c' \cdot x_p, s'_p - c' \cdot r_p)$, then **abort**.

(5) Divide the two equations $g^z \cdot h^{s_{n1}} \cdot i^{s_{n2}} = Y_n \cdot (X_n)^c$ and $g^{z'} \cdot h^{s'_{n1}} \cdot i^{s'_{n2}} = Y_n \cdot (X_n)^{c'}$ from the two accepting conversations to obtain the expression:

$$g^{(z-z')} \cdot h^{(s_{n1} - s'_{n1})} \cdot i^{(s_{n2} - s'_{n2})} = X_n^{(c-c')} \ .$$

(6) Find the multiplicative inverse $(c-c')^{-1} \in \mathbb{Z}_n^*$ of $(c-c')$, and raise both sides to $(c-c')^{-1} \in \mathbb{Z}_n$, obtaining:

$$g^{(z-z')(c-c')^{-1}} \cdot h^{(s_{n1} - s'_{n1})(c-c')^{-1}} \cdot i^{(s_{n2} - s'_{n2})(c-c')^{-1}} = X_n \ .$$

(7) Set $x_n = (z - z')(c - c')^{-1}$, $r_{n1} = (s_{n1} - s'_{n1})(c - c')^{-1}$, and $r_{n2} = (s_{n2} - s'_{n2})(c - c')^{-1}$. Note that $x_n, r_{n1}, r_{n2} \in \mathbb{Z}_n$.

(8) If $(z - c \cdot x_n, s_{n1} - c \cdot r_{n1}, s_{n2} - c \cdot r_{n2}) \neq (z' - c' \cdot x_n, s'_{n1} - c' \cdot r_{n1}, s'_{n2} - c' \cdot r_{n2})$, then **abort**.

(9) Run the extractor for the Encrypted Bits Protocol Π_{egb} (see Boneh-Shoup [BS17], uses extractor for Chaum-Pedersen Protcol). Output a witness (β_j, b_j) for all $j \in \{0, \ldots, k\}$.

(10) **return** $(x_p, r_p, r_{n1}, r_{n2}, (\beta_0, b_0), \ldots, (\beta_k, b_k))$

Now, we must show that this extractor Ext extracts valid witnesses, and fails to extract with some small probability ϵ.

First, we need to show that we can extract (x_p, r_p) as in **(3)**. That is, we need to show that $(c - c')^{-1}$ exists in $\mathbb{Z}_p^*$. Note that since $b_x + b_c + b_f < b_g$, we have that $b_c < b_g$, meaning $2^{b_c} < 2^{b_g}$. Since we know that $c, c' \in \{0, \ldots, 2^{b_c} - 1\}$, we know that $|c - c'| < 2^{b_c} < 2^{b_g}$. Since b_g is the bitlength of the prime p, this means that $|c - c'| < p$. So $(c - c') \in \mathbb{Z}_p^*$, and we know it has a multiplicative inverse $(c - c')^{-1} \in \mathbb{Z}_p^*$, meaning $(c - c')^{-1} \in \mathbb{Z}_p$. So $x_p, r_p \in \mathbb{Z}_p$ are well defined. Note that this extraction fails when $c = c'$, which happens with probability 2^{-b_c}. If this is not the case, then the extractor checks in **(4)** that the openings are consistent across the two conversations, aborting if

there is an inconsistency. In this instance, the prover does know the dlog between g_p and h_p, so the prover could have altered the commitments in the two transcripts. But due to the structure of the ElGamal encryptions, it is not possible for the prover to do such a manipulation, or the encryption would no longer be well formed. So the extractor does not abort in **(4)**. So the extraction of (x_p, r_p) fails with probability 2^{-b_c}.

