Leveled Fully-Homomorphic Signatures from Batch Arguments

Abtin Afshar
UW-Madison*

Jiaqi Cheng
UW-Madison†

Rishab Goyal
UW-Madison‡

Abstract

Fully homomorphic signatures are a significant strengthening of digital signatures, enabling computations on *secretly* signed data. Today, we have multiple approaches to design fully homomorphic signatures such as from lattices, or succinct functional commitments, or indistinguishability obfuscation, or mutable batch arguments. Unfortunately, all existing constructions for homomorphic signatures suffer from one or more limitations. We do not have homomorphic signatures with features such as multi-hop evaluation, context hiding, and fast amortized verification, while relying on standard falsifiable assumptions.

In this work, we design homomorphic signatures satisfying all above properties. We construct homomorphic signatures for polynomial-sized circuits from a variety of standard assumptions such as sub-exponential DDH, standard pairing-based assumptions, or learning with errors. We also discuss how our constructions can be easily extended to the multi-key setting.

1 Introduction

Fully homomorphic signatures [JMSW02, AB09, BFKW09, BF11a, GVW15] represent a significant advancement in cryptography, akin to fully homomorphic encryption [RAD+78, Gen09, BV14, BGV14, GSW13], enabling computations on *secretly* signed data. Standard signatures create unforgeable tokens in the form of a signature σ_m for data m. Fully homomorphic signatures provide 'homomorphism' over signatures. For any efficiently computable circuit C, they enable derivation of a new signature $\sigma_{C,y}$ starting from a signature σ_m for message m. Here $\sigma_{C,y}$ is viewed as a signature on the message (C, y) certifying that this is a homomorphically evaluated signature for output y w.r.t. a circuit C. In other words, $\sigma_{C,y}$ is an unforgeable token validating possession of a signature σ_m on some data m such that $C(m) = y$.

The two main desirable properties for homomorphic signatures are *succinctness* and *unforgeability*. Succinctness states that the size of a derived signature $\sigma_{C,y}$ does not grow with the original data size, or the size of the computation. Unforgeability has to be defined a bit more carefully as homomorphic signatures are *malleable* by design, and malleability typically conflicts with unforgeability. The most well-adopted approach to capture unforgeability is to consider attackers that, given a *single* signature on dataset m, want to forge a signature σ^* for some circuit C^* and output $y^* \neq C^*(m)$. This is colloquially referred to as the single-dataset model [GVW15]. It is well-known that this can be generically extended to multi-dataset models, where an attacker can request signatures on multiple datasets.

*Email: abtin@cs.wisc.edu.
†Email: jcheng77@wisc.edu.
‡Email: rishab@cs.wisc.edu. Support for this research was provided by OVCRGE at UW–Madison with funding from the Wisconsin Alumni Research Foundation.

More properties. Beyond succinctness and unforgeability, a variety of additional desirable features for homomorphic signatures have been studied in the literature. Some of the most popular and commonly studied features are composability, context hiding, and efficient online/amortized verification. (Refer to [GVW15] for a detailed discussion.)

□*Composability.* It states that evaluated signature(s) can also be homomorphically evaluated. In words, it enables composition of several different computations over signed data. As an example, consider a server that computes ℓ circuits $C_1, \ldots, C_\ell$ over signed data m to compute signatures $\sigma_{C_1,y_1}, \ldots, \sigma_{C_\ell,y_\ell}$, where $y_i = C_i(m)$. A composable homomorphic signature lets any other user/server to take these evaluated signatures and perform additional computation C^* on the signed data $y_1, \ldots, y_\ell$ to derive another evaluated signature $\sigma_{C^* \circ C_1|\cdots|C_\ell,y^*}$[1]. This is commonly referred to as the *multi-hop* evaluation property. A rather interesting feature of composability is that the second evaluator does not need to know the original data m, or the original signature σ_m.

□*Context hiding.* A verifier in the homomorphic signature setting receives an evaluated signature $\sigma_{C,y}$ and a circuit-output pair (C, y), in addition to a verification key. Although an honest verifier does not receive the original data m *explicitly*, this does not guarantee that a malicious verifier cannot learn non-trivial information about the original data. By non-trivial we mean anything that it cannot learn from just the circuit C and output y. Context hiding captures data privacy in presence of malicious verifiers. Informally, it states that a malicious user should not learn anything from $\sigma_{C,y}$, beyond what can be simulated given C and y. *In conjunction with the composability property, one could define stronger data privacy notions, where the goal is to also hide the number of times evaluation was performed and/or whether evaluation really took place.*

□*Fast verification.* Typically, in homomorphic signatures, succinctness only states that the size of an evaluated signature does not grow with the circuit size. Since a verifier receives a circuit, that was evaluated, as an explicit input, thus the verifier's running time grows with the circuit size. While this seems unavoidable as the statement that the verifier is checking depends on both C and y[2], one could consider applications where either C is a-priori known to the verifier, or multiple signatures for the same circuit C have to be verified. The fast online/amortized verification property states that a verifier can pre-process the circuit C to compute a short digest/public key h_C, where a fast *online* verifier can check validity of signature $\sigma_{C,y}$ given just y and h_C. That is, the online verifier runs in time independent of the circuit size. This can be used to: (1) verify an evaluated signature quickly by performing circuit pre-processing offline, and (2) amortize the cost of verification when a given circuit is evaluated on many different signed datasets.

Applications and prior work. Homomorphic signatures with above properties have proven to be quite useful for numerous applications such as computing statistics on signed data [BF11a], network coding [AB09, ABBF10], proofs of retrievability [SW13], trustworthy delegation of computation [GGP10, PHGR16, GVW15], attribute-based signatures [MPR11, Tsa17], and more.

Initial research [JMSW02, ABC+07, SW13, DVW09, AKK09, AB09, BFKW09, GKKR10, BF11a, AL11, BF11b, CFW12, ABC+12, Fre12, GW13, CF13] led to homomorphic signatures

[1]Here $C^* \circ C_1|\cdots|C_\ell$ denotes the composition of circuits, i.e., $(C^* \circ C_1|\cdots|C_\ell)(m) = C^*(C_1(m), \cdots, C_\ell(m))$.

[2]If one views homomorphic signatures as a specialized succinct non-interactive proof system, then one could consider C, y (and, also vk) to define an NP instance, where the verifier is trying to check if the prover has a valid witness (i.e., σ_m and m) for it. Clearly, a verifier needs to at least read the full instance. This is what we mean when we say "the statement that the verifier is checking depends on both C and y".

with limited functionality, either because of the class of circuits class that could be homomorphicaly evaluated, or due to lack of public verifiability (i.e., vk must be private). In 2015, Gorbunov, Vaikuntanathan, and Wichs (GVW) [GVW15] proposed the first (leveled) fully homomorphic signature scheme under standard lattice assumptions [Ajt96]. Before [GVW15], the only other approach was via general purpose succinct non-interactive arguments (SNARGs) [Kil92, Mic94], which have strong implausibility results from falsifiable assumptions [GW11a]. GVW was the first standard model homomorphic signature scheme that could support homomorphic evaluation of general polynomial-sized circuits.

What is current state of homomorphic signatures? Since 2015, there has been a large body of exciting work [BFS14, LTWC18, FMNP16, Tsa17, EKK18, CFT22, BCFL23, KLVW23, GU24, WW24, Goy24] leading to many new constructions and generalizations of fully homomorphic signatures. As it stands today, we have fully homomorphic signatures from lattices [GVW15], or succinct functional commitments [CFT22, BCFL23, KLVW23, WW24], or indistinguishability obfuscation [GU24], or mutable batch arguments [Goy24]. Unfortunately, all existing constructions for homomorphic signatures suffer from one or more limitations.

- The original fully homomorphic signatures by GVW (and its follow-up variants) [GVW15, BFS14] do not achieve composability and context hiding simultaneously. They can achieve context hiding, but at the cost that the homomorphism property is broken (i.e., once you make a signature context hiding, it can no longer be homomorphically evaluated on). Moreover, they crucially rely on lattice-based homomorphic techniques, thus these constructions do not generalize to other cryptographic assumptions.

- Although homomorphic signatures based on succinct functional commitments and mutable succinct proofs [CFT22, BCFL23, KLVW23, WW24, Goy24] can be proven secure under pairing-based or lattice-based assumptions, they do not provide general composability, context hiding, or fast verification properties. For instance, these signatures support either constant-hop evaluation or sequential composition where an entire evaluated signature must be used as the input to the next circuit (i.e., only chained multi-hop evaluation).

- The final approach is a recent work by Gay-Ursu [GU24] which relies on indistinguishability obfuscation and other standard cryptographic objects (such as lossy trapdoor functions [PW08], re-randomizable encryption [GM19, Elg85, Pai99, DJ01, CLTV15]). While this construction satisfies all desired properties of succinctness, unforgeability, composability, and context hiding (and it can also be shown to have fast verification), this is still not an ideal solution for two reasons. First, similar to SNARGs, indistinguishability obfuscation is a non-falsifiable assumption [Nao03]. Second, current constructions of obfuscations either themselves rely on non-standard assumptions (see [GP21, WW21, BDGM20] and references therein), or rely on a careful combination of multiple assumptions [JLS21, JLS22].

Ideally, we want to design homomorphic signatures that satisfy *all* the aforementioned properties from *simple and falsifiable cryptographic assumptions*. Morever, we would also like to *not* rely on combinations of different cryptographic assumptions. While combining cryptographic assumptions is an interesting and successful research strategy [GQWW19, AY20, AWY20, JLS21, JLS22] to break new ground in cryptography, it is always desirable to reduce the strength of computational assumptions needed for a particular cryptographic task. In this work, we study the problem of homomorphic signatures with the same motivation.

Our results. We provide new constructions for homomorphic signatures for polynomial-sized circuits from a variety of standard assumptions such as sub-exponential DDH, standard pairing-based assumptions, and learning with errors. To summarize, we prove the following.

1. Assuming DLIN, or sub-exponential DDH (and QR), or LWE, there exists a *single-hop* homomorphic signature scheme for any boolean circuit of size S, where the signature size is $\mathsf{poly}(\lambda, \log S)$. This scheme satisfies *context hiding* and *fast verification*. *Unlike most prior works, the size of an evaluated signature does not grow polynomially with the depth or the width of the circuit.*

2. Assuming the sub-exponential hardness of either DLIN, or DDH (and QR), or LWE, there exists a *multi-hop* homomorphic signature scheme for any boolean circuit of depth d, where the signature size is $\mathsf{poly}(\lambda, d, k)$ and k denotes the number of hops. This scheme satisfies *fast verification*, and

   (a) assuming LWE, we can prove this scheme to satisfy *strong context hiding*, where the context hiding property holds even for intermediate evaluated signatures.

   (b) assuming DLIN, or sub-exponential DDH (and QR), we show that evaluated signatures can be made context hiding at the cost of sacrificing homomorphism (similar to [GVW15]).

The central toolkit that we use in our work are non-interactive batch arguments (BARGs) [RRR16, BHK17, KPY19, CJJ21a, CJJ21b], and we state our main theorems more generally in the technical overview and the main body.

Related work. General-purpose succinct non-interactive arguments [Kil92, Mic94] are a vigorous tool in cryptography. However, there are known barriers in black-box proof techniques for designing SNARGs [GW11a]. Over the last few years, BARGs have emerged as a powerful alternative to SNARGs, and received a lot of attention in cryptography. The study of this primitive has led to a lot of new constructions (including ones from standard assumptions such as LWE, DLIN, or sub-exponential DDH (and QR)) [CJJ21a, CJJ21b, KVZ21, WW22, HJKS22, DGKV22, PP22, CGJ+23, KLV23, KLVW23] as well as many important applications. BARGs have made a major contribution in solving several long-standing open problems (see the non-exhaustive list [RRR16, BHK17, KPY19, CJJ21a, CJJ21b, KVZ21, WW22, HJKS22, DGKV22, PP22, KLVW23, GSWW22, CGJ+23, KLV23, Goy24] and references therein).

Concurrent work. A concurrent work by Anthoine, Balbás, and Fiore [ABF24] designed multi-key homomorphic signatures [FMNP16] by combining batch arguments and functional commitments [LRY16]. Multi-key homomorphic signatures are a generalization of regular homomorphic signatures to the multi-signer model, where homomorphic evaluation can be performed on signatures generated by different signers. Their scheme only supports chained multi-hop evaluation, whereas the focus of our work is on general multi-hop evaluation. Moreover, as we discuss in the technical overview, our multi-hop homomorphic signatures can be generalized to the multi-key model in a straightforward manner.

2 Technical Overview

Homomorphic signatures are a strengthening of regular signatures, where one can homomorphically evaluate any signed dataset. To enable maximal flexibility for a signer, we consider the setting where every bit of the dataset can be signed independently and asynchronously. More formally, in a homomorphic signature scheme, any dataset $m = (m_1, \ldots, m_k)$ (where m_i is a single bit) can be signed bit-by-bit by running the signing algorithm $\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$. And, for any circuit C, such signatures can be homomorphically evaluated to generate a signature as $\mathsf{Eval}(\mathsf{vk}, (m_i, \sigma_i)_i, C) \to \sigma_{C,y}$, which is a signature for the evaluated value $y = C(m)$. A verifier then, takes as input the verification key vk, a circuit C and output y, and it checks whether a signature σ is a valid signature certifying computation of C to be equal to y or not.

In this work, we consider homomorphic signatures in two different settings. In the *single-hop* setting, the evaluation can only be done over the original signed dataset. Whereas, in the *multi-hop* setting, one can evaluate over the evaluated signed messages, for any polynomial number of times. Unforgeability of the homomorphic signatures requires that for any signed dataset m, no adversary can find a tuple (C^*, y^*, σ^*) where σ^* is an accepting signature for a message y^* s.t. $y^* \neq C^*(m)$. This can be naturally extended to the multi-hop setting as well, where the signature σ^* could potentially be an output of multi-hop homomorphic evaluation. As mentioned in the introduction, we provide constructions for homomorphic signatures in both single-hop and multi-hop settings.

The starting point for our homomorphic signatures is the recent work by Goyal [Goy24], who proposed a new approach to design homomorphic signatures from monotone-policy succinct non-interactive arguments for $\mathsf{batchNP}$ (henceforth referred as monotone SNARGs) [BBK+23, NWW23]. Goyal [Goy24] designed (multi-key) homomorphic signatures satisfying context hiding and fast verification. His construction could support a *constant* number of hops, and relied on standard lattice assumptions [Reg05, BBK+23]. Although his construction uses a general template (described via *mutable* batch arguments [Goy24]), one could summarize their homomorphic signature scheme using only monotone SNARGs. We start by recalling the concept of BARGs and monotone SNARGs.

Reviewing BARGs and SNARGs. BARGs allow a prover to generate a short proof (with size independent of k) for a 'batch' statement that $x_1 \in \mathcal{L} \wedge \ldots \wedge x_k \in \mathcal{L}$, for some NP language $\mathcal{L}$. That is, it creates a short proof proving validity of a batch of k instances, in space independent of batch size k. Soundness states that an attacker cannot create an accepting proof for a batch of instances containing at least one instance $x \notin \mathcal{L}$. Somewhere extractable BARGs (seBARGs) [CJJ21b, CJJ21a] are a mild strengthening of BARGs, which enables extraction of a witness for a single statement at some trapdoor index $i^* \in [k]$ (secretly embedded in the crs). This extraction is enabled by the knowledge of a trapdoor key associated with crs.

In a recent beautiful work, Brakerski et al. [BBK+23] generalized BARGs to a highly non-trivial language that enabled a non-deterministic composition of a batch of NP statements. They define it as the monotone-policy $\mathsf{batchNP}$ language $\mathcal{L}_{\tilde{C},\mathcal{R}}$, which is associated with a monotone circuit $\tilde{C}$ and an underlying NP relation $\mathcal{R}$, defined as:

$$\mathcal{L}_{\tilde{C},\mathcal{R}} = \{(x_1, \ldots, x_k) : \exists (w_1, \ldots, w_k) \text{ s.t. } \tilde{C}(b_1, \ldots, b_k) = 1 \text{ where } b_i = \mathcal{R}(x_i, w_i)\}.$$

The succinctness of monotone SNARGs requires the proof size to be independent of k, as in standard BARGs. Non-adaptive soundness requires that an adversary who chooses an instance

$(\tilde{C}, x_1, \ldots, x_n)$ before the setup, cannot generate an accepting proof upon seeing the crs. Moreover, Brakerski et al. [BBK+23] also considered a strong *argument of knowledge* (full extraction) property for monotone SNARGs. It states that there exists an extractor that given oracle access to a prover, that generates accepting proofs for an instance $(x_1, \ldots, x_k)$ with a non-negligible probability, extracts a full set of witnesses $(w_1, \ldots, w_k)$ s.t. $\tilde{C}(b_1, \ldots, b_k) = 1$ where $b_i = \mathcal{R}(x_i, w_i)$.

Current landscape for monotone SNARGs. [BBK+23] introduced the notion of monotone SNARGs and provided multiple constructions for it. At their core, all constructions started used seBARGs as core technical component. For the purposes of the current discussion, it is sufficient to focus on their first monotone SNARG construction. It is proven secure under the LWE assumption, and provides full extraction with proof size $m \cdot \mathsf{poly}(\lambda, \log(kmn)) + \mathsf{poly}(\lambda, \log(k)) \cdot \mathsf{width}(C)$ (where m, n are witness and instance lengths). They also constructed monotone SNARGs for depth-d circuits from 2^d security of seBARGs, but this does not satisfy full extractability. Recently, Nassar, Waters, and Wu [NWW23] improved the design for monotone SNARGs from [BBK+23] by using simpler hash functions and a tighter proof technique. This led to monotone SNARGs with non-adaptive soundness and proof size $\mathsf{poly}(\lambda, m, \log|\tilde{C}|)$ under either LWE, or k-LIN over pairing groups, or sub-exponential DDH assumptions; however, their scheme also does not satisfy full extractability.

2.1 Monotone SNARGs to Single-Hop Homomorphic Signatures

In a recent work [Goy24], Goyal showed that a single-hop[3] homomorphic signature scheme can be designed by combining any standard signature scheme with a fully extractable monotone SNARG. As mentioned earlier, their scheme uses a new framework for BARGs called mutable BARGs; however, for this technical overview, we can boil down the main techniques to the following two steps: (i) an evaluator can efficiently and deterministically transform a general non-monotone circuit C, that takes a string x as input, into a monotone circuit $\tilde{C}$, that takes a string $(x, x \oplus 1^{|x|})$ as the input, and (ii) to homomorphically evaluate a signature, an evaluator can create a monotone SNARG proof for such a 'monotonized' circuit instead[4] while using the bit-by-bit signatures on the dataset as the witness.

In a bit more detail, the signature of the i-th bit of the data being m_i is simply a signature σ_{i,m_i} on (i, m_i). An evaluator uses the signatures $\{\sigma_{i,m_i}\}_i$ as the witnesses for monotone SNARG prover. Recall the i-th input wire of the monotonized version of the circuit reads m_i, while $(|m|+i)$-th wire reads $m_i \oplus 1$. So, an evaluator uses σ_{i,m_i} as a witness for the i-th input wire if $m_i = 1$, otherwise it uses σ_{i,m_i} as a witness for the $(|m| + i)$-th wire. The underlying NP relation $\mathcal{R}$ for the batch language portion corresponds to the signature verification circuit, and the monotone circuit $\tilde{C}$ is just the 'monotonized' version of the actual (possibly non-monotone) evaluation circuit. Now note that every NP instance at the input layer of the monotone circuit is associated with a signature verification circuit, thus all the instances are in the language (signatures exist for all the values) and it is only computationally hard to find witnesses (in turn, dataset signatures). Thus, to argue unforgeability, one has to rely on the *full extraction* of monotone SNARGs to extract all signatures at the input level. From the extracted signature, one of them must be a forgery on the underlying signature scheme, as otherwise either monotone SNARG's extractor extracted incorrect witnesses or the adversary's signature was not a forgery to begin with. Since the above approach crucially

[3] Actually, [Goy24] showed it to be a $O(1)$-hop (i.e., constant number of hops).

[4] Goyal [Goy24] originally stated the monotonization process for log-depth circuits but, as they noticed later, the same construction also works for poly-depth circuits too. Such extensions have also been noted in prior works [NWW23].

relies on full extractability, Goyal [Goy24] was only able to design single/constant-hop (multi-key) homomorphic signatures from lattice assumptions. This is primarily because monotone SNARGs satisfying full extractability are still only known from lattice assumptions [BBK+23].

At first it might appear that the need for full extractability could be a fundamental bottleneck, and we cannot generalize the above idea to non-lattice assumptions. But in this work, we show two unrelated approaches to get around this barrier. Our first approach is inspired by the recent work of Nasser, Waters, and Wu [NWW23]. Herein we show that by replacing standard signatures with a special form of signatures called all-but-one signatures [GVW19], we could bypass the issue of "computationally hard to find witnesses", and rather make it such that there are no valid witnesses. Our second approach is more technically involved, where we show that rather than relying on full extractability, we can reduce unforgeability of homomorphic signatures to unforgeability of regular signatures, as long as we change our construction to use seBARGs and exploit the somewhere extractability to find a forgery in a careful way during the security reduction. We elaborate the second approach later when we describe our design for multi-hop signatures. Next, we dive into the first approach.

All-but-one signatures, and why they aren't enough? Nasser et al. [NWW23] recently came across a similar issue, which is how to use monotone SNARGs to prove security of an advanced signature scheme without relying on full extraction. They were interested in designing a generalization of aggregate signatures [BGLS03], called monotone-policy aggregate signatures[5] [NWW23, BCJP24]. Rather than detailing such specialized aggregate signatures, we jump right into the main technical insight. They observed that by employing all-but-one (ABO) signatures [GVW19], the extraction issue can be bypassed. Let us briefly summarize the notion of ABO signatures. These allow sampling the verification key in a "punctured" mode, where for any particular message m^*, the punctured setup generates a punctured key $\mathsf{vk}\{m^*\}$ such that there does *not* exist any signature for m^* that gets validated, as well as $\mathsf{vk}\{m^*\}$ just looks like a regular non-punctured key vk.

Now by plugging in ABO signatures at the input layer of monotone SNARGs, we can rely on the non-adaptive soundness of monotone SNARGs to argue unforgeability as now we can puncture every message bit complementary to the data m. That is, we would like to puncture $(1, m_1 \oplus 1), \ldots, (k, m_k \oplus 1)$. If we could puncture all these k messages, then there cannot exist a valid witness for the forgery circuit-output pair (C^*, y^*).

A straightforward adaptation of the above idea requires all-but-k (ABk) puncturable signatures, since for any message bit m_i, we need to puncture $(i, m_i \oplus 1)$. Unfortunately, this primitive is not so easy to design. A common trick to generically build ABk from ABO signatures will be to sample k different ABO keys, and to sign a message, we sign it under all k keys. To puncture all k messages, we can puncture each message from just one key. While this seems like an easy fix, it is not good enough. The issue is that the signature size grows with k, and this breaks the succinctness of the evaluated signature. With such an ABk signature, each original signature is of size k and this will be used as a single witness in the monotone SNARGs. We could consider using BARGs to aggregate all the signatures to make them shorter, but note that we need a statistical guarantee and any standard compression technique would turn this into a computational guarantee which will not be enough. Namely, the verification should reject all signatures for those k messages. Therefore, a

[5] As an interesting side-contribution, we show that single-hop homomorphic signatures can alternatively be designed from monotone-policy aggregate signatures in Appendix C. Although it does not satisfy context-hiding or fast verification directly, since the underlying aggregate signatures do not satisfy such properties.

cryptographic way of aggregation that only provides computational security is not enough.

One-time ABO signatures are enough! While we fail to generically build this object with short signatures, we make a rather interesting observation about our homomorphic signatures. In our homomorphic signatures, for any index i, there are just two possible messages that could be signed – either $(i, 0)$ or $(i, 1)$. Basically, a signature for a dataset m only contains k signatures out of $2k$ signatures corresponding to messages $(1, 0), (1, 1), (2, 0), \ldots$ and so on. Thus, we do not need a general ABk signature, and really just need a much weaker signature scheme.

Basically, our observation is that an ABO signature for single-bit messages is enough! And, for this special case, ABO signatures are far more easier than general ABO signatures for multi-bit messages. Consider a simple construction based on Lamport's one-time signature. Let G be a length-doubling PRG. We can design an ABO signature for single-bit messages using just the PRG G. Consider the secret key to be two random strings, i.e. $\mathsf{sk} = (x_0, x_1)$, and verification key to be its PRG evaluations, i.e. $\mathsf{vk} = (G(x_0), G(x_1))$. Here x_b serves as a signature for bit b, and to create a verification key punctured for bit b^*, we replace $G(x_{b^*})$ with a random value. Technically, this approach introduces a statistical puncturing error, but it can be avoided by using a perfectly binding commitment instead of a PRG. An added advantage of our approach is that we can instantiate this from any injective PRG; unlike ABO signatures [GVW19], which were known from assumptions such as LWE and k-LIN over pairings, but not yet from DDH. For completeness, we sketch this ABO signature in Appendix B.

Getting back to our design, our plan is to use a separate single-bit ABO signature for each index of the dataset. Unfortunately, this way the joint verification key of the signature scheme would be large as it would contain k different verification keys for single-bit ABO signature scheme. However, this is not an issue because the verification algorithm at each input wire of the monotone circuit only runs verification for a single ABO signature scheme. Technically, the evaluated signature verification, which is a monotone SNARG verifier still requires reading all the verification keys which might not be efficient, but using simple online/offline-verification techniques [CJJ21b], we can avoid this cost. That is, by hashing the verification keys and generating proofs w.r.t. the hashed values, we can use the online/offline-verification techniques [CJJ21b] from seBARGs to build a verification process that run in $\mathsf{poly}(\log k, |\mathsf{vk}_i|)$. Basically, we can create a short digest of all k verification keys during setup, and include the digest of verification keys as part of the new verification key. This way, only the evaluators needs to read the entire verification key which contains k ABO verification keys, but the verifier only needs the digest of the verification keys.

Our single-hop homomorphic signature scheme. By combining all the above ideas we construct single-hop homomorphic signatures as follows: (1) the signing algorithm is a single-bit ABO signature that signs m_i using sk_i to get σ_i, and (2) the evaluation algorithm computes a monotone SNARG proof for the statements $(1, \ldots, 2k)$, the witnesses $(\sigma_1, \ldots, \sigma_k, \sigma_1, \ldots, \sigma_k)$, a monotone circuit $\tilde{C}_y(m', m' \oplus 1^k) := \mathbb{1}(C(m') = y)$, where $\tilde{C}_y$ has $y = C(m)$ hard-coded, and the NP relation:

$$\mathcal{R} := \mathbb{1}((i \leq k \ \wedge \ \sigma_i \text{ is a valid signature for } 1) \ \vee \ (i > k \ \wedge \ \sigma_i \text{ is a valid signature for } 0)).$$

For completeness note that if $C(m) = y$ then $\tilde{C}_y(m, m \oplus 1^k) = 1$, $m_i = \mathcal{R}(x_i, w_i)$ for $i \leq k$ and $m_i \oplus 1 = \mathcal{R}(x_i, w_i)$ for $i > k$. For soundness, we first puncture the i-th ABO verification key at $m_i \oplus 1$. Now if $\tilde{C}_y(b_1, \ldots, b_{2k}) = 0$ (where $b_i = \mathcal{R}(x_i, w_i)$) and the evaluated signature verifies, then there is some index i s.t. either $\mathcal{R}(x_i, w_i) > m_i$ and $i \leq k$, or $\mathcal{R}(x_i, w_i) > m_{i-k} \oplus 1$ and $i > k$. This

only happens if $i \leq k$ (resp. $i > k$) and w_i is a valid signature for 1 (resp.0) while $m_i = 0$ (resp. $m_{i-k} = 1$), which contradicts with the puncturing property. Note that the monotonization process from general to monotone circuits leads to same depth and only a factor of two overhead on the circuit size. Thus, for succinctness, the evaluated signature size is just a monotone SNARG proof, thus it is $\mathsf{poly}(\lambda, \log |C|)$.

