Multi-Hop Multi-Key Homomorphic Signatures with Context Hiding from Standard Assumptions

Abtin Afshar
UW-Madison*

Jiaqi Cheng
UW-Madison†

Rishab Goyal
UW-Madison‡

Abstract

Homomorphic signatures are advanced systems that enable computations on authenticated data. They let us compute over a signature σ and corresponding data M to obtain an evaluated signature σ_C for any circuit C. Generally, any sequence of evaluated/non-evaluated signatures $\sigma_{C_1}, \sigma_{C_2} \ldots, \sigma_{C_\ell}$ can be homomorphically evaluated to generate a signature $\sigma_{C^* \circ\, C_1|\cdots|C_\ell}$.

Despite significant progress over the last decade, there is only a *singular* approach by Gorbunov-Vaikuntanathan-Wichs [STOC'15] for designing homomorphic signatures supporting *arbitrary homomorphic computations from standard falsifiable assumptions*. And, if we require the homomorphic signatures to satisfy context hiding, then the homomorphism property is broken! The current state of homomorphic signatures is even more dissatisfying in the multi-key setting, where we do not even have a single approach that achieves true compactness. A major open problem is to design homomorphic signatures supporting *arbitrary homomorphic computations from standard falsifiable assumptions*. Furthermore, achieving context hiding without sacrificing homomorphism will be a great plus.

We design the first homomorphic signature system that supports arbitrary homomorphic computations from a wide variety of standard falsifiable assumptions (such as decision-linear, or DDH, or learning with errors). We do not put any artificial restrictions on an evaluator. Any combination of evaluated signatures can be arbitrarily computed upon. The size of evaluated signatures and verification key grows polynomially in the circuit depth, but is otherwise independent of the circuit/data size. Our designs naturally generalize to multi-key homomorphism. This gives the first multi-key homomorphic signature system with full succinctness under the same set of assumptions. We also achieve full context hiding without sacrificing homomorphism, under the hardness of learning with errors. All our constructions satisfy full (adaptive) security.

*Email: abtin@cs.wisc.edu.
†Email: jiaqicheng@cs.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.

Contents

1 Introduction

Fully homomorphic signatures [JMSW02, AB09, BFKW09, BF11a, GVW15] enable computations on *secretly* signed data. They represent a significant advancement in cryptography, akin to fully homomorphic encryption [RAD+78, Gen09, BV14, BGV14, GSW13], and have proven to be quite useful for numerous applications. E.g., 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 verifying image transformations [DCB24]. Using homomorphic signatures, for any computable circuit C, we can derive a new signature $\sigma_{C,y}$ from a signature σ_M for data M. In words, $\sigma_{C,y}$ is an unforgeable token validating possession of a signature σ_M on some data M such that $C(M) = y$.

How to homomorphically evaluate over signed data? In most applications, we want the ability to homomorphically evaluate signatures in a continuous fashion. This means that we should also be able to evaluate any sequence of evaluated signatures $\{\sigma_{C_i,y_i}\}_{i \leq \ell}$ to generate a new evaluated signature $\sigma_{C^* \circ\, C_1|\cdots|C_\ell}$ for any circuit C^*. E.g., in one of the original motivating scenarios of network coding [AB09, BFKW09, BF11a], each router: (i) receives a sequence of signed messages from its preceding routers, (ii) homomorphically creates a random linear combination of incoming signatures, and (iii) sends the homomorphically evaluated signature to the next router. Such an operation is carried out iteratively by every router in the network. Thus, this needs continuous homomorphism over groups of evaluated signatures, each independently computed by a different router. Similar requirements are enforced by many classic applications such as certified data analysis, computation on outsourced data, certified redaction, etc [JMSW02, AB09, BFKW09, BF11a, GVW15].

On a more general note, it might be that a user cannot access the entire dataset, or its signature, or the full circuit description, at the time evaluation starts. This could be due to resource unavailability, or lack of authorization, or scalability/performance issues, etc. For example, consider a cloud provider (say Apple) storing petabytes of user data (e.g., images, documents) across a large group of independent servers. To counter storage of illicit material (e.g., copyrighted data, deepfake images), each server could store digital signatures proving user data provenance [DCB24]. Suppose an auditor (say FBI) wants the cloud provider to prove that none of its servers are storing any illicit material. Homomorphic signatures can be a powerful tool to assist such secure audits very efficiently. However, this is only possible as long as they support (distributed) evaluation where the entire signed data is not available at one place. Moreover, due to the sheer volume of data stored by even a single server, it could be impractical to run homomorphic evaluation, in one-shot, on the dataset stored even at a single server.

