# Functional Adaptor Signatures: Beyond All-or-Nothing Blockchain-based Payments

Nikhil Vanjani[1], Pratik Soni[2], and Sri AravindaKrishnan Thyagarajan[3]

[1]Carnegie Mellon University
[2]University of Utah
[3]University of Sydney

**Abstract**

In scenarios where a seller holds sensitive data $x$, like employee / patient records or ecological data, and a buyer seeks to obtain an evaluation of specific function $f$ on this data, solutions in trustless digital environments like blockchain-based Web3 systems typically fall into two categories: (1) Smart contract-powered solutions and (2) cryptographic solutions leveraging tools such as adaptor signatures. The former approach offers atomic transactions where the buyer learns the function evaluation $f(x)$ (and not $x$ entirely) upon payment. However, this approach is often inefficient, costly, lacks privacy for the seller's data, and is incompatible with systems that do not support smart contracts with required functionalities. In contrast, the adaptor signature-based approach addresses all of the above issues but comes with an "all-or-nothing" guarantee, where the buyer fully extracts $x$ and does not support functional extraction of the sensitive data. In this work, we aim to bridge the gap between these approaches, developing a solution that enables fair functional sales of information while offering improved efficiency, privacy, and compatibility similar to adaptor signatures.

Towards this, we propose *functional adaptor signatures (FAS)* a novel cryptographic primitive that achieves all the desired properties as listed above. Using FAS, the seller can publish an advertisement committing to $x$. The buyer can pre-sign the payment transaction w.r.t. a function $f$, and send it along with the transaction to the seller. The seller adapts the pre-signature into a valid (buyer's) signature and posts the payment and the adapted signature on the blockchain to get paid. Finally, using the pre-signature and the posted signature, the buyer efficiently extracts $f(x)$, and completes the sale. We formalize the security properties of FAS, among which is a new notion called *witness privacy* to capture seller's privacy, which ensures the buyer does not learn anything beyond $f(x)$. We present multiple variants of witness privacy, namely, *witness hiding, witness indistinguishability*, and *zero-knowledge*, to capture varying levels of leakage about $x$ beyond $f(x)$ to a malicious buyer.

We introduce two efficient constructions of FAS supporting linear functions (like statistics/aggregates, kernels in machine learning, etc.), that satisfy the strongest notion of witness privacy. One construction is based on prime-order groups and compatible with Schnorr signatures for payments, and the other is based on lattices and compatible with a variant of Lyubashevsky's signature scheme. A central conceptual contribution of our work lies in revealing a surprising connection between functional encryption, a well-explored concept over the past decade, and adaptor signatures, a relatively new primitive in the cryptographic landscape. On a technical level, we avoid heavy cryptographic machinery and achieve improved efficiency, by making black-box use of building blocks like *inner product functional encryption (IPFE)* while relying on certain security-enhancing techniques for the IPFE in a non-black-box manner. We implement our FAS construction for Schnorr signatures and show that for reasonably sized seller witnesses, the different operations are quite efficient even for commodity hardware.

## Contents

## 1 Introduction

The shift from centralized Web2 to decentralized blockchain-based Web3 solutions has transformed digital goods trading. Smart contracts have been pivotal in this transition, and they facilitate seamless transactions between sellers and buyers. Imagine a seller offering a solution to a computational puzzle or a problem for sale. Typical smart contracts let the seller specify the terms of their digital offering, referred to as an *advertisement*, creating a transparent and accessible mechanism for buyers. The buyers lock coins to the contract as an expression of interest to buy the solution. Now the seller publishes the solution in the form of a transaction on the blockchain that invokes the contract. The contract executes a validation procedure to verify the solution's correctness, and if successful, transfers the locked coins to the seller. If the seller fails to respond with a correct solution, the contract refunds the buyer after the expiry of a timeout. This template underpins various digital trading scenarios including Hash Timelock Contracts [PD], Bridges [bri], NFT sales [WN22], smart contract-based offline services [cha], and e-commerce [KRA$^+$22].

Despite the advantages of smart contracts, their current implementation faces significant challenges:

- *Cost and Efficiency*: Sellers often incur hefty fees for posting advertisements on smart contracts and verifying secret solutions, resulting in higher transaction costs. Ethereum measures these costs in gas, and many popular smart contract applications have significantly higher gas costs compared to regular user-to-user payment transactions [liv].

- *Privacy*: Openly disclosing the secret solution compromises privacy. Encrypting values on the contract to address this necessitates more expensive and intricate cryptographic tools, further exacerbating costs. Further, the approach affects the fungibility[1] of the system as pointed out in many previous works [EFH$^+$21, TBM$^+$20, TM21, TMMS22, HLTW24].

- *Compatibility*: Finally, the method relies on *complex* smart contracts, rendering itself incompatible with prominent blockchains like Bitcoin. This poor adaptability hampers scalability in the broader Web3 ecosystem.

**Adaptor Signatures.** As the community delves deeper into enhancing the efficiency and simplicity of smart contracts, cryptographic tools like adaptor signatures [MMSS$^+$19, EEE20a, AEE$^+$21, EFH$^+$21, TMMS22, HLTW24, QPM$^+$23] have emerged as promising solutions. Adaptor signatures help model a blockchain-based fair exchange between a buyer for its signature $\sigma$ (on a transaction) and an NP witness $x$ that is known to the seller. More formally, the seller first samples an NP statement $X$ along with its witness $x$. The statement $X$ binds the seller to its witness which it publishes to advertise its intent to sell $x$.[2] Now, adaptor signatures offers four algorithms $\mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Adapt}, \mathsf{Ext}$, that work as follows. The buyer using the pre-sign algorithm $\mathsf{PreSign}$ first generates a pre-signature $\widetilde{\sigma}$ on the payment transaction $m$ w.r.t. the signing key $\mathsf{sk}$ and seller's statement $X$. The seller verifies the validity of $\widetilde{\sigma}$ using $\mathsf{PreVerify}$ algorithm. Then, using the $\mathsf{Adapt}$ algorithm, the seller adapts $\widetilde{\sigma}$ into a full signature $\sigma$ on $m$ using its witness $x$. To redeem the transaction $m$, the seller posts $\sigma$ on the blockchain. Central to the efficacy of adaptor signatures is their unique property that allows the buyer, with access to the pre-signature $\widetilde{\sigma}$ and the posted signature $\sigma$, to efficiently *extract* the secret solution $x$ using the algorithm $\mathsf{Ext}$, thus completing the fair exchange. This innovative approach addresses the drawbacks of traditional smart contracts:

[1] A blockchain system is said to be fungible if all units/coins in the system have the same value, independent of their history.

[2] Typically, this step is not explicitly stated in the formalization of adaptor signatures. We adopt this extension to aid our functional adaptor signature formalization.

- *Improved efficiency and privacy:* Adaptor signatures require only a single transaction and signature to be posted on the blockchain, reducing transaction costs and enhancing efficiency. Moreover, with buyers locally extracting secrets, transactions resemble regular payments, thereby improving the privacy and fungibility of coins in the system.

- *Enhanced compatibility:* Signature verification, a universally supported operation, ensures compatibility across all blockchain systems, making adaptor signatures-based digital sales compatible with the broader Web3 landscape.

**Looking ahead: Granular and functional sale of digital information.** However, while adaptor signatures offer a promising solution for digital transactions, they come with certain limitations. They operate on an all-or-nothing basis, revealing the entire secret upon successful transactions or learning nothing in case the seller aborts. This lack of granularity contrasts with the concept of *functional encryption (FE)* [BSW11], where recipients can decrypt and learn specific functions applied to the encrypted message instead of the complete message. Exploring granularity not only enables more nuanced and functional sales of digital information but also ensures the privacy of the seller's data. For instance, the seller holding medical records might desire to safeguard specific functions of the records, adhering to medical data privacy regulations like the Health Insurance Portability and Accountability Act (HIPAA) in the United States. The privacy requirement here is that the buyer learns only the specific function for which the payment is made, ensuring compliance with legal frameworks. Below we expand on this intuition and give two illustrative examples of real-world scenarios where functional sales of digital information can be critical.

**Application 1: Medical information.** In this setting the seller holds a medical database and is authorized to sell functions of the database by the patients or appropriate authorities. For instance, the buyer who might be an insurance firm, can query some aggregate/statistics of the demography in the database. If it's a research organization, more complex functions like correlations of pre-existing medical conditions and cancer can be queried. The seller has the financial incentive to provide the information, which, in the long run, can contribute greatly to medical advancements.

**Application 2: ML model training information.** In this case, the seller owns a dataset, and the buyer, with an ML model, seeks to train the model on the seller's dataset. The buyer presents the model to the seller, who then trains the model over the dataset and returns the result for a price. Training may involve simple similarity measures like computing the kernel of two vectors or more complex functions with large datasets (e.g., regression analysis, clustering).

Therefore, we ask the following question,

*Can we facilitate the functional sale of digital information using adaptor signatures?*

More concretely, let the seller hold secret data $x$, and the buyer wants access to a function $f$. *Can we expand the functionality of adaptor signatures such that, at the end of the exchange, the seller obtains a payment transaction and a signature from the buyer, and the buyer learns $f(x)$, and not the entirety of $x$?*

While general functions are the ultimate goal, this work focuses on the restricted function class of linear functions (where $f$ is a linear function), arguing that they are already widely applicable. We show that practical constructions using lightweight cryptographic tools can be achieved for linear functions, with some insights into the challenges of achieving a more expressive class of functions.

## 1.1 Our Contributions

Below we summarize the contributions of this work.

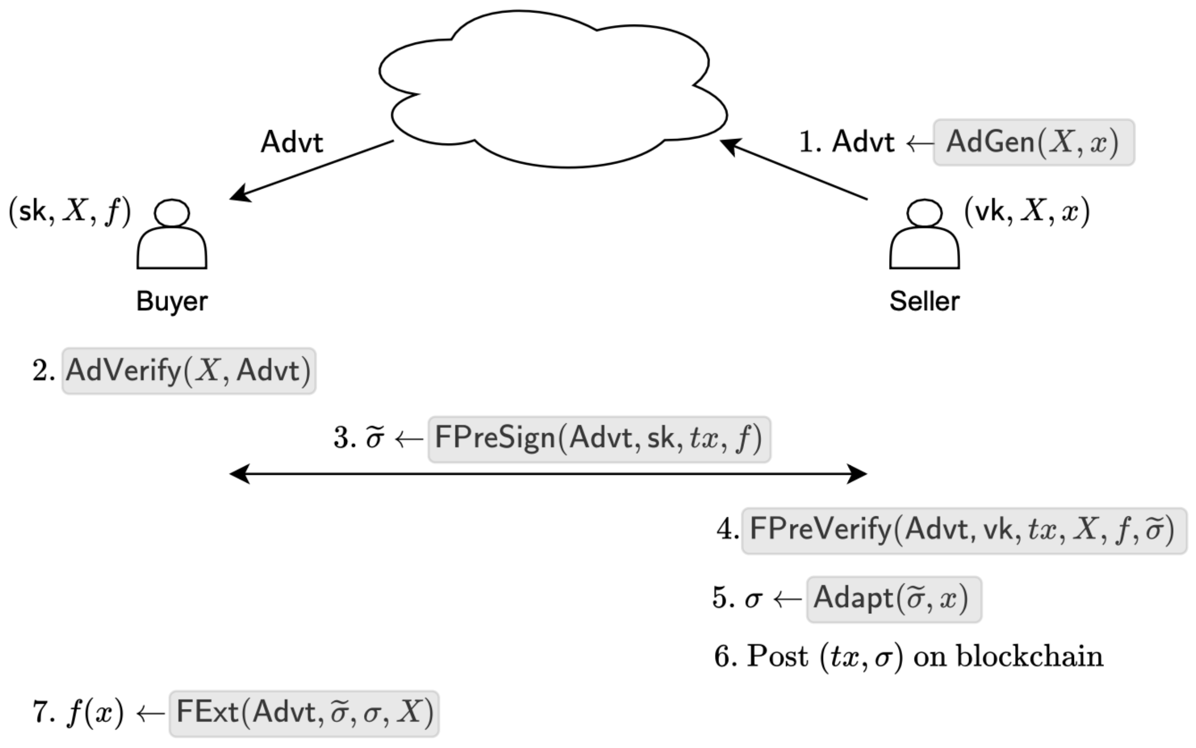
Advt
1. Advt ← AdGen$(X, x)$
$(\mathsf{sk}, X, f)$
$(\mathsf{vk}, X, x)$
Buyer
Seller
2. AdVerify$(X, \mathsf{Advt})$
3. $\widetilde{\sigma}$ ← FPreSign$(\mathsf{Advt}, \mathsf{sk}, tx, f)$
4. FPreVerify$(\mathsf{Advt}, \mathsf{vk}, tx, X, f, \widetilde{\sigma})$
5. $\sigma$ ← Adapt$(\widetilde{\sigma}, x)$
6. Post $(tx, \sigma)$ on blockchain
7. $f(x)$ ← FExt$(\mathsf{Advt}, \widetilde{\sigma}, \sigma, X)$

**Figure 1:** Functional payments via simplified FAS interface.

**New cryptographic primitive - Functional Adaptor Signatures.** To answer the above question affirmatively, we formally introduce a novel cryptographic primitive called *functional adaptor signatures (FAS)* (in Section 4). It is similar in functionality to standard adaptor signatures, except that it is additionally parameterized by a family of functions $\mathcal{F}$. This addition introduces new interfaces to FAS; we describe these below in the context of a digital trade for ease of understanding. A pictorial description is given in Figure 1.

The seller of the trade generates an advertisement $\mathsf{advt}$ using the FAS interface $\mathsf{AdGen}$, that embeds the statement $X$ and the corresponding secret witness $x$, where $(X, x) \in R$ for some NP relation $R$. Anyone can publicly verify the well-formedness of $\mathsf{advt}$ using the $\mathsf{AdVerify}$ algorithm. The buyer executes the functional pre-sign algorithm $\mathsf{FPreSign}$ to generate a pre-signature $\widetilde{\sigma}$ on the message $m$ (the payment transaction, denoted as $tx$ in Figure 1) for the statement $X$ and a function $f$. $\mathsf{FPreVerify}$ algorithm helps the seller to verify if $\widetilde{\sigma}$ is valid with respect to the message $m$, statement $X$ and function $f$. If the pre-signature is valid, the seller using the $\mathsf{Adapt}$ algorithm and the witness $x$ can adapt $\widetilde{\sigma}$ into a signature $\sigma$ on the message $m$. The seller may now post the transaction and the signature on the blockchain to get paid. Finally, the buyer can efficiently extract the value $y = f(x)$ using the functional extract algorithm $\mathsf{FExt}$ given access to $\mathsf{advt}$, $\widetilde{\sigma}$, and $\sigma$. With the above outline of FAS, we can see that we gain the same advantages of adaptor signatures over smart contracts for digital trading, while also achieving the desired granularity.

**Definitional framework.** Recent works [DOY22, GSST24] have highlighted the inherent complexities in designing the security framework for even standard adaptor signatures, with many subtle nuances that require a lot of care. In this work, we overcome these challenges and present a comprehensive definitional framework for the security of FAS (in Section 4). We establish the formal foundations of FAS, constituting a substantial contribution of this work.

Broadly speaking, for a secure FAS, we are required to upgrade the security definitions of standard adaptor signatures to handle additional constraints posed by the functional aspects of the primitive. Moreover, we introduce additional novel security properties for a secure FAS to satisfy, among which *witness privacy* is a prominent one. This property captures the leakage a malicious buyer learns about $x$ beyond $f(x)$. To formalize this, we present three different notions of witness privacy for FAS, akin to the different privacy guarantees of cryptographic proofs. To be more specific, we present (1) *witness hiding*, where the malicious buyer after learning $f(x)$ cannot output the secret witness $x$, (2) *witness indistinguishable*, where the malicious buyer learns $f(x)$ and cannot distinguish between two witnesses $x = x_0$ and $x = x_1$ (for the statement $X$) as long as $f(x_0) = f(x_1)$, and (3) *zero-knowledge*, where the malicious buyer learns no other information about $x$, other than $f(x)$.

**Efficient constructions.** From a practical perspective, we give several efficient constructions of FAS for the class of linear functions (like aggregates, statistics, kernel computation in ML, etc.) that are compatible with standard signatures like Schnorr and ECDSA. Notably, these signature schemes are widely used in blockchain systems for transaction verification, thus making our constructions ready for deployment in current systems. Towards a post-quantum instantiation of FAS, when the buyer has linearly independent function queries, we also present a FAS construction compatible with a variant of the lattice-based signature scheme of Lyubashevsky [Lyu12]. Later in Section 2.4, we discuss how the linear-independency requirement can be relaxed with minor trade-offs in efficiency.

Most importantly, the key conceptual contribution of our work is the surprising connection between two seemingly unrelated cryptographic tools, namely, functional encryption [BSW11] that has been extensively studied for over a decade and a relatively new primitive, adaptor signatures. Concretely, we construct FAS for a family of inner-product functions $\mathcal{F}_{\mathsf{IP}}$ using three lightweight building blocks:

(i) a functional encryption scheme $\mathsf{IPFE}$ for $\mathcal{F}_{\mathsf{IP}}$,

(ii) an adaptor signature scheme $\mathsf{AS}$ w.r.t. the digital signature scheme $\mathsf{DS}$ and some hard relation $R_{\mathsf{IPFE}}$ (related to $\mathsf{IPFE}$),

(iii) NIZK argument system for NP.

A nice feature of our construction is that we show it satisfies the zero-knowledge style witness privacy assuming only selective, IND-security of $\mathsf{IPFE}$ and zero-knowledge of $\mathsf{NIZK}$.

We show two ways to instantiate the construction:

- **Prime-order groups (Section 6).** Using a varaint of the DDH-based IPFE scheme by Abdalla et al. [ABDP15], we get $R_{\mathsf{IPFE}}$ is the discrete log relation. So, we use Schnorr adaptor signature [AEE$^+$21] for $\mathsf{AS}$.

- **Lattices (Section 7).** Using a variant of the LWE-based IPFE scheme by Agrawal et al. [ALS16], we get $R_{\mathsf{IPFE}}$ is the short integer solution (SIS) relation. So, we use a variant of the post-quantum adaptor signature [EEE20a] for $\mathsf{AS}$.

Throughout the paper, for conceptual clarity, we abstract out the relation $R$ to be any NP relation, and make black-box use of a NIZK [PS19] for relation $R$. However, we emphasize that in both cases, the NIZK can be efficiently instantiated depending on the exact application (i.e., for a concrete relation $R$) and plugged in without affecting other components. Lastly, for a weaker witness privacy notion of witness-indistinguishability, we present a round-optimal construction (Section 2.5 and Appendix B).

**Performance evaluation.** To assess the practicality of our FAS, we provide an open-source implementation [VST24] of our prime-order group-based instantiation satisfying zero-knowledge style witness privacy in Python. The details of our performance evaluation are given in Section 8. We also perform a series of benchmarks on our FAS scheme for a wide range of parameter settings. Our results show that our scheme is practical for a wide variety of real-world scenarios. For instance, on a MacBook Pro, for 300KB size witness, the time taken for pre-signing is 0.344 seconds, pre-verification is 0.424 seconds, adapt is 0.035 seconds, functional-extract is 1.025 seconds. We share more details of our findings in Section 8.

**Applicability of Linear Functions.** Linear functions although restricted, allow us to capture many scenarios including learning statistical information (like mean, average, weighted linear functions, select entries) of records in a medical database, employee records of an organization, weather data, or ecological records of endangered species. Linear functions also allow a buyer to measure the proximity of a vector he holds with the secret vector of the seller. Such proximity measures are quite fundamental in Machine Learning where they are referred to as computing the kernel of two vectors or graphs in the case of [ACR17].

## 2 Technical Overview

In this overview, we present our approach to fair payments for learning linear functions of a seller's witness.

To describe our techniques, we use the following setup: let $p$ and $\ell$ be integers where $p$ is prime and $L$ be an NP language containing statements $X$ with witness $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_p^\ell$. Our goal is to support linear functions over $\mathbf{x}$. Throughout this paper, we represent such linear functions by vectors $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}_p^\ell$ and the corresponding evaluation on $\mathbf{x}$ by the inner-product of the vectors $\mathbf{x}$ and $\mathbf{y}$ modulo $p$. That is, $f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{\ell} x_i \cdot y_i \mod p$. [3]

### 2.1 Functional Adaptor Signatures for Fair Functional Payments

We have a fair exchange scenario where the seller has a witness $\mathbf{x}$ which is a vector in $\mathbb{Z}_p^\ell$, and only desires to sell some linear functions $\mathbf{y}$ on input $\mathbf{x}$. To enable such an exchange FAS, like adaptor signatures, involves algorithms like $\mathsf{FPreSign}$ and $\mathsf{FPreVerify}$, $\mathsf{Adapt}$, and $\mathsf{FExt}$. However, the key distinction lies in the fact that $\mathsf{FPreSign}$ and $\mathsf{FPreVerify}$ are defined w.r.t. a function $\mathbf{y}$ from the buyer, while the functional extraction algorithm $\mathsf{FExt}$, returns $f_{\mathbf{y}}(\mathbf{x})$ instead of $\mathbf{x}$ itself, ensuring the buyer learns only the function's result.

#### 2.1.1 Security of FAS

We consider two cases, where the adversary can corrupt either the seller or the buyer. Similar to standard adaptor signatures, when the seller is corrupt, FAS must mainly guarantee (1) *unforgeability*, where an adversary who doesn't know the witness cannot forge honest buyers' signatures, and (2) *witness extractability*, where the adversary cannot publish a buyer's signature such that when the buyer tries to extract using $\mathsf{FExt}$, it gets $z' \neq f_{\mathbf{y}}(\mathbf{x})$. [4] Crucially, in the case of FAS, the adversary is allowed to get many functional pre-signatures for any choice of function $\mathbf{y}$.

[3] While our lattice-based instantiation is for computing inner products $\mod p$, the group-based instantiation is for computing integer inner products with bounded outputs, i.e., $f_{\mathbf{y}}(\mathbf{x}) \in \{0, \ldots, B\}$ for some apriori fixed bound $B \ll p$.

[4] We also require *advertisement soundness* and *pre-signature validity* properties against a corrupt seller, and defer their discussion to Section 4.

On the other hand, when the buyer is corrupt, similar to standard adaptor signatures, FAS must guarantee *pre-signature adaptability*, where if the adversary outputs a valid functional pre-signature, then adapting it using a valid witness $\mathbf{x}$ should result in a valid signature under the buyer's key. Most importantly, FAS must guarantee protection of the seller's witness $\mathbf{x}$ which we capture in the notion called *witness privacy.*

In more detail, FAS satisfies *zero-knowledge* witness privacy if a malicious buyer learns no more information about $\mathbf{x}$ than $f_{\mathbf{y}}(\mathbf{x})$ after the interaction. To formally define this in Section 4, we carefully borrow formalism from the related privacy notion of zero-knowledge for cryptographic proof systems [GMR89]. Informally, for every malicious buyer (that can sample its signing key), we require the existence of an efficient p.p.t. simulator $\mathsf{Sim}$ that can simulate the buyer's view only using $f_{\mathbf{y}}(\mathbf{x})$. Philosophically, the existence of such an efficient simulator ensures that whatever the buyer learned from interacting with the seller, it could have on its own from $f_{\mathbf{y}}(\mathbf{x})$ by running in polynomial time. While our notion shares similarities and is inspired by cryptographic proof systems, we need additional care to correctly model the fact that the buyer is allowed to ask for multiple functions which, moreover, can be adaptively chosen. To gain a comprehensive understanding of witness privacy, we also introduce two natural relaxed variants referred to as *witness hiding* and *witness indistinguishability* in Definitions 4.11 and 4.12, and study their relation with zero-knowledge witness privacy . Interestingly, in Section 2.5, we show that the relaxation to witness indistinguishability enables a round optimal construction.

For the rest of this overview, we will focus on building an efficient FAS centered around the witness privacy guarantee given it is the unique property of FAS compared to standard adaptor signatures. Moreover, our focus will be on the strongest notion of zero-knowledge witness privacy.

### 2.1.2 Strawman Construction of FAS

To build FAS, one can combine Schnorr adaptor signature (discussed above) along with a general-purpose non-interactive zero-knowledge (NIZK) proof. We present this strawman solution below and highlight a key efficiency challenge that serves as the starting point of our construction.

Recall the protocol flow in Figure 1. We describe how $\mathsf{advt}, \widetilde{\sigma}, \sigma$ are computed and the functional extraction process. The seller publishes $\mathsf{advt} = (\mathsf{ct}, \pi)$, where $\mathsf{ct}$ is a semantically-secure public-key encryption of the witness $\mathbf{x}$ and NIZK proof $\pi$ that certifies $\mathsf{ct}$ encrypts a witness for the statement $X$. The pre-signature $\widetilde{\sigma}$ is computed interactively among the buyer and the seller as follows. (i) The buyer sends $f$ to the seller. (ii) The seller computes a secret-key encryption $\mathsf{ct}_z$ of the requested evaluation $z = f(\mathbf{x})$ using a fresh secret-key $k$. Then, the seller encodes $k$ in $\mathbb{Z}_p$ and *engages with the buyer in an adaptor execution to sell the witness for the discrete logarithm of group element $K$, where $K = g^k$ for some generator $g$ of the group* $\mathbb{G}$. That is, the seller sends $(\mathsf{ct}_z, K, \pi_z)$ to the buyer, where NIZK proof $\pi_z$ certifies $\mathsf{ct}_z$ encrypts $f(\mathbf{x})$ using the key $k$ where $\mathbf{x}$ is encrypted in $\mathsf{ct}$. The buyer treats $K$ as the Schnorr adaptor statement and computes a pre-signature $\widetilde{\sigma}$ on the transaction message $m$ if and only if the NIZK proof $\pi_z$ verifies. On obtaining $\widetilde{\sigma}$, the seller adapts it into a valid signature $\sigma$ on $m$ using the key $k$ as the witness. For functional extraction, the buyer runs $\mathsf{Ext}$ algorithm of the Schnorr adaptor signature to extract $k$ from $(\widetilde{\sigma}, \sigma)$ and then decrypts $\mathsf{ct}_z$ using it to recover the function evaluation $z$.

The desired privacy of FAS is attained by relying on the zero-knowledge property of both NIZK proofs and the semantic security of $\mathsf{ct}$. That is, the seller side of interaction can be efficiently simulated just from the evaluation $z = f(\mathbf{x})$. The security against a malicious seller relies on the security of the underlying adaptor signature scheme and the soundness of the NIZK protocol.

While this approach enables fair functional payments, it requires that the seller compute NIZK proofs for every $\mathsf{FPreSign}$ interaction with a buyer. The seller's computation in computing these proofs (and their size) grows *polynomially* in the witness size $\mathbf{x}$ and is proportional to the complexity of the function $f$. While the proof sizes can be reduced by relying on general-purpose ZK-SNARKs [BCCT13], the proving cost growing super-linearly in $|\mathbf{x}|$ will become prohibitively expensive even for moderately sized databases $\mathbf{x}$. This cost also affects the scalability of the solution at the application layer. For instance, if the seller is dealing with several thousand buyers each with a different function, the seller will be computationally strained, leading to delays and a potential Denial of Service (DoS) attack vector via malicious buyers. A more efficient solution would allow (a) the seller's computation to grow linearly in the size of $\mathbf{x}$ and (b) the seller's communication to be proportional to the size of the evaluation $z$ which would be significantly smaller than $\mathbf{x}$ itself. As we explain below, we achieve these efficiency targets without relying on NIZKs, which ensures we use underlying cryptographic primitives in a *black-box* manner.

## 2.2 Our Techniques: Functional Encryption + Adaptor Signatures

The blueprint of our FAS construction is to combine adaptor signatures with another cryptographic tool called functional encryption (FE). FE was introduced as an enrichment of any (public-key) encryption scheme that allows for fine-grained decryption. In particular, in addition to algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ which are typical to any encryption scheme, a FE scheme for a function class $\mathcal{F}$ provides an additional functional key-generation algorithm $\mathsf{KGen}$. The algorithm takes as input the master secret-key $\mathsf{msk}$ and function $f \in \mathcal{F}$, and outputs a functional secret-key $\mathsf{sk}_f$ such that decrypting any encryption of $\mathbf{x}$ (w.r.t. $\mathsf{msk}$) with $\mathsf{sk}_f$ reveals $f(\mathbf{x})$ and no more. As discussed earlier, this notion of strong privacy where no information about $\mathbf{x}$ is leaked other than $f(\mathbf{x})$ is formalized using the *simulation* paradigm. For simplicity, we will henceforth refer to FE schemes satisfying this privacy property as *simulation-secure*. Numerous works have built FE schemes for rich classes of functions from a variety of hardness assumptions. More specifically, for the case of linear functions, we know of group-based schemes for prime-order groups [ABDP15, ALS16, Wee16, ALMT20] as well as for unknown-order groups [ALS16, ALMT20]; and post-quantum constructions [ABDP15, ALS16, ALMT20] are known from the hardness of the LWE problem.

**Our construction template in more detail.** In the quest of removing NIZKs from $\mathsf{FPreSign}$, we modify the above steps of the fair exchange as follows: (a) The $\mathsf{AdGen}$ remains identical except that $\mathsf{ct}$ is computed using the FE scheme's encryption algorithm, (b) In the $\mathsf{FPreSign}$ interaction, the seller instead computes the functional secret-key $\mathsf{sk}_f$ and engages with the buyer in an adaptor execution to sell $\mathsf{sk}_f$, (c) the $\mathsf{Adapt}$ algorithm remains the same, and (d) the $\mathsf{FExt}$ algorithm extracts the functional secret-key $\mathsf{sk}_f$ and decrypts $\mathsf{ct}$ directly to return $f(\mathbf{x})$. If the FE scheme satisfies simulation-security and the NIZK (in the advertisement) is zero-knowledge, then seller's interaction can be simulated only via $f(\mathbf{x})$, which ensures the desired seller privacy.

An astute reader might ask why to use an encryption of $\mathbf{x}$ in the advertisement and not a one-way function of $\mathbf{x}$, similar to how statement $X$ and witness $x$ have a discrete log relation in standard adaptor signature constructions. A crucial advantage of our construction template is that since $\mathbf{x}$ is encrypted under semantically secure encryption, we can let witness $\mathbf{x}$ be of low-entropy like user databases, which are the main applications we are targeting for FAS. A similar setting would be insecure when using one-way functions instead of encryption since $\mathbf{x}$ is not a high-entropy witness. Moreover, achieving zero-knowledge witness privacy seems hopeless when using one-way functions for setting up $X$.

**Challenges.** Unfortunately the above approach doesn't quite work as-is and runs into two main challenges:

1. *Correctness of the Adaptor Statement*: Suppose that one could encode $\mathsf{sk}_f$ as an integer $x_f$ in $\mathbb{Z}_p$

(for an appropriate $p$), and then use the Schnorr adaptor signature for the statement $X_f = g^{x_f}$. Still, for fairness, the buyer needs to gain confidence in the validity of $X_f$ (e.g., learning the incorrect functional key $\tilde{x}$ will disallow the buyer from learning the correct evaluation) during $\mathsf{FPreSign}$. Augmenting the seller's message with a NIZK proof that attests to the correctness of $X_f$ (i.e., discrete-logarithm of $X_f$ is indeed the correct $\mathsf{sk}_f$) would be sufficient. But the use of NIZKs was exactly what we were trying to avoid, and hence it seems we are back to square one.

2. *Compatibility with Adaptor Signatures*: Recall that known adaptor signature schemes only facilitate the sale of witnesses for special algebraic languages. For the above approach to work, we need a *simulation-secure* FE scheme where the functional key-generation algorithm exhibits the required structural compatibility. For e.g., to rely on Schnorr adaptor signatures that allow selling discrete-logarithms $x \in \mathbb{Z}_p$ of prime-order group elements $X = g^x$, we need a FE scheme where the functional secret-key $\mathsf{sk}_f$ w.r.t. any linear function $f$ is such that $\mathsf{sk}_f \in \mathbb{Z}_p$. In fact, [ABDP15] presents such an IPFE scheme, but the scheme satisfies only a weaker notion of IND-security (and not simulation security) which is insufficient for the template to work.