Note that our construction of single-hop homomorphic signatures, in addition to new results from non-lattice assumptions, improves existing lattice-based homomorphic signatures [GVW15, Goy24] in terms of efficiency as our signature size is independent of the circuit depth. Thus, we show the following.

Theorem 2.1 (informal, see Corollary 4.11)**.** Single-hop homomorphic signatures can be built assuming either LWE, k-LIN over pairing groups, or sub-exponential DDH over pairing-free groups.

Making it context-hiding. Context hiding for homomorphic signatures states that an evaluated signature does not reveal anything about the dataset m, beyond what can be learnt given circuit C and output y. In their seminal work, GVW [GVW15] proposed a simple template to obtain context-hiding property by applying NIZKA/PoKs as long as the homomorphic signatures were 'pre-processable'. Their core idea was that if one could preprocess the circuit C to a short digest, such that a verification algorithm only needs the short digest and not the circuit C, then one could generate a NIZK proof using the actual (non-context-hiding) evaluated signature as a witness. In words, the verification first pre-processes the circuit C to compute the digest, and then runs the NIZK verification. Now the security (unforgeability) of the system can be argued by combining NIZK extraction with the unforgeability of underlying homomorphic signatures, while context-hiding can be reduced to the zero-knowledge property of the NIZK scheme.

We follow a similar strategy. We show that by relying on fairly standardized online/offline verification features of BARGs, that is the verification algorithm of a BARG scheme can be split into a pre-verification and an online verification, we can use the GVW-style compiler to make our signatures context-hiding. Briefly, most BARG verifiers work in two stages: first, PreVerify algorithm generates a short digest of the statements whose size is independent of k, and second, the OnlineVerify algorithm takes the short digest and BARG proof and outputs 1/0 to signal validity. Here the OnlineVerify algorithm runs very fast, and the idea is to only create a NIZK proof for this computation. Our starting observation is that such an online/offline verification property can also be proven for existing monotone SNARG constructions. For example, in the monotone SNARG construction of [NWW23], the proof contains two digests (that can be viewed as a shared part among all the statements) and a BARG proof. Now one could preprocess the verification of such monotone SNARGs by creating a short digest of all instances and these shared digests, and this would correspond to the preprocessing portion for monotone SNARGs. The online verification process is simply the online verification process for the BARGs. Since the online verification is fast, we can simply generate a NIZK proof of the online verification step, given the digest of the statements and the monotone SNARG proof. We elaborate this further in the main body, and summarize the main result as follows.

Theorem 2.2 (informal, see Corollary 4.19)**.** Context-hiding single-hop homomorphic signatures can be built assuming either LWE, k-LIN over pairing groups, or sub-exponential DDH over pairing-free groups. Moreover, they support fast amortized/online verification.

2.2 Composable BARGs to Multi-Hop Homomorphic Signatures

In the multi-hop setting, the evaluator gets a sequence of evaluated signatures $\sigma_{C_1,y_1}, \ldots, \sigma_{C_\ell,y_\ell}$ for some circuit-output pairs (C_i, y_i) as inputs, in addition to a circuit C that has to be homomorphically evaluated. In order to make the notation for multi-hop evaluation cleaner, we assume (without loss of generality) that the each circuit outputs a single-bit and the evaluator (as well as the verifier) receives the evaluation circuits as directed acyclic graphs. Here by making each circuit output just one bit, it is easier to define signature for intermediate homomorphic evaluations, and by interpreting each evaluated circuit as a directed acyclic graph, it is easy to represent the structure of the multi-hop homomorphic computation so far.

A bit more formally, we define the multi-hop evaluation algorithm as $\mathsf{Eval}(\mathsf{vk}, t, (y_i, \sigma_i, G_i, \{C_{i,v}\}_{v\in V_i})_{i\in[\ell]}, C) \to \sigma$. Here t denotes the number of homomorphic evaluation hops that have been performed so far, $y = (y_1, \ldots, y_\ell)$ is the circuit output after t hops, G_i is the structure of the computation so far performed while computing σ_i (where each node v of G_i is a circuit $C_{i,v}$), and σ is the final homomorphic signature for $C(y)$. Naturally, a verifier gets an evaluated circuit in the above graph-based format along with the circuit output, and it checks validity of the evaluated signature. As discussed earlier, our main goal is to design multi-hop homomorphic signatures, satisfying context hiding and fast verification, while relying only on simple and falsifiable cryptographic assumptions. As a starting point, we shift our focus to just multi-hop evaluation, and later circle back to the remaining features.

Composing monotone SNARGs and why that is insufficient. A natural idea to design multi-hop homomorphic signatures is to apply a strategy similar to what we used in the single-hop setting. Namely, given a set of circuit output values and *evaluated* signatures, generate a fresh monotone SNARG proof for the next hop of homomorphic evaluation, while using the *evaluated* signatures as the witnesses. Such an approach was already used and proposed in [Goy24], but unfortunately, this naive approach is quite limiting because of two major issues.

(a) The first issue is the necessity of extractable monotone SNARGs for proving unforgeability of multi-hop signatures. The issue is quite similar to what we faced even while constructing single-hop homomorphic signatures from monotone SNARGs without full extraction. Namely, after the first hop, the signatures are monotone SNARG proofs, and proofs for other values will exist and it is unclear how to puncture them (unlike signatures). Thus, we either need to assume the underlying monotone SNARGs are *extractable*, or we need a powerful notion of *puncturable* monotone SNARGs. Such monotone SNARGs seem too strong, thus the first barrier that we need to bypass is to develop a new proof technique that does not need extraction or punctured security for SNARGs.

(b) Suppose we can find a new proof technique to prove unforgeability of our multi-hop signatures, there is one more hurdle that we need to get around. In a few words, the issue is due to the large blow-up in the signature size due to composition of monotone SNARGs. This was already noted in [Goy24], and this is exactly why their construction could only support constant number of evaluation hops. Simply put, the issue is that a monotone SNARG proof polynomially grows with the size of a single witness, and thus a recursive composition of such SNARG proofs will lead to cascading polynomials (i.e., $\mathsf{poly}(\mathsf{poly}(\ldots\mathsf{poly}(\cdot))))$. Therefore, monotone SNARGs can only be composed constant number of times, which is not ideal.

One might wonder whether the second issue can be resolved fairly easily by relying on recent exciting works in optimal-rate BARG proofs [DGKV22, PP22].

An optimal-rate BARG proof, also commonly referred to as rate-1 BARGs, are batch argument systems where the proof size is $|\pi_{\mathsf{BARG}}| = |w| + \mathsf{poly}(\lambda, \log k)$. That is, the size of the proof is equal to the size of a single witness, plus fixed additive polynomial terms. It is already known that such rate-1 BARGs enable arbitrary recursive composition of BARGs. That is, they can be used to design multi-hop/composable BARGs [DGKV22], where a BARG proof can be used as a witness in a future BARG and this process can be recursively carried out any polynomial number of times. Thus, one might wonder whether we could directly used rate-1 BARGs to instantiate monotone SNARGs via the current designs [BBK+23, NWW23], and that would get around the above recursive signature growth issue. As we detail later, plugging rate-1 BARGs inside the existing monotone SNARG constructions is not enough, and we really need to open up the existing monotone SNARG constructions, and make them amenable towards our goal of multi-hop homomorphic signatures. Next, we describe our approach to resolve the above issues.

Opening up monotone SNARGs: using somewhere extractable BARGs. As we realized, using monotone SNARGs as a black box is not good enough. To get around the barrier of inability to prove security while just relying on non-extractable monotone SNARGs, our approach is to open up the existing designs for monotone SNARGs [BBK+23, NWW23]. It turns out all existing constructions for monotone SNARGs follow a similar template, which is to use somewhere extractable BARGs as a core primitive and create the SNARG proof as a batch proof. Therefore, intrinsically, a monotone SNARG construction does enjoy a partial/somewhere-extraction feature that we could potentially exploit to get around the *full* extraction barrier.

In more detail, let us recall the canonical template for designing monotone SNARGs [BBK+23]. Consider a batch of instances $(x_1, \ldots, x_k)$, witnesses $(w_1, \ldots, w_k)$, an NP relation $\mathcal{R}$, and a monotone circuit $\tilde{C}$. The canonical template to create a monotone SNARG for these elements is the following two-step method:

1. Create a short (digested) commitment dig to the value of each wire, as computed during the evaluation of $\tilde{C}(\mathcal{R}(x_1, w_1), \ldots, \mathcal{R}(x_k, w_k))$.
2. Create a BARG to prove a group of two types of statements:
   (a) Each input wire is correctly computed and committed. That is, the input wire i is set to be $\mathcal{R}(x_i, w_i)$, and $\mathcal{R}(x_i, w_i)$ is correctly committed inside dig w.r.t. input wire i.
   (b) Each internal wire is correctly computed and committed. That is, if an internal wire j is the output wire of some gate g, where wires j_0, j_1 are its input wires, then the wire values committed inside dig are consistent w.r.t. gate g.

The monotone SNARG simply contains the (short) wire commmitment dig as well as a batch proof π_{BARG}, proving validity of all aforementioned statements. Since the size of the batch proof does not scale with the batch size (which is almost the number of wires in circuit $\tilde{C}$), thus the resulting monotone SNARG proof is succinct.

Brakerski et al. [BBK+23] instantiated the above template with a hash function with short local openings to create the wire commitments (e.g., Merkle tree) and a somewhere extractable BARG. They proved the above monotone SNARG proof system to be computationally sound. At a very high level, the soundness proof follows the folklore global-to-local style of reduction [GKR15]. By

this we mean, that suppose a cheating prover creates an accepting proof for an invalid statement, then one could identify at least one gate/wire in the claimed evaluation dig of the monotone circuit $\tilde{C}$ such that it proves an incorrect statement.

A bit more concretely, we can visualize this as a (guessing-based) top-down reduction, where the reduction starts from the top (i.e., the output wire) and traces a path down the monotone circuit. Its goal is to find either a gate, such that the internal wire was not correctly evaluated and/or committed, or an input wire was not correctly set and/or committed. Since the circuit under consideration is a monotone circuit, thus starting from the output layer one can argue that if, for any layer, there is an internal wire j whose value claimed in the proof is greater than its actual value in the correct computation, then the invariant of wire's actual value being greater than the claimed value will also hold for at least one of the input wires of the corresponding gate with output wire j. This way a reduction can guess a path from the output wire to the input wire catching the adversary at at least one layer along this path.

Building multi-hop directly from seBARGs. Our strategy for designing multi-hop homomorphic signatures, without using full extractability of monotone SNARGs, is to carefully instantiate the above template for multi-hop signatures. Basically, our plan is to use seBARGs as the underlying technical tool instead, and execute a similar top-down reduction where we will view a multi-hop signature as an seBARG proof along with a (short) digest of all internal wires of all evaluated circuits. Thus, we do not need to worry about full extractability, and just by exploiting somewhere extractability of seBARGs, we plan to reduce unforgeability to soundness of seBARGs, collision resistance of hash functions, and unforgeability of underlying signatures.

While the above strategy carries the right ideas, there are still two important caveats that we would like to point out. First, in such a top-down reduction approach, one cannot deterministically figure out which wire at any layer in the proof is greater than its actual value, thus the reduction must guess this at each layer. This implies that there is a factor-of-two security loss per layer in the reduction as we go down the monotone circuit. Therefore, if the circuit has depth d, then we have to rely on 2^d hardness of the underlying seBARG. By using standard complexity leveraging techniques, we can execute the proof strategy while reducing to sub-exponential security of the underlying assumptions[6].

The second (and bigger) caveat with this strategy is that we need the seBARG to be 'extractable for two instances'. That is, we need witness extractability for two instances from the batch of, say k, instances. This is essential because, to argue the soundness, we need to extract two seBARG witnesses (i.e., wire values and their openings) from two consecutive layers of the monotone circuit. If we do not extract at two locations, then the iterative top-down proof strategy does not work.

Why is extracting at two places an issue for multi-hop? Recall that to get around the large blow-up issue in the signature size due to proof composition, we are considering relying on composable/rate-1 BARGs to ensure the blow-up is controlled. This is because by using rate-1 seBARGs we might just have to pay an additive polynomial cost each time we homomorphically evaluate a set of evaluated signatures.

[6] Brakerski et al. [BBK+23] and Nasser et al. [NWW23] also considered alternate proof strategies to prove soundness without incurring this sub-exponential loss, but those do not seem to be compatible with our multi-hop homomorphic signature construction. We leave proving unforgeability of our multi-hop homomorphic signatures from polynomial hardness as an interesting open problem.

Unfortunately, the current security proof strategy heavily relies on the layer-by-layer argument and, hence requires at least one extraction on each layer. This suggests that the seBARGs must be extractable on two indices. Note that any seBARGs that is extractable on i witnesses can be generically built using i seBARGs that are extractable on a single witness. However, this brings down the proof-rate from 1 to $1/i$ since, for each single-witness-extractable seBARG, the proof size grows as $|w| + \mathsf{poly}(\lambda)$. This means that even if we start with rate-1 seBARGs (which is the optimal rate possible for straightline-extraction[7]), for the above construction we will have to use two rate-1 seBARGs (or one rate-$\frac{1}{2}$ seBARG extractable on two indices). Thus, $\pi_{\mathsf{BARG}} = 2|w_i| + \mathsf{poly}$ which would mean that by recursively composing such proofs, the resulting seBARG proofs (in turn, the multi-hop signatures) will grow as 2^t, where t is the number of hops. While this is already interesting as it improves previous homomorphic signature construction [Goy24] by enabling logarithmic number of evaluation hops, our end goal is to enable general multi-hop (leveled) evaluation over homomorphic signatures. Thus, plugging in a rate-1 seBARG directly into the above construction is not sufficient for handling an arbitrary polynomial number of hops.

Width-2 chained composition of rate-1 and rate-$\frac{1}{2}$ seBARGs. One of our main insights is the fact that there is an implicit structure in the language used for seBARGs while designing homomorphic signatures. Recall that the BARG proof, in the multi-hop signature candidate construction, checks the consistency of the internal gates in addition to the correctness of the input wires. While the witnesses proving validity of the 'input wires' could potentially be large (since the witness could be an already evaluated signature), the witnesses for the 'internal wires' (i.e., for checking gate consistency) are always *a fixed short polynomial regardless of the number of evaluation hops performed.*

This core observation drives our main modification to the current multi-hop construction. Our idea is that an evaluator now will generate two separate seBARG proofs. Instead of generating a single seBARG proof that is extractable on two indices (rate-half) for the "composed" language (validity of input wires and gate consistency), we generate two seBARG proofs — (1) for proving validity of the input wires, we use an seBARG proof system that is *extractable on just a single index (rate-1)*, while (2) for proving gate consistency, for the entire evaluated circuit, we use an seBARG proof system that is extractable on two indices (rate-half).

Now one might wonder that this actually is increasing the size of an evaluated signature! Quite clearly, now the evaluated signature contains two seBARG proofs instead of one. Moreover, one seBARG is extractable on two indices, while the other on just one. Thus, in summation, one can potentially extract three witnesses from the proofs jointly. This is unlike the original design, where there is just one seBARG extractable on two indices. While at first, this seems counter-productive, we have made great progress and this proof splitting operation enables arbitrary polynomial composition of evaluated signatures.

To understand further, let us look more carefully at our modified multi-hop signature design. By splitting the seBARG proofs into two separate proofs, we are really composing the underlying seBARG proofs in a very atypical fashion. Note that there are two seBARG proofs that are part of any evaluated signature. Now whenever an evaluated signature is used as a witness for the next level of homomorphic evaluation, then the evaluated signature is **never** used as a witness in the seBARG proof where the rate could be $\frac{1}{2}$. That is, any evaluated signature is only ever used as a

[7] A recent work by Cheng and Goyal [CG24] proves a stronger result about optimality of rate-1 BARGs, wherein they show any improvement would lead to a fully-succinct SNARK thereby inhibited by Gentry-Wichs [GW11b].

witness for another seBARG proof computation, if that BARG system is extractable at only one index (thus, can be rate-1). In words, the property that we maintain is that output of a rate-$\frac{1}{2}$ seBARG proof (extractable at two indices) is never used as a witness in another rate-$\frac{1}{2}$ seBARG proof computation, but only in some rate-1 seBARG proof computation. Thus, the blow-up due to seBARG proof composition is still capped at an additive growth each time, where the factor-of-2 never gets cascaded.

The proof strategy stays similar to the strategy that we discussed earlier, except at some points in the proof we will extract from the rate-1 seBARG, while at other points we extract from the rate-half seBARG. Namely, we perform a layer-by-layer analysis using the second seBARG to extract two witnesses, while at the input layer, we will use the first seBARG to extract a single witness. Here the single witness at the input layer could itself be an evaluated signature, thus it is recursively extracted by following the same strategy.

It is crucial to note that the technique above guarantees that the recursive composition of seBARG proofs only happens on the first seBARG which is rate-1. More specifically for the jth hop, if the signatures/proofs that we get as input are of size ℓ_σ, then the output signature/proof size would be $\ell_{\sigma'} = |\pi^{(1)}_{\mathsf{BARG}}| + |\pi^{(2)}_{\mathsf{BARG}}| + |\mathsf{com}|$. Note that the commitment is just a Merkle tree hash, hence its size is $\mathsf{poly}(\lambda)$. The first BARG proof is a rate-1 seBARG on the input signatures thus its size is $|\ell_\sigma| + \mathsf{poly}(\lambda)$, and the second seBARG only checks for the gate consistency w.r.t. the commitment openings thus its size is $\mathsf{poly}(\lambda, \log N)$. Therefore the output signature/proof size will be $\ell_{\sigma'} = |\ell_\sigma| + \mathsf{poly}(\lambda, \log N)$, as desired. Below we provide a more detailed sketch of our multi-hop signature construction.

Our multi-hop homomorphic signature construction. We use regular regular signatures, hash functions with local openings, and rate-1 seBARGs as follows:

- The signing algorithm is a regular signature Sig that sign (i, m_i) using $\mathsf{sig.sk}$ to get σ_i. We let y^t be the evaluated message at the t-th hop and y^0 is simply the messages $(m_1, \ldots, m_\ell)$ signed by regular digital signature.

- At the t-th hop, the evaluation algorithm does the following:

  1. Given C and y^{t-1}, compute $y = C(y^{t-1})$ and construct the corresponding monotone circuit $\tilde{C}_y$ (similarly to single-hop seeting), and compute all the wire values $(b_1, \ldots, b_N)$ in the evaluation of $\tilde{C}_y(y^{t-1}, y^{t-1} \oplus 1^\ell)$.
  2. Compute digest h of $(b_1, \ldots, b_N)$ and opening ρ_i using a Merkle tree hash.
  3. Compute a rate-1 seBARG proof on the statements $(1, \ldots, 2\ell)$ and the witnesses $((b_1, \rho_1, \sigma_1^{t-1}), \ldots, (b_\ell, \rho_\ell, \sigma_\ell^{t-1}), (b_{\ell+1}, \rho_{\ell+1}, \sigma_1^{t-1}), \ldots, (b_{2\ell}, \rho_{2\ell}, \sigma_\ell^{t-1}))$ for the NP relation:

$$\mathcal{R}^{(1)} := \mathbb{1}\left(\begin{array}{l} \rho_i \text{ is a valid opening for } b_i \text{ w.r.t. } \mathsf{dig} \wedge \\ b_i = \mathbb{1}\left(\begin{array}{l} (i \le \ell \ \wedge\ \sigma_i^{t-1} \text{ is a valid signature for 1}) \vee \\ (i > \ell \ \wedge\ \sigma_{i-\ell}^{t-1} \text{ is a valid signature for 0}) \end{array}\right) \end{array}\right).$$

  4. Compute a rate-half (extractable on two indices) seBARG proof on the statements $(2\ell + 1, \ldots, N)$ and the witnesses $\omega_i = (b_i, b_{i_0}, b_{i_1}, \rho_i, \rho_{i_0}, \rho_{i_1})$ (where i is the output and i_0, i_1

are the inputs of gate i) for the NP relation:

$$\mathcal{R}^{(2)} := \mathbb{1}\left(\begin{array}{l}\rho_i, \rho_{i_0}, \rho_{i_1} \text{ are valid openings for } b_i, b_{i_0}, b_{i_1} \text{ w.r.t. } h \wedge \\ b_i, b_{i_0}, b_{i_1} \text{ are consistent with gate } i \wedge \\ \text{if } i = N \text{ then } b_i = 1\end{array}\right).$$

To argue the soundness, we will proceed with a layer-by-layer analysis, similar to [BBK+23]. Let the correct evaluation of the circuit $\tilde{C}$ where $b_i = R(x_i, w_i)$ for $i \in [2k]$ be $(b_1^*, \ldots, b_N^*)$. Define the hybrid for wire i at layer L to be the following:

When $\mathsf{crs}^{(2)}_{\mathsf{BARG}}$ is extractable on the statement corresponding to some gate g in layer L, then for the committed value b_i in the digest dig, where wire i is the output of gate g, it holds that $b_i > b_i^*$.

Now the claim is that if an efficient adversary can forge a signature, that is to generate an accepting proof for a message y^* such that $y^* \neq C(y^{t-1})$, then in every layer L, there is a gate g for which the hybrid invariant holds. We will prove our claim inductively starting from the last (output) layer. Note that if $y^* \neq C(y^{t-1})$ then $\tilde{C}_y(y^{t-1}, y^{t-1} \oplus 1^\ell) = 0$. Therefore if the proof is accepted, the invariant holds for the output layer and value b_N (Note that at the beginning we let the seBARG be extractable on the output gate). Now suppose the invariant holds for some gate g in layer L, our goal is to show that it also holds for layer $L-1$.

First, using seBARG extraction, we extract a witness for the corresponding statement to gate g in $\pi^{(2)}_{\mathsf{BARG}}$. Let i_0 and i_1 be the input wires to gate g and b_{i_0} and b_{i_1} be the committed values in the digest dig. Since $b_i > b_i^*$ and gate g is monotone, by the gate consistency it holds that for some bit e, $b_{i_e} > b_{i_e}^*$. Now using the CRS indistinguishability of seBARGs, we let $\mathsf{crs}^{(2)}_{\mathsf{BARG}}$ to be extractable on gate g' whose output is wire i_e (while keeping $\mathsf{crs}^{(2)}_{\mathsf{BARG}}$ extractable on gate g). By the collision resistance property of the Merkle tree hash, overlapping parts (the openings of wire i_e) of the extracted witnesses on gate g and g' should be consistent. Thus for the extracted witness of gate g' it holds that $b_{i_e} > b_{i_e}^*$, which means that the invariant holds for level $L-1$. Therefore by the induction, the invariant should hold for some wire value in the inputs. Now we let $\mathsf{crs}^{(1)}_{\mathsf{BARG}}$ be extractable on that wire, and then extract a witness for the input layer which by the construction implies a forgery σ_i^{t-1} for a message y_i^{t-1}.

Hence, by following the same argument hop-by-hop, we can extract a forgery σ_i at the input layer on a message m_i. Finally, we will use the unforgeability of the regular signature to conclude the soundness argument. We prove the following, and provide more details later in Section 5.

Theorem 2.3 (informal, see Corollary 5.20)**.** Multi-hop homomorphic signatures can be built assuming the sub-exponential security of either LWE, k-LIN over pairing groups, or DDH over pairing-free groups.

Context-hiding and fast verification. Finally, we show that the above construction template can be easily extended to enable context-hiding and fast verification properties. As in the single-hop setting, we use the split-verification properties of (rate-1) seBARGs [DGKV22, PP22] to show fast verifiability for the multi-hop homomorphic signatures. Next, to make the above construction context-hiding, we again employ a similar strategy. Namely, we use NIZKs to hide any non-trivial information about the input dataset as well as the intermediate values during the homomorphic evaluation. However, since we want to achieve context-hiding for any evaluated signature after any

number of hops, thus must use a NIZK during every homomorphic evaluation. A straightforward application of NIZKs will not work since the NIZKs are not succinct. A NIZK proof can be as large as the underlying NP verification circuit, thus we cannot compose NIZKs as we were able to compose seBARGs.

To get around this issue, we additionally rely on rate-1 NIZKs. Gentry et al. [GGI+15] provided a generic template to build such composable (rate-1) NIZKAoK by combining fully homomorphic encryption and regular NIZKs. By using their compiler and plugging it in our multi-hop signature scheme, the composition issue is almost fixed. However, there is one last issue: the CRS size and verification time still grows in this recursive NIZK composition. Even for rate-1 NIZKs, the CRS size could grow with the statement and witness size, thus recursive composition leads to a blow-up in the verification circuit size. To handle this, we use another layer of RAM delegation to make composition of NIZK verification efficient. As in prior works [CJJ21a], the RAM delegation verifier computes the digest of the input, and assesses whether the transformation from the input hash is valid. To optimize the verifier's efficiency, we split the hash digest, and generate a short digest of the NIZK CRS in the setup stage. Thus, the verifier is no longer required generate the hash digest of the NIZK CRS during verification. Instead, the verifier only generates the digest of variable inputs, thereby ensuring verifier succinctness. For more details, we refer the reader to Section 6. We conclude the overview with the following.

Theorem 2.4 (informal, see Corollary 6.17)**.** Context-hiding multi-hop homomorphic signatures can be built assuming sub-exponential security of LWE.

2.3 Generalizing to multi-key homomorphic signatures

In multi-key homomorphic signatures the dataset can be signed using multiple different authorities. Namely, the setup algorithm now generates a set of public parameter pp, and a tuple of $(\mathsf{sk}, \mathsf{vk})$ for ℓ different users in the system. The signing algorithm uses sk_i to sign m_i, namely $\mathsf{Sign}(\mathsf{sk}_i, (i, m_i)) \to \sigma_i$. Note that we are assuming that authority i signs message i, that is fixed indexing at the first hop's input layer. The evaluation and verification algorithms remain unchanged.

The construction is nearly identical to our current construction, except we need to define homomorphic evaluation w.r.t. multiple signers. This can be easily handled by switching the NP statements at every input wire. While evaluating the signature for the first time, we will use vk_i corresponding to the appropriate user to check the validity of the associated signature. Now for evaluating an evaluated signature, we will use $\{\mathsf{vk}_i\}_i$ corresponding to the appropriate user(s) to check the validity of the associated evaluated signature. The security proof would stay the same, as by a hop-by-hop extraction we will extract a forgery on some m_i w.r.t. vk_i of some honest signer.

Although the focus of this work is on designing homomorphic signatures in the single-signer model, our results can be extended to the multi-signer model quite easily as we briefly discussed above. This leads to the first construction for multi-key homomorphic signatures with general composability, context-hiding, and fast verfiability from such wide variety of cryptographic assumptions.

3 Preliminaries

Notation. We will let PPT denote probabilistic polynomial-time. We denote the set of all positive integers up to n as $[n] \coloneqq \{1, \cdots, n\}$. Also, we use $[m, n]$ where $n \geq m$ to denote the set of all integers from m to n, i.e. $[m, n] \coloneqq \{m, \cdots, n\}$.

Throughout this paper, unless specified, all polynomials we consider are positive polynomials. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution D, $x \leftarrow D$ denotes an element x drawn from distribution D.

3.1 Puncturable (All-But-One) Signatures

Syntax. A puncturable (or all-but-one) signature (PSig) consists of the following polynomial time algorithms:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$**.** The probabilistic setup algorithm takes as input a security parameter λ and outputs a tuple of signing and verification keys $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{Setup\text{-}Punc}(1^\lambda, m^*) \to (\mathsf{sk}, \mathsf{vk})$**.** The probabilistic punctured setup algorithm takes as input a security parameter λ and a punctured message m^*, and outputs a tuple of signing and punctured verification keys $(\mathsf{sk}, \mathsf{vk})$.

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

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to 0/1$**.** The verification algorithm takes as input a verification key vk, a message m, and a signature σ. It outputs a bit to signal whether the signature is valid or not.