Capturing general homomorphism. To ensure maximum applicability and flexibility, a general approach for formalizing homomorphic signatures is to define every cryptographic operation atomically. Thus, consider that each bit of dataset $M = (m_1, \ldots)$ can be signed independently and asynchronously, i.e. $\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$. And, an evaluator can run homomorphic computation on any subset S of the signed dataset $\{\sigma_i\}_{i \in S}$, or any sequence of evaluated signatures $\{\sigma_{C_i,y_i}\}_i$ where $y_i = C_i(M)$. This gives a very general abstraction for homomorphic signatures, and is sometimes referred to as multi-hop evaluation. It enables all known applications, including those that we discussed above. It provides fully parallelizable data signing and signature evaluation, thereby resolving the challenges due to scalability, availability, and authorization.

The two core 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 states that an attacker cannot create an accepting signature σ^* for any circuit C^* and output y^*, such that $y^* \neq C^*(M)$ where M is the data that was signed.

Beyond succinctness and unforgeability, a variety of additional desirable features have been studied in the literature such as context hiding, fast verification, and multi-key evaluation (refer to [GVW15, FMNP16] for a detailed discussion.) Context hiding states that a malicious user should not learn anything about dataset M from $\sigma_{C,y}$, beyond what is revealed by C and y. The fast verification property states that a verifier can pre-process the circuit C to compute a short digest h_C, where an *online* verifier can verify signature $\sigma_{C,y}$, given y and h_C, in time independent of $|C|$. Finally, multi-key homomorphic signatures generalize regular homomorphic signatures to the multi-signer setting, where homomorphic evaluation can be performed on data signed by many different signers.

Our results. We design homomorphic signatures for arbitrary polynomial-sized circuits satisfying all aforementioned properties from a variety of standard assumptions such as DDH [DH76b, DH76a], decision-linear (DLIN) [BBS04], and learning with errors (LWE) [Reg05]. Formally, we prove:

Theorem (informal)**.** *Assuming sub-exponential hardness of* DLIN*, or* DDH*, or* LWE*, there exists a homomorphic signature scheme for all polynomial-sized circuits.*

Our signatures are (a) *fully-succinct*, support (b) *general multi-hop* and (c) *multi-key evaluation*, and satisfy (d) *fast verification*, and (e) *full (adaptive) unforgeability.* The sizes of signatures as well as the verification keys grow as $\mathsf{poly}(\lambda, d)$, where d is the maximum depth of the circuits that can be evaluated. Thus, the efficiency does not scale with the dataset size $|M|$, circuit size $|C|$, or the total number of signers n (in case of multi-key evaluation). Moreover, our LWE-based homomorphic signatures satisfy (f) *full context hiding.* While under DLIN, or DDH, we can make evaluated signatures (g) *context hiding at the cost of sacrificing homomorphism* (similar to [GVW15]).

Additionally, we show an interesting trade-off, where we can further reduce our signature size to not grow polynomially with the circuit depth/width as long as we restrict to a single evaluation over signed data. That is, for single-hop homomorphic signatures, we can obtain signature size of $\mathsf{poly}(\lambda, \log|C|)$, with only polynomial security loss. The central toolkit that we rely on is the recent exciting work on non-interactive batch arguments (BARGs) and its extensions [RRR16, BHK17, KPY19, CJJ21a, CJJ21b, DGKV22, PP22, BBK+23, KLVW23]. In the main body, we state our main theorems more generally.

Placing our results. We provide the *first* homomorphic signature system that supports *general homomorphic computations* from *(falsifiable) non-lattice assumptions such as* DLIN and DDH. All other homomorphic signatures, from standard falsifiable assumptions [Nao03], either rely on lattices [GVW15, BFS14, FMNP16], support 'chained' homomorphic evaluation [CFT22, BCFL23, KLVW23, WW24], or constant number of evaluations [Tsa17, EKK18, Goy24].

Chained homomorphic evaluation corresponds to sequential (straight-line) evaluation, where two or more evaluated signatures **cannot** be homomorphically evaluated. And, by constant number of hops, we mean that the total number of evaluations that can be performed on signed data is at most $O(1)$. Such restrictions prohibit using homomorphic signatures in many applications, including network coding and secure data auditing that we discussed earlier.