Next, we need to show that we can extract (x_n, r_{n1}, r_{n2}) as in **(7)**. That is, we need to show that $(c - c')^{-1}$ exists in $\mathbb{Z}_n^*$. From prior analysis, we know that $|c - c'| < p$. We also know that $p < q_1, q_2$. So we have that $|c - c'| < q_1, q_2$, meaning that $\mathsf{gcd}((c - c'), n) = 1$. So $(c - c') \in \mathbb{Z}_n^*$, and we know it has a multiplicative inverse $(c - c')^{-1} \in \mathbb{Z}_n^*$, meaning $(c - c')^{-1} \in \mathbb{Z}_n$. So $x_n, r_{n1}, r_{n2} \in \mathbb{Z}_n$ are well defined. Note that this extraction fails when $c = c'$, which happens with probability 2^{-b_c}. If this is not the case, then the extractor checks in **(8)** that the openings are consistent across the two conversations, aborting if there is an inconsistency. Note that this occurs only when the **prover knows the discrete log of** h, i **with respect to** g. Since the discrete log problem is hard in $\mathbb{G}_n$, this happens with probability ϵ_{DL_n}. Therefore, the extraction of (x_n, r_{n1}, r_{n2}) fails with probability $2^{-b_c} + \epsilon_{DL_n}$.

So we have that Ext fails to extract (x_p, r_p) with probability 2^{-b_c}, and it fails to extract (x_n, r_n) with probability $2^{-b_c} + \epsilon_{DL_n}$. So the probability Ext fails to extract $(x_p, r_p, x_n, r_{n1}, r_{n2})$ is $2^{-b_c} + \epsilon_{DL_n}$. This is because the same challenges c, c' were used to extract each pair, so we do not need to double count the probability that $c = c'$. Note that we also need to include the probability that the extractor fails to extract in the Encrypted Bits Protocol. Let that error be ϵ_{egb}. So the knowledge error of the protocol Π_{eq} is $\epsilon = 2^{-b_c+1} + \epsilon_{DL_n} + \epsilon_{\mathsf{egb}}$.

Now we need to show that when Ext returns $(x_p, r_p, r_{n1}, r_{n2}, (\beta_0, b_0), \ldots, (\beta_k, b_k))$ in **(10)**, that the return value is a valid witness. That is, we need to show that $x_p = x_n$, as integers. We have that $z = y + c \cdot x_p \pmod p$, meaning that $z = y + c \cdot x_p + a \cdot p$ (over the integers). Similarly, we have that $z' = y + c' \cdot x_p + a' \cdot p$ (over the integers). Subtracting the two equations, we get that:

$$(z - z') = (c - c') \cdot x_p + (a - a') \cdot p \ . \tag{42}$$

We can use a similar argument to show that:

$$(z - z') = (c - c') \cdot x_n + (b - b') \cdot n \ . \tag{43}$$

Combining Equations 42 and 43, we get that:

$$(c - c') \cdot x_p + (a - a') \cdot p = (c - c') \cdot x_n + (b - b') \cdot n \ . \tag{44}$$

Note that since $x_p < 2^{b_x}$ and $|c - c'| < 2^{b_c}$, and $(z - z')$ has bitlength less than $b_g = \log_2(p)$, we know that $a - a' = 0$. So Equation 44 simplifies to:

$$(c - c') \cdot x_p = (c - c') \cdot x_n + (b - b') \cdot n \ ,$$

which then gives that:

$$(c - c') \cdot (x_p - x_n) = (b - b') \cdot q_1 \cdot q_2 \ .$$

Note that since q_1, q_2 are prime, they must either divide $(c - c')$, or $(x_p - x_n)$. Since the bitlength of q_1, q_2 is at least b_g, and $b_g > b_c$, then q_1, q_2 are too large to divide $(c - c')$. So q_1, q_2 must both divide $(x_p - x_n)$, meaning that $n | (x_p - x_n)$. This means that we have $x_p = x_n \pmod n$. Since we have that $x_p = x_n \pmod n$, and the bitlength of x_p is less than the bitlength of n, it must be that $x_p = x_n$ over the integers as well. This means that the value $(x_p, r_p, r_{n1}, r_{n2}, (\beta_0, b_0), \ldots, (\beta_k, b_k))$ that the extractor returns is a valid witness, that is:

$$X_{p,j} = \left(g_p^{\beta_j},\ g_p^{b_j} \cdot h_p^{\beta_i}\right) \text{ for all } j \in \{0 \ldots, k\}$$
$$\wedge\ X_{p,0} \cdots X_{p,k} = \left(g_p^{r_p},\ g_p^{x} \cdot h_p^{r_p}\right)$$
$$\wedge\ X_n = g^x \cdot h^{r_{n1}} \cdot i^{r_{n2}}\ .$$

Recall that we found that the probability that the extractor fails to extract the witness is $\epsilon = 2^{-b_c+1} + \epsilon_{DL_n} + \epsilon_{\mathsf{egb}}$. Therefore, since we have found an extractor Ext that is able to extract a valid witness given two accepting conversations for a given statement (with knowledge error of ϵ), we have that the protocol Π_{eq} is computational special sound for the relation R_{eq} with error ϵ, as desired.