Definition 3.1 (Puncturable (or All-but-one) Signature)**.** A puncturable (or All-but-one) signature $\mathsf{PSig} = (\mathsf{PSig.Setup}, \mathsf{PSig.Setup\text{-}Punc}, \mathsf{PSig.Sign}, \mathsf{PSig.Verify})$ is required to satisfy the following properties:

Completeness. For all $\lambda \in \mathbb{N}$ and $m \in \{0,1\}^\lambda$ it holds that:

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 : (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)] = 1.$$

Punctured correctness. For all $\lambda \in \mathbb{N}$ and $m^* \in \{0,1\}^\lambda$ and $\sigma^* \in \{0,1\}^*$ it holds that:

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 : (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup\text{-}Punc}(1^\lambda, m^*)] = 0.$$

Verification Key Indistinguishability. For any adversary $\mathcal{A}$, any bit $b \in \{0,1\}$, define the verification key indistinguishability experiment $\mathsf{Exp}_{\mathsf{vk\text{-}ind},\mathcal{A}}(\lambda, b)$ as follows:

1. Given the security parameter λ, $\mathcal{A}$ sends $m^* \in \{0,1\}^\lambda$ to the challenger.
2. The Challenger samples $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{sk}_1, \mathsf{vk}_1) \leftarrow (1^\lambda, m^*)$ and sends vk_b to the adversary.
3. The adversary can make signing queries $m \in \{0,1\}^\lambda / \{m^*\}$ and receive $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_b, m)$.
4. The adversary outputs a bit $b' \in \{0,1\}$ which is the output of the experiment.

A puncturable signature construction satisfies verification key indistinguishability if for every $\lambda \in \mathbb{N}$, and any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t.

$$|\Pr[\mathsf{Exp}_{\mathsf{vk\text{-}ind},\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{Exp}_{\mathsf{vk\text{-}ind},\mathcal{A}}(\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda).$$

Theorem 3.2 ([GVW19])**.** Assuming LWE or k-LIN over pairing groups for any constant $k \in \mathbb{N}$, there exists all-but-one signatures.

Theorem 3.3 (Theorem B.2)**.** Assuming injective PRGs there exists all-but-one signatures for single-bit messages.

3.2 Non-Interactive Zero-Knowledge (NIZK) Arguments

Consider an NP language $\mathcal{L} = \{x \mid \exists w : \mathcal{R}(x, w) = 1\}$ defined w.r.t. a relation $\mathcal{R}$.

Syntax. A non-interactive zero-knowledge (NIZK) argument consists of the following polynomial time algorithms:

$\mathsf{Setup}(1^\lambda, 1^{n_x}) \to \mathsf{crs}$**.** The probabilistic setup algorithm takes as input a security parameter λ, a max instance length n_x, and outputs a common reference string crs.

$\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$**.** The prover algorithm takes as input a common reference string crs, an instance x, and a witness w and outputs a proof π.

$\mathsf{Verify}(\mathsf{crs}, x, \pi) \to 0/1$**.** The verifier algorithm takes as input a CRS crs, an instance x, and a proof π. It outputs a bit to signal whether the proof is valid or not.

Definition 3.4 (NIZK)**.** A non-interactive zero-knowledge proof $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ is required to satisfy the following properties:

Completeness. For all $\lambda, n_x \in \mathbb{N}$ and $(x, w) \in \mathcal{R}$ where $|x| = n_x$ we have:

$$\Pr[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{n_x}), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)] = 1.$$

Adaptive Soundness. For any PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, n_x \in \mathbb{N}$:

$$\Pr[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin \mathcal{L} : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{n_x}), (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}), |x| = n_x] \leq \mathsf{negl}(\lambda)$$

Zero-Knowledge. There exists a stateful PPT simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, n_x \in \mathbb{N}$:

$$\begin{aligned}|\Pr[\mathcal{A}^{\mathsf{Prove}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1 \ : \ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{n_x})] - \\ |\Pr[\mathcal{A}^{\mathcal{O}^{\mathcal{S}}(\cdot,\cdot)}(\mathsf{crs}) = 1 \ : \ \mathsf{crs} \leftarrow \mathcal{S}(1^\lambda, 1^{n_x})]| \leq \mathsf{negl}(\lambda)\end{aligned}$$

where $\mathcal{O}^{\mathcal{S}}(x, w)$ outputs $\mathcal{S}(x)$ if $x \in \mathcal{L}$ and $\perp$ otherwise.

Knowledge Extractor. There exists a stateful PPT extractor $\mathcal{E}$ such that for any non-uniform PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, n_x \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(\overline{\mathsf{crs}}, x, \pi) = 1 & (\overline{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathcal{E}(1^\lambda, 1^{n_x}), \\ \wedge\ \mathcal{R}(x, w) = 0 \ : & (x, \pi) \leftarrow \mathcal{A}(\overline{\mathsf{crs}}), \\ & |x| = n_x, \\ & w \leftarrow \mathcal{E}(\mathsf{td}, x, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

and $\overline{\mathsf{crs}}$ and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{n_x})$ are computationally indistinguishable.

Remark 3.5 ([CW23, BWW23, BKP+23])**.** Assuming seBARGs there exists NIZKs.

Definition 3.6 (Rate-1 NIZK)**.** A non-interactive zero-knowledge proof $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ is said to be a Rate-1 NIZKif it satisfies Definition 3.4 and the size of the proof π is $|w| + \mathsf{poly}(\lambda, \log |x|)$.

Theorem 3.7 ([GGI+15])**.** Assuming LWE there exists Rate-1 NIZKs for NP.

3.3 Hash Tree

Syntax. Syntax of hash tree is as the following:

$\mathsf{Setup}(1^\lambda) \to \mathsf{hk}$. The setup algorithm takes as input a security parameters λ, and outputs a hash key hk.

$\mathsf{Hash}(\mathsf{hk}, x) \to h$. The hash function takes as input a hash key hk, a input string $x \in \{0,1\}^N$, and outputs a hash value h, where $|h| = \mathsf{poly}(\lambda)$ for some universal polynomial $\mathsf{poly}(\cdot)$.

$\mathsf{Open}(\mathsf{hk}, x, i) \to \rho$. The opening algorithm takes as input a hash key hk, a input $x \in \{0,1\}^N$, an index $i \in [N]$, and outputs an opening ρ, where $|\rho| = \mathsf{poly}(\lambda, \log N)$ for some universal polynomial $\mathsf{poly}(\cdot, \cdot)$.

$\mathsf{Verify}(\mathsf{hk}, h, i, b, \rho) \to \{0,1\}$. The verifer algorithm takes as input a hash key hk, a hash value h, an index i, bit b, opening u, and outputs 0 or 1.

Definition 3.8. (Completeness). For every $\lambda, N \in \mathbb{N}$, $x \in \{0,1\}^N$, $i \in [N]$, the following holds:

$$\Pr\left[\mathsf{Verify}(\mathsf{hk}, h, i, x_i, \rho) = 1 \;:\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda) \\ h \leftarrow \mathsf{Hash}(\mathsf{hk}, x) \\ \rho \leftarrow \mathsf{Open}(\mathsf{hk}, x, i) \end{array}\right] = 1.$$

Definition 3.9. (Collision resistance). A Merkle Tree Hash scheme satisfies collision resistance if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{hk}, h, i, b, \rho) = 1 \\ \wedge\, \mathsf{Verify}(\mathsf{hk}, h, i, b', \rho') = 1 \\ \wedge\, b \neq b' \end{array} \;:\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda) \\ (h, i, b, b', \rho, \rho') \leftarrow \mathcal{A}(\mathsf{hk}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 3.10. ([Mer88]) Assuming existence of collision resistant hash family, there exists a hash tree as above.

3.4 Flexible RAM SNARGs with Partial Input Soundness

Syntax. A RAM delegation scheme Del for RAM machine $\mathcal{R}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, T) \to \mathsf{crs}$: The setup algorithm takes as input security parameter λ and running time bound T. It outputs CRS crs.

$\mathsf{Prove}(\mathsf{crs}, \mathsf{hk}, x_{\mathsf{exp}}, x_{\mathsf{imp}}) \to \pi$: The prover algorithm takes as input CRS crs, a hash key hk, an input $x = (x_{\mathsf{exp}}, x_{\mathsf{imp}})$, and outputs proof π.

$\mathsf{Digest}(\mathsf{hk}, x) \to h$: This is a deterministic polynomial time algorithm that takes as input a Hash Tree H key hk generated by $\mathsf{H.Setup}(1^\lambda)$, and outputs $h = \mathsf{H.Hash}(\mathsf{hk}, x)$.

$\mathsf{Verify}(\mathsf{crs}, h, x_{\mathsf{exp}}, \pi) \to \{0,1\}$: The verifier algorithm takes as input crs, a digest h, explicit input x_{exp}, and a proof π, and outputs either 0 or 1.

Completeness For every polynomial $N = N(\lambda)$, $T = T(\lambda)$, RAM machine $\mathcal{R}$, and input $x = (x_{\mathsf{exp}}, x_{\mathsf{imp}})$ such that $\mathcal{R}(x)$ accepts in T steps, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, h, x_{\mathsf{exp}}, \pi) = 1 \;:\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ \mathsf{hk} \leftarrow \mathsf{H.Setup}(1^\lambda) \\ (b, \pi) \leftarrow \mathsf{Prove}(\mathsf{crs}, x = (x_{\mathsf{exp}}, x_{\mathsf{imp}})) \\ h = \mathsf{Digest}(\mathsf{hk}, x_{\mathsf{imp}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Compactness In the above completeness experiment, $|\mathsf{crs}| \leq \mathsf{poly}(\lambda, \log T)$. Prover algorithm runs in time $\mathsf{poly}(\lambda, T)$ and outputs a proof of length $|\pi| \leq \mathsf{poly}(\lambda, \log T)$. Let n be the input size, verifier runs in time $\mathsf{poly}(\lambda, \log T, n)$.

Definition 3.11 (Partial Input Soundness)**.** For every polynomial $N = N(\lambda)$, $T = T(\lambda)$, RAM machine $\mathcal{R}$ that runs in time T, and every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{crs}, h, x_{\mathsf{exp}}, \pi) = 1 \\ \wedge\, h = \mathsf{H.Hash}(\mathsf{hk}, x_{\mathsf{imp}}) \\ \wedge\, \mathcal{R}(x_{\mathsf{exp}}, x_{\mathsf{imp}}) \text{ does not accept in } T \text{ steps} \end{array} : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ \mathsf{hk} \leftarrow \mathsf{H.Setup}(1^\lambda) \\ (x_{\mathsf{exp}}, x_{\mathsf{imp}}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We apply the above flexible RAM SNARG used in [KLVW23], [DGKV22]. The design is flexible with respect to the hash tree used to digest the memory. In particular, the RAM Machine takes two types of input: a variable input x_{exp} and a fixed input x_{imp}. The fixed input is considered as a constant string embedded in machine $\mathcal{R}$. Then, we have the hash key of the above design hk to be partitioned into $\mathsf{hk} = (\mathsf{hk}_{\mathsf{exp}}, \mathsf{hk}_{\mathsf{imp}})$ and the hash value to be partitioned into $h = (h_{\mathsf{exp}}, h_{\mathsf{imp}})$. By pre-processing h_{imp}, the running time of the verifier is $\mathsf{poly}(\lambda, \log T, |x_{\mathsf{exp}}|)$. The proof size is $\mathsf{poly}(\lambda, \log T)$. We also note that the RAM SNARG achieves a soundness notion of partial input soundness, which is stronger than the soundness results achieved by [CJJ21b].

Theorem 3.12 ([KLVW23])**.** Assuming seBARG and somewhere extractable hash family with local opening, there exists a flexible RAM SNARG.

3.5 Somewhere Extractable Batch Arguments

Syntax. A non-interactive batch argument (BARG) scheme BARG with respect to language $\mathcal{L}$ consists of the following polynomial time algorithms:

$\mathsf{Setup}(1^\lambda, 1^n, k) \to \mathsf{crs}$. The setup algorithm takes as input the security parameter λ, instance size n, number of instances k, and outputs a crs crs.

$\mathsf{Prove}(\mathsf{crs}, \{(x_i, \omega_i)\}_{i\in[k]}) \to \pi$. The prover algorithm takes as input a crs and a sequence of k instance-witness pairs (x_i, ω_i) for $i \in [k]$. It outputs a proof π.

$\mathsf{Verify}(\mathsf{crs}, \{x_i\}_{i\in[k]}, \pi) \to 0/1$. The verification algorithm takes as input a crs, a sequence of k instances x_i for $i \in [k]$, and a proof π. It outputs a bit to signal whether the proof is valid or not.

In this work, we rely on rate-1 somewhere extractable BARGs (rate-1 seBARGs) for language $\mathcal{L}$ which are defined as above, except the setup algorithm also takes a special index as an input. And, there exists an additional algorithm called $\mathsf{Extract}$ that extracts an accepting witness for the special index from any accepting batched proof. Below we provide the updated setup algorithm syntax along with the extraction algorithm.

$\mathsf{Setup}(1^\lambda, 1^n, k, i^*) \to (\mathsf{crs}, \mathsf{td})$. The setup takes an index $i^* \in [k]$ as an additional input, and outputs a trapdoor td as well.

$\mathsf{Extract}(\mathsf{td}, \{x_i\}_i, \pi) \to \omega$. The extraction algorithm takes as input the trapdoor td, k instances $\{x_i\}_i$, proof π, and outputs an extracted witness ω.

Correctness and succinctness. An rate-1 seBARG is said to be correct and succinct if for every $\lambda, k \in \mathbb{N}$, index $i^* \in [k]$, setup parameters $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, k, i^*)$, any k instances $x_1, \ldots, x_k \in \mathcal{L} \cap \{0,1\}^n$ and their corresponding witnesses ω_i for $i \in [k]$, and every proof $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \{(x_i, \omega_i)\}_i)$, the following holds:

Completeness. $\mathsf{Verify}(\mathsf{crs}, \{x_i\}_i, \pi) = 1$.

Extraction correctness. $\mathsf{Extract}(\mathsf{td}, \{x_i\}_i, \pi) = \omega_{i^*}$.

Succinctness. $|\pi| \le \mathsf{poly}(\lambda) + m$. That is, the size of the batched proof is bounded by a fixed polynomial in λ plus m where m is length of one witness.

Soundness. A BARG scheme is said to be sound if an attacker can not create a valid proof where one of the k instances being batch-proved do not belong to the language $\mathcal{L}$. For seBARGs, this can be indirectly captured by the following two properties.

Definition 3.13 (index hiding)**.** A somewhere extractable batch argument scheme seBARG satisfies index hiding if for every stateful PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\begin{array}{ll} \mathcal{A}(\mathsf{crs}) = b & (k, n, i_0^*, i_1^*) \leftarrow \mathcal{A}(1^\lambda),\ b \leftarrow \{0,1\} \\ \wedge\ i_0^*, i_1^* \in [k] & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, k, i_b^*) \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda).$$

Definition 3.14 (somewhere argument of knowledge)**.** A somewhere extractable batch argument scheme seBARG is a somewhere argument of knowledge if for every stateful PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\begin{array}{ll} \mathsf{Verify}(\mathsf{crs}, \{x_i\}_i, \pi) = 1 \wedge\ i^* \in [k] & (k, n, i^*) \leftarrow \mathcal{A}(1^\lambda) \\ \wedge\ \omega^* \text{ is not a valid witness for } x_{i^*} \in \mathcal{L} & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, k, i^*) \\ & (\{x_i\}_{i\in[k]}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ & \omega^* \leftarrow \mathsf{Extract}(\mathsf{td}, \{x_i\}_i, \pi) \end{array}\right] \le \mathsf{negl}(\lambda).$$

Remark 3.15. In our approach, the batch argument seBARG can be extracted across multiple indices, where we override the original Setup Algorithm as $\mathsf{Setup}(1^\lambda, \mathcal{L}, k, (i_0, \ldots, i_\alpha))$. The Setup algorithm generates α batch argument common reference strings, each corresponding to an individual index.

3.6 SNARGs for Monotone-Policy BatchNP (Monotone-Policy BARGs)

We will first define the monotone-policy batchNP language $\mathcal{L}_{\mathsf{MP\text{-}CSAT}}$ and then define SNARGs for this language (monotone-policy BARGs).

Definition 3.16 (Monotone Policy BatchNP)**.** A Boolean circuit $\tilde{C} : \{0,1\}^k \to \{0,1\}$ is a monotone Boolean policy if $\tilde{C}$ is a Boolean circuit comprised entirely of AND and OR gates. Let $\mathcal{R} : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean relation and $\tilde{C} : \{0,1\}^k \to \{0,1\}$ be a monotone Boolean policy. Define the monotone policy BatchNP language $\mathcal{L}_{\mathsf{MP\text{-}CSAT}}$ to be:

$$\mathcal{L}_{\mathsf{MP\text{-}CSAT}} = \left\{ (\mathcal{R}, \tilde{C}, x_1, \cdots, x_k) \ : \ \begin{array}{l} x_1, \cdots, x_k \in \{0,1\}^n, \exists w_1, \cdots, w_k \in \{0,1\}^h : \\ \tilde{C}(\mathcal{R}(x_1, w_1), \cdots, \mathcal{R}(x_k, w_k)) = 1 \end{array} \right\}.$$

Syntax. A SNARG for monotone-policy batchNP (monotone-policy BARG) scheme BARG for language $\mathcal{L}_{\mathsf{MP\text{-}CSAT}}$ consists of the following polynomial time algorithms:

$\mathsf{Gen}(1^\lambda, 1^n, 1^{\mathsf{s}_\mathcal{R}}, 1^{\mathsf{s}_{\tilde{C}}}) \to \mathsf{crs}$. The setup algorithm takes as input the security parameter λ, the instance size n, the bound on the relation (circuit) size $1^{\mathsf{s}_\mathcal{R}}$, and a bound on the monotone circuit $1^{\mathsf{s}_{\tilde{C}}}$, and outputs a crs crs.

$\mathsf{Prove}(\mathsf{crs}, \mathcal{R}, \tilde{C}, (x_1, \cdots, x_k), (w_1, \cdots, w_k)) \to \pi$. The prover algorithm takes as input a crs, an NP relation $\mathcal{R}$, a monotone circuit $\tilde{C}$, a sequence of statements $(x_1, \cdots, x_k)$, and a sequence of witnesses $(w_1, \cdots, w_k)$, and it outputs a proof π.

$\mathsf{Verify}(\mathsf{crs}, \mathcal{R}, \tilde{C}, (x_1, \cdots, x_k), \pi) \to 0/1$. The verification algorithm takes as input a crs, an NP relation $\mathcal{R}$, a monotone circuit $\tilde{C}$, a sequence of statements $(x_1, \cdots, x_k)$, and a proof π. It outputs a bit to signal whether the proof is valid or not.

Definition 3.17 (SNARGs for Monotone-Policy BatchNP (Monotone-Policy BARGs))**.** A monotone-policy BARG $\mathsf{BARG} = (\mathsf{BARG.Gen}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify})$ for $\mathcal{L}_{\mathsf{MP\text{-}CSAT}}$ is required to satisfy the following properties:

Completeness. For all $\lambda, n, \mathsf{s}_\mathcal{R}, \mathsf{s}_{\tilde{C}} \in \mathbb{N}$, all Boolean relations $\mathcal{R} : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most $\mathsf{s}_\mathcal{R}$, all monotone Boolean circuits $\tilde{C} : \{0,1\}^k \to \{0,1\}$ of size at most $\mathsf{s}_{\tilde{C}}$, all statements $x_1, \cdots, x_k \in \{0,1\}^n$ and witnesses $w_1, \cdots, w_k \in \{0,1\}^h$ where $\tilde{C}(\mathcal{R}(x_1, w_1), \cdots, (x_k, w_k)) = 1$, it holds that

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathcal{R}, \tilde{C}, (x_1, \cdots, x_k), \pi) = 1 : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, 1^n, 1^{\mathsf{s}_\mathcal{R}}, 1^{\mathsf{s}_{\tilde{C}}}) : \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathcal{R}, \tilde{C}, (x_1, \cdots, x_k), (w_1, \cdots, w_k)) \end{array}\right] = 1.$$

Succinctness. There exists a fixed polynomial $\mathsf{poly}(\cdot)$ s.t. for all $\lambda, n, \mathsf{s}_\mathcal{R}, \mathsf{s}_{\tilde{C}} \in \mathbb{N}$, all crs in the support of $\mathsf{Gen}(1^\lambda, 1^n, 1^{\mathsf{s}_\mathcal{R}}, 1^{\mathsf{s}_{\tilde{C}}})$, all Boolean relations $\mathcal{R}\{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most $\mathsf{s}_\mathcal{R}$, all monotone Boolean circuits $\tilde{C} : \{0,1\}^k \to \{0,1\}$ of size at most $\mathsf{s}_{\tilde{C}}$, the proof π output by $\mathsf{Prove}(\mathsf{crs}, \mathcal{R}, \tilde{C}, \cdot, \cdot)$ satisfies $\pi \leq \mathsf{poly}(\lambda, \mathsf{s}_\mathcal{R}, \log|\tilde{C}|)$.

Non-adaptive soundness. For any adversary $\mathcal{A}$, define the non-adaptive soundness game as follows:

1. Given the security parameter 1^λ,$\mathcal{A}$ outputs $1^n, 1^{\mathsf{s}_\mathcal{R}}, 1^{\mathsf{s}_{\tilde{C}}}$, $\mathcal{R}$ of size at most $\mathsf{s}_\mathcal{R}$, $\tilde{C}$ of size at most $\mathsf{s}_{\tilde{C}}$, and statements $x_1, \cdots, x_k \in \{0,1\}^n$.
2. The challenger sends $\mathcal{A}$ a sampled $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, 1^n, 1^{\mathsf{s}_\mathcal{R}}, 1^{\mathsf{s}_{\tilde{C}}})$.
3. $\mathcal{A}$ outputs a proof π.
4. The output of the game is $b = 1$ if $\mathsf{Verify}(\mathsf{crs}, \mathcal{R}, \tilde{C}, (x_1, \cdots, x_k), \pi) = 1$ and $(\mathcal{R}, \tilde{C}, x_1, \cdots, x_k) \notin \mathcal{L}_{\mathsf{MP\text{-}CSAT}}$.

We say that a monotone-policy BARG is non-adaptively sound if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. $\Pr[b = 1] \leq \mathsf{negl}(\lambda)$ in the non-adaptive soundness game above.

Theorem 3.18 ([NWW23])**.** Assuming either LWE, k-LIN over pairing groups for any constant $k \in \mathbb{N}$, or sub-exponential DDH over pairing-free groups, there exists monotone-policy BARGs for any polynomial depth monotone circuit $\tilde{C}$, where $|\mathsf{crs}| = \mathsf{poly}(\lambda)$, and $|\pi| = \mathsf{poly}(\lambda, \mathsf{s}_\mathcal{R}, \log \mathsf{s}_{\tilde{C}})$.

4 Single-Hop Homomorphic Signatures For General Circuits

In this section, we construct single-hop homomorphic signatures from standard assumptions, i.e., LWE, k-LIN over pairing groups for any constant $k \in \mathbb{N}$, and sub-exponential DDH over pairing-free groups.

4.1 Definition

In what follows we recall the definition of a single-hop homomorphic signature.

Syntax. A single-hop homomorphic signature Sig consists of the following polynomial time algorithms:

$\mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \to (\mathsf{pk}, \mathsf{sk})$**.** This is a probabilistic setup algorithm that takes as input a security parameter 1^λ in unary, a dataset size 1^k, and a max circuit size s_C. It outputs a public (verification and evaluation) key pk along with a signing key sk.

$\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$**.** This is a probabilistic signing algorithm that takes as input a signing key sk, a dataset index i, and a single-bit message m_i. It outputs a signature σ_i.

$\mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C) \to \sigma$**.** This is an evaluation algorithm that takes as input a public key pk, a dataset-signatures tuple $(m_i, \sigma_i)_{i\in[k]}$, and an evaluation circuit C. It outputs a signature σ of the evaluated message.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, C) \to 0/1$**.** This is a verification algorithm that takes as input a public key pk, a message y (either an original message $y = (i, m_i)$ or an evaluated message y), and an evaluation circuit C (potentially $C = \emptyset$ in case $y = (i, m_i)$). It outputs a single bit 1 (accept) or 0 (reject).

Definition 4.1 (Single-Hop Homomorphic Signature)**.** A single-hop homomorphic signature $\mathsf{HSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ is required to satisfy the following properties:

Correctness. For all $\lambda, k, \mathsf{s}_C \in \mathbb{N}$, all dataset $m \in \{0,1\}^k$, it holds that

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, (i, m_i), \sigma, \emptyset) = 1 \ : \ \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, i, m_i) \end{array}\right] = 1$$

and for all polynomial size circuits $C : \{0,1\}^k \to \{0,1\}$, it holds that:

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, C(m), \sigma, C) = 1 \ : \ \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \\ \forall i \in [k],\ \sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, i, m_i) \\ \sigma \leftarrow \mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C) \end{array}\right] = 1$$

Succinctness. There exists a fixed polynomial $\mathsf{poly}(\cdot)$ s.t. for all $\lambda, k, \mathsf{s}_C \in \mathbb{N}$, all dataset $m \in \{0,1\}^k$, and all polynomial size circuits $C : \{0,1\}^k \to \{0,1\}^{\mathsf{p}(\lambda)}$ (for some polynomial $\mathsf{p}(\cdot)$) of size at most s_C and depth d_C, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C})$ and signatures $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, i, m_i)$ for $i \in [k]$, it holds that the evaluated signature $\sigma \leftarrow \mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C)$ has size at most $\mathsf{poly}(\lambda, \log k, \log \mathsf{s}_C, d_C)$.

Selective Unforgeability. For any adversary $\mathcal{A}$ define the selective unforgeability experiment $\mathsf{Exp}_{\mathsf{SU},\mathcal{A}}(\lambda)$ as follows:

- Given the security parameter λ, $\mathcal{A}$ outputs the size of dataset 1^k, the circuit bound s_C, the dataset $m \in \{0,1\}^k$, and a forgery target (C^*, y^*).
- The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C})$.
- The challenger outputs pk and signatures $(\sigma_1, \ldots, \sigma_k)$ where $\sigma_i = \mathsf{Sign}(\mathsf{sk}, i, m_i)$.
- $\mathcal{A}$ outputs a forgery σ^*.
- The output of the experiment is 1 if $\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, C^*) = 1$ and either (1) $C^*(m) \neq y^*$, or (2) $C^* = \emptyset$ and $y^* = (i, m_i \oplus 1)$, otherwise the output is 0.

A construction satisfies selective security if for any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. for any $\lambda \in \mathbb{N}$ it holds that $\Pr[\mathsf{Exp}_{\mathsf{SU},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

The below context hiding property additionally requires a simulator $\mathcal{S}$.

Definition 4.2 (Context Hiding)**.** A single-hop homomorphic signature scheme satisfies context-hiding if there exist a stateful PPT simulator $\mathcal{S}$ such that for every stateful PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, k, \mathsf{s}_C \in \mathbb{N}$, the following holds:

$$\Pr\left[\begin{array}{l} \\ \mathcal{A}(\sigma_b) = b \ \wedge |C| = \mathsf{s}_C \ \wedge \\ \mathsf{Verify}(\mathsf{pk}_b, C(m), \sigma_0, C) = 1 \\ \\ \end{array} \ : \ \begin{array}{l} b \leftarrow \{0,1\},\ (\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \\ (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{S}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \\ ((m_i, \sigma_i)_{i\in[k]}, C) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_b, \cdot, \cdot)}(\mathsf{pk}_b) \\ \sigma_0 \leftarrow \mathsf{Eval}(\mathsf{pk}_b, (m_i, \sigma_i)_{i\in[k]}, C) \\ \sigma_1 \leftarrow \mathcal{S}(C(m), C) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

$\underline{\mathcal{T}(C, y) \to \tilde{C}_y}$

1. Let C be a circuit that takes as input a message m of size k. Define C_y to be a single-bit output circuit that takes k bits of input and has y hard-wired in it, and check whether the computation of C on its input matches the hard-wired value y. Namely:

$$C_y(m) := \mathbb{1}[C(m) = y].$$

2. Define circuit $\tilde{C}_y$ to be the monotone circuit that computes $C_y(m)$ given $(m, m \oplus 1^k)$ as input. Namely:

$$\tilde{C}_y(m, m \oplus 1^k) := C_y(m).$$

The general to monotone circuit transformation is discussed in Appendix A.