Moreover, our LWE-based construction is the *first* homomorphic signature system *with context hiding.* In the original construction by Gorbunov et al. [GVW15], one had to *turn off homomorphism to make evaluated signatures context hiding.* In our construction, we do not have such restrictions, and even our intermediate evaluated signatures are context hiding.

Finally, we give the *first* homomorphic signature system that satisfies *full-succinctness in the multi-key model from standard falsifiable assumptions* [FMNP16]. In prior works [FMNP16, FP18, SFVA21], the signature size grew polynomially with the number of signers, n. Whereas the size of our signatures does not grow with n, thereby achieving fully-succinctness from standard falsifiable assumptions. Moreover, our LWE-based multi-key homomorphic signature scheme also achieves *context hiding.*

Related work. Since early 2000s, homomorphic signatures with varying functionalities have been designed [JMSW02, ABC+07, SW13, DVW09, AKK09, AB09, BFKW09, GKKR10, BF11a, AL11, BF11b, CFW12, ABC+12, Fre12, GW13, CF13]. All of these had one or more restrictions (private verifiability, limited homomorphism). The only exception was the folklore approach based on succinct non-interactive arguments of knowledge (SNARKs) [Kil92, Mic94]. Basically, each homomorphic evaluation can be proved succinctly by a SNARK, and unforgeability follows from knowledge soundness of SNARKs. SNARKs are known to have strong implausibility results from falsifiable assumptions [GW11a, CGKS23]. In 2015, Gorbunov et al. [GVW15] designed the first fully-homomorphic signature scheme under standard lattice assumptions [Ajt96]. This was a major achievement in the study of homomorphic signatures. Since then, there has been a large body of exciting work [BFS14, LTWC18, FMNP16, Tsa17, EKK18, CFT22, BCFL23, KLVW23, GU24, WW24, Goy24] leading to many new designs and extensions.

State-of-the-art. Currently, there are four major approaches to design fully homomorphic signatures – algebraic constructions from lattices [GVW15, BFS14, FMNP16, Tsa17, EKK18], generic constructions from functional commitments and mutable batch arguments [CFT22, BCFL23, KLVW23, Goy24, WW24], from indistinguishability obfuscation [GU24], and from SNARKs [Kil92, Mic94, GW13, GVW15, LTWC18]. The first approach behind lattice-based schemes does not generalize to other cryptographic assumptions. And, as discussed earlier, they also do not satisfy context hiding or full succinctness in the multi-key model. The second approach from functional commitments and mutable batch arguments only support 'chained' or constant-hop homomorphism, thus do not achieve full homomorphism. The other two approaches, from obfuscation and SNARKs, rely on non-falsifiable assumptions [GW11a, CGKS23, GK16]. Although indistinguishability obfuscation can be constructed from a careful combination of sub-exponential-hardness of well-founded assumptions [JLS21, JLS22], our goal is to come up with direct constructions and *not* rely on combinations of multiple cryptographic assumptions[1].

Batch arguments. BARGs are powerful proof systems that provide succinct proofs for a batch of NP statements, where succinctness states that the proof size does not grow with the batch size. Over the last few years, BARGs have emerged as a powerful cryptographic tool, and we have numerous constructions from a variety of standard assumptions [CJJ21a, CJJ21b, KVZ21, WW22, HJKS22, DGKV22, PP22, CGJ+23, KLV23, KLVW23]. In this work, we rely on BARGs to design general-purpose homomorphic signatures.

Concurrent work. In a concurrent work, Anthoine, Balbás, and Fiore [ABF24] designed fully-succinct multi-key homomorphic signatures [FMNP16] by combining batch arguments and functional commitments [LRY16]. Their scheme only supports 'chained' (multi-hop) homomorphic evaluation, whereas the focus of our work is on general (multi-hop) homomorphic evaluation.

2 Technical Overview

Our starting point is the recent template for designing homomorphic signatures by Goyal [Goy24]. Their idea was to combine monotone-policy SNARGs for $\mathsf{batchNP}$ (henceforth monotone SNARGs) [BBK+23, NWW23] with vanilla signatures[2]. Below we briefly review BARGs and monotone SNARGs, as they will be essential tools throughout the sequel. Any reader familiar with these concepts can skip the next two paragraphs, and move to the simplified template for designing homomorphic signatures from monotone SNARGs.

