On Instantiating Unleveled Fully-Homomorphic Signatures from Falsifiable Assumptions

Romain Gay[1] and Bogdan Ursu[2](✉)

[1] IBM Research, Zurich, Switzerland
rga@zurich.ibm.com
[2] Consensys, Fort Worth, USA
bogdan.ursu@consensys.net

Abstract. We build the first *unleveled* fully homomorphic signature scheme in the standard model. Our scheme is not constrained by any a-priori bound on the depth of the functions that can be homomorphically evaluated, and relies on subexponentially-secure indistinguishability obfuscation, fully-homomorphic encryption and a non-interactive zero-knowledge (NIZK) proof system with composable zero-knowledge. Our scheme is also the first to satisfy the strong security notion of context-hiding for an unbounded number of levels, ensuring that signatures computed homomorphically do not leak the original messages from which they were computed. All building blocks are instantiable from falsifiable assumptions in the standard model, avoiding the need for knowledge assumptions.

The main difficulty we overcome stems from the fact that bootstrapping, which is a crucial tool for obtaining unleveled fully homomorphic encryption (FHE), has no equivalent for homomorphic signatures, requiring us to use novel techniques.

1 Introduction

Fully Homomorphic Signatures. A signature scheme is said to be homomorphic when given signatures $\sigma_1, \dots, \sigma_n$ of messages $m_1, \dots, m_n$, it is possible to publicly compute a signature σ_f of the message $f(m_1, \dots, m_n)$ for any function f. This evaluated signature σ_f is verified with respect to the verification key of the scheme, the message $m = f(m_1, \dots, m_n)$ and the function f.

Given a set of signatures $\sigma_1, \dots, \sigma_n$, unforgeability prevents an adversary from deriving a signature σ_f that verifies with respect to a function f and a message $y \neq f(m_1, \dots, m_n)$. In other words, the signature certifies that the message corresponds to the proper evaluation of the function f on the original messages.

Akin to homomorphic encryption, the signing algorithm is a homomorphism from the message space to the signature space. Computing the addition of signatures $\sigma_1 \boxplus \sigma_2$ results in the signature of the message $m_1 + m_2$, where $\boxplus$ and $+$

The corresponding author is based in Zurich, Switzerland, and this work was carried out during his time at ETH Zurich.

Q. Tang and V. Teague (Eds.): PKC 2024, LNCS 14601, pp. 74–104, 2024.
https://doi.org/10.1007/978-3-031-57718-5_3

denote the addition in the signature and message space, respectively. The same goes for multiplication. Schemes equipped with a ring homomorphism (with both addition and multiplication) are referred to as *fully* homomorphic, since these operations are sufficient to capture all possible Boolean functions.

Applications of FHS. Homomorphic signatures are applicable in a wide range of scenarios, such as:

- Integrity for Network Coding. Network performances can be improved by encoding ongoing messages into vectors and letting each node perform linear operations on these encodings, instead of simply forwarding them. Unfortunately, because these encodings are modified by every node, the integrity property is lost when using traditional signatures. Homomorphic signatures (or their secret-key counterpart, as in [AB09]) that support linear operations can be used to preserve integrity throughout the network. In particular, each node updates not only the encoded messages, but also the homomorphic signatures associated with them.
- Verifying Delegated Computations. A client that wishes to delegate some computation on his data to a cloud provider could authenticate it via homomorphic signatures, then send it away to the cloud. The cloud performs the computation and updates the signatures accordingly, then sends the result back to the client, who can then verify the evaluated signature. If verification is successful, then the client is guaranteed that the cloud computed the intended function on the data. It is the perfect complement to fully homomorphic encryption (FHE), which preserves the confidentiality of the data in use, but not its integrity.
- Anonymous Credentials. Consider the scenario where a user obtains signatures $\sigma_1, \ldots, \sigma_n$ of her credentials $m_1, \ldots, m_n$, produced by some authority (the authority is associated to the vk of the signature scheme). Later on, the user is asked by a service provider (say, an insurance company) to prove that her credentials satisfy a policy expressed by a predicate P. The user can compute the signature σ_{P} and send it to the provider. If this signature verifies successfully with respect to vk and the message 1 (the output of the predicate should be 1), then it proves the user's credentials fulfill the policy. Assuming the homomorphic signatures satisfy some mild re-randomizability property (so that evaluated signatures look fresh), this does not reveal the underlying credentials to the provider (only that they satisfy the policy). Giving the policy explicitly to the user provides some transparency (for instance, the predicate P can be signed by a trusted regulator, ensuring the insurance company is not performing some discriminatory screening). We can even evaluate a function f on the signatures that not only indicates whether the user is eligible to an insurance scheme, but also outputs the price to be paid based on the credentials.

State of the Art. The first construction of homomorphic signatures [AB09] was limited to additive homomorphism in the secret-key setting i.e. it is a message authentication (MAC) scheme. Later on, [BF11a] built the first homomorphic signature for constant-degree polynomials, subsequently improved by [CFW14].

In [GW13], the authors built the first fully homomorphic MAC from FHE, while [CF13] built an homomorphic MAC with better efficiency for a restricted class of functions. Then, [GVW15] built the first leveled fully homomorphic signature (FHS) scheme.

All existing works suffer from the fact that the depth of the functions that can be homomorphically evaluated is bounded at setup. In other words, these are *leveled* FHS. This stands in contrast with FHE, where *unleveled* schemes can be obtained via bootstrapping [Gen09] and circular security. Bootstrapping requires an FHE encryption of the secret decryption key, and relies on evaluating homomorphically the (shallow) decryption algorithm to "refresh" ciphertexts. This idea is not straightforwardly transferable to the signature case, and unleveled FHS have so far been elusive.

Another approach to building FHS is to use Succinct Arguments of Knowledge (SNARKs) for NP, but this requires the use of strong knowledge assumptions, which we discuss in more detail in the full version of this paper [GU23].

Given this state of affair, a natural question comes up:

Can we build unleveled FHS from falsifiable assumptions?

This was left as an open problem in [GVW15], and has remained unsolved until our construction.

Our Result. We answer the question positively. Namely, we build the first *unleveled* FHS from falsifiable assumptions, in the standard model. Our feasibility result relies on indistinguishability obfuscation (iO), of which promising constructions appeared recently in [BDGM20a, JLS21, GP21, WW21, AP20, BDGM20b, DQV+21, JLS22], unleveled fully homomorphic encryption and a non-interactive zero-knowledge proof system (NIZK). While iO is not a falsifiable assumption itself[1], most of the iO candidates rely on falsifiable assumptions. The second building block, fully-homomorphic encryption, can be instantiated using circularly-secure LWE [GSW13], and alternatively using indistinguishability obfuscation [CLTV15]. Instantiating the FHE scheme using [CLTV15] yields a fully homomorphic signature construction that does not require any circular security assumption.

Building Blocks. We give more details on the building blocks, and the assumptions over which they can be instantiated. To build our FHS, we use an unleveled Fully-Homomorphic Encryption (FHE) scheme, which can be chosen to be either:

- a variant of the FHE scheme from [GSW13], slightly modified to ensure that it has unique random coins (which is needed for technical reasons in the proof). This scheme can be built from circularly-secure LWE.
- the FHE scheme of [CLTV15], which is instantiable using subexponentially-secure iO and a re-randomizable public-key encryption scheme. This second type of FHE scheme does not require a circular assumption. Moreover, the

[1] Formally, the iO security game does not fulfill falsifiability because the challenger cannot efficiently check that the circuits submitted by the adversary are functionally equivalent.

re-randomizable encryption scheme can be any one of the following: Goldwasser-Micali [GM82], ElGamal [ElG85], Paillier [Pai99] or Damgard-Jurik [DJ01] (which are secure assuming QR, DDH, or DCR).

Moreover, we rely on Non-Interactive Zero Knowledge (NIZK) proof systems satisfying a proof of knowledge property and composable zero-knowledge, which can also be built from subexponentially secure iO and lossy trapdoor functions [HU19]. Lossy trapdoor functions can be based on a multitude of standard assumptions such as DDH, k-LIN, QR or DCR. Other NIZK systems also offer the properties required, but from bilinear maps [GS08].

The NIZKs above [HU19, GS08] allow that the common reference string (CRS) can be either generated honestly to be binding, which ensures soundness (i.e. the fact that only true statements can be proved), or alternatively, the CRS is generated in a hiding way, providing a simulation mode that ensures zero-knowledge. In fact, the binding CRS is generated together with an extraction trapdoor that can be used to extract efficiently a witness from any valid proof (thereby ensuring that the statement proved is indeed true). The simulated CRS is generated together with a simulation trapdoor. In this case, the simulation trapdoor can be used to generate proofs on any statement (without requiring a witness). The two modes (real or simulated) are computationally indistinguishable.

Technical Overview

Overview of Our Construction. The verification key vk of our scheme contains several FHE encryptions of an arbitrary message (for example the message equals to 0). The number of such encryptions, N, determines the arity of the functions that can be homomorphically evaluated. We require that the FHE is unleveled. This differs from the FHS scheme from [GVW15] which uses homomorphic commitments instead of FHE encryptions. They crucially rely on the fact that these commitments are non-binding, which prevents from bootstrapping and only yields leveled FHS. To produce signatures, we rely on the NIZK proof system. To sign a message m_i for $i = 1, \ldots, N$, the signer produces a simulated proof stating (falsely) that the i'th encryption from vk, which we denote by ct_i, is an FHE encryption of m_i. This can be done since the NIZK common reference string CRS is simulated with an associated simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$. Creating these simulated proofs requires the trapdoor, which is set to be the signing key. A signature is simply the ZK proof π_i stating that the ciphertext ct_i is an encryption of m_i. To homomorphically evaluate a function f on signatures $\sigma_1, \ldots, \sigma_N$ of the messages $m_1, \ldots, m_N$, we use an obfuscated circuit containing the simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$ that, given as input the tuple $(\sigma_1, m_1, \ldots, \sigma_n, m_n, f)$, first checks that the signatures σ_i are valid ZK proofs (of false statements), by running the verification algorithm of the NIZK proof system. If the check is successful, then it homomorphically evaluates the function f on the FHE encryptions $\mathsf{ct}_1, \ldots, \mathsf{ct}_N$ that are part of vk, which yields an

FHE ciphertext ct_f. It also generates a proof π that ct_f is an FHE encryption of $f(m_1, \ldots, m_n)$, using $\mathsf{td}_{\mathsf{sim}}$. The signature σ_f is set to be the proof π, which the evaluation circuit outputs. To verify a signature σ_f with respect to a function f and a value y, the verifier algorithm computes ct_f by evaluating f on the FHE encryptions $\mathsf{ct}_1, \ldots, \mathsf{ct}_N$ from vk and verifies that σ is a valid proof stating that ct_f is an FHE encryption of y.

Let us now have a look at the proof of unforgeability. For simplicity, we consider the selective setting, where the adversary first sends messages $m_1^\star, \ldots, m_n^\star$, then receives vk and the signatures $\sigma_1^\star, \ldots, \sigma_n^\star$. Finally, the adversary sends a forgery (σ_f, f, y). It wins if the signature σ_f verifies successfully with respect to vk, f, y and $y \neq f(m_1^\star, \ldots, m_n^\star)$. The first step of the proof is to switch the FHE encryptions $\mathsf{ct}_1 \ldots \mathsf{ct}_N$ of 0 in the vk to FHE encryptions of $m_1^\star, \ldots, m_n^\star$, respectively. This way, we can change the signatures $\sigma_i^\star$ to proofs that are computed using a witness (where the witness is the randomness used to compute the FHE encryptions in vk). The main implication is that we do not need to simulate proofs using $\mathsf{td}_{\mathsf{sim}}$ anymore. The intent is to get rid of $\mathsf{td}_{\mathsf{sim}}$ altogether and switch to an honestly computed CRS so that we can use the soundness of the NIZK to prevent forgeries. Unfortunately it is not clear at this point how to remove $\mathsf{td}_{\mathsf{sim}}$ from Eval, the obfuscated circuit that performs the homomorphic evaluations. What if we use proofs of knowledge? This way, if the signatures input to the Eval algorithm are valid ZK proofs, then Eval can efficiently extract witnesses (i.e. randomness of the corresponding FHE ciphertexts), which can be used to compute the randomness of the evaluated FHE ciphertext. This requires a so-called randomness homomorphism of the FHE scheme. Namely, given the secret key of the FHE sk, randomness r_1, r_2 and messages m_1, m_2 such that $\mathsf{ct}_1 = \mathsf{FHE.Enc}(\mathsf{pk}, m_1; r_1)$ and $\mathsf{ct}_2 = \mathsf{FHE.Enc}(\mathsf{pk}, m_2; r_2)$, one can compute a randomness r such that $\mathsf{FHE.EvalNAND}(\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{FHE.Enc}(\mathsf{pk}, \mathsf{NAND}; r)$. A stronger property where a randomness r can be computed only using the pk is satisfied by most lattice-based FHE schemes (e.g. [GSW13]) and the secret-key variant is satisfied by the FHE scheme from [CLTV15]. Then, Eval can use this randomness r as a witness to produce the ZK proof that constitutes the evaluated signature σ_f.