Remark 4.3. Let s_C be the size of the circuit C. Then $\mathsf{s}_{C_y} = \mathsf{s}_C + \mathsf{poly}(y)$ and $\mathsf{s}_{\tilde{C}_y} = 2\mathsf{s}_{C_y}$.

Figure 1: Construction of monotone Circuit $\tilde{C}_y$ from a general circuit C and an output y s.t. $\tilde{C}_y(m, m \oplus 1^{|m|}) = 1$ iff $C(m) = y$.

4.2 Construction

Below we describe our construction of Single-Hop Homorphic Signatures.

Construction 4.4. [Single-Hop Homomorphic Signatures for General Computation Circuits] Let $\mathsf{BARG} = (\mathsf{BARG.Gen}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify})$ be a monotone-policy BARG scheme and $\mathsf{PSig} = (\mathsf{PSig.Setup}, \mathsf{PSig.Setup\text{-}Punc}, \mathsf{PSig.Sign}, \mathsf{PSig.Verify})$ be an all-but-one signature scheme for single-bit messages. We construct a single-hop homomorphic signature for general computation circuits $\mathsf{HSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \to (\mathsf{pk}, \mathsf{sk})$**.** It runs $(\mathsf{sk}_{\mathsf{PSig},i}, \mathsf{vk}_{\mathsf{PSig},i}) \leftarrow \mathsf{PSig.Setup}(1^\lambda)$ for $i \in [k]$, and $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{\log 2k}, 1^{\mathsf{s}_\mathcal{R}}, 1^{\mathsf{s}_{\tilde{C}_y}})$ (where $\mathsf{s}_{\tilde{C}_y}$ is from Remark 4.3, and $\mathsf{s}_\mathcal{R}$ is the size of the circuit for the relation in Item 3.) and lets $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PSig},i})_{i\in[k]}$, $\mathsf{pk} = ((\mathsf{vk}_{\mathsf{PSig},i})_{i\in[k]}, \mathsf{crs}_{\mathsf{BARG}})$.

$\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$**.** It parses $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PSig},j})_{j\in[k]}$ and then it computes the signature $\sigma_i \leftarrow \mathsf{PSig.Sign}(\mathsf{sk}_{\mathsf{PSig},i}, m_i)$.

$\mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C) \to \sigma$**.** This poly-time algorithm does the following:

1. Parse $\mathsf{pk} = ((\mathsf{vk}_{\mathsf{PSig},i})_{i\in[k]}, \mathsf{crs}_{\mathsf{BARG}})$.
2. Let $y = C(m)$ and construct monotone circuit $\tilde{C}_y = \mathcal{T}(C, y)$ using Fig. 1.
3. Compute $\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \tilde{C}_y, (x_i)_{i\in[2k]}, (w_i)_{i\in[2k]})$ where
   - For $i \in [k]$, $x_i = (i, \mathsf{vk}_{\mathsf{PSig},i})$ and $w_i = \sigma_i$.
   - For $i \in [k+1, 2k]$, $x_i = (i, \mathsf{vk}_{\mathsf{PSig},i-k})$ and $w_i = \sigma_{i-k}$.

   for the NP relation $\mathcal{R}$, where $(x, w) \in \mathcal{R}$ and $x = (i, x')$ if one of the following holds:
   - $i \in [k]$, and $\mathsf{PSig.Verify}(x', 1, w) = 1$, or
   - $i \in [k+1, 2k]$, and $\mathsf{PSig.Verify}(x', 0, w) = 1$.
4. Output $\sigma = \pi_{\mathsf{BARG}}$.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, C) \to 0/1$**.** Parse $\mathsf{pk} = ((\mathsf{vk}_{\mathsf{PSig},i})_{i\in[k]}, \mathsf{crs}_{\mathsf{BARG}})$, if $C = \emptyset$, parse $y = (i, m_i)$ and run $0/1 \leftarrow \mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i}, m_i, \sigma)$, otherwise construct $\tilde{C}_y = \mathcal{T}(C, y)$ (using Fig. 1), and then run the BARG verification $0/1 \leftarrow \mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \tilde{C}_y, (x_i)_{i\in[2k]}, \sigma)$ (where x_i and $\mathcal{R}$ are defined in Item 3).

Theorem 4.5 (Correctness.)**.** If BARG is complete PSig is correct, then the homomorphic signature from construction 4.4 is complete.

Proof. Let $\sigma_i \leftarrow \mathsf{PSig}(\mathsf{sk}_{\mathsf{PSig},i}, m_i)$ for $i \in [k]$. For any message y, signature σ, and circuit C, if $C = \emptyset$, then the correctness follows directly from the correctness of PSig. Otherwise, if $y = C(m)$ then by the construction in Fig. 1 it holds that $\tilde{C}_y(m, m \oplus 1^k) = 1$. Note that for any $i \in [k]$ such that $m_i = 1$ it holds that $\mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i}, 1, \sigma_i) = 1$ and for any $i \in [k+1, 2k]$ such that $m_i \oplus 1 = 1$ it holds that $\mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i-k}, 0, \sigma_{i-k}) = 1$ both by the correctness of PSig. Hence it holds that $\tilde{C}_y((\mathcal{R}(x_i, w_i))_{i\in[2k]}) = 1$. Therefore, by the completeness of BARG it holds that $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \tilde{C}_y, (x_i)_{i\in[2k]}, \sigma) = 1$. □

Theorem 4.6 (Succinctness.)**.** If BARG is succinct, then the homomorphic signature from construction 4.4 satisfies $|\sigma| \leq \mathsf{poly}(\lambda, \log|C|)$.

Proof. The evaluated signature is just a BARG proof and since the relation $\mathcal{R}$ is just a regular signature verification, hence $\mathsf{s}_{\mathcal{R}} = \mathsf{poly}(\lambda)$, therefore $|\sigma| = |\pi_{\mathsf{BARG}}| = \mathsf{poly}(\lambda, \log|\tilde{C}|)$ □

Theorem 4.7 (Selective Unforgeability.)**.** If BARG is non-adaptively sound and PSig satisfied punctured correctness, then the homomorphic signature from construction 4.4 is selectively unforgeable.

Proof. Consider adversary $\mathcal{A}$ that breaks the unforgeability of the construction 4.4. First, we construct the following hybrids:

- hyb_i**.** For $i \in \{0, \ldots, k\}$ we define hyb_i as follows:
  - Given the security parameter λ, $\mathcal{A}$ outputs the size of dataset 1^k, the circuit bound s_C, the dataset of messages $(m_1, \ldots, m_k)$, and a forgery target (C^*, y^*).
  - The challenger runs the setup as described in the construction except that it uses $(\mathsf{sk}_{\mathsf{PSig},j}, \mathsf{vk}_{\mathsf{PSig},j}) \leftarrow \mathsf{PSig.Setup\text{-}Punc}(1^\lambda, (m_j \oplus 1))$ to generate $(\mathsf{sk}_{\mathsf{PSig},j}, \mathsf{vk}_{\mathsf{PSig},j})$ for $j \in [i]$.
  - The challenger outputs pk and signatures $(\sigma_1, \ldots, \sigma_k)$ where $\sigma_i = \mathsf{PSig.Sign}(\mathsf{sk}_{\mathsf{PSig},i}, m_i)$.
  - $\mathcal{A}$ outputs a forgery σ^*.
  - The output of the experiment is 1 if $\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, C^*) = 1$ and either (1) $C^*(m) \neq y^*$, or (2) $C^* = \emptyset$ and $y^* = (i, m_i \oplus 1)$, otherwise the output is 0.

Note that hyb_0 is the original experiment and hyb_k is where $(\mathsf{sk}_{\mathsf{PSig},j}, \mathsf{vk}_{\mathsf{PSig},j})_{j\in[k]}$ is generated using $\mathsf{PSig.Setup\text{-}Punc}(1^\lambda, m_j)$. Let $\mathsf{hyb}_i(\mathcal{A})$ denote the output of the experiment in hybrid hyb_i when run on adversary $\mathcal{A}$. We want to prove that for any computationally bounded adversary $\mathcal{A}$, (1) the outputs of any two hybrids hyb_{i-1} and hyb_i for $i \in [k]$ are indistinguishable, and (2) the output of hybrid hyb_k is 0 with all but negligible probability.

Lemma 4.8. If PSig satisfies key-indistinguishability, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_i(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $|\Pr[\mathsf{hyb}_{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_i(\mathcal{A}) = 1]| \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}_{\mathsf{vk\text{-}ind}}$ against the key indistinguishability of the puncturable signature as follows:

1. Run adversary $\mathcal{A}$ and receive $(\mathsf{s}_C, 1^k, m_1, \ldots, m_k, C^*, y^*)$.
2. Send $(m_i \oplus 1)$ to the challenger and receive $\mathsf{vk}_{\mathsf{PSig},i}$.
3. Send the signing query m_i to the challenger and receive σ_i.
4. Sample $(\mathsf{sk}_{\mathsf{PSig},j}, \mathsf{vk}_{\mathsf{PSig},j}) \leftarrow \mathsf{PSig.Setup\text{-}Punc}(1^\lambda, m_j \oplus 1)$ for $j \in [i-1]$ and $(\mathsf{sk}_{\mathsf{PSig},j}, \mathsf{vk}_{\mathsf{PSig},j}) \leftarrow \mathsf{PSig.Setup}(1^\lambda)$ for $j \in [i+1, k]$.
5. Compute $\sigma_j \leftarrow \mathsf{PSig.Sign}(\mathsf{sk}_{\mathsf{PSig},j}, m_j)$ for $j \in [k]/i$.
6. Sample $\mathsf{crs}_{\mathsf{BARG}}$ according to the setup algorithm.
7. Send $\mathsf{pk} = ((\mathsf{vk}_{\mathsf{PSig},i})_{i\in[k]}, \mathsf{crs}_{\mathsf{BARG}})$ and the set of signatures $(\sigma_1, \ldots, \sigma_k)$ to $\mathcal{A}$.
8. Receive forgery σ^* from $\mathcal{A}$.
9. Output 1 if $\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, C^*) = 1$ and either (1) $C^*(m_1, \ldots, m_k) \neq y^*$, or (2) $C^* = \emptyset$ and $y* = (i, m_i \oplus 1)$, otherwise output 0.

Note that $\mathcal{B}_{\mathsf{vk\text{-}ind}}$ perfectly simulates

- hybrid hyb_{i-1} if the challenger uses $(\mathsf{sk}_{\mathsf{PSig},i}, \mathsf{vk}_{\mathsf{PSig},i}) \leftarrow \mathsf{PSig.Setup}(1^\lambda)$, or
- hybrid hyb_i if the challenger uses $(\mathsf{sk}_{\mathsf{PSig},i}, \mathsf{vk}_{\mathsf{PSig},i}) \leftarrow \mathsf{PSig.Setup\text{-}Punc}(1^\lambda, m_i \oplus 1)$,

for adversary $\mathcal{A}$. Thus the advantage of $\mathcal{B}_{\mathsf{vk\text{-}ind}}$ is also $\epsilon(\lambda)$ which breaks the key-indistinguishability of puncturable signature. □

Lemma 4.9. If BARG is non-adaptively sound and PSig is correct w.r.t. punctured keys, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr[\mathsf{hyb}_k(\mathcal{A}) = 1] \leq \mathsf{negl}(\lambda)$$

Proof. Suppose the forgery is of the second type, namely $\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, C^*) = 1$ and $C^* = \emptyset$ and $y^* = (i, m_i \oplus 1)$. The signature verification in this case implies that $\mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i}, m_i \oplus 1, \sigma^*) = 1$, but this contradicts the punctured correctness as $\mathsf{vk}_{\mathsf{PSig},i}$ is punctured at $m_i \oplus 1$, and hence we should have $\mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i}, m_i \oplus 1, \sigma') = 0$ for any σ', i.e., $\Pr[\mathsf{hyb}_k(\mathcal{A}) = 1] = 0$.

Now suppose the forgery is of the first type, namely $\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, C^*) = 1$ and $C^*(m) \neq y^*$. Additionally, towards the contradiction suppose that $\Pr[\mathsf{hyb}_k(\mathcal{A}) = 1] \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}_{\mathsf{BARG}}$ against the non-adaptive soundness of BARG as follows:

1. Run adversary $\mathcal{A}$ and receive $(\mathsf{s}_C, 1^k, m_1, \ldots, m_k, C^*, y^*)$.
2. Sample $(\mathsf{sk}_{\mathsf{PSig},i}, \mathsf{vk}_{\mathsf{PSig},i}) \leftarrow \mathsf{PSig.Setup\text{-}Punc}(1^\lambda, m_i \oplus 1)$ for $i \in [k]$.
3. Construct $\tilde{C}_y = \mathcal{T}(C^*, y^*)$ using Fig. 1.

4. Consider the NP relation $\mathcal{R}$ defined in *Item* 3 of the evaluation's algorithm.
5. Send $(1^{\log 2k}, 1^{\mathsf{s}_{\mathcal{R}}}, 1^{\mathsf{s}_{\tilde{C}_y}}, \mathcal{R}, \tilde{C}_y)$ to the challenger and receive $\mathsf{crs}_{\mathsf{BARG}}$.
6. Compute the set of signatures $(\sigma_1, \ldots, \sigma_k)$ where $\sigma_i = \mathsf{PSig.Sign}(\mathsf{sk}_{\mathsf{PSig},i}, m_i)$
7. Let $\mathsf{pk} = ((\mathsf{vk}_{\mathsf{PSig},i})_{i\in[k]}, \mathsf{crs}_{\mathsf{BARG}})$ and send it to $\mathcal{A}$ together with $(\sigma_1, \ldots, \sigma_k)$.
8. Receive forgery σ^* from $\mathcal{A}$ and forward it to the challenger.

First note that the algorithm $\mathcal{B}_{\mathsf{BARG}}$ perfectly simulates the challenger of hyb_k for $\mathcal{A}$. Now by the assumption $\mathcal{A}$ wins with probability at least ϵ which means:

$$\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, C^*) = 1 \quad \wedge \quad C^*(m_1, \ldots, m_k) \neq y^*.$$

Now the above statement implies that:

$$\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \tilde{C}_y, (x_i)_{i\in[2k]}, \sigma^*) = 1$$

(where x_i is defined in Item 3 of the evaluation's algorithm) and

$$\tilde{C}_y(m_1, \ldots, m_k, m_1 \oplus 1, \ldots, m_k \oplus 1) = 0.$$

Now if it holds that $\tilde{C}_y((\mathcal{R}(x_i, w_i))_{i\in[2k]}) = 0$ then $\mathsf{BARG.Verify}(\cdot) = 1$ implies that $\mathcal{B}_{\mathsf{BARG}}$ breaks the non-adaptive soundness of BARG. Thus since $\tilde{C}_y$ is monotone, we only need to prove that $\mathcal{R}(x_i, w_i) \leq m_i$ for $i \in [k]$ and $\mathcal{R}(x_i, w_i) \leq m_i \oplus 1$ for $i \in [k+1, 2k]$. This is implied by punctured correctness of PSig since for any σ' it holds that $\mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i}, 1, \sigma') = 0$ if $m_i = 0$ and $\mathsf{PSig.Verify}(\mathsf{vk}_{\mathsf{PSig},i}, 0, \sigma') = 0$ if $m_i \oplus 1 = 0$. Therefore $\mathcal{B}_{\mathsf{BARG}}$ has the same advantage $\epsilon(\lambda)$ as $\mathcal{A}$ in breaking the non-adaptive soundness of the BARG scheme. □

We conclude the proof by lemmas 4.8 and 4.9 and the hybrid argument. □

Theorem 4.10. The Construction 4.4 is an unforgeable single-hop homomorphic signature with evaluated signature size $\mathsf{poly}(\lambda, \log |C|)$, assuming monotone-policy BARGs and all-but-one signatures for single-bit messages.

Proof. We conclude the proof by combining Theorems 4.5 to 4.7. □

Corollary 4.11. The Construction 4.4 is an unforgeable single-hop homomorphic signature with evaluated signature size $\mathsf{poly}(\lambda, \log |C|)$, assuming either LWE, k-LIN over pairing groups for any constant $k \in \mathbb{N}$, or sub-exponential DDH over pairing-free groups.

Proof. We conclude the proof by combining Theorems 3.2, 3.18 and 4.10 and Corollary B.4. □

4.3 Context-Hiding

Here we describe how to achieve context-hiding homomorphic signatures using NIZK and common split-verification tricks [CJJ21b, WW22, CGJ+23, CW23]. We first describe split-verification for BARGs and discuss the existence of seBARGs with split-verification. Then we will take the monotone-policy BARGs construction from [NWW23] and show if the underlying seBARGs satisfies split-verification, then their monotone-policy BARGs construction also satisfies split-verification. Finally, we show how to combine NIZKs and monotone-policy BARGs with split-verification to get context-hiding homomorphic signatures.

Definition 4.12 (Split-Verification for BARGs)**.** We say that a BARG scheme for $(\hat{x}_1, \ldots, \hat{x}_k)$ where $\hat{x}_i = (x_i, x_{\mathsf{c}})$[8] for $i \in [k]$ satisfies split-verification if the Verify algorithm can be split into the following:

$\mathsf{PreVerify}(\mathsf{crs}, \mathbf{x} = (x_1, \ldots, x_k)) \to \mathsf{dig}_{\mathbf{x}}$**.** A pre-verification algorithm that takes as input the crs and statements $\mathbf{x} = (x_1, \ldots, x_k)$, runs in $\mathsf{poly}(\lambda, |\mathbf{x}|)$, and outputs a short digest of the statements $\mathsf{dig}_{\mathbf{x}}$ of size $\mathsf{poly}(\lambda, \log k, |x_1|)$.

$\mathsf{OnlineVerify}(\mathsf{crs}, \mathcal{R}, (\mathsf{dig}_{\mathbf{x}}, x_{\mathsf{c}}), \pi) \to 0/1$**.** An online verification algorithm which takes as input the crs, the digest of inputs $\mathsf{dig}_{\mathbf{x}}$, the common input x_{c}, the NP relation $\mathcal{R}$ (as a circuit), and the proof π, runs in $\mathsf{poly}(\lambda, \log k, |\mathcal{R}|)$, and outputs 1 (accepts) or 0 (rejects).

Theorem 4.13 ([CJJ21b])**.** Somewhere-extractable BARGs with split verification can be generically constructed from any somewhere-extractable BARG.

Proof Sketch. Consider statements $(\hat{x}_1, \ldots, \hat{x}_k)$ where $\hat{x}_i = (x_i, x_{\mathsf{c}})$ and witnesses $(w_1, \ldots, w_k)$ and let BARG be a bath argument. We construct BARG' with split virification using BARG and a hash tree with local openings HT. The prover first computes a hash $\mathsf{dig}_{\mathbf{x}}$ of $(x_1, \ldots, x_k)$ and their corresponding local openings $(\mathsf{op}_1, \ldots, \mathsf{op}_k)$. Then it computes a BARG proof π for statements $(x'_i = \mathsf{dig}_{\mathbf{x}}, x_{\mathsf{c}}, i)_{i \in [k]}$ and witnesses $(w' = w_i, \mathsf{op}_i, x_i)_{i \in [k]}$ such that w'_i is a valid witness for x'_i if op_i is a valid opening of $\mathsf{dig}_{\mathbf{x}}$ to x_i, and w_i is a valid witness for (x_i, x_{c}). Now the pre-verification algorithm first computes $\mathsf{dig}_{\mathbf{x}}$ given $(x_1, \ldots, x_k)$ and the online verification given $(\mathsf{dig}_{\mathbf{x}}, x_{\mathsf{c}})$ and π runs the BARG verification on $(x'_i)_{i \in [k]}$ and π. The proof follows from the security of HT and BARG. □

Remark 4.14 (Monotone-Policy BARGs with Split-Verification)**.** Note that the prover in [NWW23] first computes two digest values dig_0 and dig_1, then w.r.t. the digest values it computes a seBARG proof $\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}_{\mathsf{dig}_0, \mathsf{dig}_1}, (x_1, \ldots, x_k), (w_1, \ldots, w_k))$ (where $\mathcal{R}_{\mathsf{dig}_0, \mathsf{dig}_1}$ has dig_0 and dig_1 hard-coded), and lets the final proof be $\pi = (\mathsf{dig}_0, \mathsf{dig}_1, \pi_{\mathsf{BARG}})$. The verification algorithm first validates the digests dig_0 and dig_1, then constructs the statements x_i for $i \in [k]$ w.r.t. the monotone predicate, and then runs the seBARG verification.

We can modify the construction so that after computing dig_0 and dig_1, the prover computes a proof using modified statements as $\pi'_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}', (\hat{x}_1, \ldots, \hat{x}_k), (w_1, \ldots, w_k))$ (where the modified statements are $\hat{x}_i = (x_i, \mathsf{dig}_0, \mathsf{dig}_1)$ and $\mathcal{R}'(\mathsf{dig}_0, \mathsf{dig}_1, \cdot) \coloneqq \mathcal{R}_{\mathsf{dig}_0, \mathsf{dig}_1}(\cdot)$) and lets the final proof to be $\pi = (\mathsf{dig}_0, \mathsf{dig}_1, \pi'_{\mathsf{BARG}})$. Now if the underlying seBARG scheme has split-verification, then the monotone-Policy BARG scheme also has split-verification. Namely, the pre-verification algorithm just constructs the statements x_i for $i \in [k]$ w.r.t. the monotone predicate, and then runs the seBARG pre-verification to output $\mathsf{dig}_{\mathbf{x}}$. The online verification algorithm first validates the digests dig_0 and dig_1, then it runs the seBARG online verification using $(\mathsf{dig}_{\mathbf{x}}, x_{\mathsf{c}} = (\mathsf{dig}_0, \mathsf{dig}_1))$. The efficiency of both pre-verification and online verification follows directly from the efficiency of the corresponding algorithm of the underlying seBARG.

Construction 4.15. [Single-Hop Homomorphic Signatures for General Circuits] We will show how to update the *Construction* 4.4 to achieve context-hiding. Here we will only mention the updated parts and avoid repeating the rest. Let $\mathsf{BARG} = (\mathsf{BARG.Gen}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify})$ be a monotone-policy BARG with additional $(\mathsf{BARG.PreVerify}, \mathsf{BARG.OnlineVerify})$ algorithms, $\mathsf{PSig} =$

[8] We can see x_{c} (resp. x_i) as the common (resp. variable) part of statement among k instances.

$(\mathsf{PSig.Setup}, \mathsf{PSig.Setup\text{-}Punc}, \mathsf{PSig.Sign}, \mathsf{PSig.Verify})$ be an all-but-one signature scheme for single-bit messages, and additionally let $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a non-interactive zero-knowledge argument of knowledge proof system. We construct a single-hop homomorphic signature for general circuits $\mathsf{HSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}.) \to (\mathsf{pk}, \mathsf{sk})$**.** Same as in Construction 4.4 except that it additionally samples a $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, n_{\mathsf{NIZK}})$[9] and appends it to pk.

$\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$**.** Same as in Construction 4.4.

$\mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C) \to \sigma$**.** Same as in Construction 4.4 except that after computing π_{BARG} it does the following:

1. Recompute the public info as $\mathsf{dig}_{\mathbf{x}} \leftarrow \mathsf{BARG.PreVerify}(\mathsf{crs}_{\mathsf{BARG}}, \tilde{C}_y, \mathbf{x} = (x_1, \ldots, x_{2k}))$, where $x_i = (i, \mathsf{vk}_{\mathsf{PSig},i})$ for $i \in [k]$, $x_i = (i, \mathsf{vk}_{\mathsf{PSig},i-k})$ for $i \in [k+1, 2k]$.
2. Compute $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}}), \pi_{\mathsf{BARG}})$[10] for the following relation:
   - $\mathsf{OnlineVerify}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}}, \pi_{\mathsf{BARG}}) = 1$.
3. Output $\sigma = \pi_{\mathsf{NIZK}}$.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, C) \to 0/1$**.** Same as in Construction 4.4 except that instead of running the verification $0/1 \leftarrow \mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \tilde{C}_y, (x_i)_{i\in[2k]}, \sigma)$, it first computes the digest as $\mathsf{dig}_{\mathbf{x}} \leftarrow \mathsf{BARG.PreVerify}(\mathsf{crs}_{\mathsf{BARG}}, \tilde{C}_y, \mathbf{x} = (x_1, \ldots, x_{2k}))$, and then it runs the verification $0/1 \leftarrow \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}}), \sigma)$.

Theorem 4.16. The Construction 4.15 is a context-hiding unforgeable single-hop homomorphic signature with evaluated signature size $\mathsf{poly}(\lambda, \log|C|)$, assuming monotone-policy BARGs, all-but-one signatures for single-bit messages, and NIZKs.

Proof. The proof is mostly similar to the proof of Theorem 4.10. Namely:

Correctness. Follows directly from the construction (by combining BARG and NIZK completeness and PSig correctness).

Succinctness. Follows from the BARG split-verification succinctness.

Selective Unforgeability. Unforgeability follows from NIZK AoK, BARG non-adaptive soundness, and PSig punctured correctness. More specifically we first apply the following lemma and then the rest of the proof is similar to the construction without context-hiding. Let hyb_0 be the original experiment and hybrid hyb_1 be similat to hyb_0 except that the $\mathsf{crs}_{\mathsf{NIZK}}$ is generated using the knowledge extractor $\mathcal{E}$, and hyb_2 be similar to hyb_1 except that that given an evaulated signature and the $\mathsf{td}_{\mathsf{NIZK}}$, and it extracts a witness w^* from the signature (that is a NIZK proof) and checks whether it is a valid witness for the corresponding statement.

Lemma 4.17. If NIZK satisfies argument of knowledge, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_1(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

[9] where $n_{\mathsf{NIZK}} = |\mathsf{crs}_{\mathsf{BARG}}| + \mathsf{s}_{\mathcal{R}} + |\mathsf{dig}_{\mathbf{x}}|$

[10] Note that here $\pi_{\mathsf{BARG}} = (\mathsf{dig}_0, \mathsf{dig}_1, \pi'_{\mathsf{BARG}})$ and $x_{\mathsf{c}} = (\mathsf{dig}_0, \mathsf{dig}_1)$.

Proof. Follows directly from the crs indistinguishability of the knowledge extractor. □

Lemma 4.18. If NIZK satisfies argument of knowledge, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_2(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Follows directly from the knowledge extractor security. □

Context-Hiding. We will construct the simulator $\mathcal{S}$ as follows. To generate $(\mathsf{pk}, \mathsf{sk})$, $\mathcal{S}$ samples the PSig signing and verification keys analogously to the setup and uses and NIZK simulator to sample $\mathsf{crs}_{\mathsf{NIZK}}$. To generated a simulated evaluated signature, $\mathcal{S}$ just runs the NIZK simulator. Note that the statement for NIZK proof is $(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}})$ where $\mathsf{crs}_{\mathsf{BARG}}$ is included in the pk, $\mathcal{R}$ is the NP relation for the input wires which only depends on PSig verification keys that are included in the pk, and $\mathsf{dig}_{\mathbf{x}}$ only depends on PSig verification keys and $\tilde{C}_y$ where $\tilde{C}_y$ can be constructed using C and y. Also note that NIZK is applied on the outermost proof and we are using adaptively zero-knowledge NIZK, thus we get an adaptively context-hiding signature. The proof is a straightforward reduction to the zero-knowledge property of the underlying NIZK. Note that the condition that the evaluated signature is verified in the context-hiding definition guarantees that σ_i is a valid signature for m_i for $i \in [k]$, which in return guarantees that the NIZK statement in the reduction is indeed in the language, satisfying the required condition for the NIZK simulator oracle.