**Our Technique 1: Augmenting IPFE with $\mathsf{PubKGen}$ algorithm.** To avoid using NIZKs during $\mathsf{FPreSign}$, we observe that the IPFE schemes for computing inner product of vectors of length $\ell$ are obtained by starting with $\ell$ instances of some PKE scheme and a functional secret key is essentially a linear combination of the secret keys of the $\ell$ PKE instances (as pointed out in [ABDP15]). Combining this observation with the fact that the public key and secret key of the underlying PKE satisfy some algebraic relation, one can envision that a similar linear combination of the public keys of the PKE schemes might give us a commitment to the functional secret key that can be directly used as the statement $X_f$. Crucially, if $X_f$ can be deterministically computed using $\mathsf{IPFE}$'s master public key and the function description $f$, then such an algorithm would be public and the buyer can compute $X_f$ without any interaction with the seller, thus avoiding NIZKs during $\mathsf{FPreSign}$ altogether. We formalize this vision by augmenting $\mathsf{IPFE}$ with a public deterministic algorithm $\mathsf{PubKGen}$. The requirement that the output of $\mathsf{PubKGen}$, i.e., $X_f$ must be a commitment to the output of $\mathsf{KGen}$, i.e., $\mathsf{sk}_f$, is formalized via a *compliance property* (Definition 3.13). Informally, $R_{\mathsf{IPFE}}$-compliance says that for a given relation $R_{\mathsf{IPFE}}$, it must be the case that $(X_f, \mathsf{sk}_f) \in R_{\mathsf{IPFE}}$.

For instance, if we set the relation $R_{\mathsf{IPFE}}$ to be the discrete-log relation, then the key structure becomes compatible with the hard relation of Schnorr adaptor signatures. As an example, the IND-secure IPFE scheme of [ABDP15] can be shown to have such a key structure. In the IND-secure IPFE scheme of [ABDP15], the master secret key is $\mathbf{s} = (s_1, \ldots, s_\ell) \in \mathbb{Z}_p^\ell$, the master public key is $g^{\mathbf{s}} = (g^{s_1}, \ldots, g^{s_\ell})$, the functional secret key for function $\mathbf{y} = (y_1, \ldots, y_\ell)$ is $\mathsf{sk}_{\mathbf{y}} = \sum_{i \in [\ell]} s_i y_i \bmod p$. Then, $\mathsf{PubKGen}$ can be defined to output $\mathsf{pk}_{\mathbf{y}} = \prod_{i \in [\ell]} (g^{s_i})^{y_i}$. Consequently, we can observe that $\mathsf{pk}_{\mathbf{y}} = g^{\mathsf{sk}_{\mathbf{y}}}$, thus $(\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}})$ satisfy the discrete log relation. Defining $\mathsf{PubKGen}$ in this way makes it compliant with the Schnorr adaptor signatures: if the buyer wants to learn function $\mathbf{y}$ evaluated on the seller's witness, he can locally compute $\mathsf{pk}_{\mathbf{y}}$ and use it as the adaptor statement. Upon receiving a pre-signature from the buyer, the seller can locally compute $\mathsf{sk}_{\mathbf{y}}$ and use it to adapt the pre-signature. Eventually, the buyer can extract $\mathsf{sk}_{\mathbf{y}}$, thus enabling functional decryption. Crucially notice that because of deterministic nature of $\mathsf{PubKGen}$, the pre-signing becomes non-interactive. But as noted earlier, IND-secure IPFE isn't sufficient for the FAS template to work, so we are not done yet.

**Our Technique 2: Simulation-Secure Compatible IPFE.** To resolve the compatibility with known adaptor signatures, we rely on the IND-secure IPFE to simulation-secure IPFE compiler of [ALMT20]. More specifically, their compiler lifts any IND-secure IPFE to achieve simulation-security without changing the structure of the functional secret keys. This is achieved by increasing the length of function vectors from $\ell$ to $2\ell$, where the first $\ell$ slots encode the function as before, and the extra $\ell$ slots encode some random coins. These random coins are used by the IPFE simulator to argue simulation security. To preserve correctness, the length of the message vector to be encrypted is also increased from $\ell$ to $2\ell$, where the first $\ell$ slots encode the message as before, and the extra $\ell$ slots are set to zero. A small caveat is that the compiler results in a stateful functional secret key generation algorithm. We make it stateless which allows us to make an optimization where we increase the vector lengths from $\ell$ to only $\ell + 1$. Instantiating their compiler with the DDH-based IND-secure IPFE scheme of [ABDP15] results in a simulation-secure FE that is compatible with Schnorr adaptor signatures. We also extend this to the post-quantum setting by using a simplified variant of the LWE-based simulation-secure IPFE scheme of [ALMT20] [5] that is compatible with the Dilithium adaptor signature [EEE20b] instantiated on unstructured lattices.

While the above technique addresses the compatibility w.r.t. known adaptor signatures, proving *zero-knowledge* witness privacy requires more care and we will introduce another technique for it. Let us first explain the challenge with proving *zero-knowledge*. Even when starting with a base $\mathsf{IPFE}$ scheme (that is IND-secure) with a deterministic functional key-generation algorithm, the IPFE compiler introduces randomness in the functional key-generation algorithm. [6] This randomization is crucial for the IPFE simulator's correctness and essential to work with the base FE scheme that is only IND-secure. In the context of FAS, during $\mathsf{FPreSign}$, if the seller could share the randomness used as part of the generating $\mathsf{sk}_f$, then the buyer can mimic the same computation using the seller's randomness and determine the validity of the $\mathsf{sk}_f$ embedded in $X_f$. While this approach makes pre-signing interactive yet it would certainly avoid the use of general-purpose NIZKs as seen in the template above because the check is *canonical* now. Unfortunately, exposing the seller's randomness would compromise simulation security of the compiled IPFE scheme as the IPFE simulator would then get stuck in the analysis. Consequently, we can't show *zero-knowledge* of FAS. Note that an apporach to determine validity of $X_f$ without having the seller share the random coins could be to use NIZKs, but as said before, we want to avoid it.

**Our Technique 3: Interactive pre-signing without NIZKs.** To address the *zero-knowledge* of FAS, we open up the IPFE compiler of [ALMT20], that is, make non-black-box use of the IPFE compiler and reveal only a part of the randomness of the functional key-generation algorithm. By carefully choosing the randomness revealed by the seller, we can guarantee the validity of the adaptor statement $X_f$ to the buyer (without expensive NIZKs) and also allow flexibility to the IPFE simulator to successfully finish the simulation (See Remarks 2.1 and 2.2 for more details).

We emphasize that the non-black-box use of the techniques from [ALMT20] is a significant technical part of this work. We elaborate on these aspects of our construction in more detail in Section 2.3.

[5]Making the functional secret key generation algorithm stateless restricts the functionality in the LWE setting. In particular, such IPFE can only handle linearly independent function queries. We show how to slightly modify our $\mathsf{FAS}$ construction to ensure that linearly dependent function queries from different buyers can be augmented with some extra slots that always guarantee linear independence of function queries to the underlying $\mathsf{IPFE}$. See Section 7.4 for more details.

[6]DDH-based IPFE scheme of [ALS16] has been shown to be simulation-secure in [ALMT20]. Interestingly, it does not introduce randomness in functional key-generation and for proving simulation-security, the simulator relies on the fact that there can be many master secret keys corresponding to a master public key. But unfortunately, it does not satisfy the IPFE compliance property as its master keys do not satisfy the discrete log relation. Hence, it is incompatible with Schnorr adaptor signatures. Despite many attempts, we could not make the two compatible.

## 2.3 Our Construction

Suppose that the relation $R$ is such that for any statement $X$ and witness $\mathbf{x}$ satisfying $(X, \mathbf{x}) \in R$, it is the case that $\mathbf{x} \in \mathbb{Z}_p^\ell$. Further, suppose that inner-product functions are of the form $\mathbf{y} \in \mathbb{Z}_p^\ell$. Then, our $\mathsf{FAS}$ construction is obtained by employing the abovementioned techniques. We remind the reader of the protocol flow in Figure 1.

- $\mathsf{Setup}$: Run by a trusted third party, it samples common reference string $\mathsf{crs}$ for NIZK and public parameters $\mathsf{pp}'$ for IPFE.
- $\mathsf{AdGen}$: The seller samples $(\mathsf{mpk}, \mathsf{msk})$ corresponding to IND-secure IPFE and random coins $\mathbf{t}$ used by the non-black-box IPFE compiler to upgrade from IND-secure IPFE to simulation-secure IPFE. It computes the elongated vector $\widetilde{\mathbf{x}}$ used by the IPFE compiler, encrypts $\widetilde{\mathbf{x}}$ to obtain $\mathsf{ct}$ and computes a NIZK proof $\pi$ certifying that $\mathsf{ct}$ encrypts a witness corresponding to $X$.
- $\mathsf{AdVerify}$: The buyer verifies the NIZK proof $\pi$.
- Interactive pre-signing: For the sake of notational simplicity, we denote the 3-round interactive functional pre-signing via three algorithms: $\mathsf{AuxGen}, \mathsf{AuxVerify}, \mathsf{FPreSign}$.
  - First round: the buyer sends the function $\mathbf{y}$ to the seller.
  - Second round: the seller runs the $\mathsf{AuxGen}$ algorithm and sends auxiliary value $\mathsf{aux}_\mathbf{y}$ along with a proof $\pi_\mathbf{y}$ validating authenticity of $\mathsf{aux}_\mathbf{y}$ to the buyer. Specifically, the seller uses the random coins $\mathbf{t}$ of the IPFE compiler to compute $f_\mathbf{y}(\mathbf{t})$. Then, it sets the elongated vector $\widetilde{\mathbf{y}} = (\mathbf{y}^T, f_\mathbf{y}(\mathbf{t}))^T$ and generates $\mathsf{pk}_\mathbf{y}$ for it. It sends $(\mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) := (\mathsf{pk}_\mathbf{y}, f_\mathbf{y}(\mathbf{t}))$.
  - Third round: the buyer verifies the auxiliary value via $\mathsf{AuxVerify}$ algorithm, i.e., it creates $\widetilde{\mathbf{y}} = (\mathbf{y}^T, \pi_\mathbf{y})^T$ and checks if $\mathsf{aux}_\mathbf{y}$ matches the output of $\mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$. If it verifies, then, it computes pre-signature $\widetilde{\sigma}$ for the adaptor statement $\mathsf{aux}_\mathbf{y}$ and sends it to the seller.
- $\mathsf{FPreVerify}$: The seller verfies that $\widetilde{\sigma}$ corresponds to $\mathbf{y}$, i.e., it verifies that $\widetilde{\sigma}$ corresponds to $\mathsf{aux}_\mathbf{y}$ and $\mathsf{aux}_\mathbf{y}$ corresponds to $\mathbf{y}$.
- $\mathsf{Adapt}$: The seller computes IPFE functional key $\mathsf{sk}_\mathbf{y}$ for function $\widetilde{\mathbf{y}}$. Since $\mathsf{sk}_\mathbf{y}$ is a witness to $\mathsf{pk}_\mathbf{y}$, it uses $\mathsf{sk}_\mathbf{y}$ to adapt $\widetilde{\sigma}$ into $\sigma$.
- $\mathsf{FExt}$: The buyer extracts the IPFE functional key $\mathsf{sk}_\mathbf{y}$ from $(\widetilde{\sigma}, \sigma)$ and uses it to decrypt $\mathsf{ct}$ and recover $f_\mathbf{y}(\mathbf{x})$.

Our formal construction is as in Figure 2. We defer the security theorem to Section 5. We alluded to in 'Our Technique 3' on how to avoid NIZKs in interactive pre-signing by making non-blackbox use of the IPFE compiler. Having described the full details of FAS construction, we now elucidate on it in Remarks 2.1 and 2.2.

**Remark 2.1** (Simulatability with leakage $f_\mathbf{y}(\mathbf{t})$ on $\mathbf{t}$)**.** *Note that while proving zero-knowledge of FAS, the FAS simulator would still have to reveal $f_\mathbf{y}(\mathbf{t})$. This helps us avoid NIZKs in the interactive pre-signing as discussed before. Also note that FAS simulator would use the IPFE simulator internally, and $f_\mathbf{y}(\mathbf{t})$ is a leakage on the random coins $\mathbf{t}$ of the IPFE (compiler's) simulator. Crucially, this does not break simulation security of IPFE. This is because the IPFE compiler of [ALMT20] — and hence the IPFE simulator too – inherently leaks $f_\mathbf{y}(\mathbf{t})$. The latter is because* $\mathsf{IPFE}$ *decryption implicitly takes $\widetilde{\mathbf{y}}$ as input and hence, the decrypter needs to know $f_\mathbf{y}(\mathbf{t})$ to ensure decryption correctness. Hence, from an honest seller's perspective, the* $\mathsf{FAS}$ *zero-knowledge simulation won't be affected by giving away $f_\mathbf{y}(\mathbf{t})$ as part of $\pi_\mathbf{y}$.*

```
$\mathsf{Setup}(1^\lambda)$
  1: Sample $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
  2: Sample $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
  3: ret $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$

$\mathsf{AdGen}(\mathsf{pp}, X, \mathbf{x})$:
  1: Sample random coins $r_0, r_1$
  2: Let $(\mathsf{mpk}, \mathsf{msk}) := \mathsf{IPFE.Setup}(\mathsf{pp}', 1^{\ell+1}; r_0)$
  3: Sample $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_p^\ell$, let $\widetilde{\mathbf{x}} := (\mathbf{x}^T, 0)^T \in \mathbb{Z}_p^{\ell+1}$
  4: Let $\mathsf{ct} := \mathsf{IPFE.Enc}(\mathsf{mpk}, \widetilde{\mathbf{x}}; r_1)$
  5: Let $\mathsf{stmt} := (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), \mathsf{wit} := (r_0, r_1, \mathbf{x})$
  6: Let $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$
  7: ret $\mathsf{advt} := (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} := (\mathsf{msk}, \mathbf{t})$

$\mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt})$
  1: ret $\mathsf{NIZK.Vf}(\mathsf{crs}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), \pi)$

$\mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, \mathbf{y})$
  1: Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} = (\mathsf{msk}, \mathbf{t})$
  2: Let $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_\mathbf{y}(\mathbf{t}))^T \in \mathbb{Z}_p^{\ell+1}$
  3: Let $\mathsf{pk}_\mathbf{y} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$
  4: ret $\mathsf{aux}_\mathbf{y} := \mathsf{pk}_\mathbf{y}$, $\pi_\mathbf{y} := f_\mathbf{y}(\mathbf{t})$

$\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y})$
  1: Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, let $\widetilde{\mathbf{y}} := (\mathbf{y}^T, \pi_\mathbf{y})^T$
  2: ret 1 iff $\mathsf{aux}_\mathbf{y} = \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$

$\mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_\mathbf{y})$
  1: ret $\widetilde{\sigma} \leftarrow \mathsf{AS.PreSign}(\mathsf{sk}, m, \mathsf{aux}_\mathbf{y})$

$\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}, \widetilde{\sigma})$
  1: ret $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) \wedge$
         $\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{aux}_\mathbf{y}, \widetilde{\sigma})$

$\mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, \mathbf{x}, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \widetilde{\sigma})$
  1: Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} = (\mathsf{msk}, \mathbf{t})$
  2: Let $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_\mathbf{y}(\mathbf{t}))^T$
  3: Let $\mathsf{sk}_\mathbf{y} := \mathsf{IPFE.KGen}(\mathsf{msk}, \widetilde{\mathbf{y}})$
  4: ret $\sigma := \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{aux}_\mathbf{y}, \mathsf{sk}_\mathbf{y}, \widetilde{\sigma})$

$\mathsf{FExt}(\mathsf{advt}, \widetilde{\sigma}, \sigma, X, \mathbf{y}, \mathsf{aux}_\mathbf{y})$
  1: Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$.
  2: Let $z := \mathsf{AS.Ext}(\widetilde{\sigma}, \sigma, \mathsf{aux}_\mathbf{y})$
  3: ret $v := \mathsf{IPFE.Dec}(z, \mathsf{ct})$
```

**Figure 2:** Construction: Functional Adaptor Signatures

**Remark 2.2** (Purpose of $\pi_\mathbf{y}$)**.** *Note that $\pi_\mathbf{y}$ enables verifying that $\mathsf{aux}_\mathbf{y}$ is functional public key corresponding to $\widetilde{\mathbf{y}} = (\mathbf{y}^T, \pi_\mathbf{y})^T$. Regarding $\widetilde{\mathbf{y}}$, the buyer knows $\mathbf{y}$ but it does not know whether the last slot of $\widetilde{\mathbf{y}}$ chosen by the seller is well-formed, i.e., is $\pi_\mathbf{y} = f_\mathbf{y}(\mathbf{t})$? In security against a malicious seller, we argue that we do not need to guarantee this. In particular, it does not affect an honest buyer whether $\pi_\mathbf{y} = f_\mathbf{y}(\mathbf{t})$ holds or not. Suppose that the malicous seller chooses arbitrary value* $\mathsf{val}$ *and sends* $(\mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) = (\mathsf{pk}_\mathbf{y}, \mathsf{val})$, *where* $\mathsf{pk}_\mathbf{y} = \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$ *and* $\widetilde{\mathbf{y}} = (\mathbf{y}^T, \mathsf{val})^T$. *This would certainly ensure that* $\mathsf{AuxVerify}$ *passes and the buyer continues the protocol with the seller. Crucially though, we argue that if the honest buyer does end up making the payment to the seller, then, the honest buyer indeed learns $f_\mathbf{y}(\mathbf{x})$ at the end of the protocol disregard of what* $\mathsf{val}$ *was chosen by the malicious seller. This is because* advertisement soundness *would guarantee the honest buyer that* $\mathsf{ct}$ *encrypts a vector $\widetilde{\mathbf{x}}$ of the form $\widetilde{\mathbf{x}} = (\mathbf{x}^T, 0)^T$, where $(X, \mathbf{x}) \in R$. Further, witness extractability would guarantee that the honest buyer extracts a functional secret key* $\mathsf{sk}_\mathbf{y}$ *corresponding to* $\widetilde{\mathbf{y}} = (\mathbf{y}^T, \mathsf{val})^T$. *Thus, IPFE correctness would guarantee that the honest buyer recovers $f_{\widetilde{\mathbf{y}}}(\widetilde{\mathbf{x}})$ which is same as $f_\mathbf{y}(\mathbf{x})$ since the last slot of $\widetilde{\mathbf{x}}$ is zero.*

## 2.4 Instantiations

In Section 6, we provide an instantiation from prime order groups for the function class of inner products over integers with output values polynomially bounded by $B \ll p$. For example, if each entry of the witness/function vector is bounded by $10^3$, and the vectors have $10^6$ entries, then, the inner product value would be bounded by $10^{12}$, so we can set $B = 10^{12}$. We explain below application scenarios where such restrictions are reasonable. We also present an instantiation from lattices in Section 7 that removes such restrictions. Specifically, it works for the function class of inner products modulo prime integer $p$. Further, it is post-quantum secure.

### 2.4.1 Prime-Order Groups based Instantiation.

To instantiate our FAS construction in Figure 2 from prime order groups, we instantiate the $\mathsf{AS}$ as Schnorr adaptor signature scheme [EFH+21] w.r.t. the hard relation $R_{\mathsf{DL}}$, where $R_{\mathsf{DL}}$ is the discrete-log relation in prime order groups (see Definition 3.3). We then set $R_{\mathsf{IPFE}} = R_{\mathsf{DL}}$ and show that a varaiant of the selective, IND-secure IPFE scheme by Abdalla et al. [ABDP15] satisfies $R_{\mathsf{DL}}$-compliance.

**Small plaintext space: reason and application scope.** This limitation is because the function output here is encoded in the exponent and functional extraction of FAS instantiation involves a discrete log computation step. For values in the exponent bounded by $B$, this incurs a running time of $O(\sqrt{B})$. This is efficient only if bound $B$ is a polynomial. We argue that this is sufficient for several realistic application scenarios. We benchmarked computation for datasets having a maximum of $10^6$ entries, resulting in a 32 MB dataset. For example, the breast cancer dataset on Kaggle [kag] has 20000 entries, total size 50 KB, sum of entries $\approx 10^6$. Assuming each entry of a function is at most $10^4$, we get $B = 10^{10}$. Thus, the benchmarks in Table 2 for $(\ell = 10^4, B = 10^{10})$ fit closest for this dataset. Other scenarios occur when the secret database is a company's employee record with their age, years of service with the employer, retirement contribution, etc. Such values while sensitive, are typically bounded (e.g., by $B = 10^{10}$) and the buyer may wish to learn statistical/aggregate information, like sum, weighted mean or average, etc. whose result is also in the bounded plaintext space. The buyer could be a research organization that is studying the workforce in companies or factories, or it could be a recruitment recommendation agency that tries to match clients with potential employers with appropriate aggregate requirements of the client.

### 2.4.2 Lattice based Instantiation.

We instantiate the $\mathsf{AS}$ as the lattice-based adaptor signature scheme by Esgin et al. [EEE20a] w.r.t. the inhomogenous short integer solution relation $R_{\mathsf{ISIS}}$ (see Definition 3.18). Our instantiation is over the ring of integers $\mathcal{R} = \mathbb{Z}$. Consequently, its security follows from plain SIS and LWE assumptions. Further, we show that a variant of the IND-secure IPFE scheme by Agrawal et al. [ALS16, Section 4.2] satisfies $R_{\mathsf{ISIS}}$-compliance. [7]

For technical reasons related to IPFE, the resulting FAS construction can only support linearly independent function request across all buyers (See Remark 7.2 for details). This could be quite restrictive since the scheme cannot support same function requests from different buyers. In Section 7.4, we show how to modify the FAS construction to remove this restriction.

## 2.5 Towards Non-Interactive Pre-Signing

One of the main features of adaptor signatures that enable scalability is the non-interactive nature of the pre-signature generation algorithm. More specifically, the buyer in an adaptor signature execution can generate a pre-signature just from the seller's advertisement and, more importantly,

[7]Note that in lattice-based adaptor signature scheme by Esgin et al. [EEE20a], the extracted witness does not satisfy $R_{\mathsf{ISIS}}$ but it satisfies an extended relation $R'_{\mathsf{ISIS}}$ such that $R_{\mathsf{ISIS}} \subset R'_{\mathsf{ISIS}}$. This is due to the rejection sampling step involved in these signatures. Using such signatures along with $\mathsf{IPFE}$ thus becomes technically more challenging as the extracted witness acts as the $\mathsf{IPFE}$ decryption key. So, we additionally require a decryption robustness property from $\mathsf{IPFE}$ (Definition 3.14).

without further interaction with the seller. Since the exchange of pre-signatures is done off-chain and hence the latency doesn't impact the blockchain eco-system, a non-interactive pre-signing phase is more preferable.

We relaxed pre-signing to be interactive to achieve strong witness privacy for FAS, namely, the malicious buyer learns no information about $\mathbf{x}$ other than $f(\mathbf{x})$. This relaxation is not new to our work and was already introduced in [QPM+23] to achieve the advanced feature of *blindness* for adaptor signatures. Despite numerous attempts, the interactive nature of pre-signing seems necessary for this notion of strong privacy. An immediate challenge towards non-interactive pre-signing would be to build a simulation-secure IPFE scheme where the buyer can itself generate $\mathsf{pk}_f \leftarrow \mathsf{IPFE.PubKGen}(\mathsf{mpk}, f)$ such that the seller can generate the corresponding functional secret-key $\mathsf{sk}_f \leftarrow \mathsf{IPFE.KGen}(\mathsf{msk}, f)$. Unfortunately, the randomized nature of the key-generation algorithm of simulation-secure scheme is a major hurdle to enabling this.

We explore whether a non-interactive pre-signing can be achieved for FAS for relaxed privacy definitions. Towards this, we study FAS that only satisfies *witness indistinguishability*. The construction of such a FAS follows from our construction template by replacing the simulation-secure IPFE with an appropriate IND-secure IPFE. To understand the intuition, we refer the reader to the IPFE compliance example discussed in our technique 1 in Section 2.2. We provide more details on this construction in Appendix B.

## 3 Preliminaries

We recall the cryptographic tools needed in this work.

**Notations.** We denote by $x \leftarrow S$ the experiment of sampling $x$ from a probability distribution $S$. We denote by $[A(\cdot)]$ the range of an algorithm $A(\cdot)$. If $p(\cdot,\cdot)$ denotes a predicate, then $\Pr[p(y,z) : x \leftarrow S, (y,z) \leftarrow A(x)]$ is the probability that the predicate $p(y,z)$ is true after the ordered sequence of events $x \leftarrow S$ followed by $(y,z) \leftarrow A(x)$. We denote scalars by lower-case alphabets such as $x$, vectors by bold-face lower-case alphabets such as $\mathbf{x}$, matrices by bold-face upper-case alphabets such as $\mathbf{X}$. $\mathbf{x} = (x_1, x_2)^T$ denotes a column-vector with elements $x_1$ and $x_2$. $\mathbf{x}^T$ denotes a row-vector. Suppose $\mathbb{G}$ is a cyclic group of prime-order $p$ and with generator $g$. For a scalar $x \in \mathbb{Z}_p$, $g^x$ denotes its group encoding. For a vector $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{Z}_p^n$, $g^{\mathbf{x}}$ denotes its group encoding $(g^{x_1}, \ldots, g^{x_n})^T$. Similarly, for a matrix $\mathbf{X}$, $g^{\mathbf{X}}$ denotes its group encoding. Given group encoding of a vector $g^{\mathbf{x}}$ and a vector $\mathbf{y}$, we can compute $g^{\mathbf{x}^T\mathbf{y}}$ efficiently as $g^{\mathbf{x}^T\mathbf{y}} = \Pi_{i\in[n]}(g^{x_i})^{y_i}$.

**Linear Functions.** We denote linear functions as inner product computation. Let $\mathcal{F}_{\mathsf{IP},\ell}$ denote the family of inner products of vectors of length $\ell$. For a prime $p$, in this work we study two restrictions of inner product computations as follows:

- *Inner products modulo $p$.* The function class $\mathcal{F}_{\mathsf{IP},\ell,p} = \{f_{\mathbf{y}} : \mathbf{y} \in \mathbb{Z}_p^\ell\}$, where $f_{\mathbf{y}} : \mathbb{Z}_p^\ell \to \mathbb{Z}_p$ is defined as $f_{\mathbf{y}}(\mathbf{x}) = \mathbf{x}^T\mathbf{y} \bmod p$.
- *Inner products with output values polynomially bounded by $B \ll p$.* The function class $\mathcal{F}_{\mathsf{IP},\ell,p,B} = \{f_{\mathbf{y}} : \mathbf{y} \in \mathbb{Z}_p^\ell\}$, where $f_{\mathbf{y}} : \mathbb{Z}_p^\ell \to \{0,\ldots,B\}$ is defined as $f_{\mathbf{y}}(\mathbf{x}) = \mathbf{x}^T\mathbf{y} \in \{0,\ldots,B\}$.

Often we will use $\mathbf{y}$ instead of $f_{\mathbf{y}}$ to denote the function.

**Digital Signature.** A digital signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ has a key generation algorithm $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$ that outputs a verification-signing key pair. Using signing key $\mathsf{sk}$ we can compute signatures on a message $m$ by running $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, which can be publicly verified using the corresponding verification key $\mathsf{vk}$ by running $\mathsf{Vf}(\mathsf{vk}, m, \sigma)$. We require the digital signature scheme to satisfy strong existential unforgeability [GMR88].

## 3.1 Hard Relation

We recall the notion of a hard relation $R$ with statement/witness pairs $(X, x)$. We denote by $L_R$ the associated language defined as $L_R := \{ X \mid \exists\, x, (X, x) \in R \}$.

**Definition 3.1** (Hardness of $R$). *We say that a relation $R$ is hard, if (i) there exists a p.p.t. sampling algorithm $\mathsf{GenR}(1^\lambda)$ that outputs a statement/witness pair $(X, x) \in R$, (ii) the relation is poly-time decidable, (iii) for every non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{G}_{\mathcal{A},R}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda) \ , \tag{1}$$

*where the game $\mathsf{G}_{\mathcal{A},R}(1^\lambda)$ is defined as follows:*

```
Game $\mathsf{G}_{\mathcal{A},R}(1^\lambda)$
1 :  $(X, x) \leftarrow \mathsf{GenR}(1^\lambda)$
2 :  $(x') \leftarrow \mathcal{A}(1^\lambda, X)$
3 :  If $(X, x') \in R$: ret 1
4 :  ret 0
```

**Definition 3.2** (Hardness of $R$ w.r.t. function class $\mathcal{F}$). *We say that a relation $R$ is $\mathcal{F}$-hard, if (i) there exists a p.p.t. sampling algorithm $\mathsf{GenR}(1^\lambda)$ that outputs a statement/witness pair $(X, x) \in R$, (ii) for every non-uniform PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{G}_{\mathcal{A},R,\mathcal{F}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda) \ , \tag{2}$$

*where the game $\mathsf{G}_{\mathcal{A},R,\mathcal{F}}(1^\lambda)$ is defined as follows:*

```
Game $\mathsf{G}_{\mathcal{A},R,\mathcal{F}}(1^\lambda)$
1 :  $(X, x) \leftarrow \mathsf{GenR}(1^\lambda)$
2 :  $(f, z) \leftarrow \mathcal{A}(1^\lambda, X)$
3 :  If $(f \in \mathcal{F}) \wedge (z \in \{f(x') : \exists\, x'$ such that $(X, x') \in R\})$: ret 1
4 :  ret 0
```

We note that in the above definition we allow the adversary $\mathcal{A}$ to choose $f$ adaptively as this hardness assumption will later be used in proving unforgeability of our functional adaptor signatures construction where the adversary is allowed to choose the challenge function adaptively. This is necessary as functional adaptor signatures are non-trivial to build only in such a setting and become trivial if the challenge function is selectively chosen upfront by the adversary. Further note that we let the adversary $\mathcal{A}$ win even if $\mathcal{A}$ outputs a function evaluation of a witness $x'$ of $X$ that is different from $x$.

**Definition 3.3** (Hard Relation $R_{\mathsf{DL}}$). *The discrete log language $L_{\mathsf{DL}}$ is defined with respect to a group $G$ with generator $g$ and order $p$ as $L_{\mathsf{DL}} := \{ X \mid \exists\, x \in \mathbb{Z}_p, X = g^x \}$ with the corresponding hard relation $R_{\mathsf{DL}} = \{ (X, x) \mid X = g^x \}$.*

## 3.2 Adaptor Signatures

Adaptor Signatures were first formally defined in [AEE+21]. [EEE20a] generalized the syntax in [AEE+21] to enable constructing adaptor signatures from lattices. More concretely, [AEE+21] defined adaptor signatures w.r.t. a digital signature scheme and a hard relation $R$. [EEE20a] generalized it to be w.r.t. two hard relations $R$ and $R'$ such that such that $R \subseteq R'$. For the group-based constructions, typically we have $R = R'$, but for the lattice based constructions, typically we have $R \neq R'$. In this work, one of our constructions will be from lattices, hence, we follow this generalized syntax.

In a different vein, [DOY22, GSST24] have strengthened the security properties of adaptor signatures, with notions like *unforgeability*, *pre-signature extractability*, and *witness extractability*. [DOY22] further added *unique extractability* and *unlinkability* as security properties and unified *unforgeability* and *witness extractability* under a single security property called *extractability*. [GSST24] further added *pre-verify soundness* as a security property. We note that not all security properties are needed always. Within the scope of this work, we will use adaptor signatures as a building block for the construction of FAS and we only need *pre-signature adaptability* and *witness extractability* security properties of adaptor signatures. Intuitively, pre-signature adaptability ensures that given a valid pre-signature and a witness for the statement, one can always adapt the pre-signature into a valid signature. Witness extractability requires that given an honestly generated pre-signature and a valid signature, one should be able to extract a witness. We define these formally below.

**Definition 3.4.** *An adaptor signature scheme* $\mathsf{AS}_{\mathsf{DS},R,R'} := (\mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Adapt}, \mathsf{Ext})$ *is defined with respect to a signature scheme* $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and hard relations $R$ and $R'$ such that $R \subseteq R'$. Here, $R$ constitutes the relation for the statement-witness pairs generated by* $\mathsf{GenR}$ *and $R'$ is an extended relation that defines the relation for* extracted *witnesses. The interfaces are described below.*

- $\widetilde{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, X)$*: The pre-signing algorithm takes as input a signing key* $\mathsf{sk}$*, a message $m$, and a statement $X$ for the language $L_R$, and outputs a pre-signature $\widetilde{\sigma}$ (we sometimes also refer to this as a partial signature).*
- $0/1 \leftarrow \mathsf{PreVerify}(\mathsf{vk}, m, X, \widetilde{\sigma})$*: The pre-signature verification algorithm takes as input a verification key* $\mathsf{vk}$*, a message $m$, a statement $X$ for the language $L_R$, and a pre-signature $\widetilde{\sigma}$, and outputs $0/1$ signifying whether $\widetilde{\sigma}$ is correctly generated.*
- $\sigma := \mathsf{Adapt}(\mathsf{vk}, m, X, x, \widetilde{\sigma})$*: The adapt algorithm transforms a pre-signature $\widetilde{\sigma}$ into a valid signature $\sigma$ given the witness $x$ for the instance $X$ of the language $L_R$.*
- $x := \mathsf{Ext}(\widetilde{\sigma}, \sigma, X)$*: The extract algorithm takes as input a pre-signature $\widetilde{\sigma}$, a signature $\sigma$, and an instance $X$, and outputs a witness $x'$ such that $(X, x') \in R'$, or $\perp$.*

Note that an adaptor signature scheme $\mathsf{AS}_{\mathsf{DS},R,R'}$ also inherits $\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf}$ algorithms from the signature scheme $\mathsf{DS}$.