This approach runs into a circular issue: while it is true that the $\sigma_i^\star$ are proofs that are computed without $\mathsf{td}_{\mathsf{sim}}$, to use the proof of knowledge property and extract witnesses, we need to first remove $\mathsf{td}_{\mathsf{sim}}$ and switch to an honestly generated CRS. To do so, we need Eval to produce the signatures σ_f without $\mathsf{td}_{\mathsf{sim}}$, but using witnesses instead, which already requires the proof of knowledge property and an honest CRS.

To solve this circular issue, our scheme uses a different NIZK proof system for each depth level of the circuit that is homomorphically evaluated. That is, to evaluate a function f, represented as a depth d circuit, we evaluate the circuit gate by gate. Starting at the level 0, signatures $\sigma_1, \ldots, \sigma_n$ of messages $m_1, \ldots, m_n$ are ZK proofs stating (falsely) that the FHE ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_N$ from vk are encryptions of $m_1, \ldots, m_n$, respectively, computed using crs_0, a simulated crs, together with a simulation trapdoor $\mathsf{td}_{\mathsf{sim}}^0$. Then Eval takes as input these level

0 signatures $\sigma_1, \ldots, \sigma_n$, the messages $m_1, \ldots, m_n$ and a n-ary gate g, verifies that the σ_i are valid proofs, computes the gate g on the messages which yields the value $y = g(m_1, \ldots, m_n)$, homomorphically evaluates g on the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ which yields ct_g, and computes a ZK proof π stating that ct_g is an FHE encryption of y using crs_1, a simulated crs, together with a simulation trapdoor $\mathsf{td}^1_{\mathsf{sim}}$. The Eval algorithm performs just one more level of the homomorphic computation. It is repeated many times to obtain the final signature σ_f for the function f. To keep track of the gate-by-gate evaluation of the circuit, each signature will be of the form $\sigma = (\pi, i, \mathsf{ct})$, where $i \in \mathbb{N}$ indicates the level of the signature, π is a proof computed using $(\mathsf{crs}_i, \mathsf{td}^i_{\mathsf{sim}})$, and ct is an homomorphically evaluated ciphertext (if $i = 0$ it is one ciphertext from vk). This way, Eval takes as input signatures of level i, and outputs signatures of level $i + 1$.

To prove the unforgeability of this scheme, as before, we start by replacing the FHE ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_N$ from the vk to encryptions of the messages $m^\star_1, \ldots, m^\star_N$ chosen by the adversary, using the semantic security of FHE. Then, we generate level 0 signatures using witnesses (the randomness used to compute the ct_i) instead of $\mathsf{td}^0_{\mathsf{sim}}$. At this point, we can switch crs_0 to a real CRS, generated along with an extraction trapdoor, since $\mathsf{td}^0_{\mathsf{sim}}$ is not used anymore. The rest of the proof proceeds using a hybrid argument over all the levels $i = 1, \ldots, d$ where d is the (unbounded) depth of the circuit chosen by the adversary. By induction, we assume crs_i is generated honestly along with an extraction trapdoor $\mathsf{td}^i_{\mathsf{ext}}$. Therefore, we can switch the way Eval computes the ZK proof for the level $i + 1$. Instead of using a simulation trapdoor $\mathsf{td}^{i+1}_{\mathsf{sim}}$ with respect to crs_{i+1} and computing simulated proofs, it instead extracts witnesses from the level i signatures using $\mathsf{td}^i_{\mathsf{ext}}$, and uses them to compute the proofs without the trapdoor $\mathsf{td}^{i+1}_{\mathsf{sim}}$. At this point $\mathsf{td}^{i+1}_{\mathsf{sim}}$ is not used anymore so we can also switch crs_{i+1} to a real CRS, and go to the next step until we reach the depth of the function f chosen by the adversary.

While using a different CRS for each level seems to solve the circularity issue, this approach creates another problem: if we simply generate all crs_i for all levels in advance and put them in vk, we necessarily have to bound the maximum depth of the functions that can be homomorphically evaluated. In other words, we have a leveled FHS. To avoid that, Eval samples the crs_i on the fly using a pseudorandom function (the key of the PRF is hard-coded in the obfuscated circuit Eval). This complicates the security proof, but it can be made to work using puncturing techniques. Namely, to switch crs_i from a simulated to real CRS and use the proof of knowledge property of the proof system associated to crs_i, we need crs_i to be generated with truly random coins, as opposed to a PRF. We simply hard-code the PRF value on i, puncture the PRF key, and switch the value to random (this is a standard technique for security proofs using iO, see for instance [SW14]). The crucial fact that makes these techniques applicable is that at any point in our security proof, we only require the CRS of one specific level to be generated with truly random coins. That is, we only need to hard-code the value of one CRS to perform the hybrid argument that goes over each level one by one. Ultimately, we show that the CRS for the last level, which corresponds to the depth of f chosen by the adversary, is generated honestly, and the soundness of the proof system directly prevents any successful FHS forgery.

High-Level Description of our FHS Scheme. In this description, $\mathsf{SimSetup}$ generates a simulated CRS with an associated simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$. In the unforgeability proof, we will use the honest variant Setup that generates a real CRS along with an extraction trapdoor $\mathsf{td}_{\mathsf{ext}}$. For simplicity, we consider an algorithm Eval that only evaluates binary NAND gates (this is without loss of generality). Our scheme is as follows:

- $\mathsf{vk} = (\mathsf{FHE.Enc}(0), \ldots, \mathsf{FHE.Enc}(0), \mathsf{crs}_0)$, where $(\mathsf{crs}_0, \mathsf{td}^0_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^\lambda)$, where $\lambda \in \mathbb{N}$ denotes the security parameter. The verification key vk contains N FHE encryptions of 0, namely $\mathsf{ct}_1 \ldots \mathsf{ct}_N$.
- $\mathsf{sk} = K$, where K is a PRF key.
- $\mathsf{EvalNAND}\Big((\sigma_0, m_0), (\sigma_1, m_1)\Big) = \widetilde{\mathcal{C}_{[\mathsf{td}^0_{\mathsf{sim}}, K]}}\Big((\sigma_0, m_0), (\sigma_1, m_1)\Big)$, where $\widetilde{\mathcal{C}_{[\mathsf{td}^0_{\mathsf{sim}}, K]}}$ denotes an obfuscation of the circuit $\mathcal{C}_{[\mathsf{td}^0_{\mathsf{sim}}, K]}$ that has the values $\mathsf{td}^0_{\mathsf{sim}}$ and K hard-coded, described in Fig. 1 below, σ_0 and σ_1 are signatures of level $i \in \mathbb{N}$ of the messages m_0 and m_1 respectively, and a binary NAND gate is homomorphically evaluated.
- $\mathsf{Verify}(\sigma_f, f, y)$: parses σ_f as (ct, π, d). Proof π is a ZK proof with respect to crs_d where d is the depth of f and $(\mathsf{crs}_d, \mathsf{td}_d) = \mathsf{SimSetup}(1^\lambda; \mathsf{PRF}_K(i))$, i.e. $\mathsf{SimSetup}$ is run on the pseudorandom coins $\mathsf{PRF}_K(d)$. Then, it homomorphically evaluates f on the ciphertexts $\mathsf{ct}_i = \mathsf{FHE.Enc}(0)$ from vk to obtain ct_f. It checks that π is a valid proof stating that ct_f is an encryption of y, with respect to crs_d (it outputs 1 if the check passes, 0 otherwise). Note that the ciphertext ct that is part of the signature is not used by Verify. It is only useful if extra homomorphically evaluation are to be performed on the evaluated signature.

$\mathcal{C}_{[\mathsf{td}_0, K]}((\sigma_0, m_0), (\sigma_1, m_1))$:

It parses $\sigma_0 = (\pi_0, i, \mathsf{ct}_0)$ and $\sigma_1 = (\mathsf{ct}_1, \pi_1, i)$ where $i \in \mathbb{N}$ denotes the level of these signatures, $\mathsf{ct}_0, \mathsf{ct}_1$ denotes FHE ciphertexts, and π_0, π_1 denotes ZK proofs.

- If $i > 0$, then it computes $(\mathsf{crs}_i, \mathsf{td}^i_{\mathsf{sim}}) = \mathsf{SimSetup}(1^\lambda; \mathsf{PRF}_K(i))$ and $(\mathsf{crs}_{i+1}, \mathsf{td}^{i+1}_{\mathsf{sim}}) = \mathsf{SimSetup}(1^\lambda; \mathsf{PRF}_K(i+1))$.
- If $i = 0$, then it only computes $(\mathsf{crs}_{i+1}, \mathsf{td}^{i+1}_{\mathsf{sim}}) = \mathsf{SimSetup}(1^\lambda; \mathsf{PRF}_K(i+1))$, since crs_0 has already been generated (it is part of vk).

Then it checks that π_b is a valid proof stating that ct_b is a ciphertext of m_b, with respect to crs_i, for all $b \in \{0, 1\}$. If any of these checks fail, it outputs $\perp$. Otherwise, it evaluates homomorphically the NAND gate on the ciphertexts ct_0 and ct_1 to obtain ct, computes $m = \mathsf{NAND}(m_0, m_1)$, and produces a proof π stating that ct is a encryption of m, using the trapdoor $\mathsf{td}^{i+1}_{\mathsf{sim}}$. It then outputs $\sigma = (\mathsf{ct}, \pi, i + 1)$.

Fig. 1. Circuit $\mathcal{C}_{[\mathsf{td}_0, K]}(\cdot, \cdot)$ used by Eval.

We summarize the unforgeability proof using the list of hybrid games presented in Fig. 2. Note that $\mathsf{G}_{3.0} = \mathsf{G}_2$, and in the last game $\mathsf{G}_{3.d}$, where d denotes the depth of the function f chosen by the adversary, security simply follows from the soundness of the level d NIZK.

Complexity Leveraging and Adaptive Security. In the overview above, we skipped over some technical details. In the unforgeability proof of our FHS scheme, the challenger that interacts with the adversary does not know in advance the depth d of the function f chosen. To solve this problem, the challenger chooses a super-polynomial e.g. $2^{\omega(\log \lambda)}$ number of levels to perform the hybrid argument sketched above. This gives a super-polynomial security loss, which is why we require subexponential security of the underlying assumptions. A similar complexity leveraging argument can be used to obtain adaptive security, where the adversary is not restricted to choose the messages $m_1^\star, \ldots, m_N^\star$ before seeing the verification key of the scheme. The challenger guesses in advance the messages and acts as though the adversary were selective. The security loss due to the guessing argument is 2^N, which we can accommodate by choosing appropriately large parameters, relying again on the subexponential security of the underlying building blocks.

Unique Randomness. For technical reasons, we require additionally that the FHE has unique randomness: given a message m and a ciphertext

- G_0: $\mathsf{vk} = \{\mathsf{FHE.Enc}(0)\}_i, (\mathsf{crs}_0, \mathsf{td}^0_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^\lambda)$, $\sigma_i^\star$ simulated with $\mathsf{td}^0_{\mathsf{sim}}$. // original security game.

- G_1: $\mathsf{vk} = \{ \mathsf{FHE.Enc}(m_i^\star) \}_i, (\mathsf{crs}_0, \mathsf{td}^0_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^\lambda)$, $\sigma_i^\star$ simulated with $\mathsf{td}^0_{\mathsf{sim}}$. // security of FHE

- G_2: $\mathsf{vk} = \{\mathsf{FHE.Enc}(m_i^\star; r_i)\}_i$, $(\mathsf{crs}_0, \mathsf{td}^0_{\mathsf{ext}}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\sigma_i^\star$ proved with r_i. // real CRS

- $\mathsf{G}_{3.\ell}$: // games defined for all $\ell = 0, \ldots, d$, where d is the depth of f

Eval uses the obfuscation of the following circuit which has
the pair $(\mathsf{crs}_\ell, \mathsf{td}^\ell_{\mathsf{ext}}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the PRF key K hard-coded:
$\mathcal{C}_{[\mathsf{crs}_\ell, \mathsf{td}^\ell_{\mathsf{ext}}, K]}((\sigma_0, m_0), (\sigma_1, m_1))$:
- Parse $\sigma_b = (\mathsf{ct}_b, \pi_b, j)$, for $b \in \{0, 1\}$
- Compute $\mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{NAND}, \mathsf{ct}_0, \mathsf{ct}_1)$
- If $j < \ell$, then compute $(\mathsf{crs}_j, \mathsf{td}^j_{\mathsf{ext}}) = \mathsf{Setup}(1^\lambda; \mathsf{PRF}_K(j))$,
  extract witnesses (r_0, r_1) from (π_0, π_1) using $\mathsf{td}^j_{\mathsf{ext}}$,
  compute r such that $\mathsf{ct} = \mathsf{FHE.Enc}(\mathsf{NAND}(m_0, m_1); r)$ using r_0, r_1, m_0, m_1,
  compute a proof π that ct encrypts $\mathsf{NAND}(m_0, m_1)$ using r.
- If $j \geq \ell$, then compute $(\mathsf{crs}_{j+1}, \mathsf{td}^{j+1}_{\mathsf{sim}}) = \mathsf{SimSetup}(1^\lambda; \mathsf{PRF}_K(j+1))$
  and compute the proof π with $\mathsf{td}^{j+1}_{\mathsf{sim}}$ instead.
- Output $\sigma = (\pi, \mathsf{ct}, j+1)$.