Reviewing BARGs and monotone SNARGs. BARGs allow a prover to generate a succinct proof π for a batch of statements that $\{x_i \in \mathcal{L}\}_{i \leq k}$ for some NP language $\mathcal{L}$, where k denotes the batch size. Succinctness is defined as the size of proof π being independent of batch size k. Soundness states that an attacker cannot create an accepting proof for a batch of instances containing at least one invalid instance $x_i \notin \mathcal{L}$. Somewhere extractable BARGs (seBARGs) [CJJ21b, CJJ21a] are a mild strengthening of BARGs, which enable 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 a trapdoor key, associated with crs.

In a recent beautiful work, Brakerski et al. [BBK+23] introduced the concept of monotone SNARGs. These SNARGs enable computation of general *monotone circuits* over the validity of a batch of statements. That is, for any monotone circuit C, and any batch of statements $(x_1, \ldots, x_k)$, a prover can create a succinct proof to prove there exist witnesses $(\omega_1, \ldots, \omega_k)$ such that $C(b_1, \ldots, b_k) = 1$, where $b_i = \mathcal{R}(x_i, w_i)$ (i.e., b_i denotes satisfiability of instance-witness pair as per $\mathcal{L}$). Succinctness is defined as the proof size being independent of k as well as circuit size, $|C|$. Soundness states that an attacker cannot create an accepting proof, for a batch of instances $\{x_i\}_i$ and monotone circuit C, for which there does not exist a sequence of witnesses $\{\omega_i\}_i$ that satisfies the monotone circuit.

[1] While combining cryptographic assumptions is a very 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. We study the problem of homomorphic signatures with the same motivation.

[2] Although [Goy24] uses a more general template, described via a new primitive called *mutable* batch arguments [Goy24], it is possible to summarize the ideas using only monotone SNARGs.

Monotone SNARGs to homomorphic signatures. To sign the i-th bit of the dataset $M = (m_1, \ldots)$, the signer uses any vanilla signature scheme. That is, it signs the message (i, m_i) to compute σ_i. In order to compute a circuit C on the signed dataset M, the plan is for the evaluator to compute a monotone SNARG proof for C using $\{\sigma_i\}_i$ as witnesses. However, C could be non-monotonic, thus their [Goy24] idea was to deterministically encode C into a monotone circuit $\tilde{C}$ of similar size. At a high level, this works by encoding the input x to circuit C into a string $\tilde{x} = (x, x \oplus 1^{|x|})$, and ensuring that $C(x) = \tilde{C}(\tilde{x})$ for every x. Such translations are well known in the literature [Vad06, GPSW06], and for completeness, we provide it in Section 3.7.

Once we can encode C into a monotone circuit $\tilde{C}$, then homomorphic evaluation is relatively straightforward. The evaluator computes a monotone SNARG for $\tilde{C}$, where it uses σ_i as a witness for the i-th statement if $m_i = 1$, otherwise it uses it for the $(|m| + i)$-th statement. (The underlying NP relation $\mathcal{R}$ for the batch language portion corresponds to the signature verification circuit.) The resulting proof is viewed as an evaluated signature. To argue unforgeability, Goyal relied on the *full extraction* property of monotone SNARGs [BBK+23]. The intuition was to extract a sequence of satisfying witnesses for the monotone circuit $\tilde{C}$ from an evaluated signature/proof, and one of those witnesses will serve as a forgery on the vanilla signature scheme.

Does this handle general homomorphic evaluation? Recall that an evaluator could want to evaluate any sequence of evaluated signatures $\{\sigma_{C_i, y_i}\}_i$ as well, and not just plain (non-evaluated) signatures. A natural approach would be to use a monotone SNARG proof as a witness during future homomorphic evaluations. 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. At first glance, it might appear that this approach will be sufficient for handling all homomorphic computations. Unfortunately, this is not the case!

Why is composing monotone SNARGs insufficient? The naive approach of composing monotone SNARGs is quite limiting. In a few words, the issue is the large blow-up in the signature size due to composition of monotone SNARGs. This was noted in [Goy24], and this is why their construction only supports $O(1)$ evaluation hops. Simply put, the issue is that the size of 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.

Similar issues were recently faced [DGKV22] in the context of composing BARGs to design (unbounded) aggregate signatures [BGLS03] and incrementally verifiable computation protocols [Val08]. Their main insight was to design optimal-rate BARG proofs [DGKV22, PP22]. An optimal-rate BARG proof, also commonly referred to as rate-1 BARGs, have a special succinctness property which states $|\pi| = |\omega| + \mathsf{poly}(\lambda, \log k)$. That is, the size of the proof is equal to the size of a single witness, plus fixed additive polynomial terms.