Typically, an adaptor signature is run between a buyer and a seller. The buyer runs $\mathsf{KGen}, \mathsf{Sign}$, $\mathsf{PreSign}, \mathsf{Ext}$ algorithms, the seller runs $\mathsf{Adapt}$ algorithm, and anyone can run $\mathsf{PreVerify}, \mathsf{Vf}$ algorithms.

**Remark 3.5.** *Looking ahead, in the discrete-log based instantiation, we will have $R' = R$, but in the lattice-based instantiation, we will have $R' \neq R$ and the reason for this extension is the* knowledge/soundness gap *inherent in* efficient *lattice-based zero-knowledge proofs.*

**Definition 3.6** (Correctness)**.** *An adaptor signature scheme* $\mathsf{AS}$ *satisfies correctness if for every* $n \in \mathbb{N}$*, every message* $m \in \{0,1\}^*$*, and every statement/witness pair* $(X, x) \in R$*, the following holds:*

$$\Pr\left[\begin{array}{c} \mathsf{PreVerify}(\mathsf{vk}, m, X, \widetilde{\sigma}) = 1 \\ \wedge\ \mathsf{Vf}(\mathsf{vk}, m, \sigma) = 1 \\ \wedge\ (X, x') \in R' \end{array} : \begin{array}{l} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda) \\ \widetilde{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, X) \\ \sigma := \mathsf{Adapt}(\mathsf{vk}, m, X, x, \widetilde{\sigma}) \\ x' := \mathsf{Ext}(\widetilde{\sigma}, \sigma, X) \end{array}\right] = 1$$

In terms of security, we want witness extractability against a malicious seller and weak pre-signature adaptability against a malicious seller. We explain these next.

Witness extractability requires that for an honestly generated pre-signature and a valid signature, one should be able to extract a witness. Intuitively, it tries to capture that an interaction between an honest buyer (i.e., generates valid pre-signature) and a malicious seller trying to get paid (i.e., adapt pre-signature into a valid signature) without revealing a witness (i.e., extraction failure) should be unlikely to occur.

**Definition 3.7** (Witness Extractability)**.** *An adaptor signature scheme* $\mathsf{AS}$ *is witness extractable if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

*where the experiment* $\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}$ *is defined as in Figure 3.*

```
Experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}$.
1 :  $\mathcal{Q} := \emptyset$
2 :  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$
3 :  $(m, X) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{pS}(\cdot,\cdot)}(\mathsf{vk})$
4 :  $\widetilde{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, X)$
5 :  $\sigma \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{pS}(\cdot,\cdot)}(\widetilde{\sigma})$
6 :  $x' := \mathsf{Ext}(\widetilde{\sigma}, \sigma, X)$
7 :  ret $((m \notin \mathcal{Q}) \wedge ((X, x') \notin R') \wedge \mathsf{Vf}(\mathsf{vk}, m, \sigma))$

Oracle $\mathcal{O}_S(m)$
1 :  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 :  ret $\sigma$

Oracle $\mathcal{O}_{pS}(m, X)$
1 :  $\widetilde{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, X)$
2 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 :  ret $\widetilde{\sigma}$
```

**Figure 3:** Witness Extractability experiment of adaptor signatures

Note that, in the above witness extractability definition, the adversary's winning condition is restricted to the extracted witness not being in $R'$. Since $R \subseteq R'$, $(X, x') \notin R'$ implies $(X, x') \notin R$. Therefore, it is sufficient to ensure that $R'$ is a hard relation, which itself implies that $R$ is also a hard relation. As a result, in our security assumptions, we make sure that $R'$ is a hard relation.

Weak pre-signature adaptability requires that given a valid pre-signature and a witness for the instance, one can always adapt the pre-signature into a valid signature. Intuitively, this tries to capture that it should be impossible for a malicious buyer to learn the witness without doing the payment.

**Definition 3.8** (Weak Pre-signature Adaptability)**.** *An adaptor signature scheme* $\mathsf{AS}$ *satisfies weak pre-signature adaptability if for any* $\lambda \in \mathbb{N}$*, any message* $m \in \{0,1\}^*$*, any key pair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow$

$\mathsf{KGen}(1^\lambda)$, *any statement/witness pair* $(X, x) \in R$, *and any pre-signature* $\widetilde{\sigma} \in \{0,1\}^*$ *with* $\mathsf{PreVerify}(\mathsf{vk}, m, X, \widetilde{\sigma}) = 1$, *we have:*

$$\Pr[\mathsf{Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, m, X, x, \widetilde{\sigma})) = 1] = 1.$$

Note that the above pre-signature adaptability is called *weak* because only statement-witness pairs satisfying $R$ are guaranteed to be adaptable, and not those satisfying $R'$. Therefore, weak pre-signature adaptability does not guarantee, for example, that an *extracted* witness can be used to adapt a pre-signature successfully. This is however guaranteed whenever $R = R'$.

We have several constructions of adaptor signatures compatible with ECDSA [AEE+21], lattice-based signatures [EEE20a, ACL+22], and dichotomic signature schemes [GSST24] which is a recently introduced abstraction that captures many popular schemes like Schnorr, CL, BBS, etc. The group-based schemes are constructed w.r.t. the discrete logarithm NP language in the respective groups, while the lattice-based schemes are constructed w.r.t. the short integer solution NP language.

## 3.3 Non-Interactive Zero-Knowledge Arguments

**Definition 3.9** (Secure NIZK Argument System). *A* $\mathsf{NIZK}$ *for language* $L_R$ *in the* common reference string (CRS) *model consists of the following possibly randomized algorithms.*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*: it takes in a security parameter* $\lambda$ *and outputs the common reference string* $\mathsf{crs}$ *which is publicly known to everyone.*
- $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*: it takes in the common reference string* $\mathsf{crs}$*, a statement* $\mathsf{stmt}$*, and a witness* $\mathsf{wit}$*, outputs a proof* $\pi$*.*
- $0/1 \leftarrow \mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi)$*: it takes in the common reference string* $\mathsf{crs}$*, a statement* $\mathsf{stmt}$*, and a proof* $\pi$*, and either accepts (with output* $1$*) the proof, or rejects (with output* $0$*) it.*

*A secure* $\mathsf{NIZK}$ *argument system must satisfy the following properties:*

- **Completeness:** *For all* $\mathsf{stmt}, \mathsf{wit}$ *where* $R(\mathsf{stmt}, \mathsf{wit}) = 1$*, if* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*, then,*
$$\Pr[\mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1] = 1.$$
- **Adaptive Soundness:** *For all non-uniform p.p.t. provers* $\mathcal{P}^*$*, there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*, if* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{stmt}, \pi) \leftarrow \mathcal{P}^*(\mathsf{crs})$*, then,*
$$\Pr[\mathsf{stmt} \notin L_R \wedge \mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1] \leq \mathsf{negl}(\lambda).$$
- **Zero-Knowledge:** *There exists a p.p.t. simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Prove}^*)$ *and there exists a negligible function* $\mathsf{negl}$ *such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, for all* $(\mathsf{stmt}, \mathsf{wit}) \in R$*,*
$$|\Pr[\mathcal{D}(\mathsf{crs}, \pi) = 1] - \Pr[\mathcal{D}(\mathsf{crs}^*, \pi^*) = 1]| \leq \mathsf{negl}(\lambda),$$
*where* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*,* $(\mathsf{crs}^*, \mathsf{td}) \leftarrow \mathsf{Setup}^*(1^\lambda)$*,* $\pi^* \leftarrow \mathsf{Prove}^*(\mathsf{crs}^*, \mathsf{td}, \mathsf{stmt})$ *and* $\mathsf{td}$ *is a trapdoor used by* $\mathsf{Sim}$ *to come up with proof* $\pi^*$ *for a statement* $\mathsf{stmt}$ *without knowing its witness.*

NIZK arguments are known from factoring [FLS90], and standard assumptions on pairings [CHK03, GOS06] and lattices [PS19, CCH+19].

## 3.4 Inner Product Functional Encryption

We will need a inner-product functional encryption scheme (IPFE). Let $\ell$ be an integer. We define IPFE for computing inner products of vectors of length $\ell$. Let the message space be $\mathcal{M} \subseteq \mathbb{Z}^{\ell}$ and function space $\mathcal{F}_{\mathsf{IP},\ell}$. Messages are denoted by $\mathbf{x} \in \mathcal{M}$, functions are denoted by $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$ and the function evaluation is denoted by $f_{\mathbf{y}}(\mathbf{x})$ (all three of them are parameterized by the prime $p \in \mathsf{pp}$ as output by $\mathsf{Gen}$ below). Formally, IPFE consists of the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^{\lambda})$: it is a *randomized* algorithm that takes in a security parameter $\lambda$ and samples public parameters $\mathsf{pp}$. We assume that $\mathsf{pp}$ contains a prime number $p$.
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}, \ell)$: it is a *randomized* algorithm that takes in the public parameters $\mathsf{pp}$ and the vector length $\ell$, outputs a master public key $\mathsf{mpk}$ and a master secret key $\mathsf{msk}$.
- $\mathsf{sk}_{\mathbf{y}} := \mathsf{KGen}(\mathsf{msk}, \mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell})$: it is a *deterministic* algorithm that takes in the master secret key $\mathsf{msk}$, and a vector $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$, outputs a functional secret key $\mathsf{sk}_{\mathbf{y}}$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x} \in \mathcal{M})$: it is a *randomized* algorithm that takes in the master public key $\mathsf{mpk}$, a plaintext vector $\mathbf{x} \in \mathcal{M}$, and outputs a ciphertext $\mathsf{ct}$.
- $v := \mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct})$: it is a *deterministic* algorithm that takes in the functional secret key $\mathsf{sk}_{\mathbf{y}}$ and a ciphertext $\mathsf{ct}$, and outputs a decrypted outcome $v$.

Further, we augment the IPFE scheme with an additional algorithm $\mathsf{PubKGen}$ defined as follows.

- $\mathsf{pk}_{\mathbf{y}} := \mathsf{PubKGen}(\mathsf{mpk}, \mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell})$: it is a *deterministic* algorithm that takes in the master public key $\mathsf{mpk}$, and a vector $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$, outputs a functional public key $\mathsf{pk}_{\mathbf{y}}$.

Next, we describe the correctness of IPFE.

**Definition 3.10** (IPFE Correctness)**.** *For any $\lambda \in \mathbb{N}$, let $\mathsf{pp} \leftarrow \mathsf{Gen}(1^{\lambda})$, then for any $\ell \in \mathbb{N}, \mathbf{x} \in \mathcal{M}, \mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$, the following holds:*

$$\Pr\left[ v = f_{\mathbf{y}}(\mathbf{x}) : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}, \ell), \\ \mathsf{sk}_{\mathbf{y}} = \mathsf{KGen}(\mathsf{msk}, \mathbf{y}), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}), \\ v = \mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct}) \end{array} \right] = 1.$$

Next, we describe the security of IPFE.

**Definition 3.11** (Selective, IND-security)**.** *We say that the $\mathsf{IPFE}$ scheme satisfies selective, IND-security, iff for any non-uniform p.p.t. admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}$ such that for all $\lambda \in \mathbb{N}, \ell \in \mathbb{N}$,*

$$|\Pr[\mathsf{IPFE\text{-}IND\text{-}SEL}^{0}_{\mathcal{A},\mathsf{IPFE}}(1^{\lambda}, \ell) = 1] - \Pr[\mathsf{IPFE\text{-}IND\text{-}SEL}^{1}_{\mathcal{A},\mathsf{IPFE}}(1^{\lambda}, \ell) = 1]| \leq \mathsf{negl}(\lambda),$$

*where experiments $\mathsf{IPFE\text{-}IND\text{-}SEL}^{b}_{\mathcal{A},\mathsf{IPFE}}(1^{\lambda})$ for $b \in \{0, 1\}$ are described in Figure 4. Here, an adversary $\mathcal{A}$ is said to be admissible iff the following holds with probability 1: for any $\mathbf{y}$ submitted in a $\mathcal{O}_{\mathsf{KGen}}$ oracle query, it must be that $f_{\mathbf{y}}(\mathbf{x}^{*}_{0}) = f_{\mathbf{y}}(\mathbf{x}^{*}_{1})$.*

```
Experiment $\mathsf{IPFE\text{-}IND\text{-}SEL}^b_{\mathcal{A},\mathsf{IPFE}}(1^\lambda, \ell)$
1 : $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$
2 : $(\mathbf{x}_0^*, \mathbf{x}_1^*) \leftarrow \mathcal{A}(\mathsf{pp})$
3 : $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}, \ell)$, $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_b^*)$
4 : $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KGen}}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^*)$
5 : ret $b'$

Oracle $\mathcal{O}_{\mathsf{KGen}}(\mathbf{y})$
1 : ret $\mathsf{sk}_{\mathbf{y}} = \mathsf{KGen}(\mathsf{msk}, \mathbf{y})$
```

**Figure 4:** Selective, IND-security experiment of Inner Product Functional Encryption

The above IND-security level is called selective because it requires the adversary to commit to the challenge message pair even before it sees $\mathsf{mpk}$. Stronger security models exist such as (i) semi-adaptive, IND-security where the adversary can commit to the challenge message pair after seeing $\mathsf{mpk}$ but before gaining access to the oracle $\mathcal{O}_{\mathsf{KGen}}$ and (ii) adaptive, IND-security where the adversary can commit to the challenge message pair after seeing $\mathsf{mpk}$ and the power to make queries to the oracle $\mathcal{O}_{\mathsf{KGen}}$ before and after committing to the challenge message pair. But, we only consider selective, IND-security as it suffices for the scope of this work. We also note that any adaptive, IND-secure IPFE is also selective, IND-secure.

**Remark 3.12.** *We note that $\mathsf{pk}_{\mathbf{y}}$ does not enable decryption by any means, so it will not change the correctness requirement. It will also not affect security of IPFE in any way. To see this, note that $\mathsf{PubKGen}$ is deterministic. So, an adversary can always compute $\mathsf{pk}_{\mathbf{y}}$ on its own. Given that $\mathsf{KGen}$ is also deterministic, one way to think of $\mathsf{pk}_{\mathbf{y}}$ is as a statistically binding and computationally hiding commitment to $\mathsf{sk}_{\mathbf{y}}$. Typically, $\mathsf{mpk}$ also has this property w.r.t. $\mathsf{msk}$ and it is always implicit in the security definitions of IPFE. We make this explicit below for $\mathsf{pk}_{\mathbf{y}}$ and $\mathsf{sk}_{\mathbf{y}}$ because our functional adaptor signatures will rely on it. We formalize this compliance requirement below.*

**Definition 3.13** ($R_{\mathsf{IPFE}}$-Compliant IPFE)**.** *For a hard relation $R_{\mathsf{IPFE}}$, we say that an IPFE scheme is $R_{\mathsf{IPFE}}$ compliant if for any $\lambda \in \mathbb{N}$, for any $\mathsf{pp} \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$, for any $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}, 1^\ell)$, for any $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$, let $\mathsf{pk}_{\mathbf{y}} := \mathsf{PubKGen}(\mathsf{mpk}, \mathbf{y})$, and $\mathsf{sk}_{\mathbf{y}} := \mathsf{KGen}(\mathsf{msk}, \mathbf{y})$. Then, it must be the case that $(\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}) \in R_{\mathsf{IPFE}}$.*

Next, we describe an additional correctness property of IPFE that is tailored specifically to make IPFE compatible with adaptor signatures when the two are used in tandem in our functional adaptor signature construction. Looking ahead, we need this property because in our functional adaptor signature construction, we will use a standard adaptor signature to sell functional secret keys. Recall that adaptor signatures are defined w.r.t. two relations $R$ and $R'$. In this case, $R$ will be $R_{\mathsf{IPFE}}$ as defined in Definition 3.13 and let us denote the corresponding extended relation $R'$ by $R'_{\mathsf{IPFE}}$. Then, the $\mathsf{AS.Ext}$ algorithm of adaptor signature scheme $\mathsf{AS}$ will extract a functional secret key $\mathsf{sk}'_{\mathbf{y}}$ as the witness for the adaptor statement $\mathsf{pk}_{\mathbf{y}}$. While $\mathsf{sk}'_{\mathbf{y}}$ will be a witness of $R'_{\mathsf{IPFE}}$, it may not be a witness of the base relation $R_{\mathsf{IPFE}}$ since $R_{\mathsf{IPFE}} \subseteq R'_{\mathsf{IPFE}}$, and we still want that $\mathsf{sk}'_{\mathbf{y}}$ should enable meaningful decryption. We formalize this robustness requirement below.

**Definition 3.14** ($R'_{\mathsf{IPFE}}$-Robust IPFE)**.** *Let $R_{\mathsf{IPFE}}$ be the hard relation as defined in Definition 3.13. Let $R'_{\mathsf{IPFE}}$ such that $R_{\mathsf{IPFE}} \subseteq R'_{\mathsf{IPFE}}$ be an an extended relation of $R_{\mathsf{IPFE}}$. We say that an IPFE scheme is $R'_{\mathsf{IPFE}}$ robust if IPFE correctness holds even w.r.t. a functional secret key satisfying $R'_{\mathsf{IPFE}}$. More formally, for any $\lambda \in \mathbb{N}$, let $\mathsf{pp} \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$, then for any $\ell \in \mathbb{N}$, $\mathbf{x} \in \mathcal{M}, \mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$, the* *following holds:*

$$\Pr\left[ v = f_{\mathbf{y}}(\mathbf{x}) : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}, \ell), \\ \mathsf{pk}_{\mathbf{y}} = \mathsf{PubKGen}(\mathsf{msk}, \mathbf{y}), \\ \mathsf{sk}'_{\mathbf{y}} \ s.t. \ (\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}'_{\mathbf{y}}) \in R'_{\mathsf{IPFE}}, \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}), \\ v = \mathsf{Dec}(\mathsf{sk}'_{\mathbf{y}}, \mathsf{ct}) \end{array} \right] = 1.$$

**Remark 3.15.** *One can think of decryption correctness (Definition 3.10) as being implicitly defined with respect to* $\mathsf{sk}_{\mathbf{y}}$ *that is witness to* $\mathsf{pk}_{\mathbf{y}}$ *for the relation* $R_{\mathsf{IPFE}}$*. And one can think of the* $R'_{\mathsf{IPFE}}$*-robustness as decryption correctness with respect to* $\mathsf{sk}'_{\mathbf{y}}$ *that is witness to* $\mathsf{pk}_{\mathbf{y}}$ *for the relation* $R'_{\mathsf{IPFE}}$*.*

Abdalla et al. [ABDP15] constructed selective, IND-secure IPFE schemes from DDH and LWE. Subsequently, Agrawal et al. [ALS16] constructed adaptive, IND-secure IPFE schemes from DDH, DCR, and LWE. Within the scope of this work, we will use the following two IPFE schemes:

- The DDH based IPFE by Abdalla et al. [ABDP15]. This scheme computes inner products with output values polynomially bounded by $B \ll p$. Specifically, the message space is $\mathcal{M} = \mathbb{Z}_p^\ell$ and the function class is $\mathcal{F}_{\mathsf{IP},\ell,p,B} = \{f_{\mathbf{y}} : \mathbf{y} \in \mathbb{Z}_p^\ell\} \subseteq \mathcal{F}_{\mathsf{IP},\ell}$, where $f_{\mathbf{y}} : \mathbb{Z}_p^\ell \to \{0, \dots, B\}$ is defined as $f_{\mathbf{y}}(\mathbf{x}) = \mathbf{x}^T\mathbf{y} \in \{0, \dots, B\}$.
- The LWE based IPFE by Agrawal et al. [ALS16, Section 4.2]. This scheme computes inner products modulo $p$. Specifically, the message space is $\mathcal{M} = \mathbb{Z}_p^\ell$ and the function class is $\mathcal{F}_{\mathsf{IP},\ell,p} = \{f_{\mathbf{y}} : \mathbf{y} \in \mathbb{Z}_p^\ell\} \subseteq \mathcal{F}_{\mathsf{IP},\ell}$, where $f_{\mathbf{y}} : \mathbb{Z}_p^\ell \to \mathbb{Z}_p$ is defined as $f_{\mathbf{y}}(\mathbf{x}) = \mathbf{x}^T\mathbf{y} \bmod p \in \mathbb{Z}_p$.

## 3.5 Lattice Preliminaries

We denote by $\mathcal{S}_c = \{\mathbf{x} \in \mathbb{Z}_q^n : ||\mathbf{x}||_\infty \leq c\}$ the set of vectors in $\mathbb{Z}_q^n$ whose maximum absolute coefficient is at most $c \in \mathbb{Z}^+$.

**Definition 3.16** ($\mathsf{SIS}_{n,m,q,\beta}$)**.** *Let* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$*. Given* $\mathbf{A}$*,* $\mathsf{SIS}$ *problem with parameters* $m > 1$ *and* $0 < \beta < q$ *asks to find a short non-zero* $\mathbf{v} \in \mathbb{Z}^m$ *such that* $\mathbf{Av} = 0 \bmod q$ *and* $||\mathbf{v}||_\infty \leq \beta$*.*

Next, we define $\mathsf{SIS}$ in Hermite Normal Form (HNF).

**Definition 3.17** ($\mathsf{HNF\text{-}SIS}_{n,m,q,\beta}$)**.** *Let* $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n \times (m-1)}$ *and* $\mathbf{A} = (\mathbf{1}||\mathbf{A}') \in \mathbb{Z}_q^{n \times m}$*. Given* $\mathbf{A}$*,* $\mathsf{HNF\text{-}SIS}$ *problem with parameters* $m > 1$ *and* $0 < \beta < q$ *asks to find a short non-zero* $\mathbf{v} \in \mathbb{Z}^m$ *such that* $\mathbf{Av} = 0 \bmod q$ *and* $||\mathbf{v}||_\infty \leq \beta$*.*

**Definition 3.18** ($\mathsf{ISIS}_{n,m,q,\beta}$)**.** *It is the inhomogeneous version of* $\mathsf{HNF\text{-}SIS}$*. Let* $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n \times (m-1)}$ *and* $\mathbf{A} = (\mathbf{1}||\mathbf{A}') \in \mathbb{Z}_q^{n \times m}$ *and let* $\mathbf{u} \xleftarrow{\$} \mathbb{Z}^m$ *s.t.* $||\mathbf{u}||_\infty \leq \beta$*. Let* $\mathbf{b} = \mathbf{Au} \in \mathbb{Z}_q^n$*. Given* $(\mathbf{A}, \mathbf{b})$*, the problem with parameters* $m > 1$ *and* $0 < \beta < q$ *asks to find a short non-zero* $\mathbf{v} \in \mathbb{Z}^m$ *such that* $\mathbf{Av} = \mathbf{b} \bmod q$ *and* $||\mathbf{v}||_\infty \leq \beta$*.*

**Definition 3.19** ($\mathsf{LWE}_{n,m,q,\Psi}$)**.** $\mathsf{LWE}$ *problem with parameters* $n, m > 0$ *and distribution* $\Psi$ *over* $\mathbb{Z}_q$ *asks to distinguish between the following two cases with non-negligible advantage: 1)* $(\mathbf{A}, \mathbf{b}) \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$*, and 2)* $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ *for* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$*, a secret* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ *and an error vector* $\mathbf{e} \leftarrow \Psi^m$*.*

**Definition 3.20** (Multi-hint extended-LWE). *Let $q, m, t$ be integers, $\alpha$ be a real and $\tau$ be a distribution over $\mathbb{Z}^{t\times m}$, all of them functions of a parameter $n$. The multi-hint extended-LWE problem $\mathsf{mheLWE}_{n,m,q,\alpha,t,\tau}$ is to distinguish between the distributions of the tuples*

$$(\mathbf{A}, \mathbf{b}, \mathbf{Z}, \mathbf{Z}\mathbf{e}) \text{ and } (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}, \mathbf{Z}, \mathbf{Z}\mathbf{e}),$$

*where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{e} \leftarrow D^m_{\mathbb{Z},\alpha q}$, $\mathbf{Z} \leftarrow \tau$.*

**Lemma 3.21** ( [ALS16], Theorem 4). *Let $n \geq 100$, $q \geq 2$, $t < n$ and $m$ with $m = \Omega(n \log n)$ and $m \leq n^{O(1)}$. There exists $\xi \leq O(n^4 m^2 \log^{5/2} n)$ and a distribution $\tau$ over $\mathbb{Z}^{t\times m}$ such that the following statements hold:*

- *There is a reduction from $\mathsf{LWE}_{n-t,m,q,D_{\mathbb{Z},\alpha q}}$ to $\mathsf{mheLWE}_{n,m,q,\alpha,t,\tau}$.*
- *It is possible to sample from $\tau$ in polynomial time in $n$.*
- *Each entry of matrix $\tau$ is an independent discrete Gaussian $\tau_{i,j} = D_{\mathbb{Z},\sigma_{i,j},c_{i,j}}$ for some $c_{i,j} \in \{0, 1\}$ and $\sigma_{i,j} \geq \Omega(\sqrt{mn \log m})$.*
- *With probability at least $1 - n^{-\omega(1)}$, all rows from a sample from $\tau$ have norms at most $\xi$.*

**Definition 3.22** (Discrete Gaussian distribution). *A discrete gaussian distribution $D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$, for $\mathbf{c} \in \mathbb{R}^n$, $\Sigma$ a positive semi-definite matrix in $\mathbb{R}^{n\times n}$, and a lattice $\Lambda \subset \mathbb{Z}^n$, is a distribution with values in $\Lambda$ and probabilities*

$$\Pr[X = \mathbf{x}] \propto \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c})^T \Sigma^+ (\mathbf{x} - \mathbf{c})\right).$$

Note that $\Sigma^+$ denotes the pseudoinverse of a matrix. If $\Lambda = \mathbb{Z}^n$, we shall just write $D_{\sqrt{\Sigma},\mathbf{c}}$. Further, if $\mathbf{c} = 0$, then we shall just write $D_{\sqrt{\Sigma}}$, and if $\sqrt{\Sigma} = \sigma I_n$ for $\sigma \in \mathbb{R}^+$ and $I_n$ an identity matrix, we shall just write $D_\sigma$.

## 4 FAS Definition

A functional adaptor signature scheme shares similar syntax to an adaptor signature scheme with few additional interfaces and parameters. Specifically, the interfaces are $\mathsf{FAS}$ := ($\mathsf{Setup}$, $\mathsf{AdGen}$, $\mathsf{AdVerify}$, $\mathsf{AuxGen}$, $\mathsf{AuxVerify}$, $\mathsf{FPreSign}$, $\mathsf{FPreVerify}$, $\mathsf{Adapt}$, $\mathsf{FExt}$). We recall that ($\mathsf{AuxGen}$, $\mathsf{AuxVerify}$, $\mathsf{FPreSign}$) essentially denote the 3-round interactive pre-signing process. We describe the formal definition below.

**Definition 4.1** (Functional adaptor signature). *A functional adaptor signature scheme $\mathsf{FAS}$ := ($\mathsf{Setup}$, $\mathsf{AdGen}$, $\mathsf{AdVerify}$, $\mathsf{AuxGen}$, $\mathsf{AuxVerify}$, $\mathsf{FPreSign}$, $\mathsf{FPreVerify}$, $\mathsf{Adapt}$, $\mathsf{FExt}$) is defined with respect to a signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, a hard relation $R$, and a family of functions $\mathcal{F}$. The interfaces are described below.*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter $1^\lambda$, and outputs public parameters* $\mathsf{pp}$.
- $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, X, x)$*: The advertisement generation algorithm takes as input a public parameter* $\mathsf{pp}$*, a statement $X$ for the language $L_R$, and a witness $x$, and outputs a public advertisement* $\mathsf{advt}$*, and a secret state* $\mathsf{st}$.

- $0/1 \leftarrow \mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt})$*: The advertisement verify algorithm takes as input a public parameter* $\mathsf{pp}$*, a statement $X$ for the language $L_R$, and a public advertisement* $\mathsf{advt}$*, and outputs $0/1$ signifying whether* $\mathsf{advt}$ *is generated correctly.*

- $(\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$*: The auxiliary value generation algorithm takes as input a public advertisement* $\mathsf{advt}$*, a* secret *state* $\mathsf{st}$*, a function $f$ belonging to the family $\mathcal{F}$, and deterministically outputs an auxiliary value* $\mathsf{aux}_f$*, and a proof $\pi_f$.*

- $0/1 \leftarrow \mathsf{AuxVerify}(\mathsf{advt}, f, \mathsf{aux}_f, \pi_f)$*: The auxiliary value verify algorithm takes as input a public advertisement* $\mathsf{advt}$*, a function $f$ belonging to the family $\mathcal{F}$, an auxiliary value* $\mathsf{aux}_f$*, and a proof $\pi_f$, and outputs $0/1$ signifying whether* $\mathsf{aux}_f$ *is generated correctly.*

- $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, f, \mathsf{aux}_f)$*: The functional pre-signing algorithm takes as input a public advertisement* $\mathsf{advt}$*, a signing key* $\mathsf{sk}$*, a message $m$, a statement $X$ for the language $L_R$, a function $f$ belonging to the family $\mathcal{F}$, and an auxiliary value* $\mathsf{aux}_f$*, and outputs a pre-signature $\widetilde{\sigma}$.*

- $0/1 \leftarrow \mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma})$*: The pre-signature verification algorithm takes as input a public advertisement* $\mathsf{advt}$*, a* secret *state* $\mathsf{st}$*, a verification key* $\mathsf{vk}$*, a message $m$, a statement $X$ for the language $L_R$, a function $f$ belonging to the family $\mathcal{F}$, an auxiliary value* $\mathsf{aux}_f$*, a proof $\pi_f$, and a pre-signature $\widetilde{\sigma}$, and outputs $0/1$ signifying whether $\widetilde{\sigma}$ is correctly generated.*

- $\sigma := \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, x, f, \mathsf{aux}_f, \widetilde{\sigma})$*: The adapt algorithm transforms a pre-signature $\widetilde{\sigma}$ into a valid signature $\sigma$ given the witness $x$ for the instance $X$ of the language $L_R$, a function $f$ belonging to the family $\mathcal{F}$, an auxiliary value* $\mathsf{aux}_f$*, a public advertisement* $\mathsf{advt}$*, and a* secret *state* $\mathsf{st}$.

- $z := \mathsf{FExt}(\mathsf{advt}, \widetilde{\sigma}, \sigma, X, f, \mathsf{aux}_f)$*: The functional extract algorithm takes as input a public advertisement* $\mathsf{advt}$*, a pre-signature $\widetilde{\sigma}$, a signature $\sigma$, an instance $X$ for the language $L_R$, and a function $f \in \mathcal{F}$, an auxiliary value* $\mathsf{aux}_f$*, and outputs a value $z$ in the range of $f$.*

In a typical functional payment application, we will have a seller who holds a witness to the statement and wants to sell a function evaluation on it, and a buyer who wants to buy a function evaluation of the witness. We refer the reader to Figure 1 for the protocol flow.

**Definition 4.2** (Correctness)**.** *A functional adaptor signature scheme* $\mathsf{FAS}$ *satisfies correctness if for every $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}^*$, every statement/witness pair $(X, x) \in R$, every function $f \in \mathcal{F}$, suppose* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, X, x)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $(\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$*,* $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}\ (\mathsf{advt}, \mathsf{sk}, m, X, f, \mathsf{aux}_f)$*,* $\sigma := \mathsf{Adapt}(\ \mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, x, f, \mathsf{aux}_f, \widetilde{\sigma})$*, and* $z := \mathsf{FExt}(\mathsf{advt}, \widetilde{\sigma}, \sigma, X, f, \mathsf{aux}_f)$*. Then the following holds:*

$$\Pr\left[\begin{array}{c} \mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt}) = 1 \\ \wedge\ \mathsf{AuxVerify}(\mathsf{advt}, f, \mathsf{aux}_f, \pi_f) = 1 \\ \wedge\ \mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \pi_f, \widetilde{\sigma}) = 1 \\ \wedge\ \mathsf{Vf}(\mathsf{vk}, m, \sigma) = 1\ \wedge\ z = f(x) \end{array}\right] = 1.$$

In terms of security, we want the following properties against malicous seller and buyer:

- **Malicious seller:** advertisement soundness, pre-signature validity, unforgeability, witness extractability.