Fig. 2. Hybrid games for the selective unforgeability proof of our FHS. We denote by $m_i^\star$ the message sent by the adversary, by $\sigma_i^\star$ the signatures it receives, by $\mathsf{SimSetup}$ the algorithm that generates a simulated CRS with a trapdoor $\mathsf{td}_{\mathsf{sim}}$, by $\mathsf{Setup}(1^\lambda)$ the honest variant that generates a real CRS together with an extraction trapdoor and by K a puncturable PRF key. We denote by $\mathsf{Setup}(1^\lambda; r)$ the algorithm Setup run with coins r (which can be truly random or pseudo random). When omitted, truly random coins are implicitly used. We use the same notations when writing $\mathsf{SimSetup}(1^\lambda; r)$ or $\mathsf{FHE.Enc}(m; r)$.

$\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; r)$ there cannot be another randomness $r' \neq r$ such that $\mathsf{Enc}(\mathsf{pk}, m; r') = \mathsf{ct}$. In the full version of this paper [GU23], we show that a slight modification of the GSW FHE scheme [GSW13] directly achieves such a property. We also show that the FHE from [CLTV15] can be adapted straightforwardly to obtain unique randomness. Simply put, their scheme relies on iO and a re-randomizable encryption scheme (such as Goldwasser Micali, ElGamal, Paillier or Damgard-Jurik). If the latter has unique randomness, then the resulting FHE also has this property.

Related Works. The work of [JMSW02] introduced a similar notion of homomorphic signature but where the verification algorithm does not take the function f as an input. That is, signatures can be manipulated homomorphically, thereby changing the underlying message being signed, but the verification does not track which function was applied. In that case, the notion of unforgeability only makes sense when the homomorphism property is limited, so that from a set of signatures, one can only get a signature on some but not all messages. Typically, the messages are vectors, and given signatures on vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, only signatures on the linear combinations of the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ can be obtained. In particular if n is less than the dimension of the vectors, then there are some vectors for which signatures cannot be generated (those outside the span of $\mathbf{v}_1, \ldots, \mathbf{v}_n$) and the unforgeability property is meaningful. These are referred to as linearly homomorphic signatures, such as in [BF11b, Fre12, ALP13, LPJY15, CFN15, CLQ16, HPP20]. This is similar to the notion of equivalence-class signatures [HS14, FHS19, FG18, KSD19], where signatures can be combined homomorphically within a given equivalence class, but forgeries outside the class are prohibited. The notion also requires a re-randomizability property, and is used in particular for anonymous credentials.

Other works [LTWC18, FP18, AP19, SBB19] consider the multi-key extension of homomorphic signatures, where the signatures to be homomorphically evaluated come from different users with different signing keys.

In [BFS14], the authors provide a fully-homomorphic signature from lattices that has the advantage of being adaptively secure (where the adversary can send the messages of her choice after receiving the verification key in the security game). In [CFN18], the authors study the security notions of homomorphic signatures in the adaptive setting, provide a simpler and stronger definition, and a compiler that generically strengthens the security of a scheme. The work of [Tsa17] establishes an equivalence between homomorphic signatures and the related notion of attribute-based signatures, and provides new constructions for both.

Another recent line of work [CFT22, BCFL23] on functional commitments also addresses the problem of homomorphic signatures. [BCFL23] instantiates the framework of [CFT22] with a functional commitment for circuits of unbounded depth, resulting in a homomorphic signature that supports circuits of unbounded depth (though the circuit width is bounded). In this way, [BCFL23] proposes schemes based on new falsifiable assumptions which rely on pairings and lattices (the pairing assumption holds in the bilinear generic group model, while the lattice one is an extension of the k-R-ISIS assumption of [ACL+22]). Comparing our work to [BCFL23], our basic scheme only relies on a bound on the input size[2]. Moreover, our scheme allows for arbitrary compositions of signatures, as was the case in [GVW15]. The signatures in [BCFL23] can be composed only sequentially, by feeding an entire signature as the input to another circuit (given a signature σ for $y = f(m)$, their scheme can compute a signature σ' for $z = g(y)$. Namely, the resulting signature σ' is with respect to z, circuit $g \circ f$ and input m).

As we mentioned already earlier, [CLTV15] builds an unleveled FHE scheme from subexponentially secure iO and re-randomizable encryption. Remarkably, their FHE does not require any circular security assumption, since it does not rely on the bootstrapping technique. Although we use a similar technical complexity leveraging argument to handle unbounded depth, the technical similarities end here.

Fully-Homomorphic Signatures from SNARKs. It was claimed in previous works [GW13,GVW15] that FHS can be built using succinct arguments of knowledge (SNARKs) for NP. This comes at a cost: in the FHS regime, that would mean using unfalsifiable assumptions (even in the random oracle model), as we explain in further details in the full version of this paper [GU23]. This stands in contrast with our scheme that can be instantiated from falsifiable assumptions, since general indistinguishability obfuscation itself can be built from falisifiable assumptions [JLS21,GP21,JLS22].

Full Context-Hiding. Our FHS scheme is also the first to achieve a strong notion of context hiding, more powerful than the one achieved by [GVW15]. Consider a signature σ for $m = f(m_1 \ldots m_N)$, which was obtained by homomorphically evaluating a function f for signature-message pairs $(\sigma_1, m_1) \ldots (\sigma_N, m_N)$. Full context-hiding[3] guarantees that the signature σ only certifies m and does not leak any information on messages $m_1 \ldots m_N$. A signature σ in [GVW15] is not context-hiding, but can be post-processed into another signature σ' that achieves context-hiding, at the cost that the homomorphism property is broken: no homomorphic operations can be applied on σ'.

In contrast, our FHS construction achieves full context hiding for signatures at all levels out-of-the-box, and context-hiding signatures can be homomorphically combined for an unbounded number of times. Our construction is the first to achieve this stronger notion of context-hiding in the standard model. More details can be found in the full version of this paper [GU23].

Roadmap. In Sect. 2 we define the building blocks used in our construction, then we describe our scheme in Sect. 3 and prove its security in Sect. 4.

Due to space limitations, some of our results are deferred to the full version of this paper [GU23] which contains:

---

[2] Our bound on the input size can be removed using random oracles, as in [GVW15].

[3] Previous work [GVW15] refers to this notion as context hiding. We use the modifier "full" to differentiate from its weak context hiding counterpart.

– a description of several schemes that satisfy unique randomness, a property needed from the FHE building block in the proof.
– a variation of the scheme that supports datasets of unbounded length, albeit by relying on the use of the random oracle model.
– an analysis of the context-hiding security of our scheme.
– a detailed description of how SNARKs can be used to build FHS. While such an approach would be much more practical in terms of the efficiency of the scheme, there would also be drawbacks with respect to the falsifiability of the assumptions used.
– a brief description of multi-data FHS, which allows for the signing of multiple datasets by associating each one with a label (the label is an arbitrary binary string, for example an encoding of a filename or a timestamp). Signing and verification is done with respect to the label, but the scheme uses the same signing and verification key for multiple labels. A generic transformation from single-data to multi-data FHS is known due to [GVW15] and is recalled in the full version of this paper [GU23].

2 Preliminaries

Notation. Throughout this paper, λ denotes the security parameter. For all $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. An algorithm is said to be *efficient* if it is a probabilistic polynomial time (PPT) algorithm. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is *negligible* if for any polynomial p there exists a bound $B > 0$ such that, for any integer $k \geq B$, $f(k) \leq 1/|p(k)|$. An event depending on λ occurs with *overwhelming probability* when its probability is at least $1 - \mathsf{negl}(\lambda)$ for a negligible function negl. Given a finite set S, the notation $x \leftarrow_{\mathrm{R}} S$ means a uniformly random assignment of an element of S to the variable x. For all probabilistic algorithms $\mathcal{A}$, all inputs x, we denote by $y \leftarrow \mathcal{A}(x)$ the process of running $\mathcal{A}$ on x and assigning the output to y. The notation $\mathcal{A}^{\mathcal{O}}$ indicates that the algorithm $\mathcal{A}$ is given an oracle access to $\mathcal{O}$. For all algorithm $\mathcal{A}, \mathcal{B}, \ldots$, all inputs $x, y, \ldots$ and all predicates P, we denote by $\Pr[a \leftarrow \mathcal{A}(x); b \leftarrow \mathcal{B}(a); \ldots : \mathsf{P}(a, b, \ldots)]$ the probability that the predicate P holds on the values $a, b, \ldots$ computed by first running $\mathcal{A}$ on x, then $\mathcal{B}$ on y and a, and so forth. For two distributions D_1, D_2, we denote by $\Delta(D_1, D_2)$ their statistical distance. We denote by $\mathcal{D}_1 \approx_c \mathcal{D}_2$ two computationally indistinguishable distribution ensembles $\mathcal{D}_1$ and $\mathcal{D}_2$. We denote by $\mathcal{D}_1 \approx_s \mathcal{D}_2$ two statistically close ensembles.

Subexponential Security. The security definitions we consider will require that for every efficient algorithm $\mathcal{A}$, there exists some negligible function negl such that for all $\lambda \in \mathbb{N}$, $\mathcal{A}$ succeeds in "breaking security" w.r.t. the security parameter λ with probability at most $\mathsf{negl}(\lambda)$. All the definitions that we consider can be extended to consider subexponential security; this is done by requiring the existence of a constant $\varepsilon > 0$, such that for every PPT algorithm $\mathcal{A}$, $\mathcal{A}$ succeeds in "breaking security" w.r.t. the security parameter λ with probability at most $2^{-\lambda^{\varepsilon}}$. The security notion of obfuscation (Sect. 2.3) and NIZK

(Sect. 2.4) are traditionally defined for *non-uniform* adversaries. We write our security definitions for uniform adversaries for simplicity, but they can be easily adapted to non-uniform adversaries.

2.1 Puncturable Pseudorandom Functions

A pseudorandom function (PRF) is a tuple of PPT algorithms $(\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ where $\mathsf{PRF.KeyGen}$ generates a key which is used by $\mathsf{PRF.Eval}$ to evaluate outputs. The core property of PRFs states that for a random choice of key, the outputs of $\mathsf{PRF.Eval}$ are pseudo-random. Puncturable PRFs (pPRFs) have the additional property that keys can be generated *punctured* at any input x in the domain. As a result, the punctured key can be used to evaluate the PRF at all inputs but x. Moreover, revealing the punctured key does not violate the pseudorandomness of the image of x. This notion can be generalized to allow they key to be punctured at multiple points.

As observed in [BW13, BGI14, KPTZ13], it is possible to construct such punctured PRFs for the original PRF construction of [GGM84], which can be based on any one-way functions [HILL99]. While this PRFs support puncturing for a polynomial number of times, in this paper we only to puncture at sets that contain at most two points.