□

Corollary 4.19. The Construction 4.15 is a context-hiding unforgeable single-hop homomorphic signature with evaluated signature size $\mathsf{poly}(\lambda, \log |C|)$, assuming either LWE, k-LIN over pairing groups for any constant $k \in \mathbb{N}$, or sub-exponential DDH over pairing-free groups.

Proof. We conclude the proof by combining Theorems 3.2, 3.18 and 4.16, Corollary B.4, and Remark 3.5. □

5 Multi-Hop Homomorphic Signature

5.1 Definition

Before we proceed to define the multi-hop homomorphic signature scheme, we first explain notations of circuit class $\mathcal{C}_\ell$ and the structured circuit $(G, \{C_v\}_{v \in V})$. The circuit class $\mathcal{C}_\ell$ is specified with parameters $|\mathcal{C}_\ell|, \ell$ and d, where ℓ represents the maximum input size, d represents the maximum depth, and $|\mathcal{C}_\ell|$ denotes the maximum size of any circuit within $\mathcal{C}_\ell$. For any circuit C within class $\mathcal{C}_\ell$, C produces a single-bit output. The structured circuit is defined by a tree $G = (V, E)$ and $\{C_v\}_{v \in V}$, where a specific circuit in class $\mathcal{C}_\ell$ is assigned to each node. The circuit takes the output from its child node's circuit as input and outputs a single bit that it sends to its parental node. For simplicity, we assume without loss of generality that all circuits at the leaf nodes receive the same input $(b_1, \ldots, b_\ell)$. Each structured circuit set produces a single bit of output.

Syntax. A homomorphic signature scheme consists of the following polynomial time algorithms:

$\mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell) \to (\mathsf{vk}, \mathsf{sk})$. The Setup Algorithm takes as input security a parameter λ, number of maximum hops, and circuit class $\mathcal{C}_\ell$, and outputs verification/secret key $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, i, b) \to \sigma$. This is a probabilistic signing algorithm that takes as input signing key sk, index i, and a single bit message b. It outputs signature σ.

$\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C) \to \sigma$. The evaluator algorithm takes as input a public verification key pk, a number of hops t, and a set of ℓ signatures with their associated structured circuits for evaluation, each denoted as $\sigma_i, G_i = (V_i, E_i), \{C_v\}_{v\in V_i}$ for $i \in [\ell]$. Additionally, it takes a circuit C from class $\mathcal{C}_\ell$ that takes ℓ bits of input. The algorithm outputs a newly generated signature σ.

$\mathsf{Verify}(\mathsf{vk}, y, \sigma, G, \{C_v\}_{v\in V}) \to 0/1$. The verifier algorithm takes as input public key vk, message y, signature σ, and structured circuit $G, \{C_v\}_{v\in V}$. It outputs a bit in $\{0, 1\}$ to indicate whether the signature σ is a valid signature for y under the structured circuit $\{C_v\}_{v\in V}$.

Completeness For any $\lambda, k, \ell, d \in \mathbb{N}$, given a circuit class $\mathcal{C}_\ell$ with a maximum input size of ℓ and depth d, and for each $b_i \in \{0, 1\}$ for $i \in [\ell]$, and for G as the new tree with C at the root, integrating all G_i, the following holds:

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(\mathsf{vk}, y, \sigma, G, \{C_v\}_{v\in V}) = 1 & \begin{array}{l}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell),\\ \mathsf{Verify}(\mathsf{vk}, b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i}) = 1 \text{ for all } i \in [\ell],\\ \sigma = \mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C),\\ y = C(b_1, \ldots, b_\ell)\end{array}\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Efficiency Let $\mathcal{C}_\ell$ be the circuit class with maximum input size ℓ and depth d. We denote the maximum size of the circuit in class $\mathcal{C}_\ell$ as $|\mathcal{C}_\ell|$. Let k be the maximized hop of our homomorphic signature scheme. For every $\lambda, k, \ell \in \mathbb{N}$, the following holds:

- For $\mathsf{vk} \leftarrow \mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell)$ and $\sigma \leftarrow \mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$ such that $t \leq k$ and all of the above circuits are in circuit class $\mathcal{C}_\ell$, there exists a universal polynomial $\mathsf{poly}(\cdot, \cdot)$ such that $|\sigma| \leq \mathsf{poly}(\lambda, k)$.
- Let $|V|$ denote the maximum number of circuits in Tree G. There exists a universal polynomial $\mathsf{poly}(\cdot, \cdot, \cdot, \cdot)$ such that the setup algorithm $\mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell)$, evaluator algorithm $\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$, and verifier algorithm $\mathsf{Verify}(\mathsf{vk}, y, \sigma, G, \{C_v\}_{v\in V})$ run in time $\mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$.
- For some universal polynomial $\mathsf{poly}(\cdot, \cdot, \cdot, \cdot)$, it holds that $|\mathsf{vk}| \leq \mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$.

Soundness Homomorphic signature scheme must satisfy selective security, as the following:

Definition 5.1 (Selective Unforgeability)**.** A multi-hop homomorphic signature scheme satisfies unforgeability if for every stateful PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[\begin{array}{l|l} \begin{array}{l}y^* \neq y\\ \wedge\ \mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\end{array} & \begin{array}{l}((b_1, \ldots, b_\ell), G, \{C_v\}_{v\in V}, y^*) \leftarrow \mathcal{A}\\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell)\\ \sigma^* \leftarrow \mathcal{A}(\mathsf{vk})\end{array}\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where in the above equation, y is the actual output of the structured circuit $(G, \{C_v\}_{v \in V})$ over input $(b_1, \ldots, b_\ell)$.

5.2 Construction

Tools Required We apply sub-exponentially secure rate-1 somewhere extractable batch argument $\mathsf{barg} = (\mathsf{barg.Setup}, \mathsf{barg.Prove}, \mathsf{barg.Verify}, \mathsf{barg.Extract})$, Hash Tree $\mathsf{H} = (\mathsf{H.Setup}, \mathsf{H.Hash}, \mathsf{H.Open}, \mathsf{H.Verify})$, public key signature scheme $(\mathsf{sig.Setup}, \mathsf{sig.Verify})$, RAM delegation scheme $\mathsf{del} = (\mathsf{del.Setup}, \mathsf{del.Prove}, \mathsf{del.Digest}, \mathsf{del.Verify})$.

Our Design Set the security parameter $\lambda' = \lambda^\epsilon \cdot d \cdot k$ for some constant ϵ. This ensures that the sub-exponentially secure primitives, parameterized by λ', are secure against adversaries with running time $\mathsf{poly}(\lambda, 2^{d \cdot k})$.

$\mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell) \to (\mathsf{vk}, \mathsf{sk})$. The Setup algorithm takes as input security parameter λ, number of maximum levels of homomorphic evaluation k, and circuit class $\mathcal{C}_\ell$. It first sets Merkle Tree hash key

$$\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'}),$$

and generates signature secret key and verification key

$$(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'}).$$

Consider that $\mathcal{C}_\ell$ has a total number of N wires and among which there are 2ℓ input wires, then the circuit has $N - 2\ell$ internal wires and $N - 2\ell$ gates. For all $i \in \{1, \ldots, k\}$, it generates seBARG common reference strings as the following:

$$(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N - 2\ell, (N - 2\ell, N - 2\ell)),$$

$$(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1).$$

It sets RAM delegation CRS with respect to RAM Machine $\mathcal{R}_i$:

$$\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T).$$

It computes the hash value of the implicit input of RAM Machine $\mathcal{R}_i$:

$$h_i^{\mathsf{imp}} = \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1)).$$

It sets $\mathsf{vk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$, and outputs

$$\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k), \mathsf{sk} = \mathsf{sk}_0.$$

Below we define circuit satisfiability language $\mathcal{L}_i^0$ and $\mathcal{L}_i^1$, RAM Machine $\mathcal{R}_i$ for $i \in [k]$:

Language $\mathcal{L}_i^0$

Instance: x_j.
Witness: ω_j.
Output: Output 1 if and only if:
For $2\ell+1 \le j \le N$ where $x_j = (j, h, \tilde{C}_y)$ and $\omega_j = (b_j, b_{j_0}, b_{j_1}, \rho_j, \rho_{j_0}, \rho_{j_1})$, let gate c in monotone circuit $\tilde{C}_y$ be the gate that takes as input the j_0-th and the j_1-th wire, and outputs the j-th wire.

- For all $\alpha \in \{j, j_0, j_1\}$, $\mathsf{H.Verify}(\mathsf{hk}, h, \alpha, b_\alpha, \rho_\alpha) = 1$.
- $c(b_0, b_1) = b$.
- For $j = N$, it additionally requires that $b = 1$.

Language $\mathcal{L}_i^1$

Instance: x_j.
Witness: ω_j.
Hardwired: vk_{i-1}.
Output: Output 1 if and only if:
For $i = 1$ where $x_j = (j, h)$ and $\omega_j = (\sigma, b, \rho)$,

- $\mathsf{H.Verify}(\mathsf{hk}, h, j, b, \rho) = 1$,
- for $1 \le j \le \ell$, the language requires $\mathsf{sig.Verify}(\mathsf{vk}_{i-1}, b, \sigma, j) = 1$,
- otherwise for $\ell + 1 \le j \le 2\ell$, it requires $\mathsf{sig.Verify}(\mathsf{vk}_{i-1}, 1-b, \sigma, j-\ell) = 1$.

For $i \ge 2$, $x_j = (j, h, G, \{C_v\}_{v\in V})$, $\omega_j = (\sigma, b, \rho)$,

- $\mathsf{H.Verify}(\mathsf{hk}, h, j, b, \rho) = 1$,
- consider extracting $(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}})$ from vk_{i-1} and $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$ from σ, for $1 \le j \le \ell$, $\mathsf{del.Verify}(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, b, h, G, \{C_v\}_{v\in V}), \mathsf{del}.\pi) = 1$,
- otherwise $\ell+1 \le j \le 2\ell$, $\mathsf{del.Verify}(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, 1-b, h, G, \{C_v\}_{v\in V}), \mathsf{del}.\pi) = 1$.

RAM Machine $\mathcal{R}_i$

Variable Input: $\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, G, \{C_v\}_{v\in V}$.
Fixed Input: $\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1$.
Output: $\mathcal{R}_i$ follows these steps:

1. It sets root as the root of the G, and tree $G_j = (V_j, E_j)$ as the sub-tree rooted as the j-th child node of root. It sets monotone circuit $\tilde{C}_y$ following from Fig. 1.
2. For $j \in [\ell]$, it sets

$$x_j = (j, h, G_j, \{C_v\}_{v\in V_i}, \tilde{C}_y), x_{j+\ell} = (j+\ell, h, G_j, \{C_v\}_{v\in V_j}, \tilde{C}_y).$$

   For $j \in \{2\ell+1, \ldots, N\}$, it sets x_j as $(j, h, \tilde{C}_y)$.
3. Machine $\mathcal{R}_i$ accepts if and only if $\mathsf{barg.Verify}(\mathsf{barg.crs}_i^0, \{x_j\}_{j\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi) = 1$ and $\mathsf{barg.Verify}(\mathsf{barg.crs}_i^1, \{x_j\}_{j\in[2\ell]}, \mathsf{barg}.\pi) = 1$.

$\mathsf{Sign}(\mathsf{sk}, i, b) \to \sigma$**.** It outputs the signature for message bit b at index i as

$$\sigma \leftarrow \mathsf{sig.Sign}(\mathsf{sk}, (i, b)).$$

$\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C) \to \sigma$. The evaluator algorithm for the t-th hop follows these steps:

1. Let y be the output of C over $(b_1, \ldots, b_\ell)$. The evaluator first generates the monotone circuit $\tilde{C}_y$ according to C, where monotone circuit $\tilde{C}_y$ has a total number of N wires. It

sets $b_i = 1 - b_{i-\ell}$ for all $i \in \{2\ell+1, \ldots, N\}$. It then simulates monotone circuit $\tilde{C}_y$ gate by gate, and records b_i as the value of the i-th wire of circuit $\tilde{C}_y$ for all $i \in \{2\ell+1, \ldots, N\}$. It sets hash value

$$h = \mathsf{H.Hash}(\mathsf{hk}, (b_1, \ldots, b_N)).$$

2. It generates the hash openings for all wires. For all $i \in [N]$, it computes

$$\rho_i = \mathsf{H.Open}(\mathsf{hk}, (b_1, \ldots, b_N), i).$$

3. It assigns an instance and a witness to each input wire. If for all $i \in [\ell]$, tree G_i is empty that does not contain any node, and the homomorphic evaluation involves the circuit C applied to newly signed messages. Then for $i \in [\ell]$,

$$x_i = (i, h), \omega_i = (\sigma_i, b_i, \rho_i),$$

$$x_{i+\ell} = (i + \ell, h), \omega_{i+\ell} = (\sigma_i, 1 - b_i, \rho_{i+\ell}).$$

For all $i \in [\ell]$,

$$x_i = (i, h, G_i, \{C_v\}_{v \in V_i}, \tilde{C}_y), \omega_i = (\sigma_i, b_i, \rho_i),$$

$$x_{i+\ell} = (i + \ell, h, G_i, \{C_v\}_{v \in V_i}, \tilde{C}_y), \omega_{i+\ell} = (\sigma_i, 1 - b_i, \rho_{i+\ell}).$$

4. It sets an instance and a witness for each internal wire. For every $i \in \{2\ell + 1, \ldots, N\}$, take the i-th wire b_j. This wire is internal and serves as the output of some gate linked with two inputs to the gate b_{i_0}, b_{i_1}. To avoid misunderstanding, we clarify that the j_0-th and j_1-th wires are inputs to the gate and are not necessarily inputs to the circuit. It sets

$$x_i = (i, h, \tilde{C}_y), \omega_i = (b_i, b_{i_0}, b_{i_1}, \rho_j, \rho_{i_0}, \rho_{i_1}).$$

5. It generates

$$\mathsf{barg}.\pi^0 \leftarrow \mathsf{barg.Prove}(\mathsf{barg.crs}_t^0, \{x_i\}_{i \in \{2\ell+1, \ldots, N\}}, \{\omega_i\}_{i \in \{2\ell+1, \ldots, N\}}),$$

$$\mathsf{barg}.\pi^1 \leftarrow \mathsf{barg.Prove}(\mathsf{barg.crs}_t^1, \{x_i\}_{i \in [2\ell]}, \{\omega_i\}_{i \in [2\ell]}).$$

6. For all $i \in [\ell]$, let root_i denote the root node of Tree G_i. It generates node root and tree $G = (V, E)$ where root is the root of G:

$$V = \bigcup_{i=1}^{\ell} V_i \cup \{\mathsf{root}\}, E = \bigcup_{i=1}^{\ell} E_i \cup \{(\mathsf{root}, \mathsf{root}_1), \ldots, (\mathsf{root}, \mathsf{root}_\ell)\}.$$

It then assigns circuit C to the node root such that $C_{\mathsf{root}} = C$. Let t be the number of levels of G. One may also take t as the total number of hops including the current hop of evaluation.

7. Next, it generates a RAM delegation proof

$$\mathsf{del}.\pi \leftarrow \mathsf{del.Prove}(\mathsf{del.crs}_t, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, G, \{C_v\}_{v \in V}), (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1)).$$

8. It outputs signature σ as $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$.

$\mathsf{Verify}(\mathsf{vk}, y, \sigma, G, \{C_v\}_{v\in V}) \to \{0,1\}$. Define t as the number of node levels in tree G, where t is the height of tree G plus one, and also represents the number of hops in the homomorphic signature. Parse σ as $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Parse vk_t as $(\mathsf{barg}.\mathsf{crs}_t^0, \mathsf{barg}.\mathsf{crs}_t^1, \mathsf{del}.\mathsf{crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$. It outputs

$$\mathsf{del}.\mathsf{Verify}(\mathsf{del}.\mathsf{crs}_t, h_t^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, G, \{C_v\}_{v\in V}), \mathsf{del}.\pi).$$

Remark 5.2. We apply a universal security parameter $\lambda' = \lambda^\epsilon \cdot d \cdot k$ for the design of the Setup algorithm above. In fact for the above design where $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$ and one may apply a tighter security parameter as $\lambda^\epsilon \cdot d \cdot i$ when generating vk_i for $i \in [k]$.

Completeness The completeness of our scheme directly follows from the completeness of public key signature scheme sig, somewhere extractable batch argument barg, RAM delegation del, and the monotone circuit transformation.

Efficiency We analyze the efficiency of the above design:

Theorem 5.3. Assume that $\mathsf{barg} = (\mathsf{barg}.\mathsf{Setup}, \mathsf{barg}.\mathsf{Prove}, \mathsf{barg}.\mathsf{Verify}, \mathsf{barg}.\mathsf{Extract})$ is a rate-1 somewhere extractable batch argument. Then our design satisfies the efficiency definition.

Proof. We analyze the signature size using a inductive proof. Let σ_t be the signature at the t-th hop, such that $\sigma_t \leftarrow \mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$.

Claim 5.4. For all $t \in [k]$, there exists a universal polynomial $\mathsf{poly}(\cdot)$ such that $|\sigma_t| \le t \cdot \mathsf{poly}(\lambda)$.

Proof. We prove the claim through induction.

Base Case ($t = 1$). By our design, $\sigma_t = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Hash value h and delegated proof $\mathsf{del}.\pi$ are local parameters such that $|h| + |\mathsf{del}.\pi| \le \mathsf{poly}(\lambda)$. Next we analyze the size of $\mathsf{barg}.\pi^0$, since the witness of language $\mathcal{L}_i^0$ only contains local hash parameters for all $i \in [k]$ and barg is rate-1, $|\mathsf{barg}.\pi^0| \le \mathsf{poly}(\lambda)$. For the first hop evaluation where $t = 1$, witness of language $\mathcal{L}_1^1$ consists of local hash parameters and basic public key signatures. Due to rate-1 seBARG, $|\mathsf{barg}.\pi^1| \le \mathsf{poly}(\lambda)$ for some universal polynomial $\mathsf{poly}(\cdot)$. Thus the claim holds for $t = 1$.

Inductive Step ($2 \le t \le k$). For $\sigma_t = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$, given the locality properties of Hash Tree and rate-1 seBARG, the overall size $|h| + |\mathsf{del}.\pi| + |\mathsf{barg}.\pi^0| \le \mathsf{poly}(\lambda)$. Additionally, it follows that $|\mathsf{barg}.\pi^1| \le |\sigma_{t-1}| + \mathsf{poly}(\lambda)$. Thus, $|\sigma_t| \le |\sigma_{t-1}| + \mathsf{poly}(\lambda)$ for some universal polynomial $\mathsf{poly}(\cdot)$. By our inductive hypothesis, $|\sigma_{t-1}| \le (t-1)\cdot\mathsf{poly}(\lambda)$. Putting the above together completes the proof for the claim. □

Claim 5.4 implies that $|\sigma| \le \mathsf{poly}(\lambda, k)$. Next, we analyze the verifier running time. For all $t \in [k]$, the verifier for the t-th hop runs a RAM delegation verifier for the RAM machine $\mathcal{R}_t$. $\mathcal{R}_t$ takes $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, G, \{C_v\}_{v\in V})$ as its variable input. By the succinctness of RAM delegation, verifier's running time depends polynomially on the size of the variable input and the security parameter λ. By Claim 5.4, we have $|\sigma| \le \mathsf{poly}(\lambda, k)$, which implies $|\mathsf{barg}.\pi^0| + |\mathsf{barg}.\pi^1| + |h| \le \mathsf{poly}(\lambda, k)$. Size of the graph $G = (V, E)$ at circuits $\{C_v\}_{v\in V}$ is at most $\mathsf{poly}(n, |\mathcal{C}_\ell|)$. Thus, verifier's running time is $\mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$.

We continue to analyze size of vk. We observe that for all $i \in [k]$, size of circuit satisfiability language $\mathcal{L}_i^0$ is at most $\mathsf{poly}(\lambda, |\mathcal{C}_\ell|)$. Size of language $\mathcal{L}_i^1$ depends on homomorphic signature size and verifier running time, which is at most in total of $\mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$. Thus by succinctness barg, $|\mathsf{barg.crs}_i^0| + |\mathsf{barg.crs}_i^1| \leq \mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$, which implies that $|\mathsf{vk}_i| \leq \mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$. Combining the above, $|\mathsf{vk}| \leq \mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$. Then the efficiency of setup and evaluator algorithms immediately follows. □

5.3 Soundness

Theorem 5.5. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property, then our construction satisfies multi-hop unforgeability.

Proof. We first define Hybrid (j_0, j_1), over which we will later present an inductive proof. Note that the hybrids are defined with respect to $2\ell + 1 \leq j_0, j_1 \leq N$.

Hybrid (j_0, j_1)

1. The attacker $\mathcal{A}$ starts by outputting a sequence of messages $(\beta_1, \ldots, \beta_\ell)$, a tree $G = (V, E)$, a set of circuits $\{C_v\}_{v \in V}$, and a message y^*.

2. Consider that the above tree G has a number of t levels. Challenger first sets hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and generates single-hop homomorphic signature key $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$.

3. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, it generates $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N - 2\ell, (N - 2\ell, N - 2\ell))$. It generates $(\mathsf{barg.crs}_t^0, \mathsf{barg.td}_t^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_t^0, N - 2\ell, (j_0 - 2\ell, j_1 - 2\ell))$. Next for all $i \in \{1, \ldots, k\}$, it sets $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, and $h_i^{\mathsf{imp}} = \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$.

4. For all $i \in [k]$, challenger sets $\mathsf{vk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. Challenger outputs $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k), \mathsf{sk} = \mathsf{sk}_0$. For all $i \in [k]$, the challenger computes and outputs $\sigma_i \leftarrow \mathsf{sig.Sign}(\mathsf{sk}, (i, \beta_i))$.

5. The attacker $\mathcal{A}$ outputs σ^*. Let y be the actual output of the structured circuits $\{C_v\}_{v \in V}$ taking $(\beta_1, \ldots, \beta_\ell)$ as input. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v \in V}) = 1$ and $y^* \neq y$.

Let root denote the root node of G. Assume without loss of generality that circuit C takes ℓ bits of input. For all $j \in [\ell]$, let $G_j = (V_j, E_j)$ represent the sub-tree rooted at the j-th child node of root. Next, for all $j \in [\ell]$, let b_j^* be the output by the structured circuit $\{C_v\}_{v \in V_i}$ over the input $(\beta_1, \ldots, \beta_\ell)$, and set $b_{j+\ell}^*$ as $1 - b_j^*$. Set circuit $\tilde{C}_{y^*}$ as the monotone circuit of circuit C_{root} ($\tilde{C}_{y^*}$ follows Fig. 1). Let $(b_{2\ell+1}^*, \ldots, b_N^*)$ be the value of the rest of the wires of circuit $\tilde{C}_{y^*}$ taking $(b_1^*, \ldots, b_{2\ell}^*)$ as input. Parse the signature by $\mathcal{A}$ as $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Next for $j \in [\ell]$, set $x_j = (j, h, G_j, \{C_v\}_{v \in V_i}, \tilde{C}_{y^*})$ and $x_{j+\ell} = (j + \ell, h, G_j, \{C_v\}_{v \in V_j}, \tilde{C}_{y^*})$. For $j \in \{2\ell + 1, \ldots, N\}$, set x_j as $(j, h, \tilde{C}_{y^*})$. Let $(\omega_{j_0}, \omega_{j_1})$ be $\mathsf{barg.Extract}(\mathsf{barg.td}_t^0, \{x_i\}_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0)$. Parse ω_{j_b} as $(b, b_0, b_1, \rho, \rho_0, \rho_1)$ and set b_{j_b} as b. For adversary $\mathcal{A}$ in Hybrid (j_0, j_1), let $\mathsf{Adv}_{\mathcal{A}}^{j_0, j_1, b}$ denote the following:

$$\mathsf{Adv}_{\mathcal{A}}^{j_0, j_1, b} = \Pr_{\mathsf{hyb}_{j_0, j_1}} [\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v \in V}) = 1 \wedge b_{j_b} > b_{j_b}^*].$$

We note that Hybrid (N, N) corresponds to the original unforgeability game for multi-hop homomorphic signature scheme. We denote $\mathcal{A}$'s winning advantage such hybrid as $\mathsf{Adv}_{\mathcal{A}}$:

$$\mathsf{Adv}_{\mathcal{A}} = \Pr_{\mathsf{hyb}_{N,N}}[\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1 \wedge y^* \neq y].$$

Lemma 5.6. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property. Assume that there exists a PPT adversary $\mathcal{A}$, a non-negligible function $\epsilon(\cdot,\cdot)$ such that $\mathsf{Adv}^{N,N,0}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d\cdot k})$. Then within the second level of circuit $\tilde{C}_{y^*}$ (the first level are input wires), there exists some internal wire indexed at j^* such that $3^{d-2} \cdot \mathsf{Adv}^{j^*,j^*,0}_{\mathcal{A}} \geq \mathsf{Adv}^{N,N,0}_{\mathcal{A}}$.

Proof. We show the following: For any node indexed at j on level i ($i > 1$), if there exists a non-negligible function $\epsilon(\cdot,\cdot)$ such that $\mathsf{Adv}^{j,j,0}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d\cdot k})$, then there exists at least one node j_α on level $i-1$, such that $3 \cdot \mathsf{Adv}^{j_\alpha,j_\alpha,0}_{\mathcal{A}} \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}}$. The above implies our lemma by a simple induction. To prove it, consider these claims:

Claim 5.7. Assume that the somewhere extractable BARG barg satisfies sub-exponentially secure index hiding, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that $\mathsf{Adv}^{j,j',0}_{\mathcal{A}} \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k})$.

Proof. We prove the claim with a reduction towards the index-hiding of barg. Define reduction algorithm $\mathcal{B}$ as the following:

$\mathcal{A}$ starts by outputting $\{\beta_i\}_{i\in[\ell]}$, $G = (V, E)$, $\{C_v\}_{v\in V}$, and y^*. $\mathcal{B}$ sets $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, $\mathcal{B}$ generates $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_i, N-2\ell, (N-2\ell, N-2\ell))$. $\mathcal{B}$ sets up $(\mathsf{barg.crs}^0_{t_0}, \mathsf{barg.td}^0_{t_0}) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_t, N-2\ell, j-2\ell)$. $\mathcal{B}$ then queries the barg index-hiding challenger using j and j' and the challenger returns with $\mathsf{barg.crs}^0_{t_1}$. $\mathcal{B}$ sets the $\mathsf{barg.crs}^0_t$ as $(\mathsf{barg.crs}^0_{t_0}, \mathsf{barg.crs}^0_{t_1})$. For all $i \in \{1, \ldots, k\}$, $\mathcal{B}$ sets $(\mathsf{barg.crs}^1_i, \mathsf{barg.td}^1_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^1_i, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, $h^{\mathsf{imp}}_i = \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$, and $\mathsf{vk}_i = (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. It outputs vk as $(\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$ and sk as sk_0. Then, $\mathcal{A}$ outputs $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Upon the output by $\mathcal{A}$, $\mathcal{B}$ extracts the bit b_j using $\mathsf{barg.Extract}(\mathsf{barg.td}^0_{t_0}, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$. If $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1$ and $b_j > b^*_j$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs 1.