- **Malicious buyer:** pre-signature adaptability, witness privacy.

Looking ahead, we will consider three notions of witness privacy: zero-knowledge, witness indistinguishability, and witness hiding. Among these, zero-knowledge is the strongest. Based on this, we define two overall security levels of functional adaptor signatures below.

**Definition 4.3** (Strongly-secure Functional Adaptor Signature Scheme)**.** *A* $\mathsf{FAS}$ *scheme is* strongly-secure *if it is advertisement sound, pre-signature valid, unforgeable, witness extractable, pre-signature adaptable and zero-knowledge.*

**Definition 4.4** (Weakly-secure Functional Adaptor Signature Scheme)**.** *A* $\mathsf{FAS}$ *scheme is* weakly-secure *if it is advertisement sound, pre-signature valid, unforgeable, witness extractable, pre-signature adaptable and witness indistinguishable.*

We describe all the security properties next.

**Advertisement Soundness.** The property requires that given $\mathsf{pp}$, it should be infeasible for a p.p.t. malicious seller to find statement $X \notin L_R$ and advertisement $\mathsf{advt}$ such that $\mathsf{advt}$ is accepted by $\mathsf{AdVerify}$.

**Definition 4.5** (Advertisement Soundness)**.** *A functional adaptor signature scheme* $\mathsf{FAS}$ *satisfies advertisement soundness if for every p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*, for all NP languages* $L_R$*,* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(X, \mathsf{advt}) \leftarrow \mathcal{A}(\mathsf{pp})$*,*

$$\Pr[X \notin L_R \ \wedge \ \mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt}) = 1] \leq \mathsf{negl}(\lambda).$$

**Pre-signature validity.** This property says that given a valid auxiliary value, one can always compute a valid pre-signature using it. Intuitively, this tries to capture that it should be impossible for a malicious seller to give out a malformed auxiliary value for the function $f$ and yet learn an honestly generated pre-signature for $f$ that is valid.

**Definition 4.6** (Pre-signature Validity)**.** *A functional adaptor signature scheme* $\mathsf{FAS}$ *satisfies pre-signature validity if for any* $\lambda \in \mathbb{N}$*, any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, any* $X, \mathsf{advt}$ *such that* $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{advt}, X) = 1$*, any message* $m \in \{0,1\}^*$*, any function class* $\mathcal{F}$*, any function* $f \in \mathcal{F}$*, any* $(\mathsf{aux}_f, \pi_f)$*, any key pair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$*, any pre-signature* $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, f, \mathsf{aux}_f)$*, we have that if* $\mathsf{AuxVerify}(\mathsf{advt}, f, \mathsf{aux}_f, \pi_f) = 1$*, then,*

$$\Pr[\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) = 1] = 1.$$

**Unforgeability.** This is similar to the unforgeability security property typically defined for adaptor signatures [AEE+21]. The property requires that even when the adversary (malicious seller) is given access to pre-signatures with respect to functions $f$ of its choice, it should not be able to forge signatures if it does not know the witness $x^*$ for the challenge statement $X^*$ in the first place. To capture this essence, the experiment (See Figure 5) samples $(X^*, x^*)$ at random. Beyond this, the adversary has all the powers of a (malicious) seller starting with generating $\mathsf{advt}$. A curious reader might ask if the adversary doesn't know $x^*$, then, wouldn't it typically fail to pass the $\mathsf{AdVerify}$ check in step 4 in Figure 5 and thus rendering the experiment after that pointless? The answer is no, because *advertisement soundness* is for statements not in the language $L_R$, but here $X^* \in L_R$ as the experiment chooses it.

**Definition 4.7** (Unforgeability)**.** *A* $\mathsf{FAS}$ *scheme is* $\mathsf{faEUF\text{-}CMA}$*-secure or simply unforgeable, if for every stateful p.p.t. adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*,* $\Pr[\mathsf{faEUF\text{-}CMA}_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{faEUF\text{-}CMA}_{\mathcal{A},\mathsf{FAS}}$ *is defined as in Figure 5.*

```
Experiment $\mathsf{faEUF\text{-}CMA}_{\mathcal{A},\mathsf{FAS}}(1^\lambda)$
1 : $\mathcal{Q} := \emptyset$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $(X^*, x^*) \leftarrow \mathsf{GenR}(1^\lambda)$
3 : $(\mathsf{advt}, m^*, f^*, \mathsf{aux}^*_f, \pi^*_f) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk}, X^*)$
4 : If $\mathsf{AdVerify}(\mathsf{pp}, X^*, \mathsf{advt}) = 0 \ \vee\ f^* \notin \mathcal{F}$
      $\vee\ \mathsf{AuxVerify}(\mathsf{advt}, f^*, \mathsf{aux}^*_f, \pi^*_f) = 0$ : ret 0
5 : $\widetilde{\sigma}^* \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m^*, X^*, f^*, \mathsf{aux}^*_f)$
6 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
7 : ret $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*))$

Oracle $\mathcal{O}_S(m)$
1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 : ret $\sigma$

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, f, \mathsf{aux}_f, \pi_f)$
1 : If $\mathsf{AuxVerify}(\mathsf{advt}, f, \mathsf{aux}_f, \pi_f) = 0$: ret $\perp$
2 : $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, f, \mathsf{aux}_f)$,
3 : $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
4 : ret $\widetilde{\sigma}$
```

**Figure 5:** Unforgeability of functional adaptor signatures

**Witness extractability.** Here we require that for any function $f \in \mathcal{F}$, for any honestly generated pre-signature w.r.t. $f$ and a valid signature, one should be able to extract the function evaluation $f$ on a witness. Intuitively, it tries to capture the interaction between an honest buyer (i.e., generates valid pre-signature) and a malicious seller trying to get paid (i.e., adapt pre-signature into a valid signature) without revealing function evaluation $f$ on a witness (i.e., extraction failure) should be unlikely to occur. This is similar to the witness extractability security property of adaptor signatures (See Definition 3.7) except that in the formal witness extractability experiment Figure 6, in step 7, computing the set $Z$ is inefficient.

**Definition 4.8** (Witness Extractability)**.** *A functional adaptor signature scheme* $\mathsf{FAS}$ *is* $\mathsf{faWitExt}$*-secure or simply witness extractable, if for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*,*

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

*where the experiment* $\mathsf{faWitExt}_{\mathcal{A},\mathsf{FAS}}$ *is defined as in Figure 6.*

```
Experiment $\mathsf{faWitExt}_{\mathcal{A},\mathsf{FAS}}(1^\lambda)$
1 : $\mathcal{Q} := \emptyset$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $(X^*, \mathsf{advt}, m^*, f^*, \mathsf{aux}^*_f, \pi^*_f) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk})$
3 : If $\mathsf{AdVerify}(\mathsf{pp}, X^*, \mathsf{advt}) = 0 \ \vee\ f^* \notin \mathcal{F}$
      $\vee\ \mathsf{AuxVerify}(\mathsf{advt}, f^*, \mathsf{aux}^*_f, \pi^*_f) = 0$: ret 0
4 : $\widetilde{\sigma}^* \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m^*, X^*, f^*, \mathsf{aux}^*_f)$
5 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
6 : $z := \mathsf{FExt}(\mathsf{advt}, \widetilde{\sigma}^*, \sigma^*, X^*, f^*, \mathsf{aux}^*_f)$
7 : Let $Z = \{f^*(x) : \exists\, x \ s.t.\ (X^*, x) \in R\}$
8 : ret $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge (z \notin Z))$

Oracle $\mathcal{O}_S(m)$
1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 : ret $\sigma$

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, f, \mathsf{aux}_f, \pi_f)$
1 : If $\mathsf{AuxVerify}(\mathsf{advt}, f, \mathsf{aux}_f, \pi_f) = 0$: ret $\perp$
2 : $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, f)$
3 : $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
4 : ret $\widetilde{\sigma}$
```

**Figure 6:** Witness Extractability of functional adaptor signatures

**Pre-signature adaptability.** This property says that given a valid pre-signature and a witness for the instance, one can always adapt the pre-signature into a valid signature. Intuitively, this tries to capture that it should be impossible for a malicious buyer to learn a function evaluation on a witness without making the payment. This is similar to the pre-signature adaptability security property of adaptor signatures (See Definition 3.8).

**Definition 4.9** (Pre-signature Adaptability)**.** *A functional adaptor signature scheme* $\mathsf{FAS}$ *satisfies pre-signature adaptability if for any* $\lambda \in \mathbb{N}$*, any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, any statement/witness pair* $(X, x) \in R$*, any* $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, X, x)$*, any message* $m \in \{0,1\}^*$*, any function class* $\mathcal{F}$*, any function* $f \in \mathcal{F}$*, any* $(\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$*, any key pair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$*, and any pre-signature* $\widetilde{\sigma} \in \{0,1\}^*$ *with* $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) = 1$*, we have:*

$$\Pr[\mathsf{Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, x, f, \mathsf{aux}_f, \widetilde{\sigma})) = 1] = 1.$$

**Witness privacy.** Here we want that in an interaction between a malicious buyer wanting to learn function evaluation $f$ on the witness and an honest seller using witness $x$ to adapt the pre-signature into a valid signature, at the end the buyer should not be able to extract any information about $x$ beyond $f(x)$. There are different ways to capture this formally. We consider various notions of witness privacy akin to those in functional encryption literature and zero-knowledge proofs literature as follows.

- **Zero-Knowledge:** for every $(X, x) \in R$, there exists a p.p.t. simulator $\mathsf{Sim}$ such that the adversary should not be able to guess whether it is interacting with a real-world challenger that is allowed to use the witness $x$ or the simulator $\mathsf{Sim}$ that is only allowed access to $f(x)$.

- **Witness Indistinguishability:** given two valid witnesses $x_0$ and $x_1$ for a statement $X \in L_R$, the adversary should not be able to guess which witness was used to adapt a pre-signature as long as $f(x_0) = f(x_1)$. This notion is meaningful only for languages that have at least two witnesses.

- **Witness Hiding:** for every $X \in L_R$, given valid pre-signature w.r.t. function $f$ and valid signature adapted from it, the adversary should not be able to guess a witness $x$ s.t. $(X, x) \in R$.

Zero-Knowledge is the strongest among all in the sense that it implies the other two. All three are meaningful for non-unique witness languages. But for unique witness languages, witness indistinguishability is not meaningful. Further, witness-indistinguishability and witness-hiding are incomparable. Next, we describe all three formally.

**Definition 4.10** (Zero-Knowledge)**.** *A* $\mathsf{FAS}$ *scheme satisfies zero-knowledge if for every p.p.t. adversary* $\mathcal{A}$*, there exists a stateful p.p.t. simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{AuxGen}^*, \mathsf{Adapt}^*)$*, there exists a negligible function* $\mathsf{negl}$ *s.t. for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, for all* $(X, x) \in R$*,* $|\Pr[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, x)) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, x)) = 1]| \leq \mathsf{negl}(\lambda)$*, where experiments* $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}$ *and* $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ *are defined in Figure 7.*

**Definition 4.11** (Witness Indistinguishability)**.** *A functional adaptor signature scheme* $\mathsf{FAS}$ *is* $\mathsf{faWI}$*-secure or simply witness indistinguishable, if for every stateful p.p.t. adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$|\Pr[\mathsf{faWI}^0_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1] - \Pr[\mathsf{faWI}^1_{\mathcal{A},\mathsf{FAS}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

*where the experiments* $\mathsf{faWI}^b_{\mathcal{A},\mathsf{FAS}}$ *for* $b \in \{0,1\}$ *are defined as in Figure 8.*

```
Experiment $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, x)$
1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, X, x)$
3 : $\mathsf{vk} := \bot$
4 : $\mathsf{vk} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AuxGen}}(\cdot), \mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{advt}, X)$
5 : $\mathbf{ret}$ view of $\mathcal{A}$

Oracle $\mathcal{O}_{\mathsf{AuxGen}}(f)$
1 : $\mathbf{ret}\ (\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$

Oracle $\mathcal{O}_{\mathsf{Adapt}}(m, f, \widetilde{\sigma})$
1 : If $\mathsf{vk} = \bot$: $\mathbf{ret}\ \bot$
2 : $(\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$
3 : If $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) = 0$:
        $\mathbf{ret}\ \bot$
4 : $\mathbf{ret}\ \sigma := \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, x, f, \mathsf{aux}_f, \widetilde{\sigma})$
```

```
Experiment $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, x)$
1 : $\mathsf{pp} \leftarrow \mathsf{Setup}^*(1^\lambda)$
2 : $\mathsf{advt} \leftarrow \mathsf{AdGen}^*(\mathsf{pp}, X)$
3 : $\mathsf{vk} := \bot$
4 : $\mathsf{vk} \leftarrow \mathcal{A}^{\mathcal{O}^*_{\mathsf{AuxGen}}(\cdot,\cdot), \mathcal{O}^*_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{advt}, X)$
5 : $\mathbf{ret}$ view of $\mathcal{A}$

Oracle $\mathcal{O}^*_{\mathsf{AuxGen}}(f, f(x))$
1 : $\mathbf{ret}\ (\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}^*(\mathsf{advt}, f, f(x))$

Oracle $\mathcal{O}^*_{\mathsf{Adapt}}(m, f, \widetilde{\sigma}, f(x))$
1 : If $\mathsf{vk} = \bot$: $\mathbf{ret}\ \bot$
2 : $(\mathsf{aux}_f, \pi_f) \leftarrow \mathsf{AuxGen}^*(\mathsf{advt}, f, f(x))$
3 : If $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) = 0$:
        $\mathbf{ret}\ \bot$
4 : $\mathbf{ret}\ \sigma := \mathsf{Adapt}^*(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \widetilde{\sigma}, f(x))$
```

**Figure 7:** Zero-Knowledge experiments of FAS

```
Experiment $\mathsf{faWI}^b_{\mathcal{A},\mathsf{FAS}}$
1 : $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{vk}, X, x_0, x_1) \leftarrow \mathcal{A}(\mathsf{crs})$
3 : If $(((X, x_0) \notin R) \vee ((X, x_1) \notin R))$: $\mathbf{ret}\ 0$
4 : $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{crs}, X, x_b)$
5 : $(m, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AuxGen}}(\cdot)}(\mathsf{advt})$
6 : If $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) = 0$: $\mathbf{ret}\ 0$
7 : If $f(x_0) \neq f(x_1)$: $\mathbf{ret}\ 0$
8 : $\sigma := \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, x_b, f, \mathsf{aux}_f, \widetilde{\sigma})$
9 : $b' \leftarrow \mathcal{A}(\sigma)$
10 : $\mathbf{ret}\ b'$

Oracle $\mathcal{O}_{\mathsf{AuxGen}}(f)$
1 : $\mathbf{ret}\ \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$
```

**Figure 8:** Witness Indistinguishability experiment of adaptor signatures

**Definition 4.12** (Witness Hiding)**.** *A functional adaptor signature scheme* FAS *is* faWH*-secure or simply witness hiding, if for every stateful p.p.t. adversary $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

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

*where the experiment $\mathsf{faWH}_{\mathcal{A},\mathsf{FAS}}$ is defined as in Figure 9.*

```
Experiment $\mathsf{faWH}_{\mathcal{A},\mathsf{FAS}}$
1 : $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(X, x) \leftarrow \mathsf{GenR}(1^\lambda)$, $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{crs}, X, x)$
2 : $(\mathsf{vk}, m, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AuxGen}}(\cdot)}(\mathsf{crs}, \mathsf{advt}, X)$
3 : If $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma}) = 0$: **ret** $0$
4 : $\sigma := \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, x, f, \mathsf{aux}_f, \widetilde{\sigma})$
5 : $x' \leftarrow \mathcal{A}(\sigma)$
6 : **ret** $1$ iff $(X, x') \in R$

Oracle $\mathcal{O}_{\mathsf{AuxGen}}(f)$
1 : **ret** $\mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, f)$
```

**Figure 9:** Witness Hiding experiment of adaptor signatures

## 5 Strongly-Secure FAS Construction

In this section, we build a generic construction of FAS w.r.t. digital signature scheme $\mathsf{DS}$, any NP relation $R$ with statement/witness pairs $(X, \mathbf{x}) \in R$ such that $\mathbf{x} \in \mathcal{M}$ for some set $\mathcal{M} \subseteq \mathbb{Z}^\ell$, and function family $\mathcal{F}_{\mathsf{IP},\ell}$ for computing inner products of vectors of length $\ell$. Our generic construction of FAS is presented in Figure 2 [8] and it uses the following building blocks:

- A selective, IND-secure $\mathsf{IPFE}$ satisfying $R_{\mathsf{IPFE}}$-compliance and $R'_{\mathsf{IPFE}}$-robustness (Definitions 3.11, 3.13 and 3.14) w.r.t. hard relations $R_{\mathsf{IPFE}}$ and $R'_{\mathsf{IPFE}}$ such that $R_{\mathsf{IPFE}} \subseteq R'_{\mathsf{IPFE}}$. The message space of $\mathsf{IPFE}$ is $\mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$ and the function family is $\mathcal{F}_{\mathsf{IP},\ell+1}$.

- An adaptor signature scheme $\mathsf{AS}$ w.r.t. a digital signature scheme $\mathsf{DS}$, and hard relations $R_{\mathsf{IPFE}}$ and $R'_{\mathsf{IPFE}}$ that satisfies witness extractability (Definition 3.7) and weak pre-signature adaptability (Definition 3.8) security properties.

- A non-interactive zero-knowledge argument system $\mathsf{NIZK}$ in the common reference string model for the NP language $L_{\mathsf{NIZK}}$:

$$L_{\mathsf{NIZK}} := \left\{ \begin{array}{c} (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}) : \\ \exists (r_0, r_1, \mathbf{x}) \text{ such that } \mathsf{pp}' \in [\mathsf{IPFE}.\mathsf{Gen}(1^\lambda)], \\ (\mathsf{mpk}, \mathsf{msk}) = \mathsf{IPFE}.\mathsf{Setup}(\mathsf{pp}', 1^{\ell+1}; r_0), \\ (X, \mathbf{x}) \in R, \mathsf{ct} = \mathsf{IPFE}.\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}^T, 0)^T; r_1) \end{array} \right\}.$$

We sketch the security proof of unforgeability and zero-knowledge witness privacy below. Correctness and formal proofs are deferred to Appendix A.

**Lemma 5.1.** *Suppose NIZK satisfies correctness,* $\mathsf{AS}$ *satisfies correctness, and* $\mathsf{IPFE}$ *satisfies* $R_{\mathsf{IPFE}}$*-compliance and* $R'_{\mathsf{IPFE}}$*-robustness. Then, the* $\mathsf{FAS}$ *construction in Figure 2 satisfies correctness.*

**Theorem 5.2.** *Let* $\mathcal{F}_{\mathsf{IP},\ell}$ *be the family of inner products functions of vectors of length* $\ell$*. Let* $R$ *be any NP relation with statement/witness pairs* $(X, \mathbf{x})$ *such that* $\mathbf{x} \in \mathcal{M}$ *for some set* $\mathcal{M} \subseteq \mathbb{Z}^\ell$*. Suppose that*

- $\mathcal{M}$ *is an additive group,* $R$ *is* $\mathcal{F}_{\mathsf{IP},\ell}$*-hard (Definition 3.2),*

- $\mathsf{NIZK}$ *is a secure NIZK argument system (Definition 3.9),*

[8] Figure 2 is presented with $\mathcal{M} = \mathbb{Z}_p^\ell$, but the scheme generalizes to $\mathcal{M} \subseteq \mathbb{Z}^\ell$ as well.

- $\mathsf{AS}$ *is an adaptor signature scheme w.r.t. digital signature scheme* $\mathsf{DS}$ *and hard relations* $R_{\mathsf{IPFE}}, R'_{\mathsf{IPFE}}$ *satisfying weak pre-signature adaptability (Definition 3.8), witness extractability (Definition 3.7),*

- $\mathsf{IPFE}$ *is a selective, IND-secure IPFE scheme (Definition 3.11) for function family* $\mathcal{F}_{\mathsf{IP},\ell+1}$ *satisfying* $R_{\mathsf{IPFE}}$*-compliance (Definition 3.13),* $R'_{\mathsf{IPFE}}$*-robustness (Definition 3.14).*

*Then, the construction in Figure 2 is a strongly-secure (Definition 4.3) functional adaptor signature scheme w.r.t. digital signature scheme* $\mathsf{DS}$*, NP relation* $R$*, and family of inner product functions* $\mathcal{F}_{\mathsf{IP},\ell}$*.*

**Proof sketch of Unforgeability.** A natural idea for proving unforgeability of $\mathsf{FAS}$ would be to somehow reduce it to unforgeability of $\mathsf{AS}$. But, this requires non-blackbox use of the underlying $\mathsf{AS}$, and thus results in a complex proof. We avoid that altogether and present a counter-intuitive yet elegant way of proving unforgeability of $\mathsf{FAS}$ by reducing it to witness extractability of $\mathsf{AS}$. Our proof technique does not need to open up the blackbox of $\mathsf{AS}$ as evident in the sequence of games below.

In a little more detail, to show that $\Pr[G_0(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$, we consider a sequence of games $G_0, G_1, G_2$ as decribed in Figure 10.

- Game $G_0$ is the original game $\mathsf{faEUF\text{-}CMA}_{\mathcal{A},\mathsf{FAS}}$, where the adversary $\mathcal{A}$ has to come up with a valid forgery on a message $m^*$ of his choice, while having access to functional pre-sign oracle $\mathcal{O}_{\mathsf{fpS}}$ and sign oracle $\mathcal{O}_S$. Here, variables $\mathsf{Bad}_1, \mathsf{Bad}_2$ do not affect the game and are used only to aide the analysis below.

- Game $G_1$ is same as $G_0$, except that before computing the pre-signature, the game checks if the NIZK statement $(X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$ is in the language $L_{NIZK}$. If no, the game sets the flag $\mathsf{Bad}_1 = \mathsf{true}$ and returns 0. Observe that $G_0$ and $G_1$ are identical until $G_1$ checks the condition for setting $\mathsf{Bad}_1$. Hence,

$$|\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| \leq \Pr[\mathsf{Bad}_1 \text{ in } G_1].$$

  Thus, for proving $|\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$, it suffices to show that in game $G_1$, $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$. $\mathsf{Bad}_1 = \mathsf{true}$ implies $\mathsf{stmt} \notin L_{NIZK}$ and $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1$. Thus, the probability bound follows from the adaptive soundness of NIZK argument system.

- Game $G_2$ is same as $G_1$, except that when $\mathcal{A}$ outputs the forgery $\sigma^*$, the game extracts the witness $z$ of the underlying adaptor signature scheme for the statement $\mathsf{pk}_{\mathbf{y}^*}$ and checks if the NIZK statement $(\mathsf{aux}^*_{\mathbf{y}}, z)$ satisfies $R'_{\mathsf{IPFE}}$. If no, the game sets the flag $\mathsf{Bad}_2 = \mathsf{true}$ and returns 0. Observe that $G_0$ and $G_1$ are identical until $G_1$ checks the condition for setting $\mathsf{Bad}_1$. Hence,

$$|\Pr[G_1(1^\lambda) = 1] - \Pr[G_2(1^\lambda) = 1]| \leq \Pr[\mathsf{Bad}_2 \text{ in } G_2].$$

  Thus, for proving $|\Pr[G_1(1^\lambda) = 1] - \Pr[G_2(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$, it suffices to show that in game $G_2$, $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$. This follows from the witness extractability of the underlying adaptor signature scheme $\mathsf{AS}$. In a little more detail, we can show that if an adversary $\mathcal{A}$ successfully causes $G_2$ to set $\mathsf{Bad}_2$, then, we can use it to come up with a reduction $\mathcal{B}$ that breaks the witness extractability of the underlying adaptor signature scheme. This is because $\mathsf{Bad}_2 = \mathsf{true}$ implies $(m^* \notin \mathcal{Q}) \wedge ((\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*)$, i.e., the winning condition for $\mathcal{B}$.

- Lastly, we can show that $\Pr[G_2(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$ assuming $\mathcal{F}_{\mathsf{IP},\ell}$-hardness of relation $R$ and $R'_{\mathsf{IPFE}}$-robustness of the $\mathsf{IPFE}$ scheme. Essentially, we show that if $\mathcal{A}$ wins $G_2$, then, we can

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

1 :  $\mathcal{Q} := \emptyset$, $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
2 :  $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}')$, $(X^*, \mathbf{x}^*) \leftarrow \mathsf{GenR}(1^\lambda)$
3 :  $(\mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk}, X^*)$
4 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
5 :  $\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$, $\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
6 :  $\mathsf{Bad}_1 := \mathsf{false}, \mathsf{Bad}_2 := \mathsf{false}$
7 :  If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \ \vee \ \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell} \ \vee \ \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$:
         **ret** $0$
8 :  $\boxed{\boxed{\text{If } \mathsf{stmt} \notin L_{NIZK}\text{: } \mathsf{Bad}_1 := \mathsf{true}, \textbf{ ret } 0}}$
9 :  $\widetilde{\sigma}^* \leftarrow \mathsf{AS.PreSign}(\mathsf{sk}, m^*, \mathsf{aux}^*_{\mathbf{y}})$, $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
10 : $\boxed{z = \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})}$
11 : $\boxed{\text{If } (m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge ((\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}})\text{:}}$
12 :       $\boxed{\mathsf{Bad}_2 := \mathsf{true}, \textbf{ ret } 0}$
13 : **ret** $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*))$

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

1 :  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, $\mathcal{Q} := \mathcal{Q} \vee \{m\}$, **ret** $\sigma$

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$

1 :  If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 0$: **ret** $\perp$
2 :  $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}})$, $\mathcal{Q} := \mathcal{Q} \vee \{m\}$, **ret** $\widetilde{\sigma}$
```

**Figure 10:** Unforgeability Proof: Games $G_0, G_1, G_2$

build a reduction $\mathcal{B}$ that breaks the $\mathcal{F}_{\mathsf{IP},\ell}$-hardness of relation $R$. $\mathcal{B}$ outputs $(\mathbf{y}^*, v)$, where $v = \mathsf{IPFE.Dec}(z, \mathsf{ct})$. To win, $\mathcal{B}$'s output must satisfy $(\mathbf{y}^* \in \mathcal{F}_{\mathsf{IP},\ell}) \wedge (v \in \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists \mathbf{x} \ s.t. \ (X, \mathbf{x}) \in R\})$. If $\mathcal{A}$ wins, then $\mathbf{y}^* \in \mathcal{F}_{\mathsf{IP},\ell}$ and $\mathsf{pk}_{\mathbf{y}^*} = \mathsf{aux}^*_{\mathbf{y}}$. Next, $\mathsf{Bad}_1 = \mathsf{false}$ implies that $\mathsf{stmt} \in L_{NIZK}$, where $\mathsf{stmt} = (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$. Hence, it follows that $\mathsf{ct}$ encrypts some vector $\widetilde{\mathbf{x}} = (\mathbf{x}^T, 0)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$ under $\mathsf{mpk}$ such that $(X^*, \mathbf{x}) \in R$. Next, $\mathsf{Bad}_2 = \mathsf{false}$ implies that $(\mathsf{aux}^*_{\mathbf{y}}, z) \in R'_{\mathsf{IPFE}}$. As $\mathsf{pk}_{\mathbf{y}^*} = \mathsf{aux}^*_{\mathbf{y}}$ and IPFE satisfies $R'_{\mathsf{IPFE}}$-robustness, it follows that $v = f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}})$. As $\widetilde{\mathbf{y}^*} = (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$ and $\widetilde{\mathbf{x}} = (\mathbf{x}^T, 0)^T$, we get that $f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}}) = f_{\mathbf{y}^*}(\mathbf{x})$. Hence, we conclude that $v \in \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists \mathbf{x} \ s.t. \ (X, \mathbf{x}) \in R\}$. Thus, $\mathcal{B}$ wins its game. This completes the proof.

**Proof sketch of Zero-Knowledge.** We first describe the stateful simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{AuxGen}^*, \mathsf{Adapt}^*)$. Let the NIZK simulator be $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Setup}^*, \mathsf{NIZK.Prove}^*)$. Then, the simulator $\mathsf{Sim}$ is as follows.

- $\mathsf{Setup}^*(1^\lambda)$: same as $\mathsf{Setup}$ except $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)$ and trapdoor $\mathsf{td}$ is stored as internal state by $\mathsf{Sim}$.

- $\mathsf{AdGen}^*(\mathsf{pp}, X)$: same as $\mathsf{AdGen}$ except $\widetilde{\mathbf{x}} := (-\mathbf{t}^T, 1)^T \in \mathbb{Z}_p^{\ell+1}$ and $\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))$.
- $\mathsf{AuxGen}^*(\mathsf{advt}, \mathbf{y}, f_{\mathbf{y}}(\mathbf{x}))$: same as $\mathsf{AuxGen}$ except $\mathsf{pk}_{\mathbf{y}}$ is computed for $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$ and thus $\pi_{\mathbf{y}} := f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x})$ as well.
- $\mathsf{Adapt}^*(\mathsf{advt}, \mathsf{vk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \widetilde{\sigma}, f_{\mathbf{y}}(\mathbf{x}))$: same as $\mathsf{Adapt}$ except $\mathsf{sk}_{\mathbf{y}}$ for $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T$ is used.

To see that adversary's views in real and ideal world ( Figure 7) are indistinguishable, consider a sequence of games $G_0, G_1, G_2, G_3$.

- Game $G_0$ corresponds to the real-world experiment where $\mathsf{Setup}$, $\mathsf{AdGen}$, $\mathsf{AuxGen}$, $\mathsf{Adapt}$ are used.
- Game $G_1$ is same as $G_0$, except the NIZK is switched to simulation mode, i.e., we change $\mathsf{Setup}$ to perform $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)$ and change $\mathsf{AdGen}$ to perform $\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))$. It follows from the zero-knowledge property of NIZK that $G_0 \approx_c G_1$.
- Game $G_2$ is same as $G_1$ except $\widetilde{\mathbf{y}}$ used in $\mathsf{AuxGen}$ and $\mathsf{Adapt}$ is switched from $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}))^T$ to $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T$. Since $f$ is a linear function, it follows that $f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}) = f_{\mathbf{y}}(\mathbf{t} + \mathbf{x})$, thus, one can view the transition to $G_2$ as a change of variables $\mathbf{t} \to \mathbf{t} + \mathbf{x}$. Since $\mathbf{t}$ is uniform random, it follows that $G_1$ and $G_2$ are identically distributed.
- Game $G_3$ is same as $G_2$ except that in $\mathsf{AdGen}$, $\mathsf{ct}$ encrypts $\widetilde{\mathbf{x}} := (-\mathbf{t}^T, 1)^T$ instead of $\widetilde{\mathbf{x}} := (\mathbf{x}^T, 0)^T$. Observe that this does not change the inner product value $\widetilde{\mathbf{x}}^T\widetilde{\mathbf{y}}$. Thus, $G_2 \approx_c G_3$ follows from IND-security of the IPFE scheme.
- Lastly observe that adversary's view in $G_3$ is same as that in the ideal-world experiment. This complete the proof.

We emphasize that in $G_3$, even the underlying IPFE only uses the information $f_{\mathbf{y}}(\mathbf{x})$ about $\mathbf{x}$. Thus, going from $G_1$ to $G_3$ is essentially building an IND-security to simulation-security IPFE compiler. The random coins of this compiler are $\mathbf{t}$. Crucially, $G_2$ and $G_3$ explicitly use $f_{\mathbf{y}}(\mathbf{t})$, which is a leakage on these random coins and for this reason the IPFE compiler is used in a non-blackbox way. We refer the reader back to Remark 2.1 on why $f_{\mathbf{y}}(\mathbf{t})$ is an acceptable leakage on the IPFE simulator's random coins $\mathbf{t}$.

## 6 FAS from Groups of Prime-Order

We provide instantiations of the FAS construction in Figure 2 from prime order groups. For this, it suffices to instantiate the building blocks $\mathsf{IPFE}$ and $\mathsf{AS}$ from prime order groups, while ensuring the two are compatible with each other. We describe these next. We can instantiate the $\mathsf{NIZK}$ for the language $L_{\mathsf{NIZK}}$ depending on the concrete relation $R$. That is, given $R$ and the above instantiation of $\mathsf{IPFE}$, we can pick the most efficient $\mathsf{NIZK}$ for $L_{\mathsf{NIZK}}$. Since this is not the main contribution of this work, we will assume $R$ to be a general NP relation and rely on the existence of $\mathsf{NIZK}$ for general NP [JJ21, PS19] and not delve into its details here.

More concretely, we instantiate the $\mathsf{AS}$ as Schnorr adaptor signature scheme [AEE+21] and thus, we have $R_{\mathsf{IPFE}} = R'_{\mathsf{IPFE}} = R_{\mathsf{DL}}$ as defined in Definition 3.3. Further, we show that the selective, IND-secure IPFE scheme by Abdalla et al. [ABDP15] satisfies $R_{\mathsf{DL}}$-compliance and $R_{\mathsf{DL}}$-robustness when appropriately augmented with a $\mathsf{PubKGen}$ algorithm.

## 6.1 Schnorr Adaptor Signature

First, we recall the Schnorr Signature Scheme $\mathsf{Sch}$ [Sch90]. Suppose $\mathsf{pp} := (\mathbb{G}, p, g) \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$ as described in Figure 11, where $\mathbb{G}$ is a cyclic group of prime order $p$ and $g$ is a generator of $\mathbb{G}$. Suppose $H : \{0,1\}^* \to \mathbb{Z}_p$ is a hash function modeled as a random oracle. Then $\mathsf{Sch}$ is described below:

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sch.KGen}(\mathsf{pp})$: choose $\delta \leftarrow \mathbb{Z}_p$ and set $\mathsf{sk} := \delta$ and $\mathsf{vk} := g^\delta$.
- $\sigma \leftarrow \mathsf{Sch.Sign}(\mathsf{sk}, m)$: sample a randomness $r \leftarrow \mathbb{Z}_p$ to compute $h := H(\mathsf{vk}||g^r||m), s := r + h\delta$ and output $\sigma := (h, s)$.
- $0/1 \leftarrow \mathsf{Sch.Vf}(\mathsf{vk}, m, \sigma)$: parse $\sigma := (h, s)$ and then compute $R := g^s/\mathsf{vk}^h$ and if $h = H(\mathsf{vk}||R||m)$ output 1, otherwise output 0.

Next, we describe the Adaptor Signature scheme $\mathsf{AS}$ with respect to the digital signature scheme $\mathsf{Sch}$ and hard relations $R = R' = R_{\mathsf{DL}}$, where $R_{\mathsf{DL}}$ is the discrete log relation. Let $L_{\mathsf{DL}}$ be the corresponding language. Then, $\mathsf{AS}$ is described below:

- $\widetilde{\sigma} \leftarrow \mathsf{AS.PreSign}(\mathsf{sk}, m, X \in L_{\mathsf{DL}})$: choose $r \xleftarrow{\$} \mathbb{Z}_p$ and compute $h := H(\mathsf{vk}||g^r \cdot X||m)$ and $\widetilde{s} := r + h\delta$. Set $\widetilde{\sigma} := (h, \widetilde{s})$.
- $0/1 \leftarrow \mathsf{AS.PreVerify}(\mathsf{vk}, m, X, \widetilde{\sigma})$: parse $\widetilde{\sigma} := (h, \widetilde{s})$ and compute $R := g^s/\mathsf{vk}^h$. If $h = H(\mathsf{vk}||R \cdot X||m)$ **ret** 1, else **ret** 0.
- $\sigma \leftarrow \mathsf{AS.Adapt}(\mathsf{vk}, m, X, x, \widetilde{\sigma})$: parse $\widetilde{\sigma} = (h, \widetilde{s})$ and compute $s := \widetilde{s} + x$. **ret** $\sigma := (h, s)$
- $x' \leftarrow \mathsf{AS.Ext}(\widetilde{\sigma}, \sigma, X)$: parse $\widetilde{\sigma} = (h, \widetilde{s})$ and $\sigma = (h, s)$. Compute $x' := s - \widetilde{s}$ and if $(X, x') \notin R_{\mathsf{DL}}$, **ret** $\perp$, else **ret** $x'$.

**Lemma 6.1** ( [AEE⁺21]). *The adaptor signature scheme in Section 6.1 satisfies witness extractability (Definition 3.7) and weak pre-signature adaptability (Definition 3.8).*

## 6.2 IPFE from Groups of Prime-Order

We first recall the IPFE scheme by Abdalla et al. [ABDP15]. Then, we show that it satisfies the additional compliance and robustness properties needed by our FAS scheme.

Suppose $p$ is a $\lambda$-bit prime number and suppose we want to compute inner products of vectors of length $\ell$. The Abdalla et al. [ABDP15] IPFE scheme computes inner products with output values polynomially bounded by $B \ll p$. Specifically, the message space is $\mathcal{M} = \mathbb{Z}_p^\ell$ and the function class is $\mathcal{F}_{\mathsf{IP},\ell,p,B} = \{f_{\mathbf{y}} : \mathbf{y} \in \mathbb{Z}_p^\ell\} \subseteq \mathcal{F}_{\mathsf{IP},\ell}$, where $f_{\mathbf{y}} : \mathbb{Z}_p^\ell \to \{0, \ldots, B\}$ is defined as $f_{\mathbf{y}}(\mathbf{x}) = \mathbf{x}^T\mathbf{y} \in \{0, \ldots, B\}$. The scheme by Abdalla et al. [ABDP15] augmented with $\mathsf{PubKGen}$ algorithm is as in Figure 11. For $d \in \mathbb{G}$, we denote the algorithm for computing the discrete log of $d$ with respect to base $g \in \mathbb{G}$ by $\mathsf{DLog}_g(d)$. As long as we are promised that the discrete log is bounded by a polynomial $B$, the $\mathsf{DLog}$ algorithm simply enumerates all the possibilities to find the discrete log. The running time is polynomial because of the bound $B$.

**Lemma 6.2** ( [ABDP15]). *Suppose the DDH assumption holds. Then, the IPFE scheme in Figure 11 is selective, IND-secure (Definition 3.11).*

Next, we show the IPFE scheme augmented with the above $\mathsf{PubKGen}$ algorithm satisfies $R_{\mathsf{DL}}$-compliance and $R_{\mathsf{DL}}$-robustness.