Definition 1 (Puncturable Pseudorandom Function). *A* puncturable pseudorandom function (pPRF) *is a triple of PPT algorithms* $(\mathsf{PRF.KeyGen}, \mathsf{PRF.Puncture}, \mathsf{PRF.Eval})$ *such that:*

- $\mathsf{PRF.KeyGen}(1^\lambda)$*: on input the security parameter, it outputs a key K in the key space $\mathcal{K}_\lambda$. It also defines a domain $\mathcal{X}_\lambda$, a range $\mathcal{Y}_\lambda$ and a punctured key space $\mathcal{K}^*_\lambda$.*
- $\mathsf{PRF.Puncture}(K, S)$*: on input a key $K \in \mathcal{K}_\lambda$, a set $S \subseteq \mathcal{X}_\lambda$, it outputs a punctured key $K\{S\} \in \mathcal{K}^*_\lambda$,*
- $\mathsf{PRF.Eval}(K, x)$*: on input a key K (punctured or not, i.e. $K \in \mathcal{K}_\lambda \cup \mathcal{K}^*_\lambda$), and a point $x \in \mathcal{X}_\lambda$, it outputs a value in $\mathcal{Y}_\lambda$.*

We require the PPR algorithms to meet the following conditions:

Functionality Preserved under Puncturing. *For all $\lambda \in \mathbb{N}$, for all subsets $S \subseteq \mathcal{X}_\lambda$,*

$$\Pr[K \leftarrow \mathsf{PRF.KeyGen}(1^\lambda), K\{S\} \leftarrow \mathsf{PRF.Puncture}(K, S) : \\ \forall x' \in \mathcal{X}_\lambda \setminus S : \mathsf{PRF.Eval}(K, x') = \mathsf{PRF.Eval}(K\{S\}, x')] = 1.$$

Pseudorandom at Punctured Points. *For every stateful* PPT *adversary $\mathcal{A}$ and every security parameter $\lambda \in \mathbb{N}$, the advantage of $\mathcal{A}$ in* $\mathsf{Exp\text{-}pPRF}$ *(described in Fig. 3) is negligible, namely:*

$$\mathsf{Adv}_{\mathsf{cPRF}}(\lambda, \mathcal{A}) := \left| \Pr[\mathsf{Exp\text{-}pPRF}(1^\lambda, \mathcal{A}) = 1] - \tfrac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

For ease of notation we often write $\mathsf{PRF}(\cdot,\cdot)$ instead of $\mathsf{PRF.Eval}(\cdot,\cdot)$. When S is a singleton set $S = \{x\}$, we denote the punctured key at S as $K\{S\} = K\{x\}$, and when $S = \{x_1, x_2\}$, we denote $K\{S\} = K\{x_1, x_2\}$.

Theorem 2. [GGM84, BW13, BGI14, KPTZ13] *Consider a fixed polynomial $p(\lambda)$, and two arbitrary polynomials $n(\lambda), m(\lambda)$ in the security parameter λ. If one-way functions exist, then there exists a puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits and which supports punctured sets S of $p(\lambda)$ size.*

As explained at the beginning of this section, in this paper we use puncturing for sets that contain at most two elements.

Experiment $\mathsf{Exp\text{-}pPRF}(1^\lambda, \mathcal{A})$
$S \leftarrow \mathcal{A}(1^\lambda)$
$b \leftarrow_{\mathrm{R}} \{0,1\}$
$K \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$
$K\{S\} \leftarrow \mathsf{PRF.Puncture}(K, S)$
$Y = \emptyset$
for all $x \in S$
$\quad y_0 \leftarrow \mathsf{PRF.Eval}(K, x)$
$\quad y_1 \leftarrow_{\mathrm{R}} \mathcal{Y}_\lambda$
$\quad Y = Y \bigcup \{y_b\}$
$b' \leftarrow \mathcal{A}(K\{S\}, Y)$
Return $b = b'$

Fig. 3. Experiment $\mathsf{Exp\text{-}pPRF}(1^\lambda, \mathcal{A})$ for the pseudo-randomness at punctured points.

2.2 Fully Homomorphic Encryption

We recall the definition of unleveled FHE here, where there is no a-priori bound on the depth of circuits that can be homomorphically evaluated. For simplicity we consider messages to be bits.

Definition 3 (Fully Homomorphic Encryption). *A fully homomorphic encryption scheme* FHE *is a tuple of* PPT *algorithms* ($\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}$)*, where:*

- $\mathsf{FHE.KeyGen}(1^\lambda)$*: outputs a public encryption/evaluation key* pk *and a secret key* sk*.*
- $\mathsf{FHE.Enc}(\mathsf{pk}, m)$*: outputs an encryption* ct *of message $m \in \{0,1\}$. We denote by $\mathcal{R}$ the randomness space of* FHE.Enc*.*
- $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct})$*: uses* sk *to decrypt* ct*. It outputs a message.*
- $\mathsf{FHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_1 \ldots \mathsf{ct}_N)$*: it is a deterministic algorithm that takes as input a circuit f of arity N, and employs* pk *to compute an evaluated ciphertext* ct_f*.*

An FHE scheme must satisfy the following requirements:

Encryption Correctness. For all $\lambda \in \mathbb{N}$, all messages $m \in \{0,1\}$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{FHE.KeyGen}(1^\lambda)$, all ciphertexts ct in the support of $\mathsf{FHE.Enc}(\mathsf{pk}, m)$, we have $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}) = m$.

Evaluation Correctness. For all $\lambda \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{FHE.KeyGen}(1^\lambda)$, all messages $m_1, \ldots, m_N \in \{0,1\}$, all ciphertexts $(\mathsf{ct}_1 \ldots \mathsf{ct}_N)$ such that $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}_i) = m_i$ for all $i \in [N]$, all circuits f of arity N, it holds that:

$$\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{FHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_1 \ldots \mathsf{ct}_N)) = f(m_1, \ldots, m_N).$$

Randomness Homomorphism. There exists an efficient deterministic algorithm $\mathsf{FHE.EvalRand}$ such that for all $\lambda \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Setup}(1^\lambda)$, all messages $m_1, \ldots, m_N \in \{0,1\}$ and randomness $r_1, \ldots, r_N \in \mathcal{R}$, all circuits f of arity N, writing $r_f = \mathsf{FHE.EvalRand}(\mathsf{sk}, \mathsf{pk}, r_1, \ldots, r_N, m_1, \ldots, m_N, f)$ and $\mathsf{ct}_i = \mathsf{FHE.Enc}(\mathsf{pk}, m_i; r_i)$ for all $i \in [N]$, we have:

$$\mathsf{FHE.Enc}(\mathsf{pk}, f(m_1, \ldots, m_N); r_f) = \mathsf{FHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_N).$$

For most lattice-based FHE schemes, such as [GSW13], a stronger property holds: $\mathsf{EvalRand}$ can be publicly evaluated from the initial randomness and messages, and does not require sk (only pk). Nevertheless, the FHE scheme based on iO from [CLTV15] does require the use of the secret key to compute the evaluated randomness (which will consist of the key of a puncturable PRF). Both variants can be used as a building block in our construction.

Unique Randomness. For all $\lambda \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{FHE.KeyGen}(1^\lambda)$, all messages $m \in \{0,1\}$, all $r \in \mathcal{R}$ where $\mathcal{R}$ denotes the randomness space, there is no $r' \in \mathcal{R}$ such that $r' \neq r$ and $\mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m; r')$.

Selective $\mathsf{IND\text{-}CPA}$ Security. For any PPT adversary $\mathcal{A}$, we require that $\mathsf{Adv}^{\mathsf{FHE}}_{\mathsf{IND\text{-}CPA}}(\lambda, \mathcal{A})$ in the experiment $\mathsf{Exp\text{-}IND\text{-}CPA}$ from Fig. 4 is negligible, namely:

$$\mathsf{Adv}^{\mathsf{FHE}}_{\mathsf{IND\text{-}CPA}}(\lambda, \mathcal{A}) := \left| \Pr[\mathsf{Exp\text{-}IND\text{-}CPA}^{\mathsf{FHE}}(1^\lambda, \mathcal{A}) = 1] - \frac{1}{2}]\right| \leq \mathsf{negl}(\lambda)$$

Experiment $\mathsf{Exp\text{-}IND\text{-}CPA}^{\mathsf{FHE}}(1^\lambda, \mathcal{A})$
$(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$;
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$
$b \leftarrow_{\mathrm{R}} \{0,1\}$
$\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct})$
Return $b = b'$

Fig. 4. Experiment $\mathsf{Exp\text{-}IND\text{-}CPA}$ for the selective indistinguishable security of FHE.

2.3 Indistinguishability Obfuscation

We recall the definition of indinstuighuishability obfuscation, originally from [BGI+01].

Definition 4 (Indistinguishability Obfuscator). *An indistinguishability obfuscator for a circuit class* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *is an efficient algorithm* iO *such that:*

- ***Perfect correctness:*** *for all* $\lambda \in \mathbb{N}$*, all* $C \in \mathcal{C}_\lambda$*, all inputs* x*, we have:*

$$\Pr[C' \leftarrow \mathsf{iO}(1^\lambda, C) : C'(x) = C(x)] = 1$$

- ***Security:*** *for all efficient algorithms* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*, all pairs of circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *such that* $C_0(x) = C_1(x)$ *for all inputs* x*, we have:*

$$\mathsf{Adv}^{\mathsf{iO}}(\lambda, \mathcal{A}) := |\Pr[\mathcal{A}(\mathsf{iO}(1^\lambda, C_0)) = 1] - \Pr[\mathcal{A}(\mathsf{iO}(1^\lambda, C_1)) = 1]| \leq \mathsf{negl}(\lambda)$$

2.4 Non-interactive Zero Knowledge Proofs

Given a binary relation $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$ defined over a set of statements $\mathcal{X}$ and a set of witnesses $\mathcal{W}$, let $\mathcal{L}_R$ be the language defined as $\mathcal{L}_R = \{x \in \mathcal{X} \mid \exists w \in \mathcal{W} : R(x, w) = 1\}$. A Non-Interactive Zero Knowledge proof system for the binary relation R (originally introduced in [BFM88]) allows a prover in possession of a statement x and a witness w such that $R(x, w) = 1$ to produce a proof that convinces a verifier of the fact that $x \in L_R$ without revealing any information about w. The soundness property ensures that no proof can convince the verifier of the validity of a false statement, i.e. a statement $x \notin L_R$. We require the existence of an extractor that efficiently gets a witness from a valid proof π of a statement x, using an extraction trapdoor. Such proof systems are called *proofs of knowledge*. We focus on NIZK for relations R where the size of all statements and witnesses are bounded, which we call *size-bounded* relation. We now give the formal definition of NIZK proof of knowledge.

Definition 5 (NIZK-PoK). *Let* R *be a size-bounded relation. A* Non-Interactive Zero-Knowledge Proof of Knowledge (NIZK-PoK) *for* R *consists of the following PPT algorithms:*

- $\mathsf{Setup}(1^\lambda)$*: on input the security parameter, it outputs a common reference string* crs *and an extraction trapdoor* $\mathsf{td}_{\mathsf{ext}}$*.*
- $\mathsf{Prove}(\mathsf{crs}, x, w)$*: on input* crs*, a statement* x *and a witness* w*, it outputs an argument* π*.*
- $\mathsf{Verify}(\mathsf{crs}, x, \pi)$*: on input* crs*, a statement* x *and an argument* π*, it deterministically outputs a bit representing acceptance (1) or rejection (0).*

The PPT algorithms satisfy the following properties.

Composable Zero-Knowledge. There exist two PPT algorithms $\mathsf{SimSetup}$ and Sim such that for all PPT adversaries $\mathcal{A}$, the following advantages $\mathsf{Adv}_{\Pi}^{\mathsf{crs}}(\lambda, \mathcal{A})$ and $\mathsf{Adv}_{\Pi}^{\mathsf{ZK}}(\lambda, \mathcal{A})$ are negligible in λ:

$$\mathsf{Adv}_{\Pi}^{\mathsf{crs}}(\lambda, \mathcal{A}) = \Big| 1/2 - \Pr \Big[(\mathsf{crs}, \mathsf{td}_{\mathsf{ext}}) \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^\lambda), \\ b \leftarrow \{0,1\}, \mathsf{crs}_0 = \mathsf{crs}, \mathsf{crs}_1 = \mathsf{crs}_{\mathsf{sim}}, b' \leftarrow \mathcal{A}(\mathsf{crs}_b) : b' = b \Big] \Big|.$$

$$\mathsf{Adv}_{\Pi}^{\mathsf{ZK}}(\lambda, \mathcal{A}) = \Big| 1/2 - \Pr \Big[(x, w) \leftarrow \mathcal{A}(1^\lambda), (\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^\lambda), \\ \pi_0 \leftarrow \mathsf{Prove}(\mathsf{crs}_{\mathsf{sim}}, x, w), \pi_1 \leftarrow \mathsf{Sim}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}, x), \\ b \leftarrow \{0,1\}, b' \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}, \pi_b) : R(x, w) = 1 \ \wedge \ b' = b \Big] \Big|.$$

Completeness on Simulated CRS. For all efficient adversaries $\mathcal{A}$, the following advantage is negligible in the security parameter $\lambda \in \mathbb{N}$: $\Pr \Big[(x, w) \leftarrow \mathcal{A}(1^\lambda), (\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}) \leftarrow \mathsf{NIZK.SimSetup}(1^\lambda), \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{sim}}, x, w) : R(x, w) = 1 \ \wedge \ \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{sim}}, x, \pi) = 0 \Big]$.