First idea and why it fails! Our intuition is that using rate-1 BARGs should be enough to improve the efficiency of existing monotone SNARGs [BBK+23, NWW23], and this will help us get around the above recursive composition issue. Unfortunately, even if we are able to design such composable monotone SNARGs, this will not be enough! The issue is that for our homomorphic signature scheme to be unforgeable, we really need our monotone SNARGs to be composable as well as *fully extractable*. Moroever, to prove full (adaptive) unforgeability, we need the underlying monotone SNARGs to be adaptively-knowledge-sound. It is well-known that adaptively-knowledge-sound BARGs are as challenging to design as SNARKs [BHK17, Section 6.1]. Since monotone SNARGs with knowledge soundness are clearly more powerful than BARGs, thus this suggests that we will face well-known black-box barriers [GW11b] in designing *fully extractable* monotone SNARGs. We did not face this issue in single/constant-hop, since composability was not an issue.

Opening up monotone SNARGs: using somewhere extractable BARGs. As we explained above, 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 seBARGs as a core primitive and create the

SNARG proof as a batch proof. Therefore, intrinsically, a monotone SNARG construction does enjoy a 'somewhere-extraction-style' feature that we plan to 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 of all wire values, 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 batch 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 digest dig and a batch proof π, 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'. Here 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 evaluations. 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[3].

[3]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

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.

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 N witnesses can be generically built using N seBARGs that are extractable on a single witness. However, this brings down the proof-rate from 1 to $1/N$. Because, for each (single-witness-extractable) seBARG, the proof size grows as $|\omega| + \mathsf{poly}(\lambda)$. This means that even if we start with rate-1 seBARGs (which is the optimal rate possible for straightline-extraction[4]), 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, $\pi = 2|\omega_i| + \mathsf{poly}$ 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 enables logarithmic number of evaluation hops, our end goal is to enable general multi-hop homomorphic evaluation. Thus, plugging in a rate-1 seBARG directly into the above construction is not sufficient for handling an arbitrary polynomial number of hops.

Our core technique: Width-2 chained composition of rate-1 and poor-rate 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 (i.e., say a rate-$\frac{1}{2}$ proof) for the "composed" language (i.e., 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 (i.e., a rate-1 proof)*, while (2) for proving gate consistency, for the entire evaluated circuit, we use an seBARG proof system that is extractable on two indices (i.e., say a rate-$\frac{1}{2}$ proof).

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 an "atypical" fashion. Note that the invariant is 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 by an seBARG prover whose the rate is lower than 1. In words, the invariant that we maintain is that output of a poor-rate seBARG proof (which

construction. We leave proving unforgeability of our multi-hop homomorphic signatures from polynomial hardness as an interesting open problem.

[4] 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 facing Gentry-Wichs black-box barriers [GW11b].

is extractable at two indices) is never used as a witness in another poor-rate 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 (due to poor-rate) never gets cascaded.

It is crucial to note that the above guarantees that the recursive composition of seBARG proofs only happens on the first seBARG, which is rate-1. Note that any evaluated signature size is simply $|\pi^{(1)}_{\mathsf{BARG}}| + |\pi^{(2)}_{\mathsf{BARG}}| + |\mathsf{com}|$. Here the commitment com is just a Merkle tree hash, hence its size is $\mathsf{poly}(\lambda)$. Suppose that the signatures that we get as inputs for homomorphic evaluation are of size $\leq \ell_\sigma$. The first BARG proof, $\pi^{(1)}_{\mathsf{BARG}}$, is a rate-1 seBARG on the input signatures, thus its size is $|\ell_\sigma| + \mathsf{poly}(\lambda)$. While the second seBARG (extractable at two places) only checks for the gate consistency w.r.t. the commitment openings, thus its size is $\mathsf{poly}(\lambda)$. Therefore the output signature/proof size will be $\ell_{\sigma'} = |\ell_\sigma| + \mathsf{poly}(\lambda)$, as desired.

Despite this modification, we observe that a similar guessing-based top-down reduction strategy is sufficient. Except, at some points in the security reduction, we will extract from the rate-1 seBARG, while at other points we extract from the poor-rate 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. Below we provide a more detailed sketch of our multi-hop signature construction.

Our multi-hop homomorphic signatures. We use vanilla signatures, hash functions with local openings, and 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 setting), 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:

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

  4. Compute a poor-rate (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^{t-1}_i for a message y^{t-1}_i.

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 4.

Context-hiding and fast verification. Next, we show that the above template can be easily extended to enable context-hiding, and fast verification.