□

Proof of Computational HVZK (see Theorem 6.3). Note that the protocol Π_{egb} was shown to be HVZK by Boneh-Shoup (Example 20.3) [BS17], so we have that HVZK of the Encrypted Bits Protocol follows from that proof.

Therefore, in order to show that the protocol is HVZK, we need to construct a simulator Sim that given some challenge c, can output a transcript for the proof of equality that is identically distributed to one generated by a real prover. We define the simulator Sim as follows:

(1) Sample $z \leftarrow\$ \{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_f} - 1\}$ uniformly at random.

(2) Sample $s_p \leftarrow\$ \mathbb{Z}_p$, $s_{n1} \leftarrow\$ \mathbb{Z}_n$, $s_{n2} \leftarrow\$ \mathbb{Z}_n$ uniformly at random.

(3) Compute $X_p = X_{p,0} \cdots X_{p,k}$

(4) Compute $Y_p = (g_p^{s_p}, g_p^z \cdot h_p^{s_p})(X_p)^{-c}$.

(5) Compute $Y_n = (g^z \cdot h^{s_{n1}} \cdot i^{s_{n2}})(X_n)^{-c}$.

(6) **return** $((Y_p, Y_n), c, \perp)$ with probability 2^{-b_f}, otherwise, **return** $((Y_p, Y_n), c, (z, s_p, s_{n1}, s_{n2}))$.

By generating the conversation backwards, the simulator can generate an accepting transcript $(Y_p, Y_n, c, (z, s_p, s_{n1}, s_{n2}))$ for any challenge c. However, during the normal protocol, the prover aborts when they obtain a badly-formed z value. But the simulator Sim, it can always choose a well-formed z, which means that the simulator would never abort in the situations that a normal prover would. Using Lemma 2 from Chase et al [COPZ22], we know that in this protocol, the prover will abort with probability 2^{-b_f}. So in order to simulate the aborting behavior of the prover, the simulator will simply abort with the same probability that a normal prover would. This means that with probability 2^{-b_f}, the simulator will "abort" by outputting the tuple $(Y_p, Y_n, c, \perp)$, which is what the verifier would see if a normal prover aborted. Otherwise, the simulator outputs the transcript $(Y_p, Y_n, c, (z, s_p, s_{n1}, s_{n2}))$, as a normal prover would. Therefore, the simulator Sim is able to mimic the behavior of a normal prover.

Now it remains to show that the real and simulated transcripts are indistinguishable. Since the simulator sampled s_p, s_n, and z randomly, we know that Y_p and Y_n are identically distributed to those generated by a real prover. Therefore, since the distribution of transcripts generated by the simulator Sim and a real prover are identical, we have that the protocol Π_{eq} is HVZK, as desired.

□

D Security Proofs for BGN-FAS

In this section, we provide formal security proofs for the BGN-FAS construction presented in Section 6. The structure of this section is as follows. Section D.1 contains the proof for correctness

(Definition 4.2). Section D.2 contains the proofs for advertisement soundness (Definition 4.3) and advertisement binding (Definition 4.4). Section D.3 contains the proofs for pre-signature adaptability (Definition 4.8) and pre-signature validity (Definition 4.5). Section D.4 contains the proof for unforgeability (Definition 4.6). Section D.5 contains the proof for witness extractability (Definition 4.7). Section D.6 contains the proof for non-abort case of zero knowledge (Definition 4.9). Section D.7 contains the proof for the abort case of zero knowledge (Definition 4.10).