Knowledge-Soundness. There exists an efficient algorithm $\mathsf{Extract}$ such that the following probability $\nu_{\mathsf{sound}}(\lambda)$ is a negligible function of $\lambda \in \mathbb{N}$, defined as:

$$\nu_{\mathsf{sound}}(\lambda) = \Pr \Big[(\mathsf{crs}, \mathsf{td}_{\mathsf{ext}}) \leftarrow \mathsf{Setup}(1^\lambda) : \exists\ \pi, x, w \in \mathsf{Supp}(\mathsf{Extract}(\mathsf{crs}, \mathsf{td}_{\mathsf{ext}}, x, \pi)) \\ s.t.\ \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \ \wedge \ R(x, w) = 0 \Big].$$

We say *subexponential* knowledge-soundness holds if ν_{sound} is subexponential in the security parameter λ.

2.5 Fully Homomorphic Signatures

We recall the definition of Fully-Homomorphic Signature (FHS), which was originally given in [BF11a]. When many datasets are present, the signing algorithm takes as an additional input a tag τ that identifies the dataset that is being signed. Only signatures issued for the same tag can be combined together. For simplicity, we focus on the single dataset setting here (where there are no tags), since [GVW15] showed how to generically transform any FHS for single dataset to many datasets. This transformation relies on regular (non-homomorphic) signature schemes. Again for simplicity, we focus on bit messages and Boolean functions.

Definition 6 (FHS, Single Dataset). *An FHS scheme is a tuple of* PPT *algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval})$*, such that:*

– $\mathsf{KeyGen}(1^\lambda, 1^N)$*: on input the security parameter* λ *and a data-size bound* N*, it generates a public verification key* vk*, along with a secret signing key* sk*.*

– $\mathsf{Sign}(\mathsf{sk}, m, i)$*: on input the secret key* sk*, a message* $m \in \{0,1\}$ *and an index* $i \in [N]$*, it outputs a signature* σ*.*
– $\mathsf{Eval}(\mathsf{vk}, f, (m_1, \sigma_1), \ldots, (m_N, \sigma_N))$*: on input the public key* vk*, a function* f *of arity* N *and pairs* (m_i, σ_i)*, it deterministically outputs an evaluated signature* σ *of the message* $f(m_1, \ldots, m_N)$*.*
– $\mathsf{Verify}(\mathsf{vk}, f, y, \sigma)$ *: on input the public key* vk*, a function* f*, a value* y *and a signature* σ*, it outputs a bit.* 0 *means the signature* σ *is deemed invalid,* 1 *means it is considered valid.*

The algorithms satisfy the following properties.

Perfect Signing Correctness. For all $\lambda, N \in \mathbb{N}$, all pairs $(\mathsf{vk}, \mathsf{sk})$ in the support of $\mathsf{KeyGen}(1^\lambda, 1^N)$, all $i \in [N]$, all messages $m \in \{0,1\}$, all signatures σ in the support of $\mathsf{Sign}(\mathsf{sk}, m, i)$, we have $\mathsf{Verify}(\mathsf{vk}, \mathsf{id}_i, m, \sigma) = 1$, where id_i is the projection function that takes N messages $m_1, \ldots, m_N \in \{0,1\}$, and outputs the i'th message m_i.

In our scheme, we achieve a weaker, computational variant of the correctness property, which roughly states that an efficient algorithm cannot find messages (with more than negligible probability) on which properly generated signatures do not verify successfully.

Computational Signing Correctness. For all efficient algorithms $\mathcal{A}$, the following probability, defined for all $\lambda, N \in \mathbb{N}$ is negligible in λ: $\Pr[(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N), (m_1, \ldots, m_N) \leftarrow \mathcal{A}(\mathsf{vk}), \forall i \in [N], \sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, m_i, i) : \exists\ i \in [N]\ s.t.\ \mathsf{Verify}(\mathsf{vk}, \mathsf{id}_i, m_i, \sigma_i) = 0]$.

Perfect Evaluation Correctness. For all $\lambda, N \in \mathbb{N}$, all pairs $(\mathsf{vk}, \mathsf{sk})$ in the support of $\mathsf{KeyGen}(1^\lambda, 1^N)$, all messages $m_1, \ldots, m_N \in \{0,1\}$, all signatures $\sigma_1, \ldots, \sigma_N$ in the support of $\mathsf{Sign}(\mathsf{sk}, m_1), \ldots, \mathsf{Sign}(\mathsf{sk}, m_N)$ respectively, for all functions f of arity N, writing $\sigma_f = \mathsf{Eval}(\mathsf{vk}, f, (\sigma_1, m_1), \ldots, (\sigma_N, m_N))$ and $y = f(m_1, \ldots, m_N)$, we have $\mathsf{Verify}(\mathsf{vk}, f, y, \sigma_f) = 1$. Moreover, it is possible to perform additional homomorphic operations on signatures that have already been evaluated on. That is, correctness holds when functions are composed. Namely, for all $\ell \in \mathbb{N}$, all functions g of arity ℓ, all tuples $(\sigma_1, f_1, m_1), \ldots, (\sigma_\ell, f_\ell, m_\ell)$ such that for all $i \in [\ell]$, $\mathsf{Verify}(\mathsf{vk}, f_i, m_i, \sigma_i) = 1$, writing $\mathsf{Eval}(\mathsf{vk}, g, (m_1, \sigma_1), \ldots, (m_\ell, \sigma_\ell)) = \sigma$ and $y = g(m_1, \ldots, m_\ell)$, we have $\mathsf{Verify}(\mathsf{vk}, g, y, \sigma) = 1$.

Similarly to signing correctness, we define a computational variant of the evaluation correctness. For simplicity, we split the property into two properties: the first is a computational evaluation correctness that only consider one-shot homomorphic evaluation, but does not take into account the possibility of performing homomorphic evaluations in several steps, i.e. composing functions. The second property, called weak context hiding, states that composing functions using Eval many times yields the same signature as using Eval once on the composed function. The (non-weak) context hiding property additionally requires that evaluated signatures be independent of the underlying dataset, apart from the output of the evaluated function.

Computational Evaluation Correctness. For all efficient algorithms $\mathcal{A}$, the following probability, defined for all $\lambda, N \in \mathbb{N}$, is negligible in λ: $\Pr[(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N), (m_1, \dots, m_N, f) \leftarrow \mathcal{A}(\mathsf{vk}), \forall i \in [N], \sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, m_i, i), \sigma_f \leftarrow \mathsf{Eval}(\mathsf{vk}, f, (m_1, \sigma_1), \dots, (m_N, \sigma_N)), y = f(m_1, \dots, m_N) : \mathsf{Verify}(\mathsf{vk}, f, y, \sigma_f) = 0]$.

Weak Context Hiding. For all $\lambda, N, t, \ell \in \mathbb{N}$, all $(\mathsf{vk}, \mathsf{sk})$ in the support of $\mathsf{Setup}(1^\lambda, 1^N)$, all messages $m_1, \dots, m_t \in \{0, 1\}$, functions $\theta_1, \dots, \theta_t$ and signatures $\sigma_1, \dots, \sigma_t$ such that for all $i \in [t]$, $\mathsf{Verify}(\mathsf{vk}, \theta_i, m_i, \sigma_i) = 1$, all t-ary functions $f_1, \dots, f_\ell$, all ℓ-ary functions g, we have:

$$\sigma_{g \circ \vec{f}} = \sigma_h,$$

where $\sigma_{g \circ \vec{f}} = \mathsf{Eval}(\mathsf{vk}, g, (\sigma_{f_1}, f_1(\vec{m})), \dots, (\sigma_{f_\ell}, f_\ell(\vec{m})))$, $\sigma_{f_j} = \mathsf{Eval}(\mathsf{vk}, f_j, (\sigma_1, m_1), \dots, (\sigma_t, m_t))$ for all $j \in [\ell]$, $\sigma_h = \mathsf{Eval}(\mathsf{vk}, h, (\sigma_1, m_1), \dots, (\sigma_t, m_t))$, h is the t-ary function defined on any input $m_1, \dots, m_t$ as $h(\vec{m}) = g(f_1(\vec{m}), \dots, f_\ell(\vec{m})))$, which we denote by $h = g \circ \vec{f}$. We are also using the notation $\vec{m} = (m_1, \dots, m_t)$.

Pre-processing. The scheme can be endowed with a pre-processing algorithm $\mathsf{Process}$. Just like the FHS scheme from [GVW15], our Verify algorithm works in two steps. The first step only depends on the inputs vk and f. Thus, it can be run offline, before knowing the signature σ and message y to verify. It produces a short processed vk, denoted by α_f (whose size is independent of the size of f). This first phase constitutes the $\mathsf{Process}$ algorithm. The second, online step takes as input α_f, y and σ and outputs a bit. The online step runs in time independent of the complexity of f.

Adaptive Unforgeability. For all stateful PPT adversaries $\mathcal{A}$ and all data bound $N \in \mathbb{N}$, the advantage $\mathsf{Adv}_\Sigma^{\mathsf{forg}}(\lambda, \mathcal{A})$ defined below is a negligible function of the security parameter $\lambda \in \mathbb{N}$:

$$\begin{aligned}\mathsf{Adv}_\Sigma^{\mathsf{forg}}(\lambda, \mathcal{A}) = \Pr\Big[&(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N), (m_1, \dots, m_N) \leftarrow \mathcal{A}(\mathsf{vk}),\\ &\forall i \in [N], \sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, m_i, i), (f, y, \sigma^\star) \leftarrow \mathcal{A}(\sigma_1, \dots, \sigma_N) :\\ &\mathsf{Verify}(\mathsf{vk}, f, y, \sigma^\star) = 1 \wedge y \neq f(m_1, \dots, m_n)\Big].\end{aligned}$$

Selective unforgeability is defined identically except the adversary $\mathcal{A}$ must send the messages $m_1, \dots, m_n$ of its choice *before* seeing the public key vk.

3 Construction

We describe our unleveled FHS scheme in Fig. 5. We choose to focus on single dataset FHS (as per Definition 6) rather that multi datasets for simplicity, since the work of [GVW15] presents a generic transformation from single to multi datasets, relying only on (non-homomorphic) signatures. Our FHS is for bit messages, and can evaluate arbitrary Boolean circuits. Without loss of generality, we focus on evaluating binary NAND gates.

We use a puncturable PRF, an indistinguishability obfuscator iO, an FHE scheme and a NIZK-PoK as building blocks, whose definition are given in the previous section. Our construction can be implemented using the dual-mode NIZK from [GS08] (from pairings) or [HU19] (from iO and lossy trapdoor functions), for instance. The FHE can be implemented using most lattice-based FHE (with bootstrapping since the FHE must be unleveled, which requires circular security), or with the construction from [CLTV15], which does not require any circularity assumption (it relies on iO and lossy trapdoor functions). Altogether, if we use the NIZK from [HU19] and the FHE from [CLTV15] we obtain our main result, which follows from Theorem 12 (unforgeability of our FHS).

Theorem 7 (Main Result). *Assume the existence of subexponentially secure iO and lossy trapdoor functions. Then subexponentially adaptively unforgeable unleveled FHS exist.*

3.1 Choice of Parameters

In our FHS, we rely on building blocks PRF, iO, NIZK, FHE that are subexponentially secure, that is, for which efficient adversaries can succeed with at most advantage $2^{-\kappa^{\varepsilon}}$ in breaking the security, for a constant $\varepsilon > 0$, where κ is the parameter chosen to run the setup of these primitives. We denote by κ_1 the parameter used for FHE and by κ_2 the parameter used for PRF, iO, and NIZK. Correctness is satisfied as long as the Eqs. (1) and (2) hold. Adaptive unforgeability is satisfied as long as the Eq. (3) holds. These equations are simultaneously satisfied when:

$$\kappa_1 = (N + \log N + 2\log^2 \lambda)^{1/\varepsilon}$$
$$\kappa_2 = \left(|\mathsf{ct}| + N + \log N + 2\log^2 \lambda + O(1)\right)^{1/\varepsilon}$$

where $|\mathsf{ct}|$ denotes the size of the FHE ciphertexts.

3.2 Correctness of the FHS

In this section we prove the computational signing correctness, the computational evaluation correctness, the weak context hiding and the pre-processing property of our scheme, all given in Definition 6.