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. Gorbunov et al. [GVW15] proposed a simple generic template to obtain context-hiding property by applying NIZKAoKs 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 for context hiding. Our approach is to rely on the fairly standardized online/offline verification features of BARGs [CJJ21a], that is the verification algorithm of a BARG scheme can be split into a (slow) pre-verification and a (fast) online verification, and then use NIZKs to make our signatures context-hiding. We point out that rate-1 seBARGs [DGKV22, PP22] also satisfy such a online/offline verification property. Thus, our starting observation is that since our template also uses seBARGs, thus it satisfies a desired online/offline verification property.

Next, to make it 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 5.

Multi-key homomorphism. Finally, we show that the above construction template can be easily extended to support multi-key homomorphism. Recall that in multi-key model, 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 $\mathsf{Sign}(\mathsf{sk}_i, (i, m_i)) \to \sigma_i$. For simplicity, we are assuming that authority i signs message i. 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. For more details, we refer the reader to Section 6.

Additional results. The above concludes a high level overview of our main results. Additionally, we also provide another homomorphic signature scheme where the signature size is even shorter (i.e., it grows only logarithmically with circuit size, instead of polynomially with circuit depth). However, we can only enable such an optimization at the cost of giving up multi-hop evaluation. That is, our homomorphic signature with shorter signatures can only support one single homomorphic evaluation, but cannot be further composed. We provide an in-depth technical overview of our single-hop homomorphic signature scheme with shorter signatures later in Appendix A.1, and provide the full construction and proof in Appendix A. Lastly, we also show an interesting connection between a recently introduced generalization of aggregate signatures, called monotone-policy aggregate signatures [NWW23, BCJP24], and single-hop homomorphic signatures. We show that single-hop homomorphic signatures can also be designed from monotone-policy aggregate signatures in Appendix B.

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 Digital Signatures

Syntax. A signature (Sig) scheme 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 $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$**.** 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 (Digital Signature)**.** A digital signature $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ is required to satisfy the following properties:

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

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

EUF-CMA Security. For any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$,

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

where $\mathcal{A}$ is an admissible adversary if it never queries m to the signing oracle.

Theorem 3.2 ([NY89, Rom90])**.** Assuming one-way functions there exists digital signatures.

3.2 Public-Key Encryption

Syntax. A public key encryption (PKE) scheme for the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(\lambda) \to (\mathsf{pk}, \mathsf{sk})$**.** The probabilistic setup algorithm takes as input a security parameter λ and outputs the public and secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$**.** The probabilistic encryption algorithm takes as input the public key pk, a message $m \in \mathcal{M}_\lambda$, and outputs the ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m'$**.** The decryption algorithm takes as input secret key sk, ciphertext ct, and outputs m'.

Definition 3.3 (PKE)**.** A public-key encryption system $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ for $m \in \mathcal{M}_\lambda$ is required to satisfy the following properties:

Correctness. For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}_\lambda$, we have that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m$ where $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\lambda)$.

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

$$\left| \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk}, \cdot)}(1^\lambda, \mathsf{pk})\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk}, 0^{|m|})}(1^\lambda, \mathsf{pk})\right] \right| \leq \mathsf{negl}(\lambda)$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\lambda)$.

3.3 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, 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 $|x| \leq n_x$ 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 $\lambda, n_x \in \mathbb{N}$:

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

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 $\lambda, n_x \in \mathbb{N}$:

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

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 $\lambda, n_x \in \mathbb{N}$:

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

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.

3.4 Hash Tree

Syntax. Syntax of hash tree is as follows:

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

$\mathsf{Hash}(\mathsf{hk}, x) \rightarrow h$. 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 $\mathsf{poly}(\cdot)$.

$\mathsf{Open}(\mathsf{hk}, x, i) \rightarrow \rho$. 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 $\mathsf{poly}(\cdot, \cdot)$.

$\mathsf{Verify}(\mathsf{hk}, h, i, b, \rho) \rightarrow \{0,1\}$. 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.

3.5 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: an explicit input x_{exp} and an implicit 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.

3.6 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. A BARG scheme is said to be almost rate-1 if $|\pi| \le (1+c/\lambda)m + \mathsf{poly}(\lambda)$ for some constant c. Throughout the paper by rate-1 we refer to the almost rate-1 property of BARGs.

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

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

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

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

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.

3.7 General to Monotone Circuit Transformation

Here, we will recall how to transform a general circuit into a monotone circuit [Vad06, GPSW06]. 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 $|\tilde{C}| \le 2|C|$.