```
IPFE.Gen($1^\lambda$)
    1 : $\mathsf{pp} := (\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$
    2 : **ret** $\mathsf{pp}$
IPFE.Setup($\mathsf{pp}, 1^\ell$)
    1 : $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^\ell$, **ret** $\mathsf{msk} := \mathbf{s}, \mathsf{mpk} := g^{\mathbf{s}}$
IPFE.Enc($\mathsf{mpk}, \mathbf{x} \in \mathbb{Z}_p^\ell$)
    1 : $r \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{ct}_0 := g^r$, $\mathsf{ct}_1 := g^{\mathbf{x}} \cdot \mathsf{mpk}^r$
    2 : **ret** $\mathsf{ct} := (\mathsf{ct}_0, \mathsf{ct}_1)$
IPFE.KGen($\mathsf{msk}, \mathbf{y} \in \mathbb{Z}_p^\ell$)
    1 : **ret** $\mathsf{sk}_{\mathbf{y}} := \mathbf{s}^T \mathbf{y} \in \mathbb{Z}_p$

IPFE.PubKGen($\mathsf{mpk}, \mathbf{y} \in \mathbb{Z}_p^\ell$)
    1 : Parse $\mathsf{mpk} = (k_1, \ldots, k_\ell)$
    2 : Parse $\mathbf{y} = (y_1, \ldots, y_\ell)$
    3 : **ret** $\mathsf{pk}_{\mathbf{y}} := \prod_{i \in [\ell]} k_i^{y_i}$
IPFE.Dec($\mathsf{sk}_{\mathbf{y}}, \mathsf{ct}$)
    1 : Parse $\mathsf{ct}_1 = (c_1, \ldots, c_\ell)$
    2 : Parse $\mathbf{y} = (y_1, \ldots, y_\ell)$
    3 : $d := \prod_{i \in [\ell]} c_i^{y_i} / \mathsf{ct}_0^{\mathsf{sk}_{\mathbf{y}}}$
    4 : **ret** $v := \mathsf{DLog}_g(d)$
```

**Figure 11:** Abdalla et al. [ABDP15] IPFE scheme augmented with PubKGen algorithm

**Lemma 6.3.** *The IPFE scheme in Figure 11 is $R_{\mathsf{DL}}$-compliant.*

*Proof.* For any $\lambda \in \mathbb{N}$, for any $\mathsf{pp} \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$ as implemented in Figure 11, for any $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}, 1^\ell)$ as implemented in Figure 11, for any $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell,p,B}$, let $\mathsf{pk}_{\mathbf{y}} := \mathsf{PubKGen}(\mathsf{mpk}, \mathbf{y})$ as implemented in Figure 11, and $\mathsf{sk}_{\mathbf{y}} := \mathsf{KGen}(\mathsf{msk}, \mathbf{y})$ as implemented in Figure 11. Suppose here $\mathsf{msk} = \mathbf{s} \in \mathbb{Z}_p^\ell$. Then, $\mathsf{sk}_{\mathbf{y}} = \mathbf{s}^T\mathbf{y} \in \mathbb{Z}_p$ and $\mathsf{pk}_{\mathbf{y}} = g^{\mathbf{s}^T\mathbf{y}} \in \mathbb{G}$. Thus, clearly, $(\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}) \in R_{\mathsf{DL}}$. □

**Lemma 6.4.** *The IPFE scheme in Figure 11 is $R_{\mathsf{DL}}$-robust.*

*Proof.* Observe that in the IPFE scheme in Figure 11, the base relation $R$ and the extended relation $R'$ are the same: $R = R' = R_{\mathsf{DL}}$. Further, $R_{\mathsf{DL}}$ is a unique witness relation. Thus, $R_{\mathsf{DL}}$-robustness follows trivially from the correctness of the IPFE scheme. □

## 6.3 FAS Construction

Instantiating FAS construction in Section 5 with IPFE from Figure 11 and AS from Section 6.1, we obtain the following corollary.

**Corollary 6.5.** *Let $p$ be a $\lambda$-bit prime number and let $\ell$ be an integer. Let $\mathcal{M} = \mathbb{Z}_p^\ell$ be an additive group. let $\mathcal{F}_{\mathsf{IP},\ell,\mathsf{p},\mathsf{B}}$ be the function family for computing inner products of vectors in $\mathbb{Z}_p^\ell$ such that the output value is bounded by some polynomial $B \ll p$. Let $R$ be any NP relation with statement/witness pairs $(X, \mathbf{x})$ such that $\mathbf{x} \in \mathbb{Z}_p^\ell$. Suppose that*

- *$R$ is $\mathcal{F}_{\mathsf{IP},\ell,p,B}$-hard (Definition 3.2),*
- NIZK *is a secure NIZK argument system (Definition 3.9),*
- AS *construction in Section 6.1 is an adaptor signature scheme w.r.t. digital signature scheme* Sch *and hard relation $R_{\mathsf{DL}}$ that satisfies weak pre-signature adaptability and witness extractability (Lemma 6.1),*
- IPFE *construction in Figure 11 is a selective, IND-secure IPFE scheme (Lemma 6.2) for function family $\mathcal{F}_{\mathsf{IP},\ell+1}$ that is $R_{\mathsf{IPFE}}$-compliant (Lemma 6.3) and $R_{\mathsf{IPFE}}$-robust (Lemma 6.4).*

*Then, the functional adaptor signature scheme w.r.t. Schnorr signature scheme* $\mathsf{Sch}$*, NP relation* $R$*, and family of inner product functions* $\mathcal{F}_{\mathsf{IP},\ell,p,B}$ *constructed in Figure 2 is strongly-secure (Definition 4.3).*

The proof of the above corollary is immediate from the proof of the Theorem 5.2 concerning the generic construction, and Lemmas 6.2 to 6.4 that show that the $\mathsf{IPFE}$ scheme in Figure 11 has the required properties.

## 7 FAS from lattices

In this section, we provide instantiations of the FAS construction in Figure 2 from lattices. For this, it suffices to instantiate the building blocks IPFE and AS from lattices, while ensuring the two are compatible with each other. We describe these next.

More specifically, we instantiate the $\mathsf{AS}$ as the lattice-based adaptor signature scheme by Esgin et al. [EEE20a]. This scheme was built using cyclotomic ring $\mathcal{R} = \mathbb{Z}[x]/(x^d + 1)$ of degree $d = 256$ under Module-SIS and Module-LWE assumptions. Here, we set the degree to be $d = 1$, thus giving us the ring of integers $\mathcal{R} = \mathbb{Z}$. Consequently, security follows from plain SIS and LWE assumptions. One can look at the resulting $\mathsf{AS}$ to be w.r.t. a digital scheme $\mathsf{Lyu}'$ that is somewhere in between Lyubashevsky's signature scheme [Lyu12] and Dilithium [DKL$^+$18] instantiated with unstructured lattices. Further, $\mathsf{AS}$ is w.r.t. hard relations $R_{\mathsf{ISIS}}$ and $R'_{\mathsf{ISIS}}$ such that $R_{\mathsf{ISIS}} \subset R'_{\mathsf{ISIS}}$.

Further, we show that the IND-secure IPFE scheme by Agrawal et al. [ALS16, Section 4.2] satisfies $R_{\mathsf{ISIS}}$-compliance and $R'_{\mathsf{ISIS}}$-robustness when appropriately augmented with a $\mathsf{PubKGen}$ algorithm. We describe the instantiations of these two building blocks in the subsequent sections.

### 7.1 Lattice-based Adaptor Signature

First, we recall the lattice-based signature scheme $\mathsf{Lyu}'$, which can be seen as a variant of Lyubashevsky's signature scheme [Lyu12] where the vector $\mathbf{y}$ sampled during signing comes from a uniform distribution over a small range instead of discrete gaussian distribution. It can also be seen as a variant of Dilithium signature scheme [DKL$^+$18] instantiated over unstructured lattices.

Suppose $\mathsf{pp} := (n, m, p, q, k, \alpha) \leftarrow \mathsf{IPFE.Gen}(1^\lambda, p)$ as described in Figure 14, where $p$ is a prime number, $n = \lambda$, $q = p^k$ and $k \in \mathbb{Z}$, $m \in \mathbb{Z}$, real $\alpha \in (0, 1)$ are chosen as described in the "Parameter choices" in Section 7.2. Let $\mathcal{H} : \{0,1\}^* \to \mathcal{C}$ be a family of hash functions (modelled as a random oracle), where $\mathcal{C} = \{\mathbf{c} \in \mathbb{Z}^h : ||\mathbf{c}||_1 = \kappa \wedge ||\mathbf{c}||_\infty = 1\}$. Let $\gamma$ be the maximum absolute coefficient of the masking randomness $\mathbf{y}$ used in the $\mathsf{Sign}$ algorithm below. Then, the digital signature scheme $\mathsf{Lyu}'$ is as in Figure 12.

```
Lyu'.KGen(1^λ, pp)
  1 : Sample $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$
  2 : Let $\mathbf{A} := (I_n || \mathbf{A}') \in \mathbb{Z}_q^{n\times(n+m)}$
  3 : Sample $\mathbf{R} \xleftarrow{\$} \mathcal{S}_1^{(n+m)\times h}$
  4 : Let $\mathbf{T} := \mathbf{AR} \bmod q \in \mathbb{Z}_q^{n\times h}$
  5 : ret $\mathsf{vk} := (\mathbf{A}, \mathbf{T}), \mathsf{sk} := \mathbf{R}$

Lyu'.Sign(sk, μ ∈ {0,1}*)
  1 : Sample $\mathbf{y} \xleftarrow{\$} \mathcal{S}_\gamma^{n+m}$
  2 : Let $\mathbf{w} := \mathbf{Ay}$
  3 : Let $\mathbf{c} := \mathsf{H}(\mathsf{vk} \,||\, \mathbf{w} \,||\, \mu) \in \mathbb{Z}^h$
  4 : Let $\mathbf{z} := \mathbf{y} + \mathbf{Rc}$
  5 : If $||\mathbf{z}||_\infty > \gamma - \kappa$: restart
  6 : ret $\sigma := (\mathbf{c}, \mathbf{z})$

Lyu'.Vf(vk, μ, σ)
  1 : Parse $\sigma = (\mathbf{c}, \mathbf{z})$
  2 : If $||\mathbf{z}||_\infty > \gamma - \kappa$: ret 0
  3 : Let $\mathbf{w}' := \mathbf{Az} - \mathbf{Tc}$
  4 : If $\mathbf{c} \neq \mathsf{H}(\mathsf{vk} \,||\, \mathbf{w}' \,||\, \mu)$: ret 0
  5 : ret 1
```

**Figure 12:** Digital signature scheme $\mathsf{Lyu}'$ from $\mathsf{SIS}$ and $\mathsf{LWE}$

Next, we describe the Adaptor Signature scheme $\mathsf{AS}$ with respect to the digital signature scheme $\mathsf{Lyu}'$ and hard relations $R_{\mathsf{ISIS}} := \mathsf{ISIS}_{n,n+m,q,\beta_0}$ and $R'_{\mathsf{ISIS}} := \mathsf{ISIS}_{n,n+m,q,\beta_1}$ such that $R_{\mathsf{ISIS}} \subseteq R'_{\mathsf{ISIS}}$. Here, $\beta_0 := \ell p B_\tau$ and $\beta_1 := 2(\gamma - \kappa) - \beta_0$ such that $\beta_1 > \beta_0$ and $\gamma - \kappa - \beta_0 > 0$. Let $L_{\mathsf{ISIS}}$ be the language corresponding to $R_{\mathsf{ISIS}}$ and $R'_{\mathsf{ISIS}}$. Notice that $L_{\mathsf{ISIS}} \subseteq \mathbb{Z}_q^n$. Hence, for $R_{\mathsf{ISIS}}$, statements are of the form $(\mathbf{A}, X) \in \mathbb{Z}_q^{n\times(n+m)} \times \mathbb{Z}_q^n$ (same for $'_{\mathsf{ISIS}}$) and witnesses are of the form $\mathbf{x} \in \mathbb{Z}^{n+m}$ such that $||\mathbf{x}||_\infty \leq \beta_0$ ($||\mathbf{x}||_\infty \leq \beta_1$ for $R'_{\mathsf{ISIS}}$). As $\mathbf{A}$ is also part of the verification key, henceforth we drop it from the statement. We next describe the $\mathsf{AS}$ construction in Figure 13.

```
AS.PreSign(sk, $\mu \in \{0,1\}^*, X \in L_{\mathsf{ISIS}}$)
  1 : Sample $\mathbf{y} \xleftarrow{\$} \mathcal{S}_\gamma^{n+m}$, let $\mathbf{w} := \mathbf{A}\mathbf{y}$
  2 : Let $\mathbf{c} := \mathsf{H}(\mathsf{vk} \,||\, \mathbf{w} + X \,||\, \mu) \in \mathbb{Z}^h$
  3 : Let $\widetilde{\mathbf{z}} := \mathbf{y} + \mathbf{R}\mathbf{c}$
  4 : If $||\widetilde{\mathbf{z}}||_\infty > \gamma - \kappa - \beta_0$: restart
  5 : ret $\widetilde{\sigma} := (\mathbf{c}, \widetilde{\mathbf{z}})$

AS.Adapt(vk, $\mu, X, x, \widetilde{\sigma}$)
  1 : If PreVerify(vk, $\mu, X, \widetilde{\sigma}$) = 0: ret $\perp$
  2 : Parse $\widetilde{\sigma} = (\mathbf{c}, \widetilde{\mathbf{z}})$
  3 : $\mathbf{z} := \widetilde{\mathbf{z}} + x$
  4 : ret $\sigma := (\mathbf{c}, \mathbf{z})$
```

```
AS.PreVerify(vk, $\mu, X, \widetilde{\sigma}$)
  1 : Parse $\widetilde{\sigma} = (\mathbf{c}, \widetilde{\mathbf{z}})$
  2 : If $||\widetilde{\mathbf{z}}||_\infty > \gamma - \kappa - \beta_0$: ret 0
  3 : Let $\mathbf{w}' := \mathbf{A}\widetilde{\mathbf{z}} - \mathbf{T}\mathbf{c}$
  4 : If $\mathbf{c} \neq \mathsf{H}(\mathsf{vk} \,||\, \mathbf{w}' \,||\, \mu)$: ret 0
  5 : ret 1

AS.Ext($\widetilde{\sigma}, \sigma, X$)
  1 : Parse $\widetilde{\sigma} = (\mathbf{c}, \widetilde{\mathbf{z}}), \sigma = (\mathbf{c}, \mathbf{z})$
  2 : $x' := \mathbf{z} - \widetilde{\mathbf{z}}$
  3 : If $X \neq \mathbf{A}x' \bmod q$: ret $\perp$
  4 : ret $x'$
```

**Figure 13:** $\mathsf{AS}$ w.r.t. $\mathsf{Lyu}'$ signature scheme and hard relations $R_{\mathsf{ISIS}}$ and $R'_{\mathsf{ISIS}}$

**Parameter choices and comparison.** Our construction can be seen as similar to [EEE20a] but with degree 1 instead of 256. Our construction is different from [EEE20a] in the following aspects though: [EEE20a] chooses $\beta_0 = 1$, whereas we choose $\beta_0 = \ell p B_\tau$, where parameters $\ell, p, B_\tau$ are chosen based on the specification in the IPFE construction in Figure 14. Further, [EEE20a] sets $\beta_1 = 2(\gamma - \kappa)$ whereas we set it a stricter bound on it $\beta_1 = 2(\gamma - \kappa) - \beta_0$ as it is sufficient for correctness and security. Further, note that for the choice of $\beta_1$, the two constraints $\beta_1 > \beta_0$ and $\gamma - \kappa - \beta_0 > 0$ are equivalent. Lastly, we note that just like in [EEE20a], we can set $\gamma = 2\kappa$ to ensure that the average number of restarts in $\mathsf{Sign}$ and $\mathsf{PreSign}$ is about $e < 3$, where $\kappa$ is the $\ell_1$ norm of values in the range set $\mathcal{C}$ of the hash family $\mathcal{H}$. Lastly, in [EEE20a], the hash outputs belong to some polynomial ring $\mathcal{R}_q$, whereas we choose the range of hash outputs to be $\mathbb{Z}_q^h$ for some integer $h$.

**Lemma 7.1** ( [EEE20a]). *The adaptor signature scheme in Figure 13 satisfies witness extractability (Definition 3.7) and weak pre-signature adaptability (Definition 3.8).*

## 7.2 IPFE from Lattices

We first recall the IPFE scheme by Agrawal et al. [ALS16]. Then, we show that it satisfies the additional compliance and robustness properties needed by our FAS construction.

Suppose $p$ is a prime number and suppose we want to compute inner products of vectors of length $\ell$. Let the set of messages be $\mathcal{M} = \mathbb{Z}_p^\ell$ and the function class be $\mathcal{F}_{\mathsf{IP},\ell,p} = \{f_{\mathbf{y}} : \mathbf{y} \in \mathbb{Z}_p^\ell\}$, where $f_{\mathbf{y}} : \mathbb{Z}_p^\ell \to \mathbb{Z}_p$ is defined as $f_{\mathbf{y}}(\mathbf{x}) = \mathbf{x}^T\mathbf{y} \bmod p$. Our IPFE scheme augmented with $\mathsf{PubKGen}$ algorithm is as in Figure 14.

**Remark 7.2.** *We note that the original construction needs* KGen *to be stateful in order to support key generation queries that are linearly dependent modulo $p$. This is problematic for us as the corresponding* PubKGen *will also have to be stateful then, but that defeats the purpose of this algorithm being public. So, in order to remedy the situation, we assume that all* KGen *queries to* IPFE *are going to be linearly independent. Naively instantiating our* FAS *construction (Figure 2) will result in* FAS *only supporting linearly independent functions, which is acceptable if it involves a single buyer but in the most general case we would want to support multiple buyers who are agnostic of each other. Specifically, a buyer may engage in* FAS *protocol to learn a function evaluation of the secret that is linearly dependent on something learnt by other some other buyers. To handle such a scenario, we show in Section 7.4 how to assign unique IDs to each buyer and make the underlying* IPFE *key generation queries linearly independent.*

```
$\mathsf{IPFE.Gen}(1^\lambda, p)$
  1 : Let $n = \lambda$, integers $m, k$, real $\alpha \in (0,1)$ as defined below
  2 : Let $q = p^k$, ret $\mathsf{pp} = (n, m, p, q, k, \alpha)$

$\mathsf{IPFE.Setup}(\mathsf{pp}, 1^\ell)$
  1 : Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $\mathbf{Z} \leftarrow \tau$ where
      distribution $\tau$ over $\mathbb{Z}^{\ell \times m}$ is as defined below
  2 : Let $\mathbf{U} := \mathbf{Z}\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$, ret $\mathsf{mpk} := (\mathbf{A}, \mathbf{U}), \mathsf{msk} := \mathbf{Z}$

$\mathsf{IPFE.Enc}(\mathsf{mpk}, \mathbf{x} \in \mathbb{Z}_p^\ell)$
  1 : Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow D_{\mathbb{Z},\alpha q}^m$, $\mathbf{e}_1 \leftarrow D_{\mathbb{Z},\alpha q}^\ell$
  2 : Let $\mathsf{ct}_0 := \mathbf{A}\mathbf{s} + \mathbf{e}_0 \in \mathbb{Z}_q^m$, $\mathsf{ct}_1 := \mathbf{U}\mathbf{s} + \mathbf{e}_1 + p^{k-1}\mathbf{x} \in \mathbb{Z}_q^\ell$
  3 : ret $\mathsf{ct} := (\mathsf{ct}_0, \mathsf{ct}_1)$

$\mathsf{IPFE.KGen}(\mathsf{msk}, \mathbf{y} \in \mathbb{Z}_p^\ell)$
  1 : ret $\mathsf{sk}_\mathbf{y} := \mathbf{Z}^T \mathbf{y} \in \mathbb{Z}^m$

$\mathsf{IPFE.PubKGen}(\mathsf{mpk}, \mathbf{y} \in \mathbb{Z}_p^\ell)$
  1 : ret $\mathsf{pk}_\mathbf{y} := \mathbf{U}^T \mathbf{y} \in \mathbb{Z}_q^n$