$\underline{\mathsf{FHS.KeyGen}(1^\lambda, 1^N)}$
$(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.Setup}(1^{\kappa_1})$
$\{\mathsf{ct}'_i \leftarrow \mathsf{FHE.Enc}(0)\}_{i \in \{1 \ldots N\}}$
$K_1, K_2 \leftarrow \mathsf{PRF.KeyGen}(1^{\kappa_2})$
$\mathsf{Obf}_{\mathsf{GenCRS}} \leftarrow \mathsf{iO}(1^{\kappa_2}, \mathsf{PubGenCRS})$
$\mathsf{Obf}_{\mathsf{Eval}} \leftarrow \mathsf{iO}(1^{\kappa_2}, \mathsf{EvalNAND})$
$\mathsf{vk} = (\mathsf{fpk}, \{\mathsf{ct}'_i\}, \mathsf{Obf}_{\mathsf{GenCRS}}, \mathsf{Obf}_{\mathsf{Eval}})$
$\mathsf{sk} = (K_1, K_2, \mathsf{fsk})$
Return $(\mathsf{vk}, \mathsf{sk})$

$\underline{\mathsf{FHS.Sign}(\mathsf{sk}, m, i)}$
$(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{GenCRS}(0)$
$\pi \leftarrow \mathsf{NIZK.Sim}(\mathsf{crs_{sim}}, \mathsf{td_{sim}}, \mathsf{stat}_{m,\mathsf{ct}'_i})$
$\sigma = (\mathsf{ct}'_i, \pi, 0)$
Return σ

$\underline{\mathsf{FHS.Verify}(\mathsf{vk}, f, y, \sigma)}$
Parse σ as $(\mathsf{ct}, \pi, \mathsf{level})$
$\mathsf{ct}_f = \mathsf{FHE.Eval}(\mathsf{fpk}, f, \mathsf{ct}'_1, \ldots, \mathsf{ct}'_N)$
$\mathsf{crs} = \mathsf{Obf}_{\mathsf{GenCRS}}(\mathsf{level})$
Return $\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{stat}_{y,\mathsf{ct}_f}, \pi)$

$\underline{\mathsf{FHS.Eval}(\mathsf{vk}, f, (m_1, \sigma_1) \ldots (m_N, \sigma_N))}$
Evaluate each NAND gate of f using $\mathsf{Obf}_{\mathsf{Eval}}$ and return the result.

$\underline{\mathsf{GenCRS}(\mathsf{level})}$
Hardcoded: key K_1
$r = \mathsf{PRF}(K_1, \mathsf{level})$
$(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; r)$
Return $(\mathsf{crs_{sim}}, \mathsf{td_{sim}})$

$\underline{\mathsf{PubGenCRS}(\mathsf{level})}$
$(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{GenCRS}(\mathsf{level})$
Return $\mathsf{crs_{sim}}$

$\underline{\mathsf{EvalNAND}((\sigma_0, m_0), (\sigma_1, m_1))}$
Hardcoded: key K_2
Parse σ_b as $(\mathsf{ct}_b, \pi_b, \mathsf{level}_b)$, for $b \in \{0, 1\}$
Return $\perp$ if $\mathsf{level}_0 \neq \mathsf{level}_1$
$\mathsf{level} = \mathsf{level}_0$
$(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{GenCRS}(\mathsf{level})$
If $\mathsf{NIZK.Verify}(\mathsf{crs_{sim}}, \mathsf{stat}_{m_b,\mathsf{ct}_b}, \pi_b) = 0$
for some $b \in \{0, 1\}$ then return $\perp$
$\mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{fpk}, \mathsf{NAND}, \mathsf{ct}_0, \mathsf{ct}_1)$
$m = \mathsf{NAND}(m_0, m_1)$
$(\mathsf{crs}'_{\mathsf{sim}}, \mathsf{td}'_{\mathsf{sim}}) = \mathsf{GenCRS}(\mathsf{level} + 1)$
$\rho = \mathsf{PRF}(K_2, (m, \mathsf{ct}, \mathsf{level} + 1))$
$\pi = \mathsf{NIZK.Sim}(\mathsf{crs}'_{\mathsf{sim}}, \mathsf{td}'_{\mathsf{sim}}, \mathsf{stat}_{m,\mathsf{ct}}; \rho)$
$\sigma = (\mathsf{ct}, \pi, \mathsf{level} + 1)$
Return σ

Fig. 5. Fully-homomorphic signature scheme $\mathsf{FHS} = (\mathsf{FHS.KeyGen}, \mathsf{FHS.Sign}, \mathsf{FHS.Verify}, \mathsf{FHS.Eval})$. PRF is a puncturable pseudo-random function, NIZK is a proof of knowledge (NIZK PoK), FHE is a fully-homomorphic encryption scheme, and iO is an indistinguishability obfuscator. By $\mathsf{stat}_{m,\mathsf{ct}}$ we denote the statement which claims that $\exists\, r \in \mathcal{R}$ such that $\mathsf{ct} = \mathsf{FHE.Enc}(\mathsf{fpk}, m; r)$, where $\mathcal{R}$ denotes the randomness space of the FHE encryption algorithm. Parameters $\kappa(\lambda) = (N + 2\log^2\lambda + 5)^{1/\varepsilon}$, where $\varepsilon > 0$ is a constant whose existence is ensure by the subexponential security of the underlying building blocks.

Lemma 8 (Computational Signing Correctness). *The FHS scheme from Fig. 5 satisfies the computational signing correctness as per Definition 6, assuming* NIZK *satisfies the subexponential composable zero-knowledge and completeness on simulated crs properties (as per Definition 5),* FHE *satisfies the subexponential (selective) IND-CPA security (as per Definition 3),* PRF *satisfies the subexponential pseudorandomness at punctured points and the functionality preservation under puncturing (as per Definition 1) and* iO *satisfies the correcntess and subexponential security properties (as per Definition 4).*

Proof. We first explain how to prove the computational signing property in the selective case, where $\mathcal{A}$ sends the messages $m_1, \ldots, m_N \in \{0, 1\}$ before receiving vk. In this case, we can prove correctness using a hybrid argument, where we first switch the ciphertexts ct'_i from vk to $\mathsf{FHE.Enc}(\mathsf{fpk}, m_i; r_i)$, using the selective IND-CPA security of FHE. Then, we want to change the way $\mathsf{FHS.Sign}(\mathsf{sk}, m_i, i)$ computes the ZK proofs, using $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{stat}_{m_i,\mathsf{ct}'_i}, r_i)$, where r_i is a witness for $\mathsf{stat}_{m_i,\mathsf{ct}'_i}$, instead of producing $\pi \leftarrow \mathsf{NIZK.Sim}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}, \mathsf{stat}_{m_i,\mathsf{ct}'_i})$. This change would be justified by the composable zero knowledge property of NIZK. Finally, we would conclude the correctness proof using the completeness of NIZK on the simulated $\mathsf{crs}_{\mathsf{sim}}$. To perform these changes, we first need to puncture the PRF key K_1 on the point 0, and hardcode the pair $(\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; \mathsf{PRF}(K_1, 0))$ in the obfuscated circuits (which relies on the functionality preservation under puncturing of PRF and the security of iO), then switch the value $\mathsf{PRF}(K_1, 0)$ to truly random (which relies on the pseudorandomness at punctured points of PRF). Then, we can switch the way the proof π is computed by $\mathsf{FHS.Sign}(\mathsf{sk}, m_i, i)$ as we explained, using the composable zero-knowledge property of NIZK. Finally use the completeness on simulated crs property of NIZK. To obtain correctness in the adaptive case, where $\mathcal{A}$ can choose the messages $m_1, \ldots, m_N$ after seeing vk, we simply guess all the messages m_i in advance, which incurs a security loss of 2^N. Since we assume subexponential security of the underlying building blocks, we know that an adversary against the selective correctness can only succeed with a probability $N \cdot 2^{-\kappa_1^{\varepsilon}} + 4 \cdot 2^{-\kappa_2^{\varepsilon}}$ for $\varepsilon > 0$ where κ_1 is the parameter used for FHE, and κ_2 is the parameter used for NIZK, PRF and iO. Note that ε does not depend on N, so we can choose κ_1, κ_2 as polynomials in the security parameter λ and the arity N such that $2^N(N \cdot 2^{-\kappa_1^{\varepsilon}} + 4 \cdot 2^{-\kappa_2^{\varepsilon}})$ is a negligible function of λ, e.g.

$$\kappa_1, \kappa_2 \geq (N + \log N + \log^2 \lambda)^{1/\varepsilon}. \tag{1}$$

Lemma 9 (Computational Evaluation Correctness). *The FHS scheme from Fig. 5 satisfies the computational evaluation correctness as per Definition 6, assuming NIZK satisfies the subexponential zero-knowledge and and completeness on simulated crs properties (as per Definition 5),* FHE *satisfies the subexponential (selective) IND-CPA security and the randomness homomorphism properties (as per Definition 3),* PRF *satisfies the subexponential pseudorandomness at punctured points and the functionality preservation under puncturing (as per Definition 1) and* iO *satisfies the subexponential security and the perfect correctness properties (as per Definition 4).*

Proof. First, we prove the evaluation correctness in the selective case where the adversary $\mathcal{A}$ sends the messages $m_1, \ldots, m_N$ and the depth d of the circuit f before seeing the public key vk. Then, $\mathcal{A}$ receives vk and chooses the circuit f of depth d. To obtain computational evaluation correctness in the adaptive setting where $\mathcal{A}$ can choose f and the messages $m_1, \ldots, m_N$ after seeing vk (as per Definition 3), we will use a guessing argument together with the subexponential security of the underlying building blocks similarly than for proving the signing correctness. Namely, we choose a superpolynomial function $L(\lambda)$, e.g. $L(\lambda) = 2^{\log^2 \lambda}$ and we guess the messages $m_1, \ldots, m_N$ at random over $\{0,1\}^N$ and the depth d at random between 1 and $L(\lambda)$. Because we choose $L(\lambda)$ superpolynomial, we know that the depth d chosen by $\mathcal{A}$ is less than $L(\lambda)$, so the guess of the depth is correct with probability $1/L(\lambda)$. Overall the guessing incurs a security loss of $2^N L(\lambda)$.

Now we prove the selective variant of computational evaluation soundness. To begin with, we switch the ciphertexts ct'_i in vk to FHE encryptions of m_i of the form $\mathsf{FHE.Enc}(\mathsf{fpk}, m_i; r_i)$, using the selective IND-CPA security of FHE, just as in the computational signing correctness proof. Moreover, by perfect correctness of iO, we know that an evaluated signature $\sigma_f = \mathsf{Eval}(\mathsf{vk}, f, (\sigma_1, m_1), \ldots, (\sigma_N, m_N))$ is of the form $\sigma_f = (\mathsf{ct}, \pi, d)$ where $\mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{fpk}, f, \mathsf{ct}'_1, \ldots \mathsf{ct}'_N)$, and d is the depth of f. By evaluation correctness of FHE, we know that ct is an encryption of the message $f(m_1, \ldots, m_N)$. In fact, by the randomness homomorphism property of FHE, we know that $\mathsf{ct} = \mathsf{FHE.Enc}(\mathsf{fpk}, f(m_1, \ldots, m_N); r_f)$ where $r_f = \mathsf{FHE.EvalRand}(\mathsf{fsk}, r_1, \ldots, r_N, m_1, \ldots, m_N, f)$. Then, we want to switch the way the proof π in σ_f is computed: using $\mathsf{NIZK.Prove}$ and the witness r_f instead of using $\mathsf{NIZK.Sim}$ and the simulation trapdoor $\mathsf{td_{sim}}$. This switch would be justified by the composable zero-knowledge property of NIZK. We would then conclude the proof using the completeness of NIZK on simulated crs. Only to use these properties of NIZK, we first need to generate $(\mathsf{crs_{sim}}, \mathsf{td_{sim}})$ of level d using truly random coins, as opposed to pseudo-random. As typical, this requires puncturing the PRF key K_1 and hardcoding the pair $(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{NIZK.Setup}(1^{\kappa_2}; \mathsf{PRF}(K_1, d))$ in the obfuscated circuits (thanks to the security of iO and the functionality preservation under puncturing of PRF), then switching the value $\mathsf{PRF}(K_1, d)$ to truly random (thanks to the pseudo-randomness at punctured points property of PRF). Afterwards, we can use the properties of NIZK to conclude the proof, as we explained.

Since we assume subexponential security of the underlying building blocks, we know that an adversary against the selective computational evaluation correctness can only succeed with a probability $N \cdot 2^{-\kappa_1^{\varepsilon}} + 4 \cdot 2^{-\kappa_2^{\varepsilon}}$ for $\varepsilon > 0$ where κ_1 is the parameter used for FHE, and κ_2 is the parameter used for NIZK, PRF and iO. Note that ε does not depend on N, so we can choose κ_1, κ_2 as polynomials in the security parameter λ and the arity N such that $2^N L(\lambda)(N \cdot 2^{-\kappa_1^{\varepsilon}} + 4 \cdot 2^{-\kappa_2^{\varepsilon}})$ is a negligible function of λ, e.g.

$$\kappa_1, \kappa_2 \geq (N + \log N + 2\log^2 \lambda)^{1/\varepsilon}. \tag{2}$$

Lemma 10 (Weak Context Hiding). *The FHS scheme from Fig. 5 satisfies the weak context hiding property as per Definition 6, assuming the perfect correctness of* iO.