Construction 3.16 (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 $\beta^j_{2i} = \beta^{j_1}_{2i_1-1}$.

Transformation $\mathcal{T}$. Below we describe transformation $\mathcal{T}(C, y) \to \tilde{C}_y$ that constructs a 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 $C(m) = y$.

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

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: $C_y(m) \coloneqq \mathbb{1}[C(m) = y]$.
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: $\tilde{C}_y(m, m \oplus 1^k) \coloneqq C_y(m)$.

Remark 3.17. 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: Description of transformation $\mathcal{T}(C, y) \to \tilde{C}_y$.

4 Multi-Hop Homomorphic Signature

Before we proceed with the definition of multi-hop homomorphic signatures we need to define structured circuits[5] that is our way of denoting evaluation of different circuits in different hops over different inputs.

Structured circuit C. Let $\mathcal{C}_{\ell,d,\mathsf{s}_C}$ be a class of single-bit output circuits where ℓ is the maximum input size, d is the maximum depth, and s_C is the maximum size of any circuit $C \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$. We define structured circuit $\mathsf{C} = (G, (C_v)_{v \in V})$ where $G = (V, E)$ is a tree[6] and a circuit $C_v \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$ is associated to each node $v \in V$. Moreover the inputs to any circuit C_v are the outputs of circuits associated to v's child nodes. Furthermore, any input wire i to any circuit C_v such that v is a leaf in graph G, is labelled with id_i. Hence any circuit C_v with n_{in} inputs such that v is a leaf in graph G is associated with $(\mathsf{id}_i)_{i \in [n_{\mathsf{in}}]}$.

By **composing** $(\mathsf{C}_i)_{i \in [\ell]}$ and some $C \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$ we mean the following operation – if C_i is empty then let C be a graph G with a single node v^* where C is associated to v^*, (i.e. let $C_{v^*} = C$) and output C. Otherwise, do the following:

1. Let $\mathsf{C}_i = ((G_i = (V_i, E_i), (C_v)_{v \in V_i}))$ and v_i be the root of G_i.
2. Construct $G = (V, E)$ where $V = \bigcup_{i=1}^{\ell} V_i \cup \{v^*\}$ and $E = \bigcup_{i=1}^{\ell} E_i \cup \{(v_1, v^*), \ldots, (v_\ell, v^*)\}$.
3. Associate C to v^*, i.e. let $C_{v^*} = C$. Output $\mathsf{C} = (G, (C_v)_{v \in V})$.

By **decomposing** C to its children and a circuit C we mean finding all $(\mathsf{C}_i)_{i \in [\ell]}$ and a circuit C s.t. $(v_i, v) \in E$ where v_i is the root of G_i and v is the root of G and C is associated to v.

4.1 Definition

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

$\mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C) \to (\mathsf{pk}, \mathsf{sk})$. The setup algorithm takes as input security a parameter λ, number of maximum hops K, a max number of circuit inputs ℓ, a max circuit depth d, and a max circuit size s_C, and outputs verification/secret key $(\mathsf{pk}, \mathsf{sk})$.

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

$\mathsf{Eval}(\mathsf{pk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i \in [\ell']}, C) \to \sigma$. The evaluator algorithm takes as input a public verification key pk, a number of hops t, and a set of ℓ bits b_i with their corresponding signatures σ_i, and structured evaluation circuits C_i where $\ell' \le \ell$. Additionally, it takes a circuit $C \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$ s.t. C takes ℓ-bits inputs. The algorithm outputs a newly generated signature σ.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, \mathsf{C}) \to 0/1$. The verification algorithm takes as input a verification key pk, a message y, a signature σ, and a structured circuit C. It outputs a bit $0/1$ to signal whether σ is a valid signature.

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

Completeness. For any $\lambda, K, \ell, d, \mathsf{s}_C \in \mathbb{N}$, and $(b_i, \sigma_i, \mathsf{C}_i)_{i \in [\ell']}$ and circuit $C \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, y, \sigma, \mathsf{C}) = 1 \;:\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C), \\ \forall i \in [\ell'], \mathsf{Verify}(\mathsf{pk}, b_i, \sigma_i, \mathsf{C}_i) = 1, \\ \sigma = \mathsf{Eval}(\mathsf{pk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i \in [\ell']}, C), \\ y = C(b_1, \ldots, b_\ell) \end{array} \right] \ge 1 - \mathsf{negl}(\lambda),$$

[5] A similar concept is often called labelled circuits in the homomorphic evaluation literature.
[6] More on this on the structure of G in Section 4.4.

where C is the output of composing $(\mathsf{C}_i)_{i\in[\ell']}$ with C.