D.1 Correctness

Theorem D.1 (Correctness for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of correctness as defined in Definition 4.2.*

Proof. We have that the NIZK schemes $\mathsf{NIZK}_{\mathsf{ad}}$ and $\mathsf{NIZK}_{\mathsf{eq}}$ satisfy completeness, the Schnorr adpator signature scheme satisfies correctness, and that BGN and ElGamal satisfy correctness. Therefore, by Theorem A.1, we have that BGN-FAS satisfies correctness, as desired. □

D.2 Advertisement Soundness and Binding

Theorem D.2 (Advertisement Soundness for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of advertisement soundness as defined in Definition 4.3.*

Proof. We have that the NIZK argument $\mathsf{NIZK}_{\mathsf{ad}}$ for the language L_{ad} satisfies adaptive soundness. Therefore, by Theorem A.2, we have that BGN-FAS satisfies advertisement soundness, as desired. □

Theorem D.3 (Advertisement Binding for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of advertisement binding as defined in Definition 4.4.*

Proof. We have that the NIZK argument $\mathsf{NIZK}_{\mathsf{ad}}$ for the language L_{ad} satisfies adaptive soundness, and that the BGN scheme satisfies binding (see Theorem 3.18). Therefore, by Theorem A.3, we have that BGN-FAS satisfies advertisement binding, as desired. □

D.3 Pre-Signature Adaptability and Validity

Theorem D.4 (Pre-Signature Adaptability for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of pre-signature adaptability as defined in Definition 4.8.*

Proof. We have that the Schnorr adaptor signature scheme satisfies pre-signature adaptability. Therefore, by Theorem A.4 we have that BGN-FAS satisfies pre-signature adaptability, as desired. □

Theorem D.5 (Pre-Signature Validity for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of pre-signature validity as defined in Definition 4.5.*

Proof. We have that the Schnorr adaptor signature scheme satisfies correctness. Therefore, by Theorem A.5 we have that BGN-FAS satisfies pre-signature validity, as desired. □

D.4 Unforgeability

Theorem D.6 (Unforgeability for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of unforgeability as defined in Definition 4.6.*

Proof. We know that the NIZK argument for L_{ad} satisfies adaptive soundness, and we proved in Lemma 6.5 that NIZK argument for L_{eq} also satisfies adaptive soundness (since it satisfies special soundness, which we proved in Theorem 6.2). Also, we know that the Schnorr adaptor signature scheme satisfies witness extractability. Further, we know that the relation R is $\mathcal{F}_{small}$-hard. Therefore, by Theorem A.6, we have that BGN-FAS satisfies unforgeability, as desired. □

D.5 Witness Extractability

Theorem D.7 (Witness Extractability for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of witness extractability as defined in Definition 4.7.*

Proof. We know that the NIZK argument for L_{ad} satisfies adaptive soundness, and we proved in Lemma 6.5 that NIZK argument for L_{eq} also satisfies adaptive soundness (since it satisfies special soundness, which we proved in Theorem 6.2). Also, we know that the Schnorr adaptor signature scheme satisfies witness extractability. Therefore, by Theorem A.12, we have that BGN-FAS satisfies witness extractability, as desired.

□

D.6 Zero-Knowledge (non-abort)

Theorem D.8 (Zero-Knowledge (non-abort) for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of zero-knowledge (non-abort) as defined in Definition 4.9.*

Proof. We know that the NIZK argument for L_{ad} satisfies zero-knowledge, and we proved in Lemma 6.6 that NIZK argument for L_{eq} satisfies computational honest verifier zero-knowledge. Further, we know that the BGN encryption scheme is semantically secure (see [BGN05]). Therefore, by Theorem A.17, we have that BGN-FAS satisfies zero-knowledge (non-abort), as desired. □

D.7 Zero-Knowledge (abort)

Theorem D.9 (Zero-Knowledge (abort) for BGN-FAS Construction)**.** *The proposed* BGN-FAS *construction in Section 6 satisfies the property of zero-knowledge (abort) as defined in Definition 4.10.*

Proof. We know that the NIZK argument for L_{ad} satisfies zero-knowledge, and we proved in Lemma 6.6 that NIZK argument for L_{eq} satisfies computational honest verifier zero-knowledge. Also, we know that the BGN encryption scheme is semantically secure (see [BGN05]). Further, we know that the ElGamal encryption scheme satisfies semantic security. Therefore, by Theorem A.23, we have that BGN-FAS satisfies zero-knowledge (abort), as desired. □