$\mathsf{IPFE.Dec}(\mathsf{sk}_\mathbf{y}, \mathsf{ct})$
  1 : Let $d := \mathsf{ct}_1^T \mathbf{y} - \mathsf{ct}_0^T \mathsf{sk}_\mathbf{y} \bmod q$
  2 : ret $v \in \mathbb{Z}_p$ that minimizes $| p^{k-1} v - d |$
```

**Figure 14:** Agrawal et al. [ALS16] IPFE scheme augmented with PubKGen algorithm. Note that KGen here is stateless.

**Parameter choices.** Let $B_\tau$ be such that with probability at most $n^{-\omega(1)}$, each row of sample from $\tau$ has $\ell_2$-norm at least $B_\tau$. Then, the parameter constraints for correctness and security are as follows.

- $\alpha^{-1} \geq \ell^2 p^3 B_\tau \omega(\sqrt{\log n})$, $q \geq \alpha^{-1}\omega(\sqrt{\log n})$,
- $\tau = D_{\mathbb{Z},\sigma_1}^{\ell \times m/2} \times (D_{\mathbb{Z}^{m/2},\sigma_2,\delta_1} \times \ldots \times D_{\mathbb{Z}^{m/2},\sigma_2,\delta_\ell})$, where $\delta_i \in Z^\ell$ denotes the $i$-th canonical vector, and the standard deviation parameters satisfy $\sigma_1 = \Theta(\sqrt{n \log m} \max(\sqrt{m}, K'))$ and $\sigma_2 = \Theta(n^{7/2} m 1/2 \max(m, K'^2) \log^{5/2} m)$, with $K' = (\sqrt{\ell} p)^\ell$.

Further, $R'_{\mathsf{ISIS}}$-robustness ( Lemma 7.5) will require the constraint $\alpha^{-1} \geq 4p\omega(\sqrt{\log n})(\ell p + m\beta_1)$, where $\beta_1$ is as chosen below.

**Lemma 7.3** ( [ALS16]). *Suppose $\ell \leq n^{O(1)}$, $m \geq 4n \log_2(q)$ and $q, \alpha, \tau$ are as described above. Suppose* $\mathsf{mheLWE}_{q,\alpha,m,\ell,\tau}$ *assumption (Definition 3.20) holds. Then, the IPFE scheme in Figure 14 is selective, IND-secure (Definition 3.11).*

Next, we show the IPFE scheme augmented with the above $\mathsf{PubKGen}$ algorithm satisfies $R_{\mathsf{ISIS}}$-compliance and $R'_{\mathsf{ISIS}}$-robustness, where $R_{\mathsf{ISIS}} = \mathsf{ISIS}_{n,m,q,\beta_0}$ and $R'_{\mathsf{ISIS}} = \mathsf{ISIS}_{n,m,q,\beta_1}$, where $\beta_0 = \ell p B_\tau$ and $\beta_1 = 2(\gamma - \kappa) - \beta_0$ s.t. $\beta_1 > \beta_0$ and $\gamma - \kappa - \beta_0 > 0$.

**Lemma 7.4.** *The IPFE scheme in Figure 14 is $R_{\mathsf{ISIS}}$-compliant, where $R_{\mathsf{ISIS}} = \mathsf{ISIS}_{n,m,q,\beta_0}$ and $\beta_0 = \ell p B_\tau$.*

*Proof.* For any $\lambda \in \mathbb{N}$, for any $\mathsf{pp} \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$ as implemented in Figure 14, for any $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}, 1^\ell)$ as implemented in Figure 14, for any $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell,p}$, let $\mathsf{pk}_{\mathbf{y}} := \mathsf{PubKGen}(\mathsf{mpk}, \mathbf{y})$ as implemented in Figure 14, and $\mathsf{sk}_{\mathbf{y}} := \mathsf{KGen}(\mathsf{msk}, \mathbf{y})$ as implemented in Figure 14. Suppose here $\mathsf{mpk} = (\mathbf{A}, \mathbf{U})$ and $\mathsf{msk} = \mathbf{Z}$. Then, $\mathsf{sk}_{\mathbf{y}} = \mathbf{Z}^T\mathbf{y} \in \mathbb{Z}^m$ and $\mathsf{pk}_{\mathbf{y}} = \mathbf{U}^T\mathbf{y} \bmod q = \mathbf{A}^T\mathsf{sk}_{\mathbf{y}} \bmod q$. Further, note that $||\mathsf{sk}_{\mathbf{y}}||_\infty \leq \ell \cdot ||\mathbf{Z}||_\infty \cdot ||\mathbf{y}||_\infty \leq \ell \cdot ||\mathbf{Z}||_2 \cdot ||\mathbf{y}||_\infty \leq \ell \cdot B_\tau \cdot p = \beta_0$. Hence, $(\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}) \in R_{\mathsf{ISIS}}$. □

**Lemma 7.5.** *If $\alpha^{-1} \geq 4p\omega(\sqrt{\log n})(\ell p + m\beta_1)$, then, the IPFE scheme in Figure 14 is $R'_{\mathsf{ISIS}}$-robust, where $R'_{\mathsf{ISIS}} = \mathsf{ISIS}_{n,m,q,\beta_1}$ and $\beta_1 = 2(\gamma - \kappa) - \beta_0$ such that $\beta_1 > \beta_0$.*

*Proof.* For any $\lambda \in \mathbb{N}$ let $\mathsf{pp} \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$ as implemented in Figure 14, for any $\ell \in \mathbb{N}$, $\mathbf{x} \in \mathbb{Z}_p^\ell$, $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell,p}$, let $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}, \ell)$ as implemented in Figure 14, let $\mathsf{pk}_{\mathbf{y}} = \mathsf{PubKGen}(\mathsf{msk}, \mathbf{y})$ as implemented in Figure 14, let $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$ as implemented in Figure 14. Then, for any $\mathsf{sk}'_{\mathbf{y}}$ such that $(\mathsf{pk}_{\mathbf{y}}, \mathsf{sk}'_{\mathbf{y}}) \in R'_{\mathsf{ISIS}}$, we want that $\mathsf{Dec}(\mathsf{sk}'_{\mathbf{y}}, \mathsf{ct})$ outputs $\mathbf{x}^T\mathbf{y} \bmod q$. Observe that the value $d$ computed by $\mathsf{Dec}$ implemented in Figure 14 simplifies to $d = p^{k-1}\mathbf{x}^T\mathbf{y} + (\mathbf{e}_1^T\mathbf{y} - \mathbf{e}_0^T\mathsf{sk}'_{\mathbf{y}}) \bmod q$. Observe that

$$\begin{aligned}|\mathbf{e}_1^T\mathbf{y} - \mathbf{e}_0^T\mathsf{sk}_{\mathbf{y}}| &\leq \ell p \alpha q \omega(\sqrt{\log n}) + m\alpha q\omega(\sqrt{\log n})||\mathsf{sk}'_{\mathbf{y}}||_\infty \\ &= \alpha q \omega(\sqrt{\log n})(\ell p + m\beta_1).\end{aligned}$$

For decryption correctness to hold, we need that $|\mathbf{e}_1^T\mathbf{y} - \mathbf{e}_0^T\mathsf{sk}_{\mathbf{y}}| \leq q/4p$. Hence it suffices to have $\alpha^{-1} \geq 4p\omega(\sqrt{\log n})(\ell p + m\beta_1)$. □

## 7.3 FAS Construction

Instantiating $\mathsf{FAS}$ construction in Section 5 with $\mathsf{IPFE}$ from Figure 14 and $\mathsf{AS}$ from Figure 13, we obtain the following corollary.

**Corollary 7.6.** *Let $p$ be a $\lambda$-bit prime number and let $\ell$ be an integer. Let $\mathcal{M} = \mathbb{Z}_p^\ell$ be an additive group. Let $\mathcal{F}_{\mathsf{IP},\ell,p}$ be the function family for computing inner products of vectors in $\mathbb{Z}_p^\ell$. Let $R$ be any NP relation with statement/witness pairs $(X, \mathbf{x})$ such that $\mathbf{x} \in \mathbb{Z}_p^\ell$. Let $R_{\mathsf{ISIS}} = \mathsf{ISIS}_{n,m,q,\beta_0}$ and $R'_{\mathsf{ISIS}} = \mathsf{ISIS}_{n,m,q,\beta_1}$, where $\beta_0 = \ell p B_\tau$ and $\beta_1 = 2(\gamma - \kappa) - \beta_0$ s.t. $\beta_1 > \beta_0$ and $\gamma - \kappa - \beta_0 > 0$. Suppose that*

- *All function queries $\mathbf{y}$ to $\mathsf{AuxGen}$ are linearly independent modulo $p$,*
- *$R$ is $\mathcal{F}_{\mathsf{IP},\ell,p}$-hard (Definition 3.2),*
- *$\mathsf{NIZK}$ is a secure NIZK argument system (Definition 3.9),*
- *$\mathsf{AS}$ construction in Figure 13 is an adaptor signature scheme w.r.t. digital signature scheme $\mathsf{Lyu}'$ and hard relations $R_{\mathsf{ISIS}}$ and $R'_{\mathsf{ISIS}}$ that satisfies weak pre-signature adaptability and witness extractability (Lemma 7.1),*

- IPFE *construction in Figure 11 is a selective, IND-secure IPFE scheme (Lemma 7.3) for function family* $\mathcal{F}_{\mathsf{IP},\ell+1}$ *that is* $R_{\mathsf{ISIS}}$*-compliant (Lemma 7.4) and* $R'_{\mathsf{ISIS}}$*-robust (Lemma 7.5).*

*Then, the functional adaptor signature scheme w.r.t. Lyubashevsky signature scheme* Lyu′*, NP relation* $R$*, and family of inner product functions* $\mathcal{F}_{\mathsf{IP},\ell,p}$ *constructed in Figure 2 is strongly-secure (Definition 4.3).*

The proof of the above corollary is immediate from the proof of the Theorem 5.2 concerning the generic construction, and Lemmas 7.3 to 7.5 that show that the IPFE scheme in Figure 14 has the required properties.

**Remark 7.7.** *Note that the linear independence modulo p restriction for all* AuxGen *queries in this lattice-based instantiation comes due to the same restriction on the underlying IPFE scheme as discussed in Remark 7.2. In practice,* AuxGen *queries can be sent by any buyer to the seller. On one hand it's reasonable to assume that all the queries by a single buyer are going to be linearly independent modulo p because by linearity of the functionality, the buyer can locally compute the function evaluation for some function* $\mathbf{y}$ *that is linear combination of the functions* $\mathbf{y}_i$*'s it previously queried as follows: if* $\mathbf{y} = \sum_i k_i \mathbf{y}_i \bmod q$*, then,* $f_{\mathbf{y}}(\mathbf{x})$ *can be computed as* $f_{\mathbf{y}}(\mathbf{x}) := \sum_i k_i f_{\mathbf{y}_i}(\mathbf{x})$*. On the other hand, it's unreasonable to assume that all buyers are working in coordination and make sure that their queries are linearly independent. In Section 7.4, we show how to remove this restriction on FAS.*

## 7.4 Modifying FAS Construction in Figure 2.

There is a simple way to ensure that even if AuxGen queries across multiple buyers are linearly dependent modulo $p$, the requests to the underlying IPFE.PubKGen are always linearly independent. Assume a polynomial bound $k$ on the number of buyers. Then, we add $k$ additional slots to the underlying IPFE resulting in a total of $\ell_k + 1$ slots and modify the FAS construction as follows:

- In AdGen on input $\mathbf{x} \in \mathbb{Z}_p^\ell$, the vector $\widetilde{\mathbf{x}} := (\mathbf{x}^T, \mathbf{0}^T)^T \in \mathbb{Z}_p^{\ell+k+1}$ is encrypted using IPFE.Enc.
- Further, the seller maintains an internal map of size $k$ from buyer verification keys $vk$'s to a unique unit vector assigned to them. In other words, $i$-th buyer with verification key denoted by $\mathsf{vk}_i$ is assigned $i$-th unit vector $\mathbf{e}_i \in \{0,1\}^k$.
- At time of AuxGen query for vector $\mathbf{y} \in \mathbb{Z}_p^\ell$ by $i$-th buyer, the seller uses $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}), \mathbf{e}_i^T)^T \in \mathbb{Z}_p^{\ell+k+1}$ to compute $\mathsf{aux}_{\mathbf{y}}$ and sets $\pi_{\mathbf{y}} := (f_{\mathbf{y}}(\mathbf{t}), i)$.
- In AuxVerify, the buyer can recompute $\widetilde{\mathbf{y}}$ on its own using $\pi_{\mathbf{y}} = (f_{\mathbf{y}}(\mathbf{t}), i)$ and return 1 iff $\mathsf{aux}_{\mathbf{y}} = \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$.
- At the time of running Adapt, the seller looks up vk in its internal map to find the index $i$ assigned to it. Then, it can recompute the same $\widetilde{\mathbf{y}}$ that it used during AuxGen query $\mathbf{y}$ by buyer vk. This will ensure that the adapted signature passes the signature verification.

Note that, the $k$ extra slots as used above do not affect correctness as the function evaluation for a function $\mathbf{y}$ is always $\widetilde{\mathbf{x}}^T\widetilde{\mathbf{y}} = \mathbf{x}^T\mathbf{y} \bmod q$. Further note that these slots remain the same in all experiments for all security properties.

**Remark 7.8** (Linear Independence)**.** *Observe that with the above modification, even if two buyers* $i$ *and* $j$ *requested for the same function* $\mathbf{y} \in \mathbb{Z}_p^\ell$*,* AuxGen *will use linearly independent queries* $\widetilde{\mathbf{y}}_i$ *and* $\widetilde{\mathbf{y}}_j$ *to the underlying* IPFE.PubKGen*, where* $\widetilde{\mathbf{y}}_i := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}), \mathbf{e}_i^T)^T \in \mathbb{Z}_p^{\ell+k+1}$ *and* $\widetilde{\mathbf{y}}_j :=$

$(\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}), \mathbf{e}_j^T)^T \in \mathbb{Z}_p^{\ell+k+1}$. *These two are linearly independent because for $i \neq j$, the unit vectors $\mathbf{e}_i$ and $\mathbf{e}_j$ are linearly independent of each other.*

**Remark 7.9** (Communication overhead)**.** *Note that in the IPFE construction in Figure 14, the size of $\mathsf{pk}_{\mathbf{y}}$ and $\mathsf{sk}_{\mathbf{y}}$ are independent of the vector length parameter $\ell$. Hence, change from $\ell+1$ to $\ell+k+1$ does not increase the size $\mathsf{aux}_{\mathbf{y}}$. The size of $\pi_{\mathbf{y}}$ increases by $\log k$ bits, but that can also optimized to be sent only once per buyer and not for every* AuxGen *query. The main communication overhead is in the ciphertext* ct *communicated as part of the advertisement* advt*. Now $\mathsf{ct} \in \mathbb{Z}_q^{m+\ell_k+1}$ instead of $\mathsf{ct} \in \mathbb{Z}_q^{m+\ell+1}$. But fortunately,* advt *needs to be communicated only once.*

## 8 Performance Evaluation

Our implementation aims to (i) show that FAS can replace smart contracts for efficient functional sales, and (ii) benchmark the computational costs for each functional sale. We provide an open-source implementation [VST24] of our prime-order group-based strongly-secure FAS in Python. We also perform a series of benchmarks on our implementation for a wide range of parameters. Our results show that our scheme is practical for variety of real-world scenarios.

We measured the costs on a Apple MacBook Pro with M2 chip, 16GB memory, 8 cores (4 cores @3.49 GHz and 4 cores @2.42 GHz). In typical applications, a seller wants to sell multiple functions of the same witness. The (application dependent) advertisement – containing NIZK proof – is just a one-time cost, while other processes are done once per sale. So, we benchmark computation costs of all algorithms except AdGen and AdVerify.

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. For compliance purpose, we implement the IPFE scheme in Figure 11 on the Secp256k1 curve. Finally, using these adaptor signatures and IPFE schemes, we implement our functional adaptor signature construction. We do not implement the NIZKs as they are needed only for AdGen and AdVerify which we do not implement here.

**Optimizations.** We make following implementation optimizations.

- For vectors of length $\ell$, we parallelize IPFE.Setup, IPFE.Enc.
- Recall that IPFE.Dec computes $\mathsf{ct}_1^T\mathbf{y} - \mathsf{ct}_0^T\mathsf{sk}_{\mathbf{y}}$ (See Figure 11). Here, $\mathsf{ct}_1$ and $\mathbf{y}$ are vectors of length $\ell$ and $\mathsf{ct}_1^T\mathbf{y}$ is computed as $\prod_{i\in\ell} c_i^{y_i}$, where $\mathsf{ct}_1 = (c_1, \ldots, c_\ell)$ is a vector of group elements and $\mathbf{y} = (y_1, \ldots, y_\ell)$ is a vector of scalars. Thus, computing $\mathsf{ct}_1^T\mathbf{y}$ is expensive as it involves $\ell$ group exponentiation operations. But, this computation can be done in an offline stage by the buyer as it does not require knowledge of $\mathsf{sk}_{\mathbf{y}}$ from the seller. This helps us make IPFE.Dec and thus, FAS.FExt very efficient in practice.
- Note that IPFE.PubKGen and IPFE.Dec involve computing a product of powers such as $\prod_{i\in\ell} k_i^{y_i}$ and $\prod_{i\in\ell} c_i^{y_i}$ respectively. We compute these via FastMult algorithm of [BGR98, Section 3.2].
- Note that IPFE.Dec involves a discrete log computation. For this task, we implement the baby-step giant-step algorithm [Sha71].
- Note that AuxGen involves computing $\mathsf{pk}_{\mathbf{y}} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$ which requires $\ell + 1$ group exponentiation operations. As AuxGen is run by the seller who knows $\mathsf{st} = (\mathsf{msk}, \mathbf{t})$, hence, we optimize this computation as $\mathsf{sk}_{\mathbf{y}} := \mathsf{IPFE.KGen}(\mathsf{msk}, \widetilde{\mathbf{y}})$, $\mathsf{pk}_{\mathbf{y}} := g^{\mathsf{sk}_{\mathbf{y}}}$. This resulting way only involves 1 group exponentiation operation and hence is very efficient.

**Table 1:** Communication efficiency of $\mathsf{FAS}$ for $\ell = 10^6$. Last column denotes if a secure channel for buyer/seller is used.

| **Variable** | **Size** | **On-chain?** | **Secure channel?** |
|---|---|---|---|
| $\mathsf{advt} = (\mathsf{IPFE.mpk}, \mathsf{IPFE.ct})$ | 128 MB | no[9] | no |
| Function $\mathbf{y}$ | 32 MB | no | yes |
| $(\mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$ | 96 bytes | no | yes |
| Pre-signature | 64 bytes | no | yes |
| Adapted-signature | 64 bytes | yes | no |

**Table 2:** Computational efficiency of $\mathsf{FAS}$.

| **Params** | | **Running Times (seconds)** | | | | | |
|---|---|---|---|---|---|---|---|
| $\ell$ | $B$ | AuxGen | AuxVerify | FPreSign | FPreVerify | Adapt | FExt |
| 1 | $10^6$ | 0.003 | 0.003 | 0.008 | 0.008 | 0.013 | 0.082 |
| $10^2$ | $10^6$ | 0.003 | 0.300 | 0.013 | 0.418 | 0.021 | 0.142 |
| $10^2$ | $10^8$ | 0.003 | 0.332 | 0.014 | 0.477 | 0.023 | 0.722 |
| $10^4$ | $10^8$ | 0.011 | 0.323 | 0.010 | 0.424 | 0.035 | 1.025 |
| $10^4$ | $10^{10}$ | 0.011 | 0.369 | 0.009 | 0.352 | 0.022 | 9.952 |
| $10^5$ | $10^{11}$ | 0.092 | 2.067 | 0.009 | 2.101 | 0.111 | 30.38 |
| $10^6$ | $10^{12}$ | 0.879 | 19.08 | 0.008 | 18.41 | 0.871 | 95.07 |
| $10^7$ | $10^{13}$ | 9.191 | 223.4 | 0.011 | 217.9 | 10.22 | 317.8 |
| $3 \cdot 10^7$ | $3 \cdot 10^{13}$ | 32.03 | 766.6 | 0.011 | 740.4 | 40.27 | 452.2 |

**Communication efficiency.** In our implementation, each group element is 64 bytes elliptic curve point and each $Z_p$ element is 32 bytes. In Table 1, we give bounds on communication cost for a witness x of dimension $\ell = 10^6$ (total size 32 MB) and also specify if the communication is done on- or off-chain. Using standard compression techniques outlined in BIP-340 [bip], advertisement size can be reduced to 66 MB by encoding 64 bytes elliptic curve points using 33 bytes. In practice, the size of $\mathsf{advt} = (\mathsf{IPFE.mpk}, \mathsf{IPFE.ct})$ can be amortized to 33 MB by reusing the same $\mathsf{IPFE.mpk}$ for all advertisements by a seller. The multi-message security of IPFE scheme should preserve FAS security and result in optimal amortized advertisement size of 1.03x the witness size.

**Computational efficiency.** We share the performance numbers in Table 2. Here, $\ell$ denotes the length of the witness vector. Each entry of the vector is 32 Bytes integer. Thus, $\ell = 10^6$ implies that the witness is of size 32MB. We note that the pre-dominant cost in the implementation is the group exponentiation operation which hasn't been optimized here. We note that for parameter $\ell$, the number of group exponentiation operations in each algorithm are as follows: in $\mathsf{AuxGen}$, $\ell + 1$ in $\mathsf{AuxVerify}$, 1 in $\mathsf{FPreSign}$, $\ell + 2$ in $\mathsf{FPreVerify}$, 0 in $\mathsf{Adapt}$, $\ell + 2$ in $\mathsf{FExt}$. Asymptotically, $\mathsf{AuxVerify}$ and $\mathsf{FPreVerify}$ run in time $O(\ell)$ and $\mathsf{FExt}$ runs in time $O(\sqrt{\ell})$. But, concretely $\mathsf{FExt}$ is the slowest for practical scenarios as highlighted in Table 2. The last two rows of Table 2 do not highlight practical scenarios as the costs of these algorithms are higher than 100 seconds each, but we benchmark them to understand at what point do the concrete running times of $\mathsf{AuxVerify}$ and $\mathsf{FPreVerify}$ start becoming a bottleneck.

[9]for example, $\mathsf{advt}$ can be published on a website.

## Acknowledgements

This work was supported in part through a gift awarded to Pratik Soni and Sri AravindaKrishnan Thyagarajan under the Stellar Development Foundation Academic Research Grants program. This work was also sponsored by the Algorand Foundation, the National Science Foundation under award numbers 2044679 and 2212746, and the Office of Naval Research under award number N000142212064. The views and conclusions contained in this document are those of the author 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

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *PKC*, 2015.

[ACL+22] Martin R Albrecht, Valerio Cini, Russell WF Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based snarks: publicly verifiable, preprocessing, and recursively composable. In *Annual International Cryptology Conference*, pages 102–132. Springer, 2022.

[ACR17] Konstantin Avrachenkov, Pavel Chebotarev, and Dmytro Rubanov. Kernels on graphs as proximity measures. In *Algorithms and Models for the Web Graph: 14th International Workshop, WAW 2017*, 2017.

[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.

[ALMT20] Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In *IACR International Conference on Public-Key Cryptography*, pages 34–64. Springer, 2020.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO*, 2016.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13. Association for Computing Machinery, 2013.

[BGR98] Mihir Bellare, Juan A Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques*, 1998.

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

[bri] Blockchain bridges. https://ethereum.org/en/bridges/.

[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.

[cha] Oracle network for smart contracts. https://shorturl.at/fpDEZ.

[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, May 4–8, 2003 Proceedings 22*, pages 255–271. Springer, 2003.

[DKL$^+$18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[DOY22] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In *International Conference on Cryptology in India*, pages 52–77. Springer, 2022.

[EEE20a] Muhammed F Esgin, Oğuzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In *European Symposium on Research in Computer Security*, pages 378–397. Springer, 2020.

[EEE20b] Muhammed F. Esgin, Oğuzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve Schneider, editors, *Computer Security – ESORICS 2020*, pages 378–397, Cham, 2020. Springer International Publishing.

[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.

[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*, pages 308–317. IEEE, 1990.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308, 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GOS06] 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. Proceedings 26*, pages 97–111. Springer, 2006.

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

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

[HLTW24] Lucjan Hanzlik, Julian Loss, Sri AravindaKrishnan Thyagarajan, and Benedikt Wagner. Sweep-uc: Swapping coins privately. In *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential ddh. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–32. Springer, 2021.

[kag] https://www.kaggle.com/datasets/erdemtaha/cancer-data.

[KRA+22] Mohammad Monirujjaman Khan, Nesat Tasneem RoJa, Faris A Almalki, Maha Aljohani, et al. Revolutionizing e-commerce using blockchain technology and implementing smart contract. *Security and Communication Networks*, 2022.

[liv] Ethereum gas cost. https://milkroad.com/ethereum/gas/.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

[MMSS+19] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019.

[PD] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments.

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

[QPM+23] Xianrui Qin, Shimin Pan, Arash Mirzaei, Zhimei Sui, Oğuzhan Ersoy, Amin Sakzad, Muhammed F Esgin, Joseph K Liu, Jiangshan Yu, and Tsz Hon Yuen. Blindhub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2462–2480. IEEE, 2023.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pages 239–252. Springer, 1990.

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

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Math. Soc., 1971*, volume 20, pages 415–440, 1971.

[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.

[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.

[VST24] Nikhil Vanjani, Pratik Soni, and Sri AravindaKrishnan Thyagarajan. Functional adaptor signatures: Implementation, 2024. https://github.com/nikhilvanjani/fas-impl.

[Wee16] Hoeteck Wee. New techniques for attribute-hiding in prime-order bilinear groups. Manuscript, 2016.

[WN22] Gang Wang and Mark Nixon. Sok: tokenization on blockchain. In *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '21, New York, NY, USA, 2022. Association for Computing Machinery.

## A Proofs for Generic Construction of FAS

Here we present the formal proofs for the generic FAS construction from Section 5.

### A.1 Correctness

First, we give the proof of correctness.

*Proof of Lemma 5.1.* For every $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}^*$, every statement/witness pair $(X, \mathbf{x}) \in R$, and every function $\mathbf{y} \in \mathcal{F}_{\mathsf{IP},\ell}$, suppose $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(\mathsf{pp}, X, \mathbf{x})$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $(\mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, \mathbf{y})$, $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_\mathbf{y})$, $\sigma := \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, \mathbf{x}, \mathbf{y}, \mathsf{aux}_\mathbf{y} \widetilde{\sigma})$, $z := \mathsf{FExt}\ (\mathsf{advt}, \widetilde{\sigma}, \sigma, X, \mathbf{y}, \mathsf{aux}_\mathbf{y})$. Then,

- From NIZK correctness, it follows that $\mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt}) = 1$.
- For determinism of $\mathsf{IPFE.PubKGen}$, it follows that $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) = 1$.
- From correctness of $\mathsf{AS}$, it follows that $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}, \widetilde{\sigma}) = 1$.
- Recall that $\widetilde{\sigma}$ is an $\mathsf{AS}$ pre-signature w.r.t. the statement $\mathsf{pk}_\mathbf{y}$. Further, recall that $\mathsf{sk}_\mathbf{y}$ is the witness computed by the $\mathsf{Adapt}$ algorithm and used by $\mathsf{AS.Adapt}$ to compute $\sigma$. From $R_{\mathsf{IPFE}}$-compliance of IPFE, it follows that $(\mathsf{pk}_\mathbf{y}, \mathsf{sk}_\mathbf{y}) \in R_{\mathsf{IPFE}}$. Hence, by the correctness of $\mathsf{AS}$, it follows that $\mathsf{Vf}(\mathsf{vk}, m, \sigma) = 1$.
- From $\mathsf{AS}$ correctness, it follows that $(\mathsf{pk}_\mathbf{y}, z) \in R'_{\mathsf{IPFE}}$, where $z$ is the extracted witness in the $\mathsf{FExt}$ algorithm. Then, from $R'_{\mathsf{IPFE}}$-robustness of IPFE, it follows that $v = f_\mathbf{y}(\mathbf{x})$, where $v$ is output of $\mathsf{FExt}$.

□

In the subsequent sub-sections, we prove the lemmas regarding advertisement soundness, unforgeability, pre-signature adaptability, witness extractability, zero-knowledge.

## A.2 Advertisement soundness

**Lemma A.1.** *Suppose NIZK satisfies adaptive soundness. Then, the functional adaptor signature construction in Figure 2 is advertisement sound.*

*Proof.* We show that if a p.p.t. adversary $\mathcal{A}$ breaks the advertisement soundness of our functional adaptor signature construction with non-negligible advantage, then, there exists a p.p.t. reduction $\mathcal{B}$ that can break the adaptive soundness of the underlying NIZK scheme with the same non-negligible advantage. Suppose the challenger for adaptive soundness of NIZK is $\mathcal{C}$. The reduction $\mathcal{B}$ is as follows.

- The reduction $\mathcal{B}$ obtains $\mathsf{crs}$ from the challenger $\mathcal{C}$, computes $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$, and sends $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$ to $\mathcal{A}$.
- The adversary $\mathcal{A}$ sends public advertisement $\mathsf{advt}$ and a statement $X$ to the reduction $\mathcal{B}$. $\mathcal{B}$ parses $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$ and sends the statement $(X, \mathsf{pp}, \mathsf{mpk}, \mathsf{ct})$ and proof $\pi$ to the challenger $\mathcal{C}$.

Observe that if $\mathcal{A}$ breaks advertisement soundness, then, $X \notin L_R$ and $\mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt}) = 1$. From the definition of $L_{\mathsf{NIZK}}$ it follows that if $X \notin L_R$, then, $(X, \mathsf{pp}, \mathsf{mpk}, \mathsf{ct}) \notin L_{\mathsf{NIZK}}$. Further, recall that the implementation of $\mathsf{AdVerify}$ simply involves a $\mathsf{NIZK}$ proof verification. Therefore, if $\mathsf{AdVerify}(\mathsf{pp}, X, \mathsf{advt}) = 1$, then, $\mathsf{NIZK.Vf}(\mathsf{crs}, (X, \mathsf{pp}, \mathsf{mpk}, \mathsf{ct}), \pi) = 1$. Hence, whenever $\mathcal{A}$ breaks advertisement soundness of $\mathsf{FAS}$, $\mathcal{B}$ breaks adaptive soundness of $\mathsf{NIZK}$.

□

## A.3 Pre-Signature Validity

**Lemma A.2.** *Suppose the adaptor signature scheme satisfies correctness (Definition 3.6). Then, the functional adaptor signature construction in Figure 2 satisfies pre-signature validity.*

*Proof.* For any $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ as computed in Figure 2, any $X, \mathsf{advt}$ such that $\mathsf{AdVerify}(\mathsf{pp}, \mathsf{advt}, X) = 1$ as computed in Figure 2, any message $m \in \{0,1\}^*$, any function $\mathbf{y} \in \mathcal{F}_{\mathsf{IP}}, \ell$, any $(\mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$, any key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$, any pre-signature $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}})$ as computed in Figure 2, suppose $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 1$. Then, to prove that $\Pr[\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, f, \mathsf{aux}_f, \pi_f, \widetilde{\sigma})] = 1$ (as computed in Figure 2), it suffices to show that $\Pr[\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{aux}_{\mathbf{y}}, \widetilde{\sigma})] = 1$. This follows from the correctness of the adaptor signature scheme $\mathsf{AS}$. □

## A.4 Unforgeability

**Lemma A.3.** *Suppose the NIZK argument system* $\mathsf{NIZK}$ *satisfies adaptive soundness ((Definition 3.9)), the Adaptor Signature scheme* $\mathsf{AS}$ *satisfies witness extractability (Definition 3.7), the relation $R$ is a $\mathcal{F}_{\mathsf{IP},\ell}$-hard (Definition 3.2), and* $\mathsf{IPFE}$ *scheme satisfies correctness and $R'_{\mathsf{IPFE}}$-robustness (Definition 3.14). Then, the functional adaptor signature instantiation above is* $\mathsf{faEUF\text{-}CMA}$*-secure.*

*Proof.* We prove the lemma by defining a sequence of games.

**Game $G_0$:** It is the original game $\mathsf{faEUF\text{-}CMA}_{\mathcal{A},\mathsf{FAS}}$, where the adversary $\mathcal{A}$ has to come up with a valid forgery on a message $m^*$ of his choice, while having access to functional pre-sign oracle $\mathcal{O}_{\mathsf{fpS}}$ and sign oracle $\mathcal{O}_S$. Game $G_0$ is formally defined in Figure 15.

**Game $G_1$:** same as game $G_0$, except that before computing the pre-signature, the game checks if the NIZK statement $(X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$ is in the language $L_{NIZK}$. If no, the game sets the flag $\mathsf{Bad}_1 = \mathsf{true}$. Game $G_1$ is formally defined in Figure 15.

**Game $G_2$:** same as game $G_1$, except that when $\mathcal{A}$ outputs the forgery $\sigma^*$, the game extracts the witness $z$ of the underlying adaptor signature scheme for the statement $\mathsf{pk}_{\mathbf{y}^*}$ and checks if the NIZK statement $(\mathsf{aux}^*_{\mathbf{y}}, z)$ satisfies $R'_{\mathsf{IPFE}}$. If no, the game sets the flag $\mathsf{Bad}_2 = \mathsf{true}$. Game $G_2$ is formally defined in Figure 15.

```
Games $G_0$, $\boxed{G_1}$, $\boxed{G_2}$
$\mathcal{Q} := \emptyset$
$\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
$\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
$\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}')$
$(X^*, \mathbf{x}^*) \leftarrow \mathsf{GenR}(1^\lambda)$
$(\mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk}, X^*)$
Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$
Let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
$\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$
$\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
$\mathsf{Bad}_1 := \mathsf{false}, \mathsf{Bad}_2 := \mathsf{false}$
If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \ \vee\ \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell}$
    $\vee\ \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$: ret $0$
$\boxed{\boxed{\text{If } \mathsf{stmt} \notin L_{NIZK}\text{: } \mathsf{Bad}_1 := \mathsf{true}, \textbf{ret } 0}}$
$\widetilde{\sigma}^* \leftarrow \mathsf{AS.PreSign}(\mathsf{sk}, m^*, \mathsf{aux}^*_{\mathbf{y}})$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
$\boxed{z = \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})}$
$\boxed{\text{if } ((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*)}$
$\boxed{\wedge((\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}}))\text{: } \mathsf{Bad}_2 := \mathsf{true}, \textbf{ret } 0}$
ret $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*))$

Oracle $\mathcal{O}_S(m)$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} := \mathcal{Q} \vee \{m\}$
ret $\sigma$

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$
If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 0$: ret $\perp$
$\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}})$
$\mathcal{Q} := \mathcal{Q} \vee \{m\}$
ret $\widetilde{\sigma}$
```

**Figure 15:** Unforgeability Proof: Games $G_0, G_1, G_2$

To prove the lemma, we need to show that

$$\Pr[G_0(1^\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Note that by triangle inequality, it follows that

$$\begin{aligned}\Pr[G_0(1^\lambda) = 1] \leq &\, |\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| \\ &+ |\Pr[G_1(1^\lambda) = 1] - \Pr[G_2(1^\lambda) = 1]| \\ &+ \Pr[G_2(1^\lambda) = 1].\end{aligned}$$

To complete the proof, we show in Claims A.4 to A.6 that each of the three terms on the right-hand-side are at most $\mathsf{negl}(\lambda)$. □

**Claim A.4.** *If the NIZK argument system* $\mathsf{NIZK}$ *satisfies adaptive soundness (Definition 3.9), then,* $|\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

*Proof.* Observe that in game $G_0$, $\mathsf{Bad}_1 = \mathsf{false}$ always and game $G_1$ differs from it when the flag $\mathsf{Bad}_1 = \mathsf{true}$ is set. Both games are identical until $G_1$ checks the condition for setting $\mathsf{Bad}_1$. Hence,

$$|\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| \leq \Pr[\mathsf{Bad}_1 \text{ in } G_1].$$

Hence, it suffices to show that in game $G_1$, $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$.

We prove the claim using a reduction to the adaptive soundness of the underlying $\mathsf{NIZK}$ scheme. More specifically, we show that if the adversary $\mathcal{A}$ causes game $G_1$ to set $\mathsf{Bad}_1 = \mathsf{true}$, then, we can use it to construct a reduction $\mathcal{B}$ that breaks the adaptive soundness of $\mathsf{NIZK}$. Let $\mathcal{C}$ be the challenger for the NIZK adaptive soundness. Then, the reduction is as in Figure 16.

```
Reduction $\mathcal{B}$ for proof of Claim A.4
1 :  $\mathcal{Q} := \emptyset$
2 :  $\mathsf{crs} \leftarrow \mathcal{C}(1^\lambda)$
3 :  $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
4 :  $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$
5 :  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}')$
6 :  $(X^*, \mathbf{x}^*) \leftarrow \mathsf{GenR}(1^\lambda)$
7 :  $(\mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk}, X^*)$
8 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
9 :  $\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$
10 : $\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
11 : $\mathsf{Bad}_1 = \mathsf{false}$
12 : If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \ \vee \ \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell} \ \vee \ \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$:
         abort
13 : If $\mathsf{stmt} \notin L_{NIZK}$: $\mathsf{Bad}_1 = \mathsf{true}$
14 : ret $(\mathsf{stmt}, \pi)$ to $\mathcal{C}$

Oracle $\mathcal{O}_S(m)$
1 :  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 :  ret $\sigma$
```

**Figure 16:** Reduction $\mathcal{B}$ for proof of Claim A.4

Observe that if $\mathcal{A}$ causes game $G_1$ to not abort, then, it must be the case that $\pi$ is a valid proof for $\mathsf{stmt} = (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$. Further, if $\mathcal{A}$ causes game $G_1$ to set $\mathsf{Bad}_1 = \mathsf{true}$, then, it must be the case that $\mathsf{stmt} \notin L_{NIZK}$. This means that in step 13, the reduction $\mathcal{B}$ successfully returns a valid proof $\pi$ to $\mathcal{C}$ for a statement $\mathsf{stmt}$ not in the language $L_{NIZK}$ and thus breaks the adaptive soundness of the $\mathsf{NIZK}$. Thus, $\Pr[\mathsf{Bad}_1]$ is same as the probability that $\mathcal{B}$ breaks adaptive soundness of the $\mathsf{NIZK}$. Hence, we can conclude that $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$. □

**Claim A.5.** *If the Adaptor Signature scheme* AS *satisfies witness extractability (Definition 3.7), then,* $|\Pr[G_1(1^\lambda) = 1] - \Pr[G_2(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

*Proof.* Observe that in game $G_1$, $\mathsf{Bad}_2 = \mathsf{false}$ always and game $G_2$ differs from it when the flag $\mathsf{Bad}_2 = \mathsf{true}$ is set. Both games are identical until $G_2$ checks the condition for setting $\mathsf{Bad}_2$. Hence,

$$|\Pr[G_1(1^\lambda) = 1] - \Pr[G_2(1^\lambda) = 1]| \leq \Pr[\mathsf{Bad}_2 \text{ in } G_2].$$

Then, it suffices to show that in game $G_2$, $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$.

We prove the claim using a reduction to the witness extractability of the underlying AS scheme. More specifically, we show that if the adversary $\mathcal{A}$ causes $\mathsf{Bad}_2$ in game $G_2$, then, we can use it to construct a reduction $\mathcal{B}$ that breaks the witness extractability of AS. For the AS witness extractability game, let $\mathcal{C}$ be the challenger and let $\mathsf{AS}.\mathcal{O}_S, \mathsf{AS}.\mathcal{O}_{pS}$ be the signing and pre-signing oracles that the reduction $\mathcal{B}$ has access to. Then, the reduction is as in Figure 17.