We note that the index-hiding challenger tosses a random coin $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it returns $\mathsf{barg.crs}^0_{t_0} \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_t, N-2\ell, j-2\ell)$ and otherwise for $\beta = 1$, challenger returns $\mathsf{barg.crs}^0_{t_0} \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_t, N-2\ell, j'-2\ell)$. Thus for $\beta = 0$, the above experiment corresponds to Hybrid (j, j) and otherwise it corresponds to Hybrid (j, j'). Assume towards contradiction that $|\mathsf{Adv}^{j,j',0}_{\mathcal{A}} - \mathsf{Adv}^{j,j,0}_{\mathcal{A}}| \geq \epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible. Then, the above reduction algorithm $\mathcal{B}$ breaks the sub-exponentially secure index-hiding property of barg. □

Claim 5.8. Assume that the RAM delegation scheme del satisfies sub-exponential soundness, then for any polynomial time adversary $\mathcal{A}$ in Hybrid (j, j'), there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b^*_j\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}^0_t, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\end{array}\right] \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Consider for Hybrid (j, j'), adversary $\mathcal{A}$ outputs $\{\beta_i\}_{i\in[\ell]}$, $G = (V, E)$, $\{C_v\}_{v\in V}$, and y^* in step 1, and outputs $(h, \mathsf{barg}.\pi, \mathsf{del}.\pi)$ in step 3. Let t denote the number of levels of tree G. According to the output of $\mathcal{A}$, for all $i \in \{2\ell+1, \ldots, N\}$, let x_i be the instance with respect to language $\mathcal{L}_t^0$. Assume towards contradiction that with probability of $\epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible, it satisfies that $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1$ and $\mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) \neq 1$.

We design a simple reduction algorithm $\mathcal{B}$ that breaks the soundness property of RAM delegation scheme del. $\mathcal{A}$ starts by outputting $\{\beta_i\}_{i\in[\ell]}$, $G = (V, E)$, $\{C_v\}_{v\in V}$, and y^*. $\mathcal{B}$ sets $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, $\mathcal{B}$ generates $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N-2\ell, \{N-2\ell, N-2\ell\})$ and $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$. $\mathcal{B}$ sets $(\mathsf{barg.crs}_t^0, \mathsf{barg.td}_t^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_t^0, N-2\ell, (j-2\ell, j'-2\ell))$. $\mathcal{B}$ queries the RAM delegation challenger with $(1^{\lambda'}, T)$ and the challenger outputs $\mathsf{del.crs}_t$. For all $i \in \{1, \ldots, k\}$, $\mathcal{B}$ sets $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1)$, $h_i^{\mathsf{imp}} \leftarrow \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$, $\mathsf{vk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. $\mathcal{B}$ outputs vk as $(\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$ and sk as sk_0. Then, $\mathcal{A}$ outputs $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Upon the output by $\mathcal{A}$, $\mathcal{B}$ outputs $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, G, \{C_v\}_{v\in V})$, $(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1)$, and $\mathsf{del}.\pi$.

By the above assumption where $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1$, it implies that $\mathsf{del.Verify}(\mathsf{del.crs}_t, h_t^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, G, \{C_v\}_{v\in V}), \mathsf{del}.\pi) = 1$. Since the assumption says that $\mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i^0\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) \neq 1$, $\mathcal{B}$ breaks the RAM delegation soundness with an advantage of $\epsilon(\lambda, 2^{d\cdot k})$. Our claim holds by the reduction with Claim 5.7. □

Claim 5.9. Assume that the seBARG barg satisfies sub-exponentially secure somewhere argument of knowledge property, then for any PPT adversary $\mathcal{A}$ in Hybrid (j, j') and $(\omega_j, \omega_{j'}) = \mathsf{barg.Extract}(\mathsf{barg.td}_t^0, \{x_i^0\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b_j^*\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\\ \wedge\ \omega_j, \omega_{j'} \text{ are valid witnesses for } \mathcal{L}_t^0\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Assume towards contradiction that with probability larger than $\epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible, $\mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1$ and at least one of $\omega_j, \omega_{j'}$ is not a valid witness for $\mathcal{L}_t^0$.

There exists a reduction algorithm $\mathcal{B}$ that breaks the somewhere argument of knowledge property of seBARG scheme barg. $\mathcal{A}$ first outputs $\{\beta_i\}_{i\in[\ell]}$, $G = (V, E)$, $\{C_v\}_{v\in V}$, and y^*. $\mathcal{B}$ sets $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, $\mathcal{B}$ generates $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N-2\ell, (N-2\ell, N-2\ell))$. $\mathcal{B}$ queries the barg challenger with $(1^{\lambda'}, \mathcal{L}_t^0, N-2\ell, (j-2\ell, j'-2\ell))$ and the challenger outputs $\mathsf{barg.crs}_t^0$. For all $i \in \{1, \ldots, k\}$, $\mathcal{B}$ generates $\mathsf{barg.crs}_i^1 \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, $h_i^{\mathsf{imp}} = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$, and sets vk_i as $(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. $\mathcal{B}$ outputs vk as $(\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$ and sk as sk_0. Next, $\mathcal{A}$ outputs $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. $\mathcal{B}$ outputs $\{x_i\}_{i\in\{2\ell+1,\ldots,N\}}$ and $\mathsf{barg}.\pi^0$.

By the contradictory assumption, $\mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1$ and either ω_j or $\omega_{j'}$ is not a valid witness for $\mathcal{L}_t^0$, where $(\omega_j, \omega_{j'}) = \mathsf{barg.Extract}(\mathsf{barg.td}^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$. Thus, with an advantage of $\epsilon(\lambda, 2^{d\cdot k})$, $\mathcal{B}$ breaks the sub-exponentially secure somewhere argument of knowledge property of barg. Our claim holds by the above reduction with Claim 5.8. □

Let c be the gate in circuit $\tilde{C}_{y^*}$ such that the j_0-th and j_1-th wires are the input wires to gate c and the j-th wire is the output wire. We require the j-th wire to be positioned on level i of $\tilde{C}_{y^*}$ where $3 \leq i \leq d$, so that j_0-th and $j-1$-th wires are internal. For $\alpha \in \{0,1\}$, in Hybrid (j, j_α), parse adversary $\mathcal{A}$'s output signature σ^* as $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi_0, \mathsf{barg}.\pi_1, \mathsf{del}.\pi)$. Set $(\omega_j, \omega_{j_\alpha}) = \mathsf{barg.Extract}(\mathsf{barg.td}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$. Parse ω_j as $(b_j, b_0, b_1, \rho_j, \rho_0, \rho_1)$, and ω_{j_α} as $(b_{j_\alpha}, b'_0, b'_1, \rho_{j_\alpha}, \rho'_0, \rho'_1)$.

Claim 5.10. Assume that the Hash Tree H satisfies sub-exponentially secure collision-resistance property, then for $\alpha \in \{0,1\}$ and PPT adversary $\mathcal{A}$ in Hybrid (j, j_α), there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j_\alpha}}\left[\begin{array}{l} \mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1 \\ \wedge\ b_j > b_j^* \\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1 \\ \wedge\ \omega_j, \omega_{j_\alpha} \text{ are valid witnesses for } \mathcal{L}_t^0 \\ \wedge\ b_{j_\alpha} = b_\alpha \end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Assume towards contradiction that with probability larger than $\epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible, both ω_j and $\omega_{j'}$ are valid witnesses for $\mathcal{L}_t$ and $b_{j_\alpha} \neq b_\alpha$.

There exists a reduction algorithm $\mathcal{B}$ that breaks the collision-resistance of H. $\mathcal{A}$ starts by outputting $\{\beta_i\}_{i\in[\ell]}$, $G = (V, E)$, $\{C_v\}_{v\in V}$, and y^*. $\mathcal{B}$ queries the hash challenger with $\mathsf{H.Setup}(1^{\lambda'})$ and the challenger returns with hash key hk. Next, $\mathcal{B}$ sets $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$. For all $i \in \{1,\ldots,k\} \setminus \{t\}$, $\mathcal{B}$ generates $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N-2\ell, (N-2\ell, N-2\ell))$. $\mathcal{B}$ additionally sets $\mathsf{barg.crs}_t^0$ as $\mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_t^0, N-2\ell, (j-2\ell, j'-2\ell))$. For all $i \in \{1,\ldots,k\}$, $\mathcal{B}$ generates $\mathsf{barg.crs}^1 \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, $h_i^{\mathsf{imp}} = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$, and sets $\mathsf{vk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. $\mathcal{B}$ outputs vk as $(\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$ and sk as sk_0. Next, $\mathcal{A}$ sends $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$ to $\mathcal{B}$. $\mathcal{B}$ extracts $(\omega_j, \omega_{j_\alpha})$ using $\mathsf{barg.Extract}(\mathsf{barg.td}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$ and outputs $h, j_\alpha, b_\alpha, \rho_\alpha, b_{j_\alpha}, \rho_{j_\alpha}$.

According to the contradictory assumption, $\omega_j, \omega_{j_\alpha}$ are valid witnesses for $\mathcal{L}_t$ which implies that $\mathsf{H.Verify}(\mathsf{hk}, h, j_\alpha, b_\alpha, \rho_\alpha) = \mathsf{H.Verify}(\mathsf{hk}, h, j_\alpha, b_{j_\alpha}, \rho_{j_\alpha}) = 1$. Then by the assumption that $b_{j_\alpha} \neq b_\alpha$, $\mathcal{B}$ breaks collision-resistance property of H with an advantage of $\epsilon(\lambda, 2^{d\cdot k})$. By the above reduction with Claim 5.9, our claim holds. □

Claim 5.11. For any PPT adversary $\mathcal{A}$, there exists some $\alpha \in \{0,1\}$, such that

$$\Pr_{\mathsf{hyb}_{j,j_\alpha}}\left[\begin{array}{l} \mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1 \\ \wedge\ b_j > b_j^* \\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1 \\ \wedge\ \omega_j, \omega_{j_\alpha} \text{ are valid witnesses for } \mathcal{L}_t^0 \\ \wedge\ b_{j_\alpha} = b_\alpha \\ \wedge\ b_{j_\alpha} > b_{j_\alpha}^* \end{array}\right] \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}(\lambda, 2^{d\cdot k})}{2}.$$

Proof. By property of monotone circuit, $b_j > b_j^*$ implies that $b_{j_\alpha} > b_{j_\alpha}^*$ for some $\alpha \in \{0,1\}$. Since $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible, the claim follows immediately from Claim 5.10. □

Claim 5.12. For any PPT adversary $\mathcal{A}$, if there exists a non-negligible function $\epsilon(\cdot,\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda, 2^{d\cdot k})$, then there exists some $\alpha \in \{0,1\}$, such that

$$\mathsf{Adv}_{\mathcal{A}}^{j_\alpha,j_\alpha,0} \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{3}.$$

Proof. Reducing to the index-hiding property of barg and using Claim 5.11, we obtain:

$$\Pr_{\mathsf{hyb}_{j_\alpha,j_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1 \\ \wedge\ b_{j_\alpha} > b^*_{j_\alpha}\end{array}\right] \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{2} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

We omit the proof for the above as it is almost the same as the proof of Claim 5.7. Next, $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda, 2^{d\cdot k})$ implies that $\frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{2} - \mathsf{negl}(\lambda, 2^{d\cdot k}) \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{3}$. □

By a simple inductive proof using Claim 5.12, our lemma immediately follows. □

Next, we switch to Hybrid (j^*, j^*_α) for $2\ell+1 \leq j^* \leq N$ and $1 \leq j^*_\alpha \leq 2\ell$. In Hybrid (j^*, j^*_α), j^* represents an internal wire at the second level of $\tilde{C}_{y^*}$, while j^*_α denotes the index of an input wire at the bottom level. This input wire is one of the two that connect to the gate producing the j^*-th wire.

Hybrid (j^*, j^*_α)

1. The attacker $\mathcal{A}$ starts by outputting a sequence of messages $(\beta_1, \ldots, \beta_\ell)$, a tree $G = (V, E)$, a set of circuits $\{C_v\}_{v\in V}$, and a message y^*.

2. Consider that the above tree G has a number of t levels. Challenger first sets hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and generates single-hop homomorphic signature key $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$.

3. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, it generates $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_i, N-2\ell, (N-2\ell, N-2\ell))$ and $(\mathsf{barg.crs}^1_i, \mathsf{barg.td}^1_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^1_i, 2\ell, 1)$. It sets $(\mathsf{barg.crs}^0_t, \mathsf{barg.td}^0_t) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_t, N-2\ell, (j^*, j^*))$ and $(\mathsf{barg.crs}^1_t, \mathsf{barg.td}^1_t) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^1_t, 2\ell, j^*_\alpha)$. Then for all $i \in [k]$, challenger sets $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$ and $h^{\mathsf{imp}}_i = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$.

4. For all $i \in [k]$, challenger sets $\mathsf{vk}_i = (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. Challenger outputs $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k), \mathsf{sk} = \mathsf{sk}_0$. For all $i \in [k]$, the challenger computes and outputs $\sigma_i \leftarrow \mathsf{sig.Sign}(\mathsf{sk}, (i, \beta_i))$.

5. The attacker $\mathcal{A}$ outputs σ^*. Let y be the actual output of the structured circuits $\{C_v\}_{v\in V}$ taking $(\beta_1, \ldots, \beta_\ell)$ as input. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1$ and $y^* \neq y$.

As the previous hybrids, we let root denote the root node of G. For all $j \in [\ell]$, let $G_j = (V_j, E_j)$ represent the sub-tree rooted at the j-th child node of root, let b^*_j be the output by the structured circuit $\{C_v\}_{v\in V_i}$ over the input $(\beta_1, \ldots, \beta_\ell)$, and let $b^*_{j+\ell}$ be $1 - b^*_j$. Again consider circuit $\tilde{C}_{y^*}$ as the monotone circuit from C_{root}. Let $(b^*_{2\ell+1}, \ldots, b^*_N)$ be the values of the rest of the wires of circuit $\tilde{C}_{y^*}$ taking $(b^*_1, \ldots, b^*_{2\ell})$ as input. Consider σ^* as $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Let (ω^*_j, ω'_j) be $\mathsf{barg.Extract}(\mathsf{barg.td}^0_t, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$. We parse ω^*_j as $(b_{j^*}, b_0, b_1, \rho_{j^*}, \rho_0, \rho_1)$. Let $\omega_{j^*_\alpha}$ be $\mathsf{barg.Extract}(\mathsf{barg.td}^1_t, \{x_i\}_{i\in\{1,\ldots,2\ell\}}, \mathsf{barg}.\pi^1)$ where $\omega_{j^*_\alpha} = (\sigma, b_{j^*_\alpha}, \rho_{j^*_\alpha})$.

Lemma 5.13. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property. Assume that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \epsilon(\lambda, 2^{d\cdot k})$ for some non-negligible function $\epsilon(\cdot,\cdot)$ and for wire j^* at the second level of circuit $\tilde{C}_{y^*}$ (the bottom internal wires excluding input wires), then there exists some input wire at j^*_α such that

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ \omega_{j^*_\alpha} \text{ is a valid witness for } \mathcal{L}^1_t\\ \wedge\ b_{j^*_\alpha} > b^*_{j^*_\alpha}\end{array}\right] \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0}}{3}.$$

Proof. We prove the lemma using the following hybrid claims:

Claim 5.14. Assume that the seBARG barg satisfies sub-exponentially secure index hiding, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_{j^*} > b^*_{j^*}\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. We omit the proof as it is nearly identical to the proof of Claim 5.7. □

Claim 5.15. Assume that RAM delegation scheme del satisfies sub-exponential soundness, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_{j^*} > b^*_{j^*}\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}^0_t, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}^1_t, \{x_i\}_{i\in[2\ell]}, \mathsf{barg}.\pi^1) = 1\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. The proof is nearly identical to the proof of Claim 5.8. □

Claim 5.16. Assume that seBARG scheme barg satisfies sub-exponentially secure somewhere argument of knowledge property, then for any PPT adverary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_{j^*} > b^*_{j^*}\\ \wedge\ \omega_{j^*} \text{ is a valid witness for } \mathcal{L}^0_t\\ \wedge\ \omega_{j^*_\alpha} \text{ is a valid witness for } \mathcal{L}^1_t\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. The proof is nearly identical to the proof of Claim 5.9. □

Claim 5.17. Assume that Hash Tree H satisfies sub-exponentially secure collision-resistance property, then for any PPT adversary $\mathcal{A}$, for $\alpha \in \{0,1\}$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_{j^*} > b^*_{j^*}\\ \wedge\ \omega_{j^*} \text{ is a valid witness for } \mathcal{L}^0_t\\ \wedge\ \omega_{j^*_\alpha} \text{ is a valid witness for } \mathcal{L}^1_t\\ \wedge\ b_{j^*_\alpha} = b_\alpha\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. (Omitted) Follows by the proof of Claim 5.10. □

Now, since we are assuming that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible. By the above claim, we conclude that our lemma holds. There exists some $\alpha \in \{0,1\}$, such that

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1 \\ \wedge\ \omega_{j^*_\alpha} \text{ is a valid witness for } \mathcal{L}_t^1 \\ \wedge\ b_{j^*_\alpha} > b^*_{j^*_\alpha}\end{array}\right] \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} - \mathsf{negl}(\lambda, 2^{d\cdot k})}{2} \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0}}{3}.$$

□

Recall that we define $\mathsf{Adv}_{\mathcal{A}}$ as the overall winning probability of an adversarial attacker $\mathcal{A}$ before introducing Lemma 5.18, where

$$\mathsf{Adv}_{\mathcal{A}} = \Pr_{\mathsf{hyb}_{N,N}}[\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1 \wedge y^* \neq y].$$

Lemma 5.18. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property. Assume that there exists an PPT adversary $\mathcal{A}$ and a non-negligible function $\epsilon(\cdot,\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(T(\lambda), 2^{d(\lambda)})$. There exists some internal wire indexed at j^* at the second level of circuit $\tilde{C}_{y^*}$, such that the following holds:

$$\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \frac{\mathsf{Adv}_{\mathcal{A}}}{3^{d-1}}.$$

Proof. Using a reduction to the soundness of del and somewhere argument of knowledge property of barg, the following holds (similar to the proofs of Claim 5.8 and 5.9):

$$\mathsf{Adv}_{\mathcal{A}}^{N,N,0} \geq \mathsf{Adv}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Thus by Lemma 5.18,

$$\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \frac{\mathsf{Adv}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k})}{3^{d-2}}.$$

Recall that we assume $\mathsf{Adv}_{\mathcal{A}}^{N,N,0} \geq \epsilon(T(\lambda), 2^{d(\lambda)})$, thus our lemma immediately follows. □

Recall that $\omega_{j^*_\alpha} = (\sigma, b_{j^*_\alpha}, \rho_{j^*_\alpha})$. Note that in Lemma 5.13, $\omega_{j^*_\alpha}$ as a valid witness for $\mathcal{L}_t^1$ and $b_{j^*_\alpha} > b^*_{j^*_\alpha}$ indicates that σ is an adversarial signature for message $1 - b_{j^*}$ corresponding to sub-tree G_{j^*} $(1 \leq j^* \leq \ell)$ or $G_{j^*-\ell}$ $(\ell \leq j^* \leq 2\ell)$. Combining Lemma 5.13 and Lemma 5.18, we conclude the following: for any PPT adversary $\mathcal{A}$ that breaks the unforgeability of our scheme with advantage $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d\cdot k})$ at the t-th hop, there exists a reduction algorithm that breaks the unforgeability of the signature scheme at the $t-1$-th hop with advantage greater than or equal to $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^d}$. Thus our theorem holds, following from a simple induction over Lemma 5.13 and Lemma 5.18: $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d\cdot k})$ implies a polynomial time reduction algorithm which breaks the unforgeability of sig with advantage at least $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^{d\cdot k}}$, which is non-negligible.

Theorem 5.19 (Theorem 3.1 of [PP22])**.** Assume the existence of a $T(\cdot)$-secure non-interactive batch argument for the index language, a $T(\cdot)$-secure somewhere extractable hash with additive overhead $\alpha(\lambda, \ell) = \frac{\ell}{\lambda} + \mathsf{poly}(\lambda)$, a $T(\cdot)$-secure non-interactive delegation scheme for RAM, then there exists a $T(\cdot)$-secure non-interactive batch argument for the indexl language with unbounded witness length and with additive overhead $\sigma(\lambda, m) = \frac{3m}{\lambda} + \mathsf{poly}(\lambda)$.

Corollary 5.20. Assuming either LWE, k-LIN over pairing groups, or sub-exponential DDH (and QR), there exists a multi-hop homomorphic signature scheme.

Proof. By Theorem 5.19 and the previous designs of RAM Delegation and Batch Arguments, there exists a rate-1 somewhere extractable batch argument from the above assumptions. Thus by Theorem 5.3 and Theorem 5.5, our corollary follows. □

□

6 Multi-Hop Homomorphic Signature with Context Hiding

The efficiency, completeness, and unforgeability definition of multi-hop homomorphic signature with context hiding is exactly the same as the general multi-hop homomorphic signature scheme, except for the context-hiding property. The property additionally requires a simulator $\mathcal{S}$.

Definition 6.1 (Context Hiding)**.** A single-hop homomorphic signature scheme satisfies context-hiding if there exist a stateful PPT simulator $\mathcal{S}$ such that for every stateful PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and $\mathcal{C}_\ell$, the following holds:

$$\Pr\left[\begin{array}{l} \mathcal{A}(\sigma_b) = b \ \wedge C \in \mathcal{C}_\ell \ \wedge \\ \ \forall\ v \in V_i :\ C_v \in \mathcal{C}_\ell \ \wedge \\ \mathsf{Verify}(\mathsf{vk}_b, C((b_i)_{i\in[\ell]}), \sigma_0, C) = 1 \end{array} : \begin{array}{l} b \leftarrow \{0,1\},\ (\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell) \\ (\mathsf{vk}_1, \mathsf{sk}_1) \leftarrow \mathcal{S}(1^\lambda, 1^k, \mathcal{C}_\ell) \\ ((b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})_{i\in[k]}, C) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_b,\cdot,\cdot)}(\mathsf{pk}_b) \\ \sigma_0 \leftarrow \mathsf{Eval}(\mathsf{vk}_b, (t, (b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})_{i\in[\ell]}, C)) \\ \sigma_1 \leftarrow \mathcal{S}((G_i, \{C_v\}_{v\in V_i})_{i\in[\ell]}, C((b_i)_{i\in[\ell]}), C) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

6.1 Construction

Tools Required We apply sub-exponentially secure rate-1 somewhere extractable batch argument $\mathsf{barg} = (\mathsf{barg.Setup}, \mathsf{barg.Prove}, \mathsf{barg.Verify}, \mathsf{barg.Extract})$, rate-1 NIZK $\mathsf{nizk} = (\mathsf{nizk.Setup}, \mathsf{nizk.Prove}, \mathsf{nizk.Verify})$, Hash Tree $\mathsf{H} = (\mathsf{H.Setup}, \mathsf{H.Hash}, \mathsf{H.Open}, \mathsf{H.Verify})$, public key signature scheme $(\mathsf{sig.Setup}, \mathsf{sig.Verify})$, RAM delegation scheme $\mathsf{del} = (\mathsf{del.Setup}, \mathsf{del.Prove}, \mathsf{del.Verify})$, and perfectly complete public key encryption scheme $\mathsf{pke} = (\mathsf{pke.Setup}, \mathsf{pke.Enc}, \mathsf{pke.Dec})$

Our Design Set the security parameter $\lambda' = \lambda^\epsilon \cdot d \cdot k$ for some constant ϵ. This ensures that the sub-exponentially secure primitives, parameterized by λ', are secure against adversaries with running time $\mathsf{poly}(\lambda, 2^{d\cdot k})$.

$\mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell) \to (\mathsf{vk}, \mathsf{sk})$. The Setup algorithm takes as input security parameter λ, number of maximum levels of homomorphic evaluation k, and circuit class $\mathcal{C}_\ell$. It first sets Merkle Tree hash key

$$\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'}),$$

and generates signature secret key and verification key

$$(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'}).$$

Consider that $\mathcal{C}_\ell$ has a total number of N wires and among which there are 2ℓ input wires, then the circuit has $N-2\ell$ internal wires and $N-2\ell$ gates. For all $i \in \{1, \ldots, k\}$, it generates seBARG common reference strings as the following:

$$(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N-2\ell, (N-2\ell, N-2\ell)),$$

$$(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1).$$

It sets NIZK CRS

$$(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{nizk.Setup}(1^{\lambda'}, \mathcal{L}_i^2).$$

It sets public key encryption key

$$(\mathsf{pke.pk}, \mathsf{pke.sk}) \leftarrow \mathsf{pke.Setup}(1^{\lambda'}).$$

It sets RAM delegation CRS with respect to RAM Machine $\mathcal{R}_i$:

$$\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T).$$

It computes the hash value of the implicit input of RAM Machine $\mathcal{R}_i$:

$$h_i^{\mathsf{imp}} = \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1)).$$

It sets $\mathsf{vk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$, and outputs

$$\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k), \mathsf{sk} = \mathsf{sk}_0.$$

Below we define circuit satisfiability language $\mathcal{L}_i^0$, $\mathcal{L}_i^1$, and $\mathcal{L}_i^2$, RAM Machine $\mathcal{R}_i$ for $i \in [k]$:

Language $\mathcal{L}_i^0$

Instance: x_j.
Witness: ω_j.
Output: Output 1 if and only if:
For $2\ell+1 \le j \le N$ where $x_j = (j, h, \tilde{C}_y)$ and $\omega_j = (b_j, b_{j_0}, b_{j_1}, \rho_j, \rho_{j_0}, \rho_{j_1})$, let gate c in monotone circuit $\tilde{C}_y$ be the gate that takes as input the j_0-th and the j_1-th wire, and outputs the j-th wire.

- For all $\alpha \in \{j, j_0, j_1\}$, $\mathsf{H.Verify}(\mathsf{hk}, h, \alpha, b_\alpha, \rho_\alpha) = 1$.
- $c(b_0, b_1) = b$.
- For $j = N$, it additionally requires that $b = 1$.

Language $\mathcal{L}_i^1$

Instance: x_j.
Witness: ω_j.
Hardwired: vk_{i-1}.
Output: Output 1 if and only if:
For $i = 0$ where $x_j = (j, h)$ and $\omega_j = (\sigma, b, \rho)$,

- $\mathsf{H.Verify}(\mathsf{hk}, h, j, b, \rho) = 1$,
- for $1 \le j \le \ell$, the language requires $\mathsf{sig.Verify}(\mathsf{vk}_{i-1}, b, \sigma, j) = 1$,
- otherwise for $\ell + 1 \le j \le 2\ell$, it requires $\mathsf{sig.Verify}(\mathsf{vk}_{i-1}, 1 - b, \sigma, j - \ell) = 1$.

For $i \ge 1$ where $x_j = (j, h, G, \{C_v\}_{v \in V})$ and $\omega_j = (\sigma, b, \rho)$,

- $\mathsf{H.Verify}(\mathsf{hk}, h, j, b, \rho) = 1$,
- consider extracting $(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}})$ from vk_{i-1} and $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$ from σ, for $1 \le j \le \ell$, $\mathsf{del.Verify}(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}}, (\mathsf{nizk}.\pi, b, \mathsf{ct}_h, G, \{C_v\}_{v \in V}), \mathsf{del}.\pi) = 1$,
- for $\ell + 1 \le j \le 2\ell$, $\mathsf{del.Verify}(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}}, (\mathsf{nizk}.\pi, 1 - b, \mathsf{ct}_h, G, \{C_v\}_{v \in V}), \mathsf{del}.\pi) = 1$.

Language $\mathcal{L}_i^2$

Instance: $\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}$.
Witness: $\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h$.
Output:

1. It first checks if $\mathsf{pke.Enc}(\mathsf{pke.pk}, r, h) = \mathsf{ct}_h$. If the check fails, it outputs 0 and aborts.
2. It sets root as the root of the G, and tree $G_j = (V_j, E_j)$ as the sub-tree rooted as the j-th child node of root. It sets $\tilde{C}_y$ as the monotone circuit of C_{root}.
3. For $j \in [\ell]$, it sets
$$x_j = (j, h, G_j, \{C_v\}_{v \in V_i}, \tilde{C}_y), x_{j+\ell} = (j + \ell, h, G_j, \{C_v\}_{v \in V_j}, \tilde{C}_y).$$
For $j \in \{2\ell + 1, \ldots, N\}$, it sets x_j as $(j, h, \tilde{C}_y)$.
4. It outputs 1 if and only if $\mathsf{barg.Verify}(\mathsf{barg.crs}_i^0, \{x_j\}_{j \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi) = 1$ and $\mathsf{barg.Verify}(\mathsf{barg.crs}_i^1, \{x_j\}_{j \in [2\ell]}, \mathsf{barg}.\pi) = 1$.