Proof. This property follows straightforwardly from the description of the Eval algorithm and the correctness of iO. Indeed, Eval evaluates circuits gate by gate, using the $\mathsf{EvalNAND}$ algorithm (see Fig. 5), which performs deterministic evaluation on the FHE ciphertext, and then derive a ZK proof deterministically from the statement and the depth level (using PRF on the key K_2). Thus, we have $\sigma_{g \circ \vec{f}} = \sigma_h$.

Lemma 11 (Pre-processing). *The FHS scheme from Fig. 5 satisfies the pre-processing property as per Definition 6.*

Proof. This simply follows from the description of $\mathsf{FHS.Verify}$. First, during a pre-processing phase, it computes the values ct_f and crs from vk and f. This can be performed offline, since it does not require to know the message y and the signature σ. The result is a short pre-processed key $\alpha_f = (\mathsf{ct}_f, \mathsf{crs})$. Then, during the online phase, $\mathsf{FHS.Verify}$ uses α_f, σ and y to run the $\mathsf{NIZK.Verify}$ algorithm. The running time of this online phase is independent from the size or depth of f.

4 Proof of Unforgeability

Theorem 12 (Adaptive Unforgeability). *Assuming subexponential security of* PRF, FHE, iO, *and* NIZK, *the FHS from Fig. 5 satisfies subexponential unforgeability as per Definition 6.*

Proof of Theorem 12. We first prove the selective unforgeability (as per Definition 6), where the adversary $\mathcal{A}$ must send the messages $m_1, \ldots, m_N$ before receiving vk. Then we show how to obtain adaptive unforgeability using a guessing argument and the subexponential security of the underlying building blocks (just as in the proof of computational signing and evaluation correctness in the previous section).

To prove unforgeability in the selective setting, we use a sequence of hybrid games, starting with G_0, defined exactly as the selective unforgeability game from Definition 6. For any game G_i, we denote by $\mathsf{Adv}_i(\mathcal{A})$ the advantage of $\mathcal{A}$ in G_i, that is, $\Pr[\mathsf{G}_i(1^\lambda, \mathcal{A}) = 1]$, where the probability is taken over the random coins of G_i and $\mathcal{A}$. Before we proceed to describe the other hybrids, we make several technical remarks.

Remark 13. When we hardcode a value in a subprogram, it is understood that this value is also hardcoded in all the programs that run it, and if a PRF key K is punctured in a subprogram, it is also punctured in all the programs that run it.

Remark 14 (Padding the programs). The security of iO can only be invoked for programs of the same size. For brevity, we assume without loss of generality that all programs in the security proof are padded to the size of the longest program. Since our hybrids extend up to a superpolynomial level $L(\lambda) = 2^{\omega(\log \lambda)}$, this implies a small increase in the programs contained in the real verification key (since the last hybrid must keep track of the level, and its bit representation requires $\omega(\log \lambda)$ bits). For example, choosing $L(\lambda) = 2^{\log^2 \lambda}$ would only incur a multiplicative increase by a factor of $\log^2 \lambda$ bits.

Remark 15 (Bounding the Sizes of Punctured PRF Keys). The security proof will require that PRF keys K_1 and K_2 are punctured at levels $i = 0 \ldots L(\lambda)$, where $L(\lambda) = 2^{\log^2 \lambda}$. Puncturing increases the size of the keys. In existing constructions of PRFs (e.g. [GGM84]), the size of the punctured keys only grows logarithmically with the number of levels This results in a size-increase of the keys (and therefore of the programs) of up to $O(\log^2 \lambda)$. In particular, it is important to note that this size increase is independent of the value of the specific level at which the adversary will output a forgery.

– **Game** G_1: same as G_0, except that we change the $\mathsf{FHS.KeyGen}$ algorithm. Instead of computing the ct'_i in the verification key as encryptions of 0, we compute $\mathsf{ct}'_i \leftarrow \mathsf{FHE.Enc}(m_i; r_i)$, where m_i are the messages sent by $\mathcal{A}$. The randomness r_i used to compute the ciphertext ct'_i is stored in the secret key sk.

Lemma 16 (From G_0 to G_1). *For every* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$, such that:* $|\mathsf{Adv}_0(\mathcal{A}) - \mathsf{Adv}_1(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{FHE}}_{\mathsf{IND\text{-}CPA}}(\kappa_1, \mathcal{B})$.

Proof. The reduction $\mathcal{B}$ starts by sending $(0 \dots 0)$ and $(m_1 \dots m_N)$ to the IND-CPA challenger. It receives $(\mathsf{ct}'_1 \dots \mathsf{ct}'_N)$, which it embeds in the vk. During the execution of $\mathsf{FHS.KeyGen}$, all the other obfuscated programs in vk are generated as before, but using the ciphertexts received from the challenger.

– **Game** G_2: same as G_1, except that we change the $\mathsf{FHS.Sign}$ algorithm and replace it with $\mathsf{HybridSign}$, defined in Fig. 6. The latter computes the signatures $\sigma_1, \dots, \sigma_N$ sent to $\mathcal{A}$ (after $\mathcal{A}$ sends the messages $m_1, \dots, m_N$) as $\sigma_i = (\mathsf{ct}'_i, \pi_i, 0)$ where $\mathsf{ct}'_i = \mathsf{FHE.Enc}(\mathsf{fpk}, m_i; r_i)$ is the i'th FHE encryption contained in vk, 0 indicates the level, and π_i is computed using the witness r_i (which is stored in sk), instead of using a simulation trapdoor.

Lemma 17 (From G_1 to G_2). *For every PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ such that:*

$$|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_2(\mathcal{A})| \leq 2\big(\mathsf{Adv}_{\mathsf{cPRF}}(\kappa_2, \mathcal{B}_1) + \mathsf{Adv}_{\mathsf{iO}}(\kappa_2, \mathcal{B}_2)\big) + N \cdot \mathsf{Adv}_{\mathsf{ZK}}(\kappa_2, \mathcal{B}_3).$$

Proof. To switch from proofs π_i generated using $\mathsf{NIZK.Sim}$ and the simulation trapdoor $\mathsf{td_{sim}}$ to proofs generated using $\mathsf{NIZK.Prove}$ and the witnesses r_i, as described in Fig. 6, we want to use the composable zero-knowledge property of NIZK. To do so, we first have to hard-code the pair $(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; \mathsf{PRF}(K_1, 0))$ in the obfuscated circuit instead of using the key K_1 on the point 0. To generate the pairs $(\mathsf{crs_{sim}}, \mathsf{td_{sim}})$ for all other levels $i \neq 0$, we compute $(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; \mathsf{PRF}(K_1\{0\}, i))$, where $K_1\{0\}$ is a key punctured at the point 0. Because puncturing preserves the functionality of PRF (as per Definition 1), this does not change the input/output behavior of the obfuscated circuit. Thus we can use the iO security to argue that this change is computational undetectable by the adversary. Then, we switch the hardcoded pair $(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; \mathsf{PRF}(K_1, 0))$ to $(\mathsf{crs_{sim}}, \mathsf{td_{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; r_0)$, where r_0 is truly random. This is possible by the pseudorandomness property at punctured points of PRF. Then, we use the composable zero-knowledge property of NIZK to switch π_i to $\pi_i \leftarrow$

$\mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{stat}_{\mathsf{ct}'_i, m_i}, r_i)$ for all $i \in [N]$. Finally we switch back the generation of the pairs $(\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}})$ using pseudo-random coins for all levels (instead of using truly random coins for the level 0) and we unpuncture the key K_1.

$\underline{\mathsf{HybridSign}(\mathsf{sk}, m_i, i)}$

$\mathsf{crs}_{\mathsf{sim}} = \mathsf{PubGenCRS}(0)$

$\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{stat}_{m_i, \mathsf{ct}'_i}, r_i)$

$\sigma_i = (\mathsf{ct}'_i, \pi_i, 0)$

Return σ_i

Fig. 6. In G_2, we replace the $\mathsf{FHS.Sign}$ algorithm with $\mathsf{HybridSign}$. Changes are highlighted in gray.

- **Game** $\mathsf{G}_{3,\ell}$: At this point, the proof proceeds in a series of $L(\lambda) = 2^{\log^2 \lambda}$ hybrids where $\mathsf{G}_{3,\ell}$ is defined for all $\ell = \{0, \ldots, L(\lambda)\}$ identically to G_2, except that:
  1. the program GenCRS is replaced by $\mathsf{HybridGenCRS}^\ell$, described in Fig. 7. The latter generates a crs with an extraction trapdoor using $\mathsf{NIZK.Setup}$ on any level $< \ell$, and generates a simulated crs with a simulation trapdoor using $\mathsf{NIZK.SimSetup}$ on any level $\geq \ell$.
  2. the program $\mathsf{EvalNAND}$ is replaced by $\mathsf{HybridEvalNAND}^\ell$, described in Fig. 7. For any level $< \ell$, the latter generates proofs for the next level using witnesses obtained using an extraction trapdoor and the randomness homomorphic property of FHE. For any level $\geq \ell$, it generates proofs for the next level using a simulation trapdoor.

  Note that $\mathsf{G}_{3,0} = \mathsf{G}_2$. In Theorem 18, we prove that for all $\ell \in \{0, \ldots, L(\lambda) - 1\}$, $\mathsf{G}_{3,\ell} \approx_c \mathsf{G}_{3,\ell+1}$.
- **Game** G_4: same as $\mathsf{G}_{3,L(\lambda)}$, except the game guesses the depth of the function f chosen by the adversary $\mathcal{A}$ for his forgery, by sampling $d^\star \leftarrow_{\mathrm{R}} \{1, \ldots, L(\lambda)\}$. At the end of the game, $\mathcal{A}$ sends the forgery $(f, y, \sigma^\star)$. If $d^\star \neq d$, then the game G_4 outputs 0. Otherwise it proceeds as in $\mathsf{G}_{3,L(\lambda)}$. Since $L(\lambda)$ has been chosen super polynomial in λ, we know that the function f has depth $d \leq L(\lambda)$. Thus, with probability $1/L(\lambda)$, the guess is correct, i.e. we have $d^\star = d$. Therefore,

$$\mathsf{Adv}_4(\mathcal{A}) = \frac{\mathsf{Adv}_{3,L(\lambda)}(\mathcal{A})}{L(\lambda)}.$$

- **Game** G_5: same as G_4, except we puncture the key K_1 at $d^\star$ and hardcode the value $\mathsf{PRF}(K_1, d^\star)$ in the obfuscated circuit. Since puncturing preserve the functionality, we can use the security of iO to argue that there exists a PPT adversary $\mathcal{B}_5$ such that:

$$|\mathsf{Adv}_5(\mathcal{A}) - \mathsf{Adv}_4(\mathcal{A})| = \mathsf{Adv}_{\mathsf{iO}}(\kappa_2, \mathcal{B}_5).$$

– **Game** G_6: same as G_5, except we change the value $\mathsf{PRF}(K_1, d^\star)$ hardcoded in the obfuscated circuit is turned to a truly random value. By the pseudorandomness of PRF on punctured points, we know there exists a PPT $\mathcal{B}_6$ such that:

$$|\mathsf{Adv}_6(\mathcal{A}) - \mathsf{Adv}_5(\mathcal{A})| = \mathsf{Adv}_{\mathsf{cPRF}}(\kappa_2, \mathcal{B}_6).$$

We now proceed to bound $\mathsf{Adv}_6(\mathcal{A})$. By the knowledge soundness property of NIZK, we know that $\mathsf{Adv}_6(\mathcal{A}) \leq \nu_{\mathsf{sound}}(\kappa_2)$. Putting things together, we have $\mathsf{Adv}_4(\mathcal{A}) \leq \nu_{\mathsf{sound}}(\kappa) + \mathsf{Adv}_{\mathsf{cPRF}}(\kappa_2, \mathcal{B}_6) + \mathsf{Adv}_{\mathsf{iO}}(\kappa_2, \mathcal{B}_5)$ and $\mathsf{Adv}_3(\mathcal{A}) = L(\lambda)\mathsf{Adv}_4(\mathcal{A})$. Together with the result of Theorem 18, we have:

$$\begin{aligned}\mathsf{Adv}_0(\mathcal{A}) \leq &(2^{|\mathsf{ct}|+2} + L(\lambda) + 8)\mathsf{Adv}_{\mathsf{iO}}(\kappa_2, \mathcal{B}_1) + (2^{|\mathsf{ct}|+2} + L(\lambda) + 6)\mathsf{Adv}_{\mathsf{cPRF}}(\kappa_2, \mathcal{B}_2)\\ &+ \mathsf{Adv}_{\mathsf{crs}}(\kappa_2, \mathcal{B}_3) + (2^{|\mathsf{ct}|+1} + N)\mathsf{Adv}_{\mathsf{ZK}}(\kappa_2, \mathcal{B}_4)\\ &+ (L(\lambda) + 2)\nu_{\mathsf{sound}}(\kappa_2) + \mathsf{Adv}^{\mathsf{FHE}}_{\mathsf{IND\text{-}CPA}}(\kappa_1, \mathcal{B}_5).\end{aligned}$$