```
Reduction $\mathcal{B}$ for proof of Claim A.5
1 :  $\mathcal{Q} := \emptyset$
2 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
3 :  $(\mathsf{pp}', \mathsf{vk}) \leftarrow \mathcal{C}(1^\lambda)$
4 :  $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$
5 :  $(X^*, \mathbf{x}^*) \leftarrow \mathsf{GenR}(1^\lambda)$
6 :  $(\mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk}, X^*)$
7 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$
8 :  Let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
9 :  $\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$
10 : $\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
11 : $\mathsf{Bad}_1 := \mathsf{false}, \mathsf{Bad}_2 := \mathsf{false}$
12 : If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \ \vee \ \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell}$
     $\vee\ \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$: **abort**
13 : If $\mathsf{stmt} \notin L_{NIZK}$: $\mathsf{Bad}_1 := \mathsf{true}$, **abort**
14 : $\widetilde{\sigma}^* \leftarrow \mathcal{C}(m^*, \mathsf{aux}^*_{\mathbf{y}})$
15 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
16 : $z = \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})$
17 : If $(m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge ((\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}})$:
         $\mathsf{Bad}_2 = \mathsf{true}$, **ret** $\sigma^*$ to $\mathcal{C}$
18 : Else: **abort**

Oracle $\mathcal{O}_S(m)$
1 :  $\sigma \leftarrow \mathsf{AS}.\mathcal{O}_S(m)$
2 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 :  **ret** $\sigma$

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$
1 :  If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 0$: **ret** $\bot$
2 :  $\widetilde{\sigma} \leftarrow \mathsf{AS}.\mathcal{O}_{pS}(m, \mathsf{aux}_{\mathbf{y}})$
3 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
4 :  **ret** $\widetilde{\sigma}$
```

**Figure 17:** Reduction $\mathcal{B}$ for proof of Claim A.5

Observe that the reduction $\mathcal{B}$ perfectly simulates the game $G_2$ to the adversary $\mathcal{A}$ as it appropriately forwards the signing and functional pre-signing queries to its the signing and pre-signing oracles of AS. Thus, to complete the proof it suffices to argue that if $\mathcal{B}$ returns $\sigma^*$ to $\mathcal{C}$ in step 17, then, it breaks the witness extractability of AS. Suppose that the query set maintained by $\mathcal{C}$ is denoted by $\mathsf{AS}.\mathcal{Q}$. Recall then that to break the witness extractability of AS, the signature $\sigma^*$ on message $m^*$ must satisfy $(m^* \notin \mathsf{AS}.\mathcal{Q}) \wedge ((\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*)$, where $z = \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})$. The last of the three conditions holds because the same is present in the pre-condition for returning

$\sigma^*$ to $\mathcal{C}$. Hence, it remains to argue that the first two conditions also hold true. We note that $m^* \notin \mathcal{Q}$ implies $m^* \notin \mathsf{AS}.\mathcal{Q}$ as from the description of the reduction $\mathcal{B}$, it follows that $\mathcal{Q} = \mathsf{AS}.\mathcal{Q}$. Next, $(\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}}$ because $\mathsf{Bad}_2 = \mathsf{true}$. Therefore, if the adversary $\mathcal{A}$ causes $\mathsf{Event}_2$ in game $G_2$, then, $\mathcal{B}$ that breaks the witness extractability of AS. Hence, it follows that $\Pr[\mathsf{Event}_2] \leq \mathsf{negl}(\lambda)$. □

**Claim A.6.** *Suppose relation $R$ is a $\mathcal{F}_{\mathsf{IP},\ell}$-hard (Definition 3.2), IPFE scheme satisfies $R'_{\mathsf{IPFE}}$-robustness (Definition 3.14). Then,* $\Pr[G_2(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

```
Reduction $\mathcal{B}$ for proof of Claim A.6
1 :  $\mathcal{Q} := \emptyset$
2 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
3 :  $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
4 :  $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$
5 :  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}')$
6 :  $X^* \leftarrow \mathcal{C}(1^\lambda)$
7 :  $(\mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk}, X^*)$
8 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
9 :  $\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$
10 : $\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
11 : $\mathsf{Bad}_1 := \mathsf{false}, \mathsf{Bad}_2 := \mathsf{false}$
12 : If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \ \vee \ \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell}$
     $\vee \ \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$: **abort**
13 : If $\mathsf{stmt} \notin L_{NIZK}$: $\mathsf{Bad}_1 := \mathsf{true}$, **abort**
14 : $\widetilde{\sigma}^* \leftarrow \mathsf{AS.PreSign}(\mathsf{sk}, m^*, \mathsf{aux}^*_{\mathbf{y}})$
15 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
16 : $z = \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})$
17 : If $(m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge ((\mathsf{aux}^*_{\mathbf{y}}, z) \notin R'_{\mathsf{IPFE}})$:
         $\mathsf{Bad}_2 = \mathsf{true}$, **abort**
18 : $v = \mathsf{IPFE.Dec}(z, \mathsf{ct})$
19 : If $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*))$:
         **ret** $(\mathbf{y}^*, v)$
20 : Else: abort game with $\mathcal{C}$

Oracle $\mathcal{O}_S(m)$
1 :  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 :  **ret** $\sigma$

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$
1 :  If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 0$: **ret** $\bot$
2 :  $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}})$
3 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
4 :  **ret** $\widetilde{\sigma}$
```

**Figure 18:** Reduction $\mathcal{B}$ for proof of Claim A.6

*Proof.* We prove the claim using a reduction to the $\mathcal{F}_{\mathsf{IP},\ell}$-hardness of the relation $R$. More specifically, we show that if an adversary $\mathcal{A}$ causes game $G_2$ to return 1, then, we can use it to construct a reduction $\mathcal{B}$ that breaks the $\mathcal{F}_{\mathsf{IP},\ell}$-hardness of the relation $R$. Let $\mathcal{C}$ be the challenger for the $\mathcal{F}_{\mathsf{IP},\ell}$-hardness of the relation $R$. The reduction $\mathcal{B}$ is as in Figure 18.

Observe that the reduction $\mathcal{B}$ perfectly simulates game $G_2$ to the adversary $\mathcal{A}$ as it obtains $X^*$ from the challenger $\mathcal{C}$ who computes it as $(X^*, \cdot) \leftarrow \mathsf{GenR}(1^\lambda)$. Then, if $\mathcal{A}$ causes $G_2$ to return 1, it must be the case that the if condition in step 19 of the reduction must be true. In such a case, the reduction returns $(\mathbf{y}^*, v)$ to the challenger $\mathcal{C}$. To complete the proof, it remains to show that this results in $\mathcal{B}$ winning the game against $\mathcal{C}$.

To win the game against $\mathcal{C}$, the reduction $\mathcal{B}$'s output must satisfy $(\mathbf{y}^* \in \mathcal{F}_{\mathsf{IP},\ell}) \wedge (v \in \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists \mathbf{x}\ s.t.\ (X, \mathbf{x}) \in R\})$. Note that since the reduction did not abort, it implies $\mathbf{y}^* \in \mathcal{F}_{\mathsf{IP},\ell}$ and $\mathsf{pk}_{\mathbf{y}^*} = \mathsf{aux}^*_{\mathbf{y}}$. Next, $\mathsf{Bad}_1 = \mathsf{false}$ implies that $\mathsf{stmt} \in L_{NIZK}$, where $\mathsf{stmt} = (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$. Hence, it follows that $\mathsf{ct}$ encrypts some vector $\widetilde{\mathbf{x}} = (\mathbf{x}^T, 0)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$ under $\mathsf{mpk}$ such that $(X^*, \mathbf{x}) \in R$, where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}', 1^{\ell+1})$. Next, $\mathsf{Bad}_2 = \mathsf{false}$ implies that $(\mathsf{aux}^*_{\mathbf{y}}, z) \in R'_{\mathsf{IPFE}}$. As $\mathsf{pk}_{\mathbf{y}^*} = \mathsf{aux}^*_{\mathbf{y}}$ and IPFE satisfies $R'_{\mathsf{IPFE}}$-robustness, it follows that $v = f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}})$. As $\widetilde{\mathbf{y}^*} = (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$ and $\widetilde{\mathbf{x}} = (\mathbf{x}^T, 0)^T$, we get that $f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}}) = f_{\mathbf{y}^*}(\mathbf{x})$. Hence, we conclude that $v \in \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists \mathbf{x}\ s.t.\ (X, \mathbf{x}) \in R\}$. This completes the proof.

□

## A.5 Witness Extractability

**Lemma A.7.** *Suppose* AS *satisfies witness extractability,* NIZK *satisfies adaptive soundness, and* IPFE *satisfies* $R'_{\mathsf{IPFE}}$*-robustness. Then, the functional adaptor signature construction in Figure 2 is* faWitExt*-secure.*

*Proof.* We prove the lemma by defining a sequence of games.

**Game $G_0$:** It is the original game $\mathsf{faWitExt}_{\mathcal{A},\mathsf{FAS}}$, where the adversary $\mathcal{A}$ who is given access to a pre-signature on a message, must come up with a full signature such that it does not reveal the function evaluation on a witness. The adversary also has access to functional pre-sign oracle $\mathcal{O}_{\mathsf{fpS}}$ and sign oracle $\mathcal{O}_S$. Game $G_0$ is formally defined in Figure 19.

**Game $G_1$:** same as game $G_0$, except that when $\mathcal{A}$ outputs a public advertisement $\mathsf{advt}$, the game checks if the NIZK statement $(X^*, \mathsf{mpk}, \mathsf{ct})$ is in the language $L_{\mathsf{NIZK}}$. If no, the game sets the flag $\mathsf{Bad}_1 = \mathsf{true}$. Game $G_1$ is formally defined in Figure 19.

To prove the lemma, we need to show that

$$\Pr[G_0(1^\lambda) = 1] \le \mathsf{negl}(\lambda).$$

Note that by triangle inequality, it follows that

$$\Pr[G_0(1^\lambda) = 1] \le |\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| + \Pr[G_1(1^\lambda) = 1].$$

To complete the proof, we show in Claims A.8 and A.9 that each of the two terms on the right-hand-side are at most $\mathsf{negl}(\lambda)$. □

**Claim A.8.** *If the NIZK argument system* NIZK *satisfies adaptive soundness, then,* $|\Pr[G_0(1^\lambda) = 1] - \Pr[G_1(1^\lambda) = 1]| \le \mathsf{negl}(\lambda)$.

*Proof.* Similar to proof of Claim A.4. □

**Claim A.9.** *Suppose the Adaptor Signature scheme* AS *satisfies witness extractability and the IPFE scheme* IPFE *satisfies* $R'_{\mathsf{IPFE}}$*-robustness. Then,* $\Pr[G_1(1^\lambda) = 1] \le \mathsf{negl}(\lambda)$.

*Proof.* To prove $\Pr[G_1(1^\lambda) = 1] \le \mathsf{negl}(\lambda)$, we show that if there exists a p.p.t. adversary $\mathcal{A}$ such that it wins game $G_1$ with non-negligible advantage, then, we can construct a p.p.t. reduction $\mathcal{B}$ that breaks the witness extractability of the underlying adaptor signature scheme AS. For the AS witness extractability game, let $\mathcal{C}$ be the challenger and let $\mathsf{AS}.\mathcal{O}_S, \mathsf{AS}.\mathcal{O}_{pS}$ be the signing and pre-signing oracles that the reduction $\mathcal{B}$ has access to. Then, the reduction is as in Figure 20.

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

```
1 :  $\mathcal{Q} := \emptyset$
2 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
3 :  $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
4 :  $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$
5 :  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}')$
6 :  $(X^*, \mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk})$
7 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
8 :  $\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$
9 :  $\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
10 : $\mathsf{Bad}_0 := \mathsf{false}, \mathsf{Bad}_1 := \mathsf{false}, \mathsf{Bad}_2 := \mathsf{false}$
11 : If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \vee \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell} \vee \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$:
        $\mathsf{Bad}_0 := \mathsf{true}$
12 : $\boxed{\text{If } \mathsf{stmt} \notin L_{\mathsf{NIZK}}\text{: } \mathsf{Bad}_1 := \mathsf{true}}$
13 : $\widetilde{\sigma}^* \leftarrow \mathsf{AS.PreSign}(\mathsf{sk}, m^*, \mathsf{aux}^*_{\mathbf{y}})$
14 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
15 : $z := \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})$
16 : $v := \mathsf{IPFE.Dec}(z, \mathsf{ct})$
17 : If $v \in \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists\ \mathbf{x}\ s.t.\ (X^*, \mathbf{x}) \in R\}$: $\mathsf{Bad}_2 = \mathsf{true}$
18 : ret $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge \neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1 \wedge \neg\mathsf{Bad}_2)$
```

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

```
1 :  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 :  ret $\sigma$
```

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}})$

```
1 :  If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 0$: ret $\perp$
2 :  $\widetilde{\sigma} \leftarrow \mathsf{FPreSign}(\mathsf{advt}, \mathsf{sk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}})$
3 :  $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
4 :  ret $\widetilde{\sigma}$
```

**Figure 19:** Witness Extractability proof: Games $G_0$ and $G_1$

From the description of reduction $\mathcal{B}$, we can observe that whenever the challenger $\mathcal{C}$ obtains the signature $\sigma^*$ on message $m^*$ obtained, it is the case that $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge \neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1 \wedge \neg\mathsf{Bad}_2)$. Suppose that the query set maintained by $\mathcal{C}$ is denoted by $\mathsf{AS}.\mathcal{Q}$. For $\mathcal{B}$ to succeed, the signature $\sigma^*$ must satisfy $((m^* \notin \mathsf{AS}.\mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge ((\mathsf{pk}_{\mathbf{y}^*}, z) \notin R'_{\mathsf{IPFE}}))$. We note that $m^* \notin \mathcal{Q}$ implies $m^* \notin \mathsf{AS}.\mathcal{Q}$ as from the description of the reduction $\mathcal{B}$, it follows that $\mathcal{Q} = \mathsf{AS}.\mathcal{Q}$. Note that $\sigma^*$ already satisfies the second condition and it suffices to show that if $\neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1 \wedge \neg\mathsf{Bad}_2$, then, $(\mathsf{pk}_{\mathbf{y}^*}, z) \notin R'_{\mathsf{IPFE}}$.

Note that $\mathsf{Bad}_0 = \mathsf{false}$ implies $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1$ and $\mathbf{y}^* \in \mathcal{F}_{\mathsf{IP},\ell}$ and $\mathsf{aux}^*_{\mathbf{y}} = \mathsf{pk}_{\mathbf{y}^*}$. Next, $\mathsf{Bad}_1 = \mathsf{false}$ implies that $\mathsf{stmt} \in L_{\mathsf{NIZK}}$, where $\mathsf{stmt} = (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$. Hence, it follows that $\mathsf{ct}$ encrypts some vector $\widetilde{\mathbf{x}^*} = (\mathbf{x}^{*T}, 0)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$ under $\mathsf{mpk}$ such that $(X^*, \mathbf{x}^*) \in R$, where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}', 1^\ell)$. Next, $\mathsf{Bad}_2 = \mathsf{false}$ implies that $v \notin \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists\ \mathbf{x}\ s.t.\ (X^*, \mathbf{x}) \in R\}$, where $v$ is computed as $v := \mathsf{IPFE.Dec}(z, \mathsf{ct})$. This implies $v \neq f_{\mathbf{y}^*}(\mathbf{x}^*)$. Note that for $\widetilde{\mathbf{y}^*} = (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$, we have $f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}^*}) = f_{\mathbf{y}^*}(\mathbf{x}^*)$. Hence, it follows that $v \neq f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}^*})$.

To complete the proof, we argue that if $v \neq f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}^*})$, then, $(\mathsf{pk}_{\mathbf{y}^*}, z) \notin R'_{\mathsf{IPFE}}$. This follows from $R'_{\mathsf{IPFE}}$-robustness of IPFE. In particular, the contra-positive form of $R'_{\mathsf{IPFE}}$-robustness says that if $v \neq f_{\widetilde{\mathbf{y}^*}}(\widetilde{\mathbf{x}^*})$, then either $\mathsf{pp}'$ is not computed honestly, or $\mathsf{mpk}$ is not computed honestly or $\mathsf{ct}$ is not computed honestly or $\mathsf{pk}_{\mathbf{y}^*}$ is not computed honestly or $v$ is not computed honestly or $(\mathsf{pk}_{\mathbf{y}^*}, z) \notin R'_{\mathsf{IPFE}}$. Observe that the challenger $\mathcal{C}$ samples $\mathsf{pp}'$ honestly, the NIZK proof $\pi$ attests that $\mathsf{mpk}$ and $\mathsf{ct}$ are computed honestly, the reduction $\mathcal{B}$ computes $\mathsf{pk}_{\mathbf{y}^*}$ and $v$ honestly. Thus, it must be the case that $(\mathsf{pk}_{\mathbf{y}^*}, z) \notin R'_{\mathsf{IPFE}}$.

Reduction $B$ for proof of Claim A.9

```
1 :  $\mathcal{Q} := \emptyset$
2 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
3 :  $(\mathsf{pp}', \mathsf{vk}) \leftarrow \mathcal{C}(1^\lambda)$
4 :  $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$
5 :  $(X^*, \mathsf{advt}, m^*, \mathbf{y}^*, \mathsf{aux}^*_{\mathbf{y}}, \pi^*_{\mathbf{y}}) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot)}(\mathsf{pp}, \mathsf{vk})$
6 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, let $\mathsf{stmt} := (X^*, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})$
7 :  $\widetilde{\mathbf{y}^*} := (\mathbf{y}^{*T}, \pi^*_{\mathbf{y}})^T$
8 :  $\mathsf{pk}_{\mathbf{y}^*} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}^*})$
9 :  $\mathsf{Bad}_0 := \mathsf{false}, \mathsf{Bad}_1 := \mathsf{false}, \mathsf{Bad}_2 := \mathsf{false}$
10 : If $\mathsf{NIZK.Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 0 \vee \mathbf{y}^* \notin \mathcal{F}_{\mathsf{IP},\ell} \vee \mathsf{pk}_{\mathbf{y}^*} \neq \mathsf{aux}^*_{\mathbf{y}}$:
         $\mathsf{Bad}_0 := \mathsf{true}$
11 : If $\mathsf{stmt} \notin L_{\mathsf{NIZK}}$: $\mathsf{Bad}_1 = \mathsf{true}$
12 : $\widetilde{\sigma}^* \leftarrow \mathcal{C}(m^*, \mathsf{aux}^*_{\mathbf{y}})$
13 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot), \mathcal{O}_{\mathsf{fpS}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\widetilde{\sigma}^*)$
14 : $z := \mathsf{AS.Ext}(\widetilde{\sigma}^*, \sigma^*, \mathsf{aux}^*_{\mathbf{y}})$
15 : $v := \mathsf{IPFE.Dec}(z, \mathsf{ct})$
16 : If $v \in \{f_{\mathbf{y}^*}(\mathbf{x}) : \exists\, \mathbf{x}\ s.t.\ (X^*, \mathbf{x}) \in R\}$: $\mathsf{Bad}_2 = \mathsf{true}$
17 : If $((m^* \notin \mathcal{Q}) \wedge \mathsf{Vf}(\mathsf{vk}, m^*, \sigma^*) \wedge \neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1 \wedge \neg\mathsf{Bad}_2)$:
         $\mathbf{ret}\ \sigma^*$
18 : Else: abort game with $\mathcal{C}$
```

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

```
1 : $\sigma \leftarrow \mathsf{AS}.\mathcal{O}_S(m)$
2 : $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
3 : $\mathbf{ret}\ \sigma$
```

Oracle $\mathcal{O}_{\mathsf{fpS}}(m, X, f)$

```
1 : If $\mathsf{advt} = \perp$: $\mathbf{ret}\ \perp$
2 : $\mathsf{pk}_{\mathbf{y}} = \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \mathbf{y})$
3 : $\widetilde{\sigma} \leftarrow \mathsf{AS}.\mathcal{O}_{pS}(m, \mathsf{pk}_{\mathbf{y}})$
4 : $\mathcal{Q} := \mathcal{Q} \vee \{m\}$
5 : $\mathbf{ret}\ \widetilde{\sigma}$
```

**Figure 20:** Reduction $B$ for proof of Claim A.9

□

## A.6 Pre-Signature Adaptability

**Lemma A.10.** *Suppose* IPFE *satisfies* $R_{\mathsf{IPFE}}$*-compliance (Definition 3.13) and suppose that* AS *satisfies weak pre-signature adaptability (Definition 3.8). Then, the functional adaptor signature construction in Figure 2 is pre-signature adaptable (Definition 4.9).*

*Proof.* For any $\lambda \in \mathbb{N}$, let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ be as computed in Figure 2. For any statement/witness pair $(X, \mathbf{x}) \in R$, let $(\mathsf{advt}, \mathsf{st}) \leftarrow \mathsf{AdGen}(1^\ell, \mathsf{crs}, X, \mathbf{x})$ be as computed in Figure 2. For any message $m \in \{0,1\}^*$, any function $\mathbf{y} \in \mathcal{F}_{IP,\mathbb{Z}_p^\ell}$, any $(\mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) \leftarrow \mathsf{AuxGen}(\mathsf{advt}, \mathsf{st}, \mathbf{y})$ as computed in Figure 2, any key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$ as computed in Figure 2, any pre-signature $\widetilde{\sigma} \in \{0,1\}^*$ such that $\mathsf{FPreVerify}(\mathsf{advt}, \mathsf{vk}, m, X, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}, \widetilde{\sigma}) = 1$ as computed in Figure 2. This implies $\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{aux}_{\mathbf{y}}, \widetilde{\sigma}) = 1$. By $R_{\mathsf{IPFE}}$-compliance of the IPFE scheme, we know that $(\mathsf{aux}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}) \in R_{\mathsf{IPFE}}$, where $\mathsf{sk}_{\mathbf{y}}$ is as computed in the Adapt algorithm in Figure 2. Then, it follows by the weak pre-signature adaptability of AS that $\Pr[\mathsf{AS.Vf}(\mathsf{vk}, m, \sigma) = 1] = 1$, where $\sigma = \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{aux}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}, \widetilde{\sigma})$. Then, from the implementation of the Adapt algorithm of our functional adaptor signature scheme as in Figure 2, it follows that

$$\Pr[\mathsf{Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{advt}, \mathsf{st}, \mathsf{vk}, m, X, \mathbf{x}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \widetilde{\sigma})) = 1] = 1.$$

□

## A.7 Zero-Knowledge

**Lemma A.11.** *Suppose $\mathcal{M}$ is an additive group,* NIZK *satisfies zero-knowledge (Definition 3.9) and* IPFE *satisfies selective, IND-security (Definition 3.11). Then, the functional adaptor signature construction in Figure 2 is zero-knowledge (Definition 4.10).*

*Proof.* To prove the lemma, we need to show that for every stateful p.p.t. adversary $\mathcal{A}$, there exists a stateful p.p.t. simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{AuxGen}^*, \mathsf{Adapt}^*)$ and there exists a negligible function $\mathsf{negl}$ such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, for all $(X, \mathbf{x}) \in R$,

$$|\Pr[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| \leq \mathsf{negl}(\lambda).$$

We first describe the stateful simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{AdGen}^*, \mathsf{AuxGen}^*, \mathsf{Adapt}^*)$. Let $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Setup}^*, \mathsf{NIZK.Prove}^*)$ be the NIZK simulator. Then, the simulator $\mathsf{Sim}$ is as in Figure 21.

```
Setup*(1^λ)
1 : Let $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)$
2 : Sample $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
3 : Store internal state $\mathsf{td}$, ret $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$

AdGen*(pp, X)
1 : Sample $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}', 1^{\ell+1})$
2 : Sample $\mathbf{t} \xleftarrow{\$} \mathcal{M}$
3 : Let $\widetilde{\mathbf{x}} := (-\mathbf{t}^T, 1)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$
4 : Let $\mathsf{ct} \leftarrow \mathsf{IPFE.Enc}(\mathsf{mpk}, \widetilde{\mathbf{x}})$
5 : Let $\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))$
6 : Store internal state $\mathsf{st} := (\mathsf{msk}, \mathbf{t})$
7 : ret $\mathsf{advt} := (\mathsf{mpk}, \mathsf{ct}, \pi)$

AuxGen*(advt, y, f_y(x))
1 : Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$
2 : Let $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$
3 : Let $\mathsf{pk}_{\mathbf{y}} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$
4 : ret $\mathsf{aux}_{\mathbf{y}} := \mathsf{pk}_{\mathbf{y}}$, $\pi_{\mathbf{y}} := f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x})$

Adapt*(advt, vk, m, X, y, aux_y, σ̃, f_y(x))
1 : Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, and $\mathsf{st} = (\mathsf{msk}, \mathbf{t})$
2 : Let $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$
3 : Let $\mathsf{sk}_{\mathbf{y}} := \mathsf{IPFE.KGen}(\mathsf{msk}, \widetilde{\mathbf{y}})$
4 : ret $\sigma := \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{aux}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}, \widetilde{\sigma})$
```

**Figure 21:** Zero-Knowledge Simulator $\mathsf{Sim}$

Having described the Simulator $\mathsf{Sim}$, the experiments $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ are as in Figure 22. Observe that $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ is same as $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}$ except that NIZK is switched to simulation mode and IPFE ciphertext $\mathsf{ct}$ encrypts $\widetilde{\mathbf{x}} := (-\mathbf{t}^T, 1)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$ and functional keys $\mathsf{pk}_{\mathbf{y}}$ and $\mathsf{sk}_{\mathbf{y}}$ correspond to $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$. Observe that $f_{\mathbf{y}}(\mathbf{x}) = f_{\widetilde{\mathbf{y}}}(\widetilde{\mathbf{x}})$. To prove zero-knowledge, we introduce an intermediate hybrid experiment $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$ which is same as $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}$ except that only NIZK is switched to simulation mode. Note that by triangle inequality, it follows that

$$\begin{aligned}
&|\Pr[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| \\
&\quad \leq |\Pr[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| \\
&\quad\quad + |\Pr[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]|.
\end{aligned}$$

To complete the proof, we show in Claims A.12 and A.13 that each of the two terms on the right-hand-side are at most $\mathsf{negl}(\lambda)$. □

```
Experiments $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})$, $\boxed{\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})}$ , $\boxed{\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})}$ .

1 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ $\boxed{\boxed{(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)}}$
2 :  $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$, $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}')$, sample random coins $r_0$
3 :  Let $(\mathsf{mpk}, \mathsf{msk}) := \mathsf{IPFE.Setup}(\mathsf{pp}', 1^{\ell+1}; r_0)$ $\boxed{\boxed{(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IPFE.Setup}(\mathsf{pp}', 1^{\ell+1})}}$
4 :  Sample $\mathbf{t} \xleftarrow{\$} \mathcal{M}$
5 :  Let $\widetilde{\mathbf{x}} := (\mathbf{x}^T, 0)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$ $\boxed{\widetilde{\mathbf{x}} := (-\mathbf{t}^T, 1)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}}$
6 :  Sample random coins $r_1$, $\mathsf{ct} := \mathsf{IPFE.Enc}(\mathsf{mpk}, \widetilde{\mathbf{x}}; r_1)$ $\boxed{\boxed{\mathsf{ct} \leftarrow \mathsf{IPFE.Enc}(\mathsf{mpk}, \widetilde{\mathbf{x}})}}$
7 :  $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), (r_0, r_1, \mathbf{x}))$ $\boxed{\boxed{\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))}}$
8 :  $\mathsf{advt} := (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} := (\mathsf{msk}, \mathbf{t})$, $\mathsf{vk} := \bot$
9 :  $\mathsf{vk} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AuxGen}}(\cdot), \mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{advt}, X)$ $\boxed{\mathsf{vk} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AuxGen}^*}(\cdot,\cdot), \mathcal{O}_{\mathsf{Adapt}^*}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{advt}, X)}$
10 : ret view of $\mathcal{A}$

Oracles $\mathcal{O}_{\mathsf{AuxGen}}(\mathbf{y})$, $\boxed{\mathcal{O}^*_{\mathsf{AuxGen}}(\mathbf{y}, f_{\mathbf{y}}(\mathbf{x}))}$ .

1 :  Described in Figure 23

Oracles $\mathcal{O}_{\mathsf{Adapt}}(m, \mathbf{y}, \widetilde{\sigma})$, $\boxed{\mathcal{O}^*_{\mathsf{Adapt}}(m, \mathbf{y}, \widetilde{\sigma}, f_{\mathbf{y}}(\mathbf{x}))}$ .

1 :  Described in Figure 23
```

**Figure 22:** Zero-knowledge security of FAS: $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}$ is the real world experiment, $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$ is an intermediate hybrid experiment, $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ is the ideal world experiment.

**Claim A.12.** *Suppose* NIZK *satisfies zero-knowledge. Then, for every stateful p.p.t. adversary $\mathcal{A}$, for the p.p.t. simulator* NIZK.Sim, *there exists a negligible function* negl *such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, for all $(X, \mathbf{x}) \in R$,*

$$|\Pr[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| \leq \mathsf{negl}(\lambda).$$

*Proof.* Suppose towards a contradiction that for the stateful p.p.t. adversary $\mathcal{A}$ that makes no queries to the oracles $\mathcal{O}_{\mathsf{AuxGen}}$ and $\mathcal{O}_{\mathsf{Adapt}}$, there exists a p.p.t. distinguisher $\mathcal{D}$ and a non-negligible value $\epsilon$ such that

$$|\Pr[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| = \epsilon.$$

Then, we build a p.p.t. distinguisher $\mathcal{D}_{\mathsf{NIZK}}$ using $\mathcal{D}$ which breaks the zero-knowledge of NIZK for the NP language $L_{\mathsf{NIZK}}$. This should complete the proof.

Suppose the inputs to the distinguisher $D_{\mathsf{NIZK}}$ are the NIZK common reference string $\mathsf{crs}$ and a NIZK proof $\pi$ for some statement $(X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}) \in L_{\mathsf{NIZK}}$. Then, $\mathcal{D}_{\mathsf{NIZK}}$ is as follows. It sets $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$ and $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$. As $\mathcal{A}$ makes no oracle queries, the view of the adversary $\mathcal{A}$ against the the zero-knowledge security of functional adaptor signatures is simply $(\mathsf{pp}, \mathsf{advt}, X)$.