RAM Machine $\mathcal{R}_i$

Variable Input: $\mathsf{nizk}.\pi, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}$.
Fixed Input: $\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1$.
Output: $\mathcal{R}_i$ follows these steps:

1. $\mathcal{R}_i$ accepts iff $\mathsf{nizk.Verify}(\mathsf{nizk.crs}_i, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}), \mathsf{nizk}.\pi) = 1$.

$\mathsf{Sign}(\mathsf{sk}, i, b) \to \sigma$**.** It outputs the signature for message bit b at index i as

$$\sigma \leftarrow \mathsf{sig.Sign}(\mathsf{sk}, (i, b)).$$

$\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C) \to \sigma$. The evaluator algorithm for the t-th hop follows these steps:

1. Let y be the output of C over $(b_1, \ldots, b_\ell)$. The evaluator first generates the monotone circuit $\tilde{C}_y$ according to C, where monotone circuit $\tilde{C}_y$ has a total number of N wires. It sets $b_i = 1 - b_{i-\ell}$ for all $i \in \{2\ell+1, \ldots, N\}$. It then simulates monotone circuit $\tilde{C}_y$ gate by

gate, and records b_i as the value of the i-th wire of circuit $\tilde{C}_y$ for all $i \in \{2\ell+1, \ldots, N\}$. It sets hash value

$$h = \mathsf{H.Hash}(\mathsf{hk}, (b_1, \ldots, b_N)).$$

2. It generates the hash openings for all wires. For all $i \in [N]$, it computes

$$\rho_i = \mathsf{H.Open}(\mathsf{hk}, (b_1, \ldots, b_N), i).$$

3. It assigns an instance and a witness to each input wire. If for all $i \in [\ell]$, tree G_i is empty that does not contain any node, and the homomorphic evaluation involves the circuit C applied to newly signed messages. Then for $i \in [\ell]$,

$$x_i = (i, h), \omega_i = (\sigma_i, b_i, \rho_i),$$

$$x_{i+\ell} = (i+\ell, h), \omega_{i+\ell} = (\sigma_i, 1-b_i, \rho_{i+\ell}).$$

For all $i \in [\ell]$,

$$x_i = (i, h, G_i, \{C_v\}_{v \in V_i}, \tilde{C}_y), \omega_i = (\sigma_i, b_i, \rho_i),$$

$$x_{i+\ell} = (i+\ell, h, G_i, \{C_v\}_{v \in V_i}, \tilde{C}_y), \omega_{i+\ell} = (\sigma_i, 1-b_i, \rho_{i+\ell}).$$

4. It sets an instance and a witness for each internal wire. For every $i \in \{2\ell+1, \ldots, N\}$, take the i-th wire b_j. This wire is internal and serves as the output of some gate linked with two inputs to the gate b_{i_0}, b_{i_1}. To avoid misunderstanding, we clarify that the j_0-th and j_1-th wires are inputs to the gate and are not necessarily inputs to the circuit. It sets

$$x_i = (i, h, \tilde{C}_y), \omega_i = (b_i, b_{i_0}, b_{i_1}, \rho_j, \rho_{i_0}, \rho_{i_1}).$$

5. It generates

$$\mathsf{barg}.\pi^0 \leftarrow \mathsf{barg.Prove}(\mathsf{barg.crs}_t^0, \{x_i\}_{i \in \{2\ell+1,\ldots,N\}}, \{\omega_i\}_{i \in \{2\ell+1,\ldots,N\}}),$$

$$\mathsf{barg}.\pi^1 \leftarrow \mathsf{barg.Prove}(\mathsf{barg.crs}_t^1, \{x_i\}_{i \in [2\ell]}, \{\omega_i\}_{i \in [2\ell]}).$$

6. For all $i \in [\ell]$, let root_i denote the root node of Tree G_i. It generates node root and tree $G = (V, E)$ where root is the root of G:

$$V = \bigcup_{i=1}^{\ell} V_i \cup \{\mathsf{root}\}, E = \bigcup_{i=1}^{\ell} E_i \cup \{(\mathsf{root}, \mathsf{root}_1), \ldots, (\mathsf{root}, \mathsf{root}_\ell)\}.$$

It then assigns circuit C to the node root such that $C_{\mathsf{root}} = C$. Let t be the number of levels of G. One may also take t as the total number of hops including the current hop of evaluation.

7. It chooses randomness r and encrypts hash value h as $\mathsf{ct}_h = \mathsf{pke.Enc}(\mathsf{pke.pk}, h, r)$. It generates a NIZK proof

$$\mathsf{nizk}.\pi \leftarrow \mathsf{nizk.Prove}(\mathsf{nizk.crs}_i, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}), (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h)).$$

Next, it generates a RAM delegation proof

$$\mathsf{del}.\pi \leftarrow \mathsf{del.Prove}(\mathsf{del.crs}_t, (\mathsf{nizk}.\pi, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}), (\mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1)).$$

8. It outputs signature σ as $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$.

$\mathsf{Verify}(\mathsf{vk}, y, \sigma, G, \{C_v\}_{v\in V}) \to \{0,1\}$. Define t as the number of node levels in tree G, where t is the height of tree G plus one, and also represents the number of hops in the homomorphic signature. Parse σ as $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$. Parse vk_t as $(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{del.crs}_t, \mathsf{hk})$. It outputs

$$\mathsf{del.Verify}(\mathsf{del.crs}_t, h_t^{\mathsf{imp}}, (\mathsf{nizk}.\pi, y, \mathsf{ct}_h, G, \{C_v\}_{v\in V}), \mathsf{del}.\pi).$$

Completeness The completeness of our scheme directly follows from the completeness of public key encryption scheme pke, NIZK scheme nizk, public key signature scheme sig, somewhere extractable batch argument barg, RAM delegation del, and the monotone circuit transformation.

Efficiency We argue that efficiency requirements hold for the above design:

Theorem 6.2. Assume that $\mathsf{barg} = (\mathsf{barg.Setup}, \mathsf{barg.Prove}, \mathsf{barg.Verify}, \mathsf{barg.Extract})$ is a rate-1 somewhere extractable batch argument and $\mathsf{nizk} = (\mathsf{nizk.Setup}, \mathsf{nizk.Prove}, \mathsf{nizk.Verify})$ is also rate-1. Let $\mathcal{C}_\ell$ be the circuit class with maximum input size ℓ and depth d. We denote the maximum size of the circuit in class $\mathcal{C}_\ell$ as $|\mathcal{C}_\ell|$. Let k be the maximized hop of our homomorphic signature scheme. Then for every $\lambda, k, \ell \in \mathbb{N}$, the following holds:

- For $\mathsf{vk} \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$ and $\sigma \leftarrow \mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$ such that $t \le k$ and all of the above circuits are in circuit class $\mathcal{C}_\ell$, there exists a universal polynomial $\mathsf{poly}(\cdot,\cdot)$ such that $|\sigma| \le \mathsf{poly}(\lambda, k)$.
- Let $|V|$ denote the maximum number of circuits in Tree G. There exists a universal polynomial $\mathsf{poly}(\cdot,\cdot,\cdot,\cdot)$ such that the setup algorithm $\mathsf{Setup}(1^\lambda, 1^k, \mathcal{C}_\ell)$, evaluator algorithm $\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$, and verifier algorithm $\mathsf{Verify}(\mathsf{vk}, y, \sigma, G, \{C_v\}_{v\in V})$ run in time $\mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$.
- For some universal polynomial $\mathsf{poly}(\cdot,\cdot,\cdot,\cdot,\cdot)$, it holds that $|\mathsf{vk}| \le \mathsf{poly}(\lambda, k, |V|, |\mathcal{C}_\ell|)$.

Proof. Our proof is nearly identical to the proof of Theorem 5.3, since the constructions are very similar. The only difference is that we replace some inputs of RAM Machine $\mathcal{R}_i$ with a NIZK proof $\mathsf{nizk}.\pi$. Since the NIZK we applied is of rate-1, we conclude that the theorem follows from the proof of Theorem 5.3. □

6.2 Soundness

Theorem 6.3. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, nizk satisfies sub-exponential argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property, then our construction satisfies multi-hop unforgeability.

Proof. We first define Hybrid (j_0, j_1), over which we will later present an inductive proof. Note that the following is defined $2\ell + 1 \le j_0, j_1 \le N$.

Hybrid (j_0, j_1)

1. The attacker $\mathcal{A}$ starts by outputting a sequence of messages $(\beta_1, \ldots, \beta_\ell)$, a tree $G = (V, E)$, a set of circuits $\{C_v\}_{v \in V}$, and a message y^*.

2. Consider that the above tree G has a number of t levels. Challenger first sets hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and generates single-hop homomorphic signature key $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$.

3. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, it generates $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^0, N - 2\ell, (N - 2\ell, N - 2\ell))$. It generates $(\mathsf{barg.crs}_t^0, \mathsf{barg.td}_t^0) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_t^0, N - 2\ell, (j_0 - 2\ell, j_1 - 2\ell))$. Next for all $i \in \{1, \ldots, k\}$, it sets $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}_i^1, 2\ell, 1)$, $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{nizk.Setup}(1^{\lambda'}, \mathcal{L}_i^2)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, $h_i^{\mathsf{imp}} = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$.

4. For all $i \in [k]$, challenger sets $\mathsf{vk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. Challenger outputs $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k), \mathsf{sk} = \mathsf{sk}_0$. For all $i \in [k]$, the challenger computes and outputs $\sigma_i \leftarrow \mathsf{sig.Sign}(\mathsf{sk}, (i, \beta_i))$.

5. The attacker $\mathcal{A}$ outputs σ^*. Let y be the actual output of the structured circuits $\{C_v\}_{v \in V}$ taking $(\beta_1, \ldots, \beta_\ell)$ as input. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v \in V}) = 1$ and $y^* \neq y$.

Let root denote the root node of G. Assume without loss of generality that circuit C takes ℓ bits of input. For all $j \in [\ell]$, let $G_j = (V_j, E_j)$ represent the sub-tree rooted at the j-th child node of root. Next, for all $j \in [\ell]$, let b_j^* be the output by the structured circuit $\{C_v\}_{v \in V_i}$ over the input $(\beta_1, \ldots, \beta_\ell)$, and set $b_{j+\ell}^*$ as $1 - b_j^*$. Set circuit $\tilde{C}_{y^*}$ as the monotone circuit of circuit C_{root}. Let $(b_{2\ell+1}^*, \ldots, b_N^*)$ be the value of the rest of the wires of circuit $\tilde{C}_{y^*}$ taking $(b_1^*, \ldots, b_{2\ell}^*)$ as input. Parse the signature by $\mathcal{A}$ as $\sigma^* = (\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$. Next for $j \in [\ell]$, set $x_j = (j, h, G_j, \{C_v\}_{v \in V_i}, \tilde{C}_{y^*})$ and $x_{j+\ell} = (j + \ell, h, G_j, \{C_v\}_{v \in V_j}, \tilde{C}_{y^*})$. For $j \in \{2\ell + 1, \ldots, N\}$, set x_j as $(j, h, \tilde{C}_{y^*})$. Let the output of $\mathsf{nizk}.\mathcal{E}(\mathsf{nizk.td}_t, \mathsf{nizk}.\pi)$ be $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h)$. Let $(\omega_{j_0}, \omega_{j_1})$ be $\mathsf{barg.Extract}(\mathsf{barg.td}_t^0, \{x_i\}_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0)$. Parse ω_{j_b} as $(b, b_0, b_1, \rho, \rho_0, \rho_1)$ and set b_{j_b} as b. For adversary $\mathcal{A}$ in Hybrid (j_0, j_1), let $\mathsf{Adv}_{\mathcal{A}}^{j_0, j_1, b}$ denote the following:

$$\mathsf{Adv}_{\mathcal{A}}^{j_0, j_1, b} = \Pr_{\mathsf{hyb}_{j_0, j_1}}[\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v \in V}) = 1 \wedge b_{j_b} > b_{j_b}^*].$$

We note that Hybrid (N, N) corresponds to the original unforgeability game for multi-hop homomorphic signature scheme. We denote $\mathcal{A}$'s winning advantage such hybrid as $\mathsf{Adv}_{\mathcal{A}}$:

$$\mathsf{Adv}_{\mathcal{A}} = \Pr_{\mathsf{hyb}_{N,N}}[\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v \in V}) = 1 \wedge y^* \neq y].$$

Lemma 6.4. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, nizk satisfies sub-exponentially secure argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property. Assume that there exists a PPT adversary $\mathcal{A}$, a non-negligible function $\epsilon(\cdot, \cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{N,N,0} \geq \epsilon(\lambda, 2^{d \cdot k})$. Then within the second level of circuit $\tilde{C}_{y^*}$ (the first level are input wires), there exists some internal wire indexed at j^* such that $3^{d-2} \cdot \mathsf{Adv}_{\mathcal{A}}^{j^*, j^*, 0} \geq \mathsf{Adv}_{\mathcal{A}}^{N,N,0}$.

Proof. We show the following: For any node indexed at j on level i ($i > 1$), if there exists a non-negligible function $\epsilon(\cdot,\cdot)$ such that $\mathsf{Adv}^{j,j,0}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d\cdot k})$, then there exists at least one node j_α on level $i-1$, such that $3 \cdot \mathsf{Adv}^{j_\alpha,j_\alpha,0}_{\mathcal{A}} \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}}$. The above implies our lemma by a simple induction. To prove it, consider these claims:

Claim 6.5. Assume that the somewhere extractable BARG barg satisfies sub-exponentially secure index hiding, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that $\mathsf{Adv}^{j,j',0}_{\mathcal{A}} \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k})$.

Proof. Omitted as it is nearly identical to the proof of Claim 5.7. □

Claim 6.6. Assume that the RAM delegation scheme del satisfies sub-exponential soundness, then for any polynomial time adversary $\mathcal{A}$ in Hybrid (j, j'), there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b^*_j\\ \wedge\ \mathsf{nizk.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}^0_t, \mathsf{barg.crs}^1_t, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v\in V}), \mathsf{nizk}.\pi) = 1\end{array}\right] \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Omitted by proof of Claim 5.8. □

Claim 6.7. Assume that the NIZK scheme nizk satisfies sub-exponential secure argument of knowledge, then for any polynomial time adversary $\mathcal{A}$ in Hybrid (j, j'), there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b^*_j\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}^0_t, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\end{array}\right] \geq \mathsf{Adv}^{j,j,0}_{\mathcal{A}} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Assume towards contradiction that with probability larger than $\epsilon(\lambda, 2^{d\cdot k})$ where $\epsilon(\cdot,\cdot)$ is non-negligible, $\mathsf{barg.Verify}(\mathsf{barg.crs}^0_t, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 0$ and $\mathsf{nizk.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}^0_t, \mathsf{barg.crs}^1_t, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v\in V}), \mathsf{nizk}.\pi) = 1$.

There exists a reduction algorithm $\mathcal{B}$ that breaks the argument of knowledge property of NIZK scheme nizk. $\mathcal{A}$ first outputs $\{\beta_i\}_{i\in[\ell]}$, $G = (V, E)$, $\{C_v\}_{v\in V}$, and y^*. $\mathcal{B}$ sets $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{sig.Setup}(1^{\lambda'})$. For all $i \in \{1,\ldots,k\} \setminus \{t\}$, $\mathcal{B}$ generates $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_i, N-2\ell, (N-2\ell, N-2\ell))$ and $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{nizk.Setup}(1^{\lambda'}, \mathcal{L}^2_i)$. $\mathcal{B}$ generates $(\mathsf{barg.crs}^0_t, \mathsf{barg.td}^0_t) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_t, N-2\ell, (j-2\ell, j'-2\ell))$ and the challenger outputs $\mathsf{barg.crs}^0_t$. $\mathcal{B}$ queries the NIZK challenger with language $\mathcal{L}^2_t$ and the challenger replies with $\mathsf{nizk.crs}_t$. For all $i \in \{1,\ldots,k\}$, $\mathcal{B}$ generates $\mathsf{barg.crs}^1_i \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^1_i, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, $h^{\mathsf{imp}}_i = \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$, and sets vk_i as $(\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{nizk.crs}_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. $\mathcal{B}$ outputs vk as $(\mathsf{vk}_0,\ldots,\mathsf{vk}_k)$ and sk as sk_0. Next, $\mathcal{A}$ outputs $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$. $\mathcal{B}$ outputs $\mathsf{nizk}.\pi$.

By the contradictory assumption, $\mathsf{barg.Verify}(\mathsf{barg.crs}^0_t, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 0$ and thus the extracted witness is not a valid witness for $\mathcal{L}^2_t$. We conclude that with an advantage of $\epsilon(\lambda, 2^{d\cdot k})$, $\mathcal{B}$ breaks the sub-exponentially secure argument of knowledge property of nizk. Our claim holds by the above reduction with Claim 6.6. □

Claim 6.8. Assume that the seBARG barg satisfies sub-exponentially secure somewhere argument of knowledge property, then for any PPT adversary $\mathcal{A}$ in Hybrid (j, j') and $(\omega_j, \omega_{j'}) = \mathsf{barg.Extract}(\mathsf{barg.td}_t^0, \{x_i^0\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$, there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b_j^*\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\\ \wedge\ \omega_j, \omega_{j'} \text{ are valid witnesses for } \mathcal{L}_t^0\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Omitted by proof of Claim 5.9. □

Claim 6.9. Assume that the Hash Tree H satisfies sub-exponentially secure collision-resistance property, then for $\alpha \in \{0,1\}$ and PPT adversary $\mathcal{A}$ in Hybrid (j, j_α), there exists a negligible function $\mathsf{negl}(\cdot,\cdot)$ such that the following holds:

$$\Pr_{\mathsf{hyb}_{j,j_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b_j^*\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\\ \wedge\ \omega_j, \omega_{j_\alpha} \text{ are valid witnesses for } \mathcal{L}_t^0\\ \wedge\ b_{j_\alpha} = b_\alpha\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}(\lambda, 2^{d\cdot k}).$$

Proof. Omitted by proof of Claim 5.10. □

Claim 6.10. For any PPT adversary $\mathcal{A}$, there exists some $\alpha \in \{0,1\}$, such that

$$\Pr_{\mathsf{hyb}_{j,j_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v\in V}) = 1\\ \wedge\ b_j > b_j^*\\ \wedge\ \mathsf{barg.Verify}(\mathsf{barg.crs}_t^0, \{x_i\}_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\\ \wedge\ \omega_j, \omega_{j_\alpha} \text{ are valid witnesses for } \mathcal{L}_t^0\\ \wedge\ b_{j_\alpha} = b_\alpha\\ \wedge\ b_{j_\alpha} > b_{j_\alpha}^*\end{array}\right] \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}(\lambda, 2^{d\cdot k})}{2}.$$

Proof. Omitted by proof of Claim 5.11. □

Claim 6.11. For any PPT adversary $\mathcal{A}$, if there exists a non-negligible function $\epsilon(\cdot,\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda, 2^{d\cdot k})$, then there exists some $\alpha \in \{0,1\}$, such that

$$\mathsf{Adv}_{\mathcal{A}}^{j_\alpha, j_\alpha, 0} \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{3}.$$

Proof. Omitted by proof of Claim 5.12. □

Our lemma follows from a induction using Claim 6.11. □

Next, we switch to Hybrid (j^*, j_α^*) for $2\ell + 1 \leq j^* \leq N$ and $1 \leq j_\alpha^* \leq 2\ell$. In Hybrid (j^*, j_α^*), j^* represents an internal wire at the second level of $\tilde{C}_{y^*}$, while j_α^* denotes the index of an input wire at the bottom level. This input wire is one of the two that connect to the gate producing the j^*-th wire. The hybrid is exactly the same as Hybrid (j^*, j^*), except for the third item:

Hybrid (j^*, j^*_α)

3. For all $i \in \{1, \ldots, k\} \setminus \{t\}$, it generates $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_i, N - 2\ell, (N - 2\ell, N - 2\ell))$ and $(\mathsf{barg.crs}^1_i, \mathsf{barg.td}^1_i) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^1_i, 2\ell, 1)$. It sets $(\mathsf{barg.crs}^0_t, \mathsf{barg.td}^0_t) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^0_t, N - 2\ell, (j^*, j^*))$ and $(\mathsf{barg.crs}^1_t, \mathsf{barg.td}^1_t) \leftarrow \mathsf{barg.Setup}(1^{\lambda'}, \mathcal{L}^1_t, 2\ell, j^*_\alpha)$. Then for all $i \in [k]$, challenger sets $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{nizk.Setup}(1^{\lambda'}, \mathcal{L}^2_i)$, $\mathsf{del.crs}_i \leftarrow \mathsf{del.Setup}(1^{\lambda'}, T)$, $h^{\mathsf{imp}}_i = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$.

As the previous hybrids, we let root denote the root node of G. For all $j \in [\ell]$, let $G_j = (V_j, E_j)$ represent the sub-tree rooted at the j-th child node of root. Next, for all $j \in [\ell]$, let b^*_j be the output by the structured circuit $\{C_v\}_{v \in V_i}$ over the input $(\beta_1, \ldots, \beta_\ell)$, and set $b^*_{j+\ell}$ as $1 - b^*_j$. Set circuit $\tilde{C}_{y^*}$ to be the monotone circuit from C_{root}. Let $(b^*_{2\ell+1}, \ldots, b^*_N)$ be the value of the rest of the wires of circuit $\tilde{C}_{y^*}$ taking $(b^*_1, \ldots, b^*_{2\ell})$ as input. Consider σ^* as $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$. Next for $j \in [\ell]$, set $x_j = (j, h, G_j, \{C_v\}_{v \in V_i}, \tilde{C}_{y^*})$ and $x_{j+\ell} = (j + \ell, h, G_j, \{C_v\}_{v \in V_j}, \tilde{C}_{y^*})$. For $j \in \{2\ell + 1, \ldots, N\}$, set x_j as $(j, h, \tilde{C}_{y^*})$. Let the output of $\mathsf{nizk}.\mathcal{E}(\mathsf{nizk.td}_t, \mathsf{nizk}.\pi)$ be $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h)$. Let (ω^*_j, ω'_j) be $\mathsf{barg.Extract}(\mathsf{barg.td}^0_t, \{x_i\}_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0)$. We parse ω^*_j as $(b_{j^*}, b_0, b_1, \rho_{j^*}, \rho_0, \rho_1)$. Let $\omega_{j^*_\alpha}$ be $\mathsf{barg.Extract}(\mathsf{barg.td}^1_t, \{x_i\}_{i \in \{1, \ldots, 2\ell\}}, \mathsf{barg}.\pi^1)$ where $\omega_{j^*_\alpha} = (\sigma, b_{j^*_\alpha}, \rho_{j^*_\alpha})$.

Lemma 6.12. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, nizk satisfies sub-exponentially secure argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property. Assume that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{j^*, j^*, 0}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d \cdot k})$ for some non-negligible function $\epsilon(\cdot, \cdot)$ and for wire j^* at the second level of circuit C_{y^*} (the bottom internal wires excluding input wires), then there exists some input wire at j^*_α such that

$$\Pr_{\mathsf{hyb}_{j^*, j^*_\alpha}} \left[\begin{array}{l} \mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, G, \{C_v\}_{v \in V}) = 1 \\ \wedge\ \omega_{j^*_\alpha} \text{ is a valid witness for } \mathcal{L}^1_t \\ \wedge\ b_{j^*_\alpha} > b^*_{j^*_\alpha} \end{array} \right] \geq \frac{\mathsf{Adv}^{j^*, j^*, 0}_{\mathcal{A}}}{3}.$$

Proof. We omit the proof here as the proof proceeds identically to the proof of Lemma 5.13, except that we rely on the argument of knowledge property of nizk, as what we did in Claim 6.7. □

Lemma 6.13. Assume that barg satisfies sub-exponentially secure index hiding and somewhere argument of knowledge, nizk satisfies sub-exponentially secure argument of knowledge, del satisfies sub-exponential soundness, and H satisfies sub-exponentially secure collision-resistance property. Assume that there exists an PPT adversary $\mathcal{A}$ and a non-negligible function $\epsilon(\cdot, \cdot)$ such that $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(T(\lambda), 2^{d(\lambda)})$. There exists some internal wire indexed at j^* at the second level of circuit $\tilde{C}_{y^*}$, such that the following holds:

$$\mathsf{Adv}^{j^*, j^*, 0}_{\mathcal{A}} \geq \frac{\mathsf{Adv}_{\mathcal{A}}}{3^{d-1}}.$$

Proof. Omitted by proof of Lemma 5.18. □

Combining Lemma 6.12 and Lemma 6.13, we conclude: for any PPT adversary $\mathcal{A}$ that breaks the unforgeability of our scheme with advantage $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d \cdot k})$ at the t-th hop, there exists a reduction algorithm that breaks the unforgeability of the signature scheme at the $t-1$-th hop with advantage greater than or equal to $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^d}$. With a simple induction, $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda, 2^{d \cdot k})$ implies a polynomial time reduction algorithm which breaks the unforgeability of sig with advantage at least $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^{d \cdot k}}$, which is non-negligible. We conclude that our theorem follows. □

Next, we prove the context-hiding property:

Theorem 6.14. Assume that nizk satisfies zero knowledge property and pke satisfies semantic security then our construction satisfies multi-hop unforgeability.

Proof. We first define our simulated evaluator $\mathcal{S}$ as the following: $\mathcal{S}$ takes as input $(t, \{(G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$. It first sets up $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$. Next, it computes σ following from evaluator Eval step by step except for the following: At step 1, it sets hash value as all-zeros string 0. It skips step 2. For step 3 and 4, it only generates instance x_i. Next, it skips step 5. Then, at step 7, the simulator instead sets

$$\mathsf{ct}_h = \mathsf{pke.Enc}(\mathsf{pke.pk}, 0, r)$$

and generates

$$(\mathsf{nizk.crs}', \mathsf{nizk}.\pi') \leftarrow \mathsf{nizk}.\mathcal{S}(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v\in V}).$$

Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, simulator replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, \mathsf{hk})$.

To prove that the simulator satisfies context-hiding, we consider the following hybrids experiments:

Hybrid 0

1. The attacker $\mathcal{A}$ starts by outputting $(t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$. The challenger tosses a coin $b \in \{0, 1\}$.
2. For $b = 0$, the challenger sets up $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$ and computes
$$\sigma \leftarrow \mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C).$$
3. For $b = 1$, the challenger first sets up $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$. Challenger then computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v\in V_i})\}_{i\in[\ell]}, C)$ step by step, except for step 7. At step 7, the challenger instead sets
$$\mathsf{ct}_h = \mathsf{pke.Enc}(\mathsf{pke.pk}, 0, r)$$
and generates
$$(\mathsf{nizk.crs}', \mathsf{nizk}.\pi') \leftarrow \mathsf{nizk}.\mathcal{S}(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v\in V}).$$
Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, challenger replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$.
4. Challenger outputs (vk, σ). $\mathcal{A}$ outputs b' and wins if and only if $b' = b$.

Hybrid 1

1. The attacker $\mathcal{A}$ starts by outputting $t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C$. The challenger tosses a coin $b \in \{0,1\}$. The challenger tosses a coin $b \in \{0,1\}$.

2. For $b = 0$, the challenger first sets up $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$. Challenger then computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C)$ step by step, except for step 7. At step 7, the challenger instead generates

$$(\mathsf{nizk.crs}', \mathsf{nizk}.\pi') \leftarrow \mathsf{nizk}.\mathcal{S}(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}).$$

   Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, challenger replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$.