The subexponential security of the building blocks implies that there exists a constant $\varepsilon > 0$ such that $\mathsf{Adv}_{\mathsf{iO}}(\kappa_2, \mathcal{B}_1), \mathsf{Adv}_{\mathsf{cPRF}}(\kappa_2, \mathcal{B}_2), \mathsf{Adv}_{\mathsf{crs}}(\kappa_2, \mathcal{B}_3)$, $\mathsf{Adv}_{\mathsf{ZK}}(\kappa_2, \mathcal{B}_4), \nu_{\mathsf{sound}}(\kappa_2) \leq 2^{-\kappa_2^\varepsilon}$ and $\mathsf{Adv}^{\mathsf{FHE}}_{\mathsf{IND\text{-}CPA}}(\kappa_1, \mathcal{B}_5) \leq 2^{-\kappa_1^\varepsilon}$. Thus, we have

$$\mathsf{Adv}_0(\mathcal{A}) \leq 2^{-\kappa_2^\varepsilon}(5 \cdot 2^{|\mathsf{ct}|+1} + 3L(\lambda) + N + 17) + 2^{-\kappa_1^\varepsilon}.$$

Since we chose $L(\lambda) = \log^2 \lambda$, selective security can be achieved by choosing for instance

$$\begin{aligned}\kappa_2 &\geq (|\mathsf{ct}| + \log N + 2\log^2 \lambda + O(1))^{1/\varepsilon},\\ \kappa_1 &\geq (\log^2 \lambda)^{1/\varepsilon}.\end{aligned}$$

To achieve adaptive unforgeability, we use the same guessing technique as for the proof of computation correctness (both signing and evaluation) in the previous section. Namely, we simply guess the messages $m_1^\star, \ldots, m_N^\star \leftarrow_{\mathrm{R}} \{0,1\}$ in advance, then proceed as in the selective game (but with the guesses $m_i^\star$ instead of the real messages chosen by the adversary). If the guess is correct, we have the same advantage as in the selective security game. If the guess is incorrect, the game outputs 0. This guessing argument incurs a security loss of 2^N. That is, the advantage of an adaptive adversary $\mathcal{A}$ against the unforgeability of our FHS is less than 2^N times the security loss in the selective setting written above. Therefore, adaptive unforgeability can be achieved by choosing for instance

$$\kappa_2 \geq (|\mathsf{ct}| + N + \log N + 2\log^2 \lambda + O(1))^{1/\varepsilon},\ \kappa_1 \geq (N + \log^2 \lambda)^{1/\varepsilon} \qquad (3)$$

This concludes the unforgeability proof. □

Theorem 18 (From $\mathsf{G}_{3,\ell}$ to $\mathsf{G}_{3,\ell+1}$). *For every* PPT *adversary $\mathcal{A}$, there exist* PPT *adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, such that:*

$$|\mathsf{Adv}_{3,\ell}(\mathcal{A}) - \mathsf{Adv}_{3,\ell+1}(\mathcal{A})| \leq (2^{|\mathsf{ct}|+2} + 6)\mathsf{Adv}_{\mathsf{iO}}(\kappa_2, \mathcal{B}_1) + (2^{|\mathsf{ct}|+2} + 4)\mathsf{Adv}_{\mathsf{cPRF}}(\kappa_2, \mathcal{B}_2) + 2^{|\mathsf{ct}|+1}\mathsf{Adv}_{ZK}(\kappa_2, \mathcal{B}_3) + \mathsf{Adv}_{\mathsf{crs}}(\kappa_2, \mathcal{B}_4) + 2\nu_{\mathsf{sound}}(\kappa_2).$$

$\underline{\mathsf{HybridGenCRS}^{\ell}(\mathsf{level})}$
Hardcoded: key K_1
$s = \mathsf{PRF}(K_1, \mathsf{level})$
Return $(\mathsf{crs}, \mathsf{td}_{\mathsf{ext}}) = \mathsf{NIZK.Setup}(1^{\kappa_2}; s)$ for $\mathsf{level} < \ell$
Return $(\mathsf{crs}_{\mathsf{sim}}, \mathsf{td}_{\mathsf{sim}}) = \mathsf{NIZK.SimSetup}(1^{\kappa_2}; s)$ for $\mathsf{level} \geq \ell$

$\underline{\mathsf{HybridEvalNAND}^{\ell}((\sigma_0, m_0), (\sigma_1, m_1))}$
Hardcoded: key K_2
Parse σ_b as $(\mathsf{ct}_b, \pi_b, \mathsf{level}_b)$, for $b \in \{0, 1\}$
Return $\perp$ if $\mathsf{level}_0 \neq \mathsf{level}_1$
$j = \mathsf{level}_0$
$(\mathsf{crs}_j, \mathsf{td}_j) = \mathsf{HybridGenCRS}^{\ell}(j)$
If $\mathsf{NIZK.Verify}(\mathsf{crs}_j, \mathsf{stat}_{m_b, \mathsf{ct}_b}, \pi_b) = 0$ for some $b \in \{0, 1\}$ then output $\perp$
$\mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{fpk}, \mathsf{NAND}, \mathsf{ct}_0, \mathsf{ct}_1)$
$m = \mathsf{NAND}(m_0, m_1)$
$(\mathsf{crs}_{j+1}, \mathsf{td}_{j+1}) = \mathsf{HybridGenCRS}^{\ell}(j + 1)$
$\rho = \mathsf{PRF}(K_2, (m, \mathsf{ct}, j + 1))$
If $j < \ell$
  $r_b = \mathsf{NIZK.Extract}(\mathsf{crs}_j, \mathsf{td}_j, \mathsf{stat}_{m_b, \mathsf{ct}_b}, \pi_b)$ for $b \in \{0, 1\}$
  $r = \mathsf{FHE.EvalRand}(\mathsf{fsk}, \mathsf{NAND}, r_1, r_2)$
  $\pi = \mathsf{NIZK.Prove}(\mathsf{crs}_{j+1}, \mathsf{stat}_{m, \mathsf{ct}}, r; \rho)$

If $j \geq \ell$
  $\pi = \mathsf{NIZK.Sim}(\mathsf{crs}_{j+1}, \mathsf{td}_{j+1}, \mathsf{stat}_{m, \mathsf{ct}}; \rho)$

$\sigma = (\mathsf{ct}, \pi, j + 1)$
Return σ

Fig. 7. Algorithms $\mathsf{HybridGenCRS}^{\ell}$ and $\mathsf{HybridEvalNAND}^{\ell}$, used in the games $\mathsf{G}_{3,\ell}$, for all $\ell \in \{0, \ldots, L(\lambda)\}$.

Due to space constraints, we provide the technical proof of this theorem in the full version of the paper [GU23].

Acknowledgements. We would like to thank Geoffroy Couteau and Dennis Hofheinz for their input during discussions that led to this work.

References

AB09. Agrawal, S., Boneh, D.: Homomorphic MACs: MAC-based integrity for network coding. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 292–305. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01957-9_18

ACL+22. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.: Lattice-based SNARKs: publicly verifiable, preprocessing, and recursively composable. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology, CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 102–132. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15979-4_4

ALP13. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_24

AP19. Aranha, D.F., Pagnin, E.: The simplest multi-key linearly homomorphic signature scheme. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 280–300. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30530-7_14

AP20. Agrawal, S., Pellet-Mary, A.: Indistinguishability obfuscation without maps: attacks and fixes for noisy linear FE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 110–140. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_5

BCFL23. Balbás, D., Catalano, D., Fiore, D., Lai, R.W.F.: Chainable functional commitments for unbounded-depth circuits. In: Rothblum, G., Wee, H. (eds.) Theory of Cryptography, TCC 2023. LNCS, vol. 14371, pp. 363–393. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-48621-0_13

BDGM20a. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Candidate iO from homomorphic encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 79–109. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_4

BDGM20b. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Factoring and pairings are not necessary for iO: circular-secure LWE suffices. Cryptology ePrint Archive (2020)

BF11a. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_10

BF11b. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_1

BFM88. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC, May 1988, pp. 103–112. ACM Press (1988)

BFS14. Boyen, X., Fan, X., Shi, E.: Adaptively secure fully homomorphic signatures based on lattices. Cryptology ePrint Archive, Paper 2014/916 (2014). https://eprint.iacr.org/2014/916

BGI+01. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1

BGI14. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_29

BW13. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_15

CF13. Catalano, D., Fiore, D.: Practical homomorphic MACs for arithmetic circuits. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 336–352. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_21

CFN15. Catalano, D., Fiore, D., Nizzardo, L.: Programmable hash functions go private: constructions and applications to (homomorphic) signatures with shorter public keys. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 254–274. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_13

CFN18. Catalano, D., Fiore, D., Nizzardo, L.: On the security notions for homomorphic signatures. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 183–201. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_10

CFT22. Catalano, D., Fiore, D., Tucker, I.: Additive-homomorphic functional commitments and applications to homomorphic signatures. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology, ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 159–188. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22972-5_6

CFW14. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 371–389. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_21

CLQ16. Chen, W., Lei, H., Qi, K.: Lattice-based linearly homomorphic signatures in the standard model. Theoret. Comput. Sci. **634**, 47–54 (2016)

CLTV15. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_19

DJ01. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_9

DQV+21. Devadas, L., Quach, W., Vaikuntanathan, V., Wee, H., Wichs, D.: Succinct LWE sampling, random polynomials, and obfuscation. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 256–287. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90453-1_9

ElG85. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theor. **31**(4), 469–472 (1985)

FG18. Fuchsbauer, G., Gay, R.: Weakly secure equivalence-class signatures from standard assumptions. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 153–183. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_6

FHS19. Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. J. Cryptol. **32**(2), 498–546 (2019)

FP18. Fiore, D., Pagnin, E.: Matrioska: a compiler for multi-key homomorphic signatures. In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 43–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_3

Fre12. Freeman, D.M.: Improved security for linearly homomorphic signatures: a generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_41

Gen09. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: 41st ACM STOC, May/June 2009, pp. 169–178. ACM Press (2009)

GGM84. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS, October 1984, pp. 464–479. IEEE Computer Society Press (1984)

GM82. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: 14th ACM STOC, May 1982, pp. 365–377. ACM Press (1982)

GP21. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: 53rd ACM STOC, June 2021, pp. 736–749. ACM Press (2021)

GS08. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24

GSW13. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5

GU23. Gay, R., Ursu, B.: On instantiating unleveled fully-homomorphic signatures from falsifiable assumptions. Cryptology ePrint Archive, Paper 2023/1818 (2023). https://eprint.iacr.org/2023/1818

GVW15. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: 47th ACM STOC, June 2015, pp. 469–477. ACM Press (2015)

GW13. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 301–320. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_16

HILL99. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999)

HPP20. Hébant, C., Phan, D.H., Pointcheval, D.: Linearly-homomorphic signatures and scalable mix-nets. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 597–627. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_21

HS14. Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 491–511. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_26

HU19. Hofheinz, D., Ursu, B.: Dual-mode NIZKs from obfuscation. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 311–341. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_12

JLS21. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: 53rd ACM STOC, June 2021, pp. 60–73. ACM Press (2021)

JLS22. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology, EUROCRYPT 2022. LNCS, vol. 13275, pp. 670–699. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06944-4_23

JMSW02. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45760-7_17

KPTZ13. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM CCS 2013, November 2013, pp. 669–684. ACM Press (2013)

KSD19. Khalili, M., Slamanig, D., Dakhilalian, M.: Structure-preserving signatures on equivalence classes from standard assumptions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 63–93. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_3

LPJY15. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. Des. Codes Crypt. **77**(2), 441–477 (2015)

LTWC18. Lai, R.W.F., Tai, R.K.H., Wong, H.W.H., Chow, S.S.M.: Multi-key homomorphic signatures unforgeable under insider corruption. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 465–492. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_16

Pai99. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16

SBB19. Schabhüser, L., Butin, D., Buchmann, J.: Context hiding multi-key linearly homomorphic authenticators. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 493–513. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_25

SW14. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: 46th ACM STOC, May/June 2014, pp. 475–484. ACM Press (2014)

Tsa17. Tsabary, R.: An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 489–518. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_16

WW21. Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 127–156. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_5