```
Oracles $\mathcal{O}_{\mathsf{AuxGen}}(\mathbf{y})$, $\boxed{\mathcal{O}^*_{\mathsf{AuxGen}}(\mathbf{y}, f_{\mathbf{y}}(\mathbf{x}))}$.
1 : Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$
2 : Compute $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$ $\boxed{\text{Compute } \widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}}$
3 : Compute $\mathsf{pk}_{\mathbf{y}} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$
4 : ret $\mathsf{aux}_{\mathbf{y}} := \mathsf{pk}_{\mathbf{y}}$, $\pi_{\mathbf{y}} := f_{\mathbf{y}}(\mathbf{t})$ $\boxed{\textbf{ret } \mathsf{aux}_{\mathbf{y}} := \mathsf{pk}_{\mathbf{y}},\ \pi_{\mathbf{y}} := f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x})}$

Oracles $\mathcal{O}_{\mathsf{Adapt}}(m, \mathbf{y}, \widetilde{\sigma})$, $\boxed{\mathcal{O}^*_{\mathsf{Adapt}}(m, \mathbf{y}, \widetilde{\sigma}, f_{\mathbf{y}}(\mathbf{x}))}$.
1 : Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, and $\mathsf{st} = (\mathsf{msk}, \mathbf{t})$. If $\mathsf{vk} = \bot$: ret $\bot$
2 : $(\mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) \leftarrow \mathcal{O}_{\mathsf{AuxGen}}(\mathbf{y})$ $\boxed{(\mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) \leftarrow \mathcal{O}^*_{\mathsf{AuxGen}}(\mathbf{y}, f_{\mathbf{y}}(\mathbf{x}))}$
3 : If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_{\mathbf{y}}, \pi_{\mathbf{y}}) = 0 \lor \mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{aux}_{\mathbf{y}}, \widetilde{\sigma}) = 0$ : ret $\bot$
4 : Compute $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$ $\boxed{\text{Compute } \widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}}$
5 : Compute $\mathsf{sk}_{\mathbf{y}} := \mathsf{IPFE.KGen}(\mathsf{msk}, \widetilde{\mathbf{y}})$
6 : ret $\sigma := \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{aux}_{\mathbf{y}}, \mathsf{sk}_{\mathbf{y}}, \widetilde{\sigma})$
```

**Figure 23:** Oracle descriptions for experiments $\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$, $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$.

Then, $\mathcal{D}_{\mathsf{NIZK}}$ runs the distinguisher $\mathcal{D}$ on inputs $(\mathsf{pp}, \mathsf{advt}, X)$ and returns whatever $\mathcal{D}$ returns. Then, observe that

$$\Pr\begin{bmatrix} \mathcal{D}_{\mathsf{NIZK}}(\mathsf{crs}, \pi) = 1 : \\ \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda), \\ \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{pp}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), (r_0, r_1, \mathbf{x})) \end{bmatrix}$$
$$= \Pr\begin{bmatrix} \mathcal{D}(\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}'), \mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi), X) = 1 : \\ \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda), \\ \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{pp}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), (r_0, r_1, \mathbf{x})) \end{bmatrix}$$
$$= \Pr\left[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1\right].$$

Similarly, we get that

$$\Pr\begin{bmatrix} \mathcal{D}_{\mathsf{NIZK}}(\mathsf{crs}, \pi) = 1 : \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda) \\ \pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, \mathsf{stmt} = (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})) \end{bmatrix}$$
$$= \Pr\begin{bmatrix} \mathcal{D}(\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}'), \mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi), X) = 1 : \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda) \\ \pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, \mathsf{stmt} = (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})) \end{bmatrix}$$
$$= \Pr\left[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1\right].$$

This implies that

$$\begin{aligned}
&\left|\begin{array}{l} \Pr\left[\begin{array}{c} \mathcal{D}_{\mathsf{NIZK}}(\mathsf{crs}, \pi) = 1 : \\ \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda), \\ \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{pp}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), (\mathbf{x}, r)) \end{array}\right] \\ - \Pr\left[\begin{array}{c} \mathcal{D}_{\mathsf{NIZK}}(\mathsf{crs}, \pi) = 1 : \\ (\mathsf{crs}, \pi) \leftarrow \mathsf{NIZK.Sim}(1^\lambda, \mathsf{stmt} = (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct})) \end{array}\right] \end{array}\right| \\
&= \left|\begin{array}{l} \Pr\left[\mathcal{D}(\mathsf{faZKReal}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1\right] \\ - \Pr\left[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1\right] \end{array}\right| \\
&= \epsilon.
\end{aligned}$$

As $\epsilon$ is non-negligible, hence, $\mathcal{D}_{\mathsf{NIZK}}$ breaks the zero-knowledge of the underlying NIZK scheme. □

**Claim A.13.** *Suppose $\mathcal{M}$ is an additive group and* IPFE *satisfies selective, IND-security. Then, for every stateful p.p.t. adversary $\mathcal{A}$, for the p.p.t. simulators* NIZK.Sim *and* Sim *(described in Figure 21), there exists a negligible function* negl *such that for all p.p.t. distinguishers $\mathcal{D}$, for all $\lambda \in \mathbb{N}$, for all $(X, \mathbf{x}) \in R$,*

$$|\Pr[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| \leq \mathsf{negl}(\lambda).$$

*Proof.* Observe that experiments $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$ and $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$ differ in the following two aspects:

- The vector $\widetilde{\mathbf{x}}$ that ciphertext $\mathsf{ct}$ encrypts: In $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$, we have $\widetilde{\mathbf{x}} := (\mathbf{x}^T, 0)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$, and in $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$, we have $\widetilde{\mathbf{x}} := (-\mathbf{t}^T, 1)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$.
- The vector $\widetilde{\mathbf{y}}$ used for computing $\mathsf{pk}_{\mathbf{y}}$ and $\mathsf{sk}_{\mathbf{y}}$ for every $\mathbf{y}$ queried to $\mathsf{AuxGen}$ and $\mathsf{Adapt}$ oracles: In $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$, we have $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$, and in $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$, we have $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_{\mathbf{y}}(\mathbf{t}) + f_{\mathbf{y}}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$.

Consider an intermediate hybrid $\overline{\mathsf{Hyb}}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$ that uses $\widetilde{\mathbf{x}}$ as in $\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}$ and $\widetilde{\mathbf{y}}$ as in $\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}$.

We argue that $\mathsf{Hyb}$ and $\overline{\mathsf{Hyb}}$ are identically distributed. The two differ in how $\widetilde{\mathbf{y}}$ is computed. Notice that by linearity, it follows that $f_{\mathbf{y}}(\mathbf{t}) + f_{\underset{\sim}{\mathbf{y}}}(\mathbf{x}) = f_{\underset{\sim}{\mathbf{y}}}(\mathbf{t} + \mathbf{x})$. Hence, one can view the change from $\mathsf{Hyb}$ to $\overline{\mathsf{Hyb}}$ as simply a change of variables $\widetilde{\mathbf{t}} = \mathbf{t} \to \widetilde{\mathbf{t}} = \mathbf{t} + \mathbf{x}$. As $\mathcal{M}$ is an additive group and $\mathbf{t}, \mathbf{x} \in \mathcal{M}$, hence, $\mathbf{t} + \mathbf{x} \in \mathcal{M}$. As $\mathbf{t} \overset{\$}{\leftarrow} \mathcal{M}$, hence, it follows that $\mathbf{t}$ and $\mathbf{t} + \mathbf{x}$ are identically distributed. Therefore, $\mathsf{Hyb}$ and $\overline{\mathsf{Hyb}}$ are identically distributed. In other words,

$$|\Pr[\mathcal{D}(\mathsf{Hyb}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\overline{\mathsf{Hyb}}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| = 0.$$

To complete the proof, we argue that if $\mathsf{IPFE}$ is selective, IND-secure, then,

$$|\Pr[\mathcal{D}(\overline{\mathsf{Hyb}}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| \leq \mathsf{negl}(\lambda).$$

Suppose towards a contradiction that for any stateful p.p.t. adversary $\mathcal{A}$, there exists a there exists a p.p.t. distinguisher $\mathcal{D}$ and a non-negligible value $\epsilon$ such that

$$|\Pr[\mathcal{D}(\overline{\mathsf{Hyb}}^{\mathsf{NIZK.Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1] - \Pr[\mathcal{D}(\mathsf{faZKIdeal}^{\mathsf{Sim}}_{\mathcal{A},\mathsf{FAS}}(1^\lambda, X, \mathbf{x})) = 1]| = \epsilon.$$

Reduction $\mathcal{B}(1^\lambda, X, \mathbf{x})$ for proof of Claim A.13.

```
1 :  $\mathsf{pp}' \leftarrow \mathcal{C}(1^\lambda)$
2 :  Sample $\mathbf{t} \xleftarrow{\$} \mathcal{M}$
3 :  Let $\widetilde{\mathbf{x}_0} := (\mathbf{x}^T, 0)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$
4 :  Let $\widetilde{\mathbf{x}_1} := (-\mathbf{t}^T, 1)^T \in \mathcal{M}' \subseteq \mathbb{Z}^{\ell+1}$
5 :  $(\mathsf{mpk}, \mathsf{ct}) \leftarrow \mathcal{C}(\widetilde{\mathbf{x}_0}, \widetilde{\mathbf{x}_1})$
6 :  $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)$
7 :  $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$
8 :  $\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, \mathsf{td}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))$
9 :  $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} = \mathbf{t}$
10 : $\mathsf{vk} = \bot$
11 : $\mathsf{vk} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AuxGen}^*}(\cdot,\cdot)\mathcal{O}_{\mathsf{Adapt}^*}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{advt}, X)$
12 : $b \leftarrow \mathcal{D}(\text{view of } \mathcal{A})$
13 : ret bit $b$ to $\mathcal{C}$
```

Oracle $\mathcal{O}^*_{\mathsf{AuxGen}}(\mathbf{y}, f_\mathbf{y}(\mathbf{x}))$.

```
1 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$ and $\mathsf{st} = \mathbf{t}$
2 :  Compute $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_\mathbf{y}(\mathbf{t}) + f_\mathbf{y}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$
3 :  Compute $\mathsf{pk}_\mathbf{y} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \widetilde{\mathbf{y}})$
4 :  ret $\mathsf{aux}_\mathbf{y} := \mathsf{pk}_\mathbf{y}$, $\pi_\mathbf{y} := f_\mathbf{y}(\mathbf{t}) + f_\mathbf{y}(\mathbf{x})$
```

Oracle $\mathcal{O}^*_{\mathsf{Adapt}}(m, \mathbf{y}, \widetilde{\sigma}, f_\mathbf{y}(\mathbf{x}))$.

```
1 :  Parse $\mathsf{advt} = (\mathsf{mpk}, \mathsf{ct}, \pi)$, and $\mathsf{st} = \mathbf{t}$
2 :  If $\mathsf{vk} = \bot$: ret $\bot$
3 :  $(\mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) \leftarrow \mathcal{O}^*_{\mathsf{AuxGen}}(\mathbf{y}, f_\mathbf{y}(\mathbf{x}))$
4 :  If $\mathsf{AuxVerify}(\mathsf{advt}, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}) = 0 \vee$
     $\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{aux}_\mathbf{y}, \widetilde{\sigma}) = 0$ :
        ret $\bot$
5 :  Compute $\widetilde{\mathbf{y}} := (\mathbf{y}^T, f_\mathbf{y}(\mathbf{t}) + f_\mathbf{y}(\mathbf{x}))^T \in \mathcal{F}_{\mathsf{IP},\ell+1}$
6 :  Compute $\mathsf{sk}_\mathbf{y} \leftarrow \mathsf{IPFE}.\mathcal{O}_{\mathsf{KGen}}(\widetilde{\mathbf{y}})$
7 :  ret $\sigma := \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{aux}_\mathbf{y}, \mathsf{sk}_\mathbf{y}, \widetilde{\sigma})$
```

**Figure 24:** Reduction $\mathcal{B}$ for proof of Claim A.13

Then, we build a p.p.t. reduction $\mathcal{B}$ using $\mathcal{D}$ that breaks the selective, IND-security of IPFE. This should complete the proof.

Let the IPFE challenger be $\mathcal{C}$ and let its $\mathsf{KGen}$ oracle be denoted be $\mathsf{IPFE}.\mathcal{O}_{\mathsf{KGen}}$. The reduction $\mathcal{B}$ is as in Figure 24. Observe that if the challenger $\mathcal{C}$ chooses to encrypt $\widetilde{\mathbf{x}_0}$, then, $\mathcal{A}$'s view is same as in $\overline{\mathsf{Hyb}}$ and if it chooses to encrypt $\widetilde{\mathbf{x}_1}$, then $\mathcal{A}$'s view is same as in $\mathsf{faZKIdeal}$. Thus, if $\mathcal{D}$ can distinguish the two views of $\mathcal{A}$ with a non-negligible distinguishing advantage $\epsilon$ against $\mathcal{B}$, then, $\mathcal{B}$ has the same non-negligible distinguishing advantage $\epsilon$ against $\mathcal{C}$. Thus, $\mathcal{B}$ breaks the selective, IND-security of IPFE.

□

## B Weakly-Secure FAS Construction

Recall that our main FAS construction was obtained by starting with a simulation-secure IPFE scheme. The construction was presented using an IND-secure IPFE and applying the IND-secure IPFE to simulation-secure IPFE compiler of [ALMT20] in a non-black-box way within our FAS construction. Here, we show that if (i) we do not apply the compiler and just use the IND-secure IPFE, (ii) use NIZK with adaptive zero-knowledge, then, the resulting FAS construction is weakly-secure. This means that it satisfies all security properties as before except that for witness privacy, it satisfies witness indistinguishability. We first define adaptive zero-knowledge of NIZK and then describe the FAS construction next.

**Definition B.1** (Adaptively Secure NIZK Argument System)**.** *An adaptively secure NIZK argument system must satisfy completeness and adaptive soundness as before. Additionally, it must satisfy* adaptive zero-knowledge *which requires that there exists a p.p.t. simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Prove}^*)$ *and there exists a negligible function* $\mathsf{negl}$ *such that for all p.p.t. distinguishers* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}$*, for* *all* $(\mathsf{stmt}, \mathsf{wit}) \in R$,

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

*where* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$, *and* $(\mathsf{crs}^*, \mathsf{td}) \leftarrow \mathsf{Setup}^*(1^\lambda)$, $\pi \leftarrow \mathsf{Prove}^*(\mathsf{crs}^*, \mathsf{td}, \mathsf{stmt})$. *Here,* $\mathsf{td}$ *is the trapdoor for the simulated* $\mathsf{crs}^*$ *that is used by* $\mathsf{Prove}^*$ *to generate an accepting proof for* $\mathsf{stmt}$ *without knowing the witness.*

Our generic construction of weakly-secure functional adaptor signatures will use the following building blocks:

- An inner product functional encryption scheme $\mathsf{IPFE}$ that satisfies selective, IND-security (Definition 3.11), $R_{\mathsf{IPFE}}$-compliance (Definition 3.13) and $R'_{\mathsf{IPFE}}$-robustness (Definition 3.14) w.r.t. hard relations $R_{\mathsf{IPFE}}$, $R'_{\mathsf{IPFE}}$ such that $R_{\mathsf{IPFE}} \subseteq R'_{\mathsf{IPFE}}$. The message space of $\mathsf{IPFE}$ is $\mathcal{M}' \subseteq \mathbb{Z}^\ell$ and the function family is $\mathcal{F}_{\mathsf{IP},\ell}$.
- An adaptor signature scheme $\mathsf{AS}$ w.r.t. a digital signature scheme $\mathsf{DS}$, and hard relations $R_{\mathsf{IPFE}}$ and $R'_{\mathsf{IPFE}}$ that satisfies witness extractability (Definition 3.7) and weak pre-signature adaptability (Definition 3.8) security properties.
- An adaptively secure $\mathsf{NIZK}$ (Definition B.1) for the NP language $L_{\mathsf{NIZK}}$ defined as

$$L_{\mathsf{NIZK}} := \left\{ (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}) : \begin{array}{c} \exists (r_0, r_1, \mathbf{x}) \text{ such that} \\ \mathsf{pp}' \in [\mathsf{IPFE.Gen}(1^\lambda)], \\ (\mathsf{mpk}, \mathsf{msk}) = \mathsf{IPFE.Setup}(\mathsf{pp}', 1^{\ell+1}; r_0), \\ (X, \mathbf{x}) \in R, \\ \mathsf{ct} = \mathsf{IPFE.Enc}(\mathsf{mpk}, \mathbf{x}; r_1) \end{array} \right\}.$$

Our construction is as in Figure 25. Observe that it uses $\ell$ slots instead of $\ell + 1$ slots in the underlying $\mathsf{IPFE}$, thus making the construction slightly more efficient than before. Further, compared to before, $\mathsf{AuxGen}$ and $\mathsf{AuxVerify}$ are redundant, thus, can be skipped in the protocol resulting in a non-interactive pre-signing stage. The construction can be instantiated from prime-order groups and lattices as before.

**Lemma B.2.** *Suppose NIZK satisfies correctness,* $\mathsf{AS}$ *satisfies correctness, and* $\mathsf{IPFE}$ *satisfies* $R_{\mathsf{IPFE}}$*-compliance and* $R'_{\mathsf{IPFE}}$*-robustness. Then, the functional adaptor signatures construction in Figure 25 satisfies correctness.*

*Proof.* Similar to the proof of Lemma 5.1. □

**Theorem B.3.** *Let* $\mathcal{F}_{\mathsf{IP},\ell}$ *be the function family for computing inner products of vectors of length* $\ell$*. Let* $R$ *be any NP relation with statement/witness pairs* $(X, \mathbf{x})$ *such that* $\mathbf{x} \in \mathcal{M}$ *for some set* $\mathcal{M} \subseteq \mathbb{Z}^\ell$*. Suppose that*

- $\mathcal{M}$ *is an additive group,*
- $R$ *is* $\mathcal{F}_{\mathsf{IP},\ell}$*-hard (Definition 3.2),*
- $\mathsf{NIZK}$ *is a secure NIZK argument system (Definition 3.9),*
- $\mathsf{AS}$ *is an adaptor signature scheme w.r.t. digital signature scheme* $\mathsf{DS}$ *and hard relations* $R_{\mathsf{IPFE}}, R'_{\mathsf{IPFE}}$ *that satisfies weak pre-signature adaptability (Definition 3.8) and witness extractability (Definition 3.7),*

```
Setup(1^λ)
  1 : Sample crs ← NIZK.Setup(1^λ)
  2 : Sample pp' ← IPFE.Gen(1^λ)
  3 : ret pp := (crs, pp')

AdGen(pp, X, x):
  1 : Sample random coins r_0, r_1
  2 : Let (mpk, msk) := IPFE.Setup(pp', 1^ℓ; r_0)
  3 : Let ct := IPFE.Enc(mpk, x; r_1)
  4 : Let π ← NIZK.Prove(crs, (X, pp', mpk, ct), (r_0, r_1, x))
  5 : ret advt := (mpk, ct, π), and st := msk

AdVerify(pp, X, advt)
  1 : ret NIZK.Vf(crs, (X, pp', mpk, ct), π)

AuxGen(advt, st, y)
  1 : ret aux_y = ⊥, π_y = ⊥

AuxVerify(advt, y, aux_y, π_y)
  1 : ret 1

FPreSign(advt, sk, m, X, y, aux_y)
  1 : Parse advt = (mpk, ct, π)
  2 : Let pk_y := IPFE.PubKGen(mpk, y)
  3 : ret σ̃ ← AS.PreSign(sk, m, pk_y)

FPreVerify(advt, vk, m, X, y, aux_y, π_y, σ̃)
  1 : Parse advt = (mpk, ct, π)
  2 : Let pk_y := IPFE.PubKGen(mpk, y)
  3 : ret (AuxVerify(advt, y, aux_y, π_y))∧
            (AS.PreVerify(vk, m, pk_y, σ̃))

Adapt(advt, st, vk, m, X, x, y, aux_y, σ̃)
  1 : Parse advt = (mpk, ct, π), and st = (msk, t)
  2 : Let pk_y := IPFE.PubKGen(mpk, y)
  3 : Let sk_y := IPFE.KGen(msk, y)
  4 : ret σ := AS.Adapt(vk, m, pk_y, sk_y, σ̃)

FExt(advt, σ̃, σ, X, y, aux_y)
  1 : Parse advt = (mpk, ct, π).
  2 : Let pk_y := IPFE.PubKGen(mpk, y)
  3 : Let z := AS.Ext(σ̃, σ, pk_y)
  4 : ret v := IPFE.Dec(z, ct)
```

**Figure 25:** Construction: Weakly-secure FAS

- IPFE *is a selective, IND-secure IPFE scheme (Definition 3.11) for function family* $\mathcal{F}_{\mathsf{IP},\ell}$ *that is* $R_{\mathsf{IPFE}}$*-compliant (Definition 3.13) and* $R'_{\mathsf{IPFE}}$*-robust (Definition 3.14).*

*Then, the functional adaptor signature scheme w.r.t. digital signature scheme* DS*, NP relation* $R$*, and family of inner product functions* $\mathcal{F}_{\mathsf{IP},\ell}$ *constructed in Figure 25 is weakly-secure (Definition 4.4).*

*Proof.* Follows from Lemmas A.1 to A.3, A.7, A.10 and B.4. □

## B.1 Witness Indistinguishability

**Lemma B.4.** *Suppose* NIZK *satisfies adaptive zero-knowledge and* IPFE *satisfies selective, IND-security. Then, the functional adaptor signature construction in Figure 2 is witness indistinguishable.*

*Proof.* We proof this through a sequence of games $G_0^b$ and $G_1^b$ for $b \in \{0, 1\}$.

**Game $G_0^b$:** This game corresponds to the original game $\mathsf{faWI}^b_{\mathcal{A},\mathsf{FAS}}$. Formally, the game is defined in Figure 26. For sake of simplicity, we drop the oracle $\mathcal{O}_{\mathsf{AuxGen}}$ as it is redundant in context of the construction.

**Game $G_1^b$:** This game is same as $G_0^b$ except that NIZK simulator is used instead of NIZK prover. Formally, the game is defined in Figure 26.

```
Games $G_0^b$, $\boxed{G_1^b}$ .
1 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$  $\boxed{\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)}$
2 :  $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
3 :  $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$
4 :  $(\mathsf{vk}, X, \mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}(\mathsf{pp})$
5 :  if $(((X, \mathbf{x}_0) \notin R) \vee ((X, \mathbf{x}_1) \notin R))$ : $\mathbf{ret} 0$
6 :  Sample random coins $r_0, r_1$
7 :  $(\mathsf{mpk}, \mathsf{msk}) := \mathsf{IPFE.Setup}(\mathsf{pp}', 1^\ell; r_0)$
8 :  $\mathsf{ct} := \mathsf{IPFE.Enc}(\mathsf{mpk}, \mathbf{x}_b; r_1)$
9 :  $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), (r_0, r_1, \mathbf{x}_b))$
     $\boxed{\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))}$
10 : $\mathsf{advt} := (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} := \mathsf{msk}$
11 : $(m, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}, \widetilde{\sigma}) \leftarrow \mathcal{A}(\mathsf{advt})$
12 : $\mathsf{pk}_\mathbf{y} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \mathbf{y})$
13 : If $\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{pk}_\mathbf{y}, \widetilde{\sigma}) = 0$: $\mathbf{ret}\ 0$
14 : If $f_\mathbf{y}(\mathbf{x}_0) \neq f_\mathbf{y}(\mathbf{x}_1)$: $\mathbf{ret}\ 0$
15 : $\mathsf{sk}_\mathbf{y} := \mathsf{IPFE.KGen}(\mathsf{msk}, \mathbf{y})$
16 : $\sigma = \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{pk}_\mathbf{y}, \mathsf{sk}_\mathbf{y}, \widetilde{\sigma})$
17 : $b' \leftarrow \mathcal{A}(\sigma)$
18 : $\mathbf{ret}\ b'$
```

**Figure 26:** Witness Indistinguishability proof: Games $G_0^b$ and $G_1^b$ for $b \in \{0, 1\}$

To prove the lemma, we need to show that

$$|\Pr[G_0^0(1^\lambda) = 1] - \Pr[G_0^1(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

Note that by triangle inequality, it follows that

$$\begin{aligned} &|\Pr[G_0^0(1^\lambda) = 1] - \Pr[G_0^1(1^\lambda) = 1]| \\ &\quad \leq |\Pr[G_0^0(1^\lambda) = 1] - \Pr[G_1^0(1^\lambda) = 1]| \\ &\qquad + |\Pr[G_1^0(1^\lambda) = 1] - \Pr[G_1^1(1^\lambda) = 1]| \\ &\qquad + |\Pr[G_1^1(1^\lambda) = 1] - \Pr[G_0^1(1^\lambda) = 1]|. \end{aligned}$$

To complete the proof, we show in Claim B.5 that the first and third terms on the right-hand-side are $\leq \mathsf{negl}(\lambda)$ and in Claim B.6 that the second term on the right-hand-side is $\leq \mathsf{negl}(\lambda)$.

□

**Claim B.5.** *If* NIZK *satisfies adaptive zero-knowledge, then, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, for all $b \in \{0, 1\}$,*

$$|\Pr[G_0^b(1^\lambda) = 1] - \Pr[G_1^b(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

*Proof.* We show that if there exists a p.p.t. adversary $\mathcal{A}$ can distinguish its view in games $G_0^b$ and $G_1^b$, then, we can create a distinguisher $\mathcal{D}^b$ that breaks the adaptive zero-knowledge of the underlying $\mathsf{NIZK}$. Let $\mathcal{C}$ be the adaptive zero-knowledge challenger for $\mathsf{NIZK}$. $\mathcal{C}$ samples a random bit $\beta \overset{\$}{\leftarrow} \{0,1\}$ and if $\beta = 0$, it samples $\mathsf{crs}$ and $\pi$ as in the real-world and if $\beta = 1$, it samples $\mathsf{crs}$ and $\pi$ using the $\mathsf{NIZK}$ simulator algorithms $\mathsf{Setup}^*$ and $\mathsf{Prove}^*$ respectively. Note that as the adaptive zero-knowledge of the NIZK proof system holds for all valid statement-witness pairs, we will allow the distinguisher $\mathcal{D}^b$ to choose a valid statement-witness pair adaptively after seeing $\mathsf{crs}$. The distinguisher $\mathcal{D}^b$ must output a bit at the end and it will use the adversary $\mathcal{A}$ for this task. The distinguisher $\mathcal{D}^b$ interacts with $\mathcal{A}$ and either presents it the view as in $G_0^b$ or as in $G_1^b$ and at the end $\mathcal{A}$ outputs its guess bit $\beta' \in \{0,1\}$. The distinguisher $\mathcal{D}^b$ is as in Figure 27.

```
Distinguisher $\mathcal{D}^b$ for proof of Claim B.5
1 : $\mathsf{crs} \leftarrow \mathcal{C}(1^\lambda)$
2 : $\mathsf{pp}' \leftarrow \mathsf{IPFE.Gen}(1^\lambda)$
3 : $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$
4 : $(\mathsf{vk}, X, \mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}(\mathsf{pp})$
5 : if $(((X, \mathbf{x}_0) \notin R) \vee ((X, \mathbf{x}_1) \notin R))$ : $\mathbf{ret} 0$
6 : Sample random coins $r_0, r_1$
7 : $(\mathsf{mpk}, \mathsf{msk}) := \mathsf{IPFE.Setup}(\mathsf{pp}', 1^\ell; r_0)$
8 : $\mathsf{ct} := \mathsf{IPFE.Enc}(\mathsf{mpk}, \mathbf{x}_b; r_1)$
9 : $\pi \leftarrow \mathcal{C}((X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}), (r_0, r_1, \mathbf{x}_b))$
10 : $\mathsf{advt} := (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} := \mathsf{msk}$
11 : $(m, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}, \widetilde{\sigma}) \leftarrow \mathcal{A}(\mathsf{advt})$
12 : $\mathsf{pk}_\mathbf{y} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \mathbf{y})$
13 : If $\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{pk}_\mathbf{y}, \widetilde{\sigma}) = 0$: $\mathbf{ret}$ 0
14 : If $f_\mathbf{y}(\mathbf{x}_0) \neq f_\mathbf{y}(\mathbf{x}_1)$: $\mathbf{ret}$ 0
15 : $\mathsf{sk}_\mathbf{y} := \mathsf{IPFE.KGen}(\mathsf{msk}, \mathbf{y})$
16 : $\sigma = \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{pk}_\mathbf{y}, \mathsf{sk}_\mathbf{y}, \widetilde{\sigma})$
17 : $\beta' \leftarrow \mathcal{A}(\sigma)$
18 : $\mathbf{ret}$ $\beta'$
```

**Figure 27:** Distinguisher $\mathcal{D}^b$ for proof of Claim B.5

Observe that if $\mathcal{C}$ chose bit $\beta$, then, the view observed by $\mathcal{A}$ is that of game $G_\beta^b$. Hence, it follows that

$$\begin{aligned}&|\Pr[\mathcal{D}^b(\mathsf{crs}, \pi) = 1 | \beta = 0] - \Pr[\mathcal{D}^b(\mathsf{crs}, \pi) = 1 | \beta = 1]| \\ &= |\Pr[G_0^b(1^\lambda) = 1] - \Pr[G_0^b(1^\lambda) = 1]|.\end{aligned}$$

Hence, if $\mathcal{A}$ has a noticeable distinguishing advantage $\epsilon$, then, $\mathcal{D}^b$ also has a noticeable distinguishing advantage $\epsilon$ in its game with challenger $\mathcal{C}$. This completes the proof. □

**Claim B.6.** *If* $\mathsf{IPFE}$ *satisfies selective, IND-security, then, there exists a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$,

$$|\Pr[G_1^0(1^\lambda) = 1] - \Pr[G_1^1(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

*Proof.* We show that if there exists a p.p.t. adversary $\mathcal{A}$ can distinguish its view in games $G_1^0$ and $G_1^1$, then, we can create a reduction $\mathcal{B}$ that breaks the selective, IND-security of the underlying $\mathsf{IPFE}$. For the $\mathsf{IPFE}$ selective, IND-security game, let $\mathcal{C}$ be the challenger and let $\mathsf{IPFE}.\mathcal{O}_{\mathsf{KGen}}$ be the key generation oracle that the reduction $\mathcal{B}$ has access to. The reduction $\mathcal{B}$ is as in Figure 28.

```
Reduction $\mathcal{B}$ for proof of Claim B.6

1 :  $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}^*(1^\lambda)$
2 :  $\mathsf{pp}' \leftarrow \mathcal{C}(1^\lambda)$
3 :  $\mathsf{pp} = (\mathsf{crs}, \mathsf{pp}')$
4 :  $(\mathsf{vk}, X, \mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}(\mathsf{pp})$
5 :  if $(((X, \mathbf{x}_0) \notin R) \vee ((X, \mathbf{x}_1) \notin R))$ : $\mathbf{ret} 0$
6 :  $(\mathsf{mpk}, \mathsf{ct}) \leftarrow \mathcal{C}(\mathbf{x}_0, \mathbf{x}_1)$
7 :  $\pi \leftarrow \mathsf{NIZK.Prove}^*(\mathsf{crs}, (X, \mathsf{pp}', \mathsf{mpk}, \mathsf{ct}))$
8 :  $\mathsf{advt} := (\mathsf{mpk}, \mathsf{ct}, \pi)$, $\mathsf{st} := \mathsf{msk}$
9 :  $(m, \mathbf{y}, \mathsf{aux}_\mathbf{y}, \pi_\mathbf{y}, \widetilde{\sigma}) \leftarrow \mathcal{A}(\mathsf{advt})$
10 : $\mathsf{pk}_\mathbf{y} := \mathsf{IPFE.PubKGen}(\mathsf{mpk}, \mathbf{y})$
11 : If $\mathsf{AS.PreVerify}(\mathsf{vk}, m, \mathsf{pk}_\mathbf{y}, \widetilde{\sigma}) = 0$: $\mathbf{ret}\ 0$
12 : If $f_\mathbf{y}(\mathbf{x}_0) \neq f_\mathbf{y}(\mathbf{x}_1)$: $\mathbf{ret}\ 0$
13 : $\mathsf{sk}_\mathbf{y} = \mathsf{IPFE}.\mathcal{O}_{\mathsf{KGen}}(\mathbf{y})$
14 : $\sigma = \mathsf{AS.Adapt}(\mathsf{vk}, m, \mathsf{pk}_\mathbf{y}, \mathsf{sk}_\mathbf{y}, \widetilde{\sigma})$
15 : $b' \leftarrow \mathcal{A}(\sigma)$
16 : $\mathbf{ret}\ b'$
```

**Figure 28:** Reduction $\mathcal{B}$ for proof of Claim B.6

Observe that when $\mathcal{C}$ plays game $\mathsf{IPFE\text{-}Expt\text{-}SEL}^b_\mathcal{A}(1^\lambda, n)$ with the reduction $\mathcal{B}$ for some $b \in \{0, 1\}$, then, $\mathsf{ct}$ encrypts $\mathbf{x}_b$ and thus, $\mathcal{A}$'s view is same as $G_1^b$. Further, observe that $\mathcal{B}$ makes query to the key generation oracle only if $f_\mathbf{y}(\mathbf{x}_0) = f_\mathbf{y}(\mathbf{x}_1)$. Thus, $\mathcal{B}$ is an admissible adversary in the game with $\mathcal{C}$. Hence, it follows that for all $b \in \{0, 1\}$,

$$\begin{aligned}\Pr[\mathsf{IPFE\text{-}Expt\text{-}SEL}^b_\mathcal{B}(1^\lambda, n) = 1] &= \Pr[b' = 1] \\ &= \Pr[G_1^b(1^\lambda) = 1].\end{aligned}$$

Therefore, we get that

$$\begin{aligned}&|\Pr[\mathsf{IPFE\text{-}Expt\text{-}SEL}^0_\mathcal{A}(1^\lambda, n) = 1] - \Pr[\mathsf{IPFE\text{-}Expt\text{-}SEL}^1_\mathcal{A}(1^\lambda, n) = 1]| \\ &\quad = |\Pr[G_1^0(1^\lambda) = 1] - \Pr[G_1^1(1^\lambda) = 1]|.\end{aligned}$$

Hence, if $\mathcal{A}$ has a noticeable distinguishing advantage $\epsilon$ in its game with reduction $\mathcal{B}$, then, $\mathcal{B}$ also has a noticeable distinguishing advantage $\epsilon$ in its game with challenger $\mathcal{C}$. This completes the proof. □