3. For $b = 1$, the challenger first sets up $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$. Challenger then computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C)$ step by step, except for step 7. At step 7, the challenger instead sets

$$\mathsf{ct}_h = \mathsf{pke.Enc}(\mathsf{pke.pk}, 0, r)$$

   and generates

$$(\mathsf{nizk.crs}', \mathsf{nizk}.\pi') \leftarrow \mathsf{nizk}.\mathcal{S}(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}).$$

   Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, challenger replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$.

4. Challenger outputs (vk, σ). $\mathcal{A}$ outputs b' and wins if and only if $b' = b$.

Hybrid 2

1. The attacker $\mathcal{A}$ starts by outputting $t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C$.

2. The challenger first sets up $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^k, \mathcal{C}_\ell)$. Challenger then computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C)$ step by step, except for step 7. At step 7, the challenger instead sets

$$\mathsf{ct}_h = \mathsf{pke.Enc}(\mathsf{pke.pk}, 0, r)$$

   and generates

$$(\mathsf{nizk.crs}', \mathsf{nizk}.\pi') \leftarrow \mathsf{nizk}.\mathcal{S}(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}).$$

   Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, challenger replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$.

3. Challenger outputs (vk, σ). $\mathcal{A}$ outputs b' and wins if and only if $b' = b$.

Denote $\mathcal{A}$'s advantage in hybrid j as $\mathsf{Adv}_{\mathcal{A}}^j$.

Lemma 6.15. Assume that nizk satisfies zero-knowledge, then for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^0 - \mathsf{Adv}_{\mathcal{A}}^1| \leq \mathsf{negl}(\lambda')$.

Proof. We prove the lemma using a reduction algorithm $\mathcal{B}$ that breaks the zero-knowledge property of nizk.

$\mathcal{A}$ starts by outputting $t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C$. Then, $\mathcal{B}$ samples $b \in \{0,1\}$. If $b = 0$, $\mathcal{B}$ sends $\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V}$ to the zero-knowledge challenger, where the challenger returns $(\mathsf{nizk.crs}, \mathsf{nizk}.\pi)$. Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, $\mathcal{B}$ replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$. Otherwise if $b = 1$, $\mathcal{B}$ proceeds as the challenger of step 3 at Hybrid 0 and 1. $\mathcal{B}$ sends (vk, σ) to $\mathcal{A}$ and $\mathcal{A}$ outputs b'. $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise.

Assuming towards contradiction that $|\mathsf{Adv}_{\mathcal{A}}^0 - \mathsf{Adv}_{\mathcal{A}}^1| \geq \epsilon(\lambda')$ for some non-negligible function $\epsilon(\cdot)$, $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise breaks the zero-knoweldge property of the underlying nizk. □

Lemma 6.16. Assume that pke satisfies semantic-security, then for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^1 - \mathsf{Adv}_{\mathcal{A}}^2| \leq \mathsf{negl}(\lambda')$.

Proof. We prove the lemma using a reduction algorithm $\mathcal{B}$ that breaks the zero-knowledge property of nizk.

$\mathcal{A}$ starts by outputting $t, \{(b_i, \sigma_i, G_i, \{C_v\}_{v \in V_i})\}_{i \in [\ell]}, C$. Then, $\mathcal{B}$ samples $b \in \{0,1\}$. If $b = 0$, $\mathcal{B}$ queries the pke challenger with 0 and h and challenger replies ct_h. Next, $\mathcal{B}$ generates $(\mathsf{nizk.crs}', \mathsf{nizk}.\pi') \leftarrow \mathsf{nizk}.\mathcal{S}(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, G, \{C_v\}_{v \in V})$. Next, for $\mathsf{vk} = (\mathsf{vk}_0, \ldots, \mathsf{vk}_k)$, $\mathcal{B}$ replaces $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}_t, \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ using $\mathsf{vk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{nizk.crs}', \mathsf{pke.pk}, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$. Otherwise if $b = 1$, $\mathcal{B}$ proceeds as the challenger of step 3 at Hybrid 1. $\mathcal{B}$ sends (vk, σ) to $\mathcal{A}$ and $\mathcal{A}$ outputs b'. $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise.

Assuming towards contradiction that $|\mathsf{Adv}_{\mathcal{A}}^1 - \mathsf{Adv}_{\mathcal{A}}^2| \geq \epsilon(\lambda')$ for some non-negligible function $\epsilon(\cdot)$, $\mathcal{B}$ breaks the semantic security pke. □

In Hybrid 3, any adversary $\mathcal{A}$ has at most $1/2$ winning advantage, which implies $|\mathsf{Adv}_{\mathcal{A}}^0 - 1/2| \leq \mathsf{negl}(\lambda')$. We conclude that our simulator satisfies context hiding.

□

Corollary 6.17. Assuming sub-exponential LWE, there exists a multi-hop homomorphic signature scheme satisfying context hiding.

Proof. [GGI+15] proposed a design of rate-1 NIZK of argument of knowledge using homomoprhic encryption scheme which is only known to be built from the learning with error assumption. [DGKV22] proposed a design of rate-1 somewhere extractable BARGs from learning with error. Thus by Theorem 6.2, Theorem 6.3, and Theorem 6.14, our corollary follows. □

References

[AB09] Shweta Agrawal and Dan Boneh. Homomorphic macs: Mac-based integrity for network coding. In *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings 7*, pages 292–305. Springer, 2009.

[ABBF10] Shweta Agrawal, Dan Boneh, Xavier Boyen, and David Mandell Freeman. Preventing pollution attacks in multi-source network coding. In *Public Key Cryptography–PKC*

2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13, pages 161–176. Springer, 2010.

[ABC+07] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609, 2007.

[ABC+12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. In *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings 9*, pages 1–20. Springer, 2012.

[ABF24] Gaspard Anthoine, David Balbás, and Dario Fiore. Fully-succinct multi-key homomorphic signatures from standard assumptions. *Cryptology ePrint Archive*, 2024.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

[AKK09] Giuseppe Ateniese, Seny Kamara, and Jonathan Katz. Proofs of storage from homomorphic identification protocols. In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15*, pages 319–333. Springer, 2009.

[AL11] Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In *Public Key Cryptography–PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings 14*, pages 17–34. Springer, 2011.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from lwe and pairings in the standard model. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, pages 149–178. Springer, 2020.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and lwe. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 13–43. Springer, 2020.

[BBK+23] Zvika Brakerski, Maya Farber Brodsky, Yael Tauman Kalai, Alex Lombardi, and Omer Paneth. Snargs for monotone policy batch np. In *Annual International Cryptology Conference*, pages 252–283. Springer, 2023.

[BCFL23] David Balbás, Dario Catalano, Dario Fiore, and Russell WF Lai. Chainable functional commitments for unbounded-depth circuits. In *Theory of Cryptography Conference*, pages 363–393. Springer, 2023.

[BCJP24] Maya Farber Brodsky, Arka Rai Choudhuri, Abhishek Jain, and Omer Paneth. Monotone-policy aggregate signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 168–195. Springer, 2024.

[BDGM20] Zvika Brakerski, Nico Doettling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In *39th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 79–109. Springer Basel AG, 2020.

[BF11a] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*, pages 149–168. Springer, 2011.

[BF11b] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *International Workshop on Public Key Cryptography*, pages 1–16. Springer, 2011.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography–PKC 2009: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings 12*, pages 68–87. Springer, 2009.

[BFS14] Xavier Boyen, Xiong Fan, and Elaine Shi. Adaptively secure fully homomorphic signatures based on lattices. *Cryptology ePrint Archive*, 2014.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 416–432. Springer, 2003.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Kalai. Non-interactive delegation and batch np verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 474–482, 2017.

[BKP+23] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. *Cryptology ePrint Archive*, 2023.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871, 2014.

[BWW23] Eli Bradley, Brent Waters, and David J Wu. Batch arguments to nizks from one-way functions. *Cryptology ePrint Archive*, 2023.

[CF13] Dario Catalano and Dario Fiore. Practical homomorphic macs for arithmetic circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 336–352. Springer, 2013.

[CFT22] Dario Catalano, Dario Fiore, and Ida Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 159–188. Springer, 2022.

[CFW12] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In *Public Key Cryptography–PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings 15*, pages 680–696. Springer, 2012.

[CG24] Jiaqi Cheng and Rishab Goyal. Boosting snarks and rate-1 barrier in extractable proofs. *Unpublished manuscript*, 2024.

[CGJ+23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and snargs from sub-exponential ddh. In *Annual International Cryptology Conference*, pages 635–668. Springer, 2023.

[CJJ21a] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for np from standard assumptions. In *Annual International Cryptology Conference*, pages 394–423. Springer, 2021.

[CJJ21b] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Snargs for $\mathcal{P}$ from lwe. Cryptology ePrint Archive, Paper 2021/808, 2021. https://eprint.iacr.org/2021/808.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography: 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II 12*, pages 468–497. Springer, 2015.

[CW23] Jeffrey Champion and David J Wu. Non-interactive zero-knowledge from non-interactive batch arguments. In *Annual International Cryptology Conference*, pages 38–71. Springer, 2023.

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd Annual Symposium on Foundations of Computer Science*, pages 1057–1068. IEEE Computer Society Press, October / November 2022.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4*, pages 119–136. Springer, 2001.

[DVW09] Yevgeniy Dodis, Salil Vadhan, and Daniel Wichs. Proofs of retrievability via hardness amplification. In *Theory of Cryptography: 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings 6*, pages 109–127. Springer, 2009.

[EKK18] Ali El Kaafarani and Shuichi Katsumata. Attribute-based signatures for unbounded circuits in the rom and efficient instantiations from lattices. In *Public-Key Cryptography–PKC 2018: 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II 21*, pages 89–119. Springer, 2018.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In *International conference on the theory and application of cryptology and information security*, pages 499–530. Springer, 2016.

[Fre12] David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *Public Key Cryptography–PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings 15*, pages 697–714. Springer, 2012.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[GGI+15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, 28(4):820–843, 2015.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 465–482. Springer, 2010.

[GKKR10] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In *Public Key Cryptography–PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13*, pages 142–160. Springer, 2010.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.

[GM19] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pages 173–201. 2019.

[Goy24] Rishab Goyal. Mutable batch arguments and applications. Cryptology ePrint Archive, Paper 2024/737, 2024. https://eprint.iacr.org/2024/737.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749, 2021.

[GQWW19] Rishab Goyal, Willy Quach, Brent Waters, and Daniel Wichs. Broadcast and trace with ciphertext size from standard assumptions. In *Annual International Cryptology Conference*, pages 826–855. Springer, 2019.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[GSWW22] Rachit Garg, Kristin Sheridan, Brent Waters, and David J Wu. Fully succinct batch arguments for np from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 526–555. Springer, 2022.

[GU24] Romain Gay and Bogdan Ursu. On instantiating unleveled fully-homomorphic signatures from falsifiable assumptions. In *IACR International Conference on Public-Key Cryptography*, pages 74–104. Springer, 2024.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 469–477, 2015.

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In *IACR International Workshop on Public Key Cryptography*, pages 3–33. Springer, 2019.

[GW11a] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 99–108, 2011.

[GW11b] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, June 2011.

[GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 301–320. Springer, 2013.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. Snargs for p from sub-exponential ddh and qr. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 520–549. Springer, 2022.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over f p, dlin, and prgs in nc 0. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 670–699. Springer, 2022.

[JMSW02] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *Cryptographers' track at the RSA conference*, pages 244–262. Springer, 2002.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[KLV23] Yael Tauman Kalai, Alex Lombardi, and Vinod Vaikuntanathan. Snargs and ppad hardness from the decisional diffie-hellman assumption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 470–498. Springer, 2023.

[KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and ram delegation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1545–1552, 2023.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1115–1124, 2019.

[KVZ21] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and snargs. In *Theory of Cryptography Conference*, pages 330–368. Springer, 2021.

[LRY16] Benoît Libert, Somindu C Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In *43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, 2016.

[LTWC18] Russell WF Lai, Raymond KH Tai, Harry WH Wong, and Sherman SM Chow. Multi-key homomorphic signatures unforgeable under insider corruption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 465–492. Springer, 2018.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, Heidelberg, August 1988.

[Mic94] S. Micali. Cs proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 436–453, 1994.

[MPR11] Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *Cryptographers' track at the RSA conference*, pages 376–392. Springer, 2011.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003.

[NWW23] Shafik Nassar, Brent Waters, and David J Wu. Monotone policy bargs from bargs and additively homomorphic encryption. *Cryptology ePrint Archive*, 2023.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[PHGR16] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 59(2):103–112, 2016.

[PP22] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *63rd Annual Symposium on Foundations of Computer Science*, pages 1045–1056. IEEE Computer Society Press, October / November 2022.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 187–196, 2008.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th annual ACM symposium on Theory of computing*, 2005.

[RRR16] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 49–62, 2016.

[SW13] Hovav Shacham and Brent Waters. Compact proofs of retrievability. *Journal of cryptology*, 26(3):442–483, 2013.

[Tsa17] Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *Theory of Cryptography Conference*, pages 489–518. Springer, 2017.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–156. Springer, 2021.

[WW22] Brent Waters and David J Wu. Batch arguments for np and more from standard bilinear group assumptions. In *Annual International Cryptology Conference*, pages 433–463. Springer, 2022.

[WW24] Hoeteck Wee and David J Wu. Succinct functional commitments for circuits from k-lin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 280–310. Springer, 2024.

A General to Monotone Circuit Transformation

In this section, we will discuss how to transform a general circuit into a monotone circuit. Let ‘$C(m)$ be a general circuit. We will construct a monotone circuit $\tilde{C}$ s.t. $\tilde{C}(m, m \oplus 1^{|m|}) = C(m)$ and $|\tilde{C}| \leq 2|C|$.

Construction A.1 (General Circuit C to Monotone Circuit $\tilde{C}$ transformation)**.** For any circuit C^* let $\ell^j_{C^*}$ be the number of wires in the jth layer of C^* and let the input layer be layer 1, and the output layer be layer n_{C^*}.

Now consider any general circuit C with wires $(\alpha^j_i)_{j\in[n_{C^*}],i\in[\ell^j_{C^*}]}$ (where α^j_i is the ith wire in the jth layer of the circuit). We will construct a circuit $\tilde{C}$ with wires β^j_i where $\ell^j_{\tilde{C}} = 2\ell^j_C$. More specifically we will construct $\tilde{C}$ s.t. $\beta^j_{2i-1} = \alpha^j_i$, and $\beta^j_{2i} = \alpha^j_i \oplus 1$. We will construct this inductively (on the layer j).

Induction Base. For the input layer $j = 1$, since in addition to m_i we get $m_i \oplus 1$, we just need to do the following:

$$\beta^1_{2i-1} = m_i \ \wedge \ \beta^1_{2i} = m_i \oplus 1$$

Induction Step. Suppose the assumption holds for every layer $j_1, j_2 \in [j-1]$, we will show how to construct β^j_{2i-1} and β^j_{2i} for any α^j_i.

- Let α^j_i be the output of $\mathsf{AND}(\alpha^{j_1}_{i_1}, \alpha^{j_2}_{i_2})$. Then construct $\beta^j_{2i-1} = \mathsf{AND}(\beta^{j_1}_{2i_1-1}, \beta^{j_2}_{2i_2-1})$ and $\beta^j_{2i} = \mathsf{OR}(\beta^{j_1}_{2i_1}, \beta^{j_2}_{2i_2})$.
- Let α^j_i be the output of $\mathsf{OR}(\alpha^{j_1}_{i_1}, \alpha^{j_2}_{i_2})$. Then construct $\beta^j_{2i-1} = \mathsf{OR}(\beta^{j_1}_{2i_1-1}, \beta^{j_2}_{2i_2-1})$ and $\beta^j_{2i} = \mathsf{AND}(\beta^{j_1}_{2i_1}, \beta^{j_2}_{2i_2})$.
- Let α^j_i be the output of $\mathsf{NOT}(\alpha^{j_1}_{i_1})$. Then construct $\beta^j_{2i-1} = \beta^{j_1}_{2i_1}$ and $\beta^j_{2i} = \beta^{j_1}_{2i_1-1}$.

B Puncturable (All-But-One) Signatures for Single-Bit Messages

A puncturable signature for single-bit messages is defined the same as Definition 3.1 except that the message space is a single-bit. Note that the the signing algorithm is deterministic, thus w.l.o.g. we can consider the in the security game the adversary is allowed to make a single signing query before generating a forgery.

Construction B.1 (Single-Bit Puncturable Signatures)**.** Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be an injective PRG. We construct a single-bit puncturable signature $\mathsf{PSig} = (\mathsf{Setup}, \mathsf{Setup\text{-}Punc}, \mathsf{Sign}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$**.** It samples $x_1, x_2 \leftarrow \{0,1\}^\lambda$, computes $\mathsf{vk}_0 = G(x_0)$ and $\mathsf{vk}_1 = G(x_1)$, and lets $\mathsf{sk} = (x_0, x_1)$ and $\mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1)$.

$\mathsf{Setup\text{-}Punc}(1^\lambda, b^*) \to (\mathsf{vk}, \mathsf{sk})$**.** It samples $x_1, x_2 \leftarrow \{0,1\}^\lambda$, computes $G(x_0)$ and $G(x_1)$, and lets $\mathsf{sk} = (x_0, x_1)$ and $\mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1)$ where $\mathsf{vk}_b = G(x_b)$ if $b = 1 - b^*$ and $\mathsf{vk}_b \leftarrow \{0,1\}^{2\lambda}$ otherwise.

$\mathsf{Sign}(\mathsf{sk}, b) \to \sigma$**.** It parses $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$ and outputs sk_b.

$\mathsf{Verify}(\mathsf{vk}, b, \sigma) \to 0/1$**.** It parses $\mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1)$ and outputs 1 if $G(\sigma) = \mathsf{vk}_b$ and 0 otherwise.

Theorem B.2. Assuming injective PRGs, the Construction B.1 is a puncturable signatures for single-bit messages.

Proof. **Correctness of Setup.** Follows directly from the construction correctness of the PRG.

Correctness of Punctured Setup. For a punctured message b^*, $G(\sigma) = \mathsf{vk}_{b^*}$ for any σ can only happen if the randomly sampled vk_{b^*} is in the span on $G(x)$ which happens with probability $2^{\lambda}/2^{2\lambda} = 2^{-\lambda}$.

Verification Keys indistinguishability. The verification keys indistinguishability follows directly from the indistinguishability of the optput of the PRG from a random value.

□

Remark B.3. In the above approach if we replace the injective PRG with a perfectly binding commitment, the statisitical error of the $2^{-\lambda}$ goes away and we get perfect correctness of punctured setup.

Corollary B.4. Assuming either LWE, DLIN, or DDH, single-bit puncturable signatures exist.

C Single-Hop Homomorphic Signature from Monotone-Policy Aggregate Signatures

In this section we show how to construct single-hop homomorphic signatures generically from aggregate signatures. Let us first define aggregate signatures.

C.1 Aggregate Signatures

Syntax. Let $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme with message space $\{0,1\}^{\lambda}$. A monotone-policy aggregate signature AggSig consists of the following polynomial time algorithms:

$\mathsf{Setup}(1^{\lambda}, 1^{k}, 1^{\mathsf{s}_{\tilde{C}}}) \to \mathsf{crs}$**.** On input a security parameter λ, the number of signers k, and a bound $\mathsf{s}_{\tilde{C}}$ on the policy size, the setup algorithm outputs a common reference string crs.

$\mathsf{Aggregate}(\mathsf{crs}, m, \tilde{C}, (\mathsf{vk}_i, \sigma_i)_{i\in[k]}) \to \sigma$**.** The Aggregate algorithm takes as input common reference string crs, a message $m\{0,1\}^{\lambda}$, a policy circuit $\tilde{C} : \{0,1\}^{k} \to \{0,1\}$, verifcation key/signature pairs $(\mathsf{vk}_i, \sigma_i)$, the aggregation algorithm produces an aggregate signature σ.

$\mathsf{AggVerify}(\mathsf{crs}, m, \tilde{C}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), \sigma) \to 0/1$**.** The Verify algorithm takes as input common reference string crs, a message m, a policy circuit $\tilde{C} : \{0,1\}^{k} \to \{0,1\}$, a tuple of k verification keys and an aggregate signature σ. The aggregate verification algorithm outputs a bit $b \in \{0,1\}$.

Definition C.1 (Aggregate Signatures)**.** An aggregate signature $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is required to satisfy the following properties:

Correctness. For all $\lambda \in \mathbb{N}$ and all $m \in \{0,1\}^\lambda$, all monotone circuits $\tilde{C} : \{0,1\}^k \to \{0,1\}$ and all key and signature tuples $\{(i, \mathsf{vk}_i, \sigma_i)\}_{i\in[k]}$ where $\tilde{C}(\mathsf{Verify}(\mathsf{vk}_1, m, \sigma_1), \ldots, \mathsf{Verify}(\mathsf{vk}_k, m, \sigma_l)) = 1$ it holds that

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\mathsf{AggVerify}(\mathsf{crs}, m, \tilde{C}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), \sigma) = 1 \;:\; \begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_{\tilde{C}}}) \\ \sigma \leftarrow \mathsf{Aggregate}(\mathsf{crs}, m, \tilde{C}, (\mathsf{vk}_1, \sigma_1, \ldots, (\mathsf{vk}_k, \sigma_k))\end{array}\right] = 1.$$

Succinctness. There exists a universal polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda, k, \mathsf{s}_{\tilde{C}} \in \mathbb{N}$, all messages $m \in \{0,1\}^\lambda$, all monotone circuits $\tilde{C} : \{0,1\}^k \to \{0,1\}$ and all pairs $\{(\mathsf{vk}_i, \sigma_i)\}$ where $i \in [k]$, the size of the aggregate signature σ in the correctness experiment satisfies that $|\sigma| = \mathsf{poly}(\lambda + \log|\tilde{C}|)$.

Static Security. For any adversary $\mathcal{A}$, define the static unforgeability experiment $\mathsf{exp}_{\mathcal{A}}(\lambda)$ as follows:

1. On input the security parameter λ, the adversary $\mathcal{A}$ outputs the number of parties 1^k, and a monotone policy $\tilde{C} : \{0,1\}^k \to \{0,1\}$.
2. The challenger samples key paris $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for all $i \in [n]$ and sends $\mathsf{vk}_1, \ldots, \mathsf{vk}_k$ to the adversary.
3. The adversary $\mathcal{A}$ can now issue signing queries. Each signing query consists of an index $i \in [n]$ and a message $m \in \{0,1\}^\lambda \backslash \{m^*\}$. The challenger responds with $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m)$.
4. After the adversary is finished making signing queries, it outputs a tuple of verification keys $(vk_1^*, \ldots, \mathsf{vk}_k^*)$.
5. The challenger replies with common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_{\tilde{C}}})$.
6. The adversary $\mathcal{A}$ can continue to make signing queries. The challenger responds to these exactly as before.
7. The adversary outputs the aggregate signature σ^*.
8. The output of the experiment is 1 if all of the following holds:
   - For all $i \in [k]$, let $b_i = 0$ if $\mathsf{vk}_i^* = \mathsf{vk}_j$ for some $j \in [n]$. Otherwise, let $b_i = 1$. Then, it holds that $\tilde{C}(b_1, \ldots, b_k) = 0$.
   - $\mathsf{AggVerify}(\mathsf{crs}, m^*, \tilde{C}, (\mathsf{vk}_1^*, \ldots, \mathsf{vk}_k^*), \sigma^*) = 1$.

   Otherwise, it outputs 0.

We say that the aggregate signature scheme satisfies static security if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr[\mathsf{exp}_{\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$.

Theorem C.2 ([NWW23])**.** Assuming monotone-policy BARGs there exists monotone-policy aggregate signatures when the underlying digital signatures is instantiated using puncturable signatures.

C.2 Construction

Construction C.3 (Single-Hop Homomorphic Signatures for General Circuits)**.** Let $\mathsf{AggSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Aggregate}, \mathsf{AggVerify})$ be a monotone-policy aggregate signature w.r.t. a puncturable signature $\mathsf{PSig} = (\mathsf{PSig.Setup}, \mathsf{PSig.Setup\text{-}Punc}, \mathsf{PSig.Sign}, \mathsf{PSig.Verify})$. We construct a single-hop homomorphic signature $\mathsf{HSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \to (\mathsf{pk}, \mathsf{sk})$**.** It runs $(\mathsf{sk}_{i,b}, \mathsf{vk}_{i,b}) \leftarrow \mathsf{PSig.Setup}(1^\lambda)$ for $i \in [k]$ and $b \in \{0,1\}$, and $\mathsf{crs} \leftarrow \mathsf{AggSig.Setup}(1^\lambda, 1^{2k}, 1^{\mathsf{s}_{\tilde{C}_y}})$ (where $\mathsf{s}_{\tilde{C}_y}$ is from Remark 4.3) and lets $\mathsf{sk} = (\mathsf{sk}_{i,b})_{i\in[k],b\in\{0,1\}}$, $\mathsf{pk} = ((\mathsf{vk}_{i,b})_{i\in[k],b\in\{0,1\}}, \mathsf{crs})$.

$\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$**.** It parses $\mathsf{sk} = (\mathsf{sk}_{i,b})_{i\in[k],b\in\{0,1\}}$ and then it computes the signature $\sigma_i = \mathsf{PSig.Sign}(\mathsf{sk}_{i,m_i}, 1)$.

$\mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C) \to \sigma$**.** It parses $\mathsf{pk} = ((\mathsf{vk}_{i,b})_{i\in[k],b\in\{0,1\}}, \mathsf{crs})$, then construct $\tilde{C}_y = \mathcal{T}(C, y)$ using Fig. 1 where $y = C(m)$, and outputs $\sigma = \mathsf{AggSig.Aggregate}(\mathsf{crs}, 1, \tilde{C}_y, (\mathsf{vk}_{i,b}, \sigma_i)_{i\in[k],b\in\{0,1\}})$.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, C) \to 0/1$**.** It parses $\mathsf{pk} = ((\mathsf{vk}_{i,b})_{i\in[k],b\in\{0,1\}}, \mathsf{crs})$, if $C = \emptyset$, parse $y = (i, m_i)$ and output whatever $\mathsf{PSig.Verify}(\mathsf{vk}_{i,m_i}, 1, \sigma)$ outputs, otherwise construct $\tilde{C}_y = \mathcal{T}(C, y)$ using Fig. 1, and outputs whatever $\mathsf{AggVerify}(\mathsf{crs}, 1, \tilde{C}_y, (\mathsf{vk}_{i,b})_{i\in[k],b\in\{0,1\}}, \sigma)$ outputs.

Theorem C.4. The Construction C.3 is an unforgeable single-hop homomorphic signature with evaluated signature size $\mathsf{poly}(\lambda, \log|C|)$, assuming aggregate signatures, and puncturable signatures.

Proof. **Correctness.** If $C(m) = 1$, then $\tilde{C}_y(m, m \oplus 1^k) = 1$. Thus it is sufficient to show that for $i \in [k]$ if $m_i = 1$, then $\mathsf{PSig.Verify}(\mathsf{vk}_{i,1}, 1, \sigma_i) = 1$ and for $i \in [k+1, 2k]$ if $m_{i-k} = 0$, then $\mathsf{PSig.Verify}(\mathsf{vk}_{i-k,0}, 1, \sigma_i) = 1$. Both of the above follow from the fact that $\sigma_i = \mathsf{PSig.Sign}(\mathsf{sk}_{i,m_i}, 1)$.

Succinctness. follows directly from the succinctness of AggSig.

Selective Unforgeability. Follows directly from the unforgeability of AggSig w.r.t. PSig.

□

Corollary C.5. The Construction C.3 is an unforgeable single-hop homomorphic signature with evaluated signature size $\mathsf{poly}(\lambda, \log|C|)$, assuming either LWE, or k-LIN over pairing groups for any constant $k \in \mathbb{N}$.

Proof. We conclude the proof by combining Theorems 3.2, 3.18, C.2 and C.4. □

Contents