Efficiency. For the completeness experiment above the following hold:

- $|\mathsf{pk}|, |\sigma| \leq \mathsf{poly}(\lambda, d, K)$.
- Setup runs in time $\mathsf{poly}(\lambda, d, K)$, and evaluation and verification run in time $\mathsf{poly}(\lambda, |V|, \mathsf{s}_C)$ where $|V|$ denote the number of nodes in G where $G \in \mathsf{C}$.

Adaptive 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 $\lambda, \ell, d, \mathsf{s}_C \in \mathbb{N}$, the following holds:

$$\Pr\left[\begin{array}{l} y^* \neq y \\ \wedge\ \mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, \mathsf{C}^*) = 1 \end{array} : \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C) \\ (\mathcal{I}, (b_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}}) \leftarrow \mathcal{A}(\mathsf{pk}) \\ \forall \mathsf{id} \in \mathcal{I}, \sigma_{\mathsf{id}} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{id}, b_i) \\ (\mathsf{C}^*, y^*, \sigma^*) \leftarrow \mathcal{A}((\sigma_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where y is the actual output of the structured circuit C given $(b_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}}$.

4.2 Construction

Notation and parameters. For ease of exposition (and w.l.o.g.) we assume that any $C_v \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$ has exactly ℓ inputs, namely, $\ell' = \ell$. Throughout our construction, we set the security parameter for all underlying primitives to be $\lambda' = (\lambda + 4dK)^{O(1)}$. This ensures that we rely on their sub-exponential security: We define a "strong" negligible function $\mathsf{negl}'(\cdot)$, such that $\mathsf{negl}'(\lambda) \leq 2^{-\lambda^c}$ for some $0 < c < 1$. We assume that subexponentially-secure cryptographic primitives under security parameter λ' in our work to be secure against PPT adversaries, and every PPT adversary has at most $\mathsf{negl}'(\lambda') \leq \frac{1}{2^{\lambda+4dK}}$ advantage.

Let N denote the maximum number of wires and 2ℓ denote the number of input wires in any monotone circuit $\tilde{C}_y$ where $\tilde{C}_y = \mathcal{T}(C, y)$, $C \in \mathcal{C}_{\ell,d,\mathsf{s}_C}$, and y is C's output. Thus, the circuit $\tilde{C}_y$ has $N - 2\ell$ internal wires and $N - 2\ell$ gates.

Throughout this paper we will not explicitly give out the length for setup of BAGRs (and in the following sections for set up of NIZKs) as they're clear from the context. We emphasize that for simplicity of presentation we are assuming that all circuits have exactly ℓ inputs, thus an arbitrary tree G of such structure could be of large (exponential in depth) size. Hence the exact input size would depend up on the structure of $G \in \mathsf{C}$ which directly depends upon the actual value of ℓ' for every node in G.

Let $\mathsf{BARG} = (\mathsf{BARG.Setup}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify}, \mathsf{BARG.Extract})$ be a rate-1 somewhere extractable batch argument , $\mathsf{H} = (\mathsf{H.Setup}, \mathsf{H.Hash}, \mathsf{H.Open}, \mathsf{H.Verify})$ be a hash Tree, $\mathsf{Sig} = (\mathsf{Sig.Setup}, \mathsf{Sig.Verify})$ be a digital signature scheme, and $\mathsf{Del} = (\mathsf{Del.Setup}, \mathsf{Del.Prove}, \mathsf{Del.Digest}, \mathsf{Del.Verify})$ be a RAM delegation scheme. We present our multi-hop homomomorphic signature as follows:

$\mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C) \rightarrow (\mathsf{pk}, \mathsf{sk})$. The setup algorithm first samples a hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$, and generates signature secret key and verification key $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{Sig.Setup}(1^{\lambda'})$, and let $\mathsf{pk}_0 = \mathsf{vk}_0$ and for all $i \in [K]$, it does the following

1. Generates seBARG parameters as follows for languages $\mathcal{L}_i^0, \mathcal{L}_i^1$:

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

   $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$ for language $\mathcal{L}_i^1$ (Fig. 3).

2. Samples RAM delegation CRS with respect to machine $\mathcal{R}_i$ (Fig. 4) as $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, 2^\lambda)$.
3. Computes hash of implicit input of $\mathcal{R}_i$ as $h_i^{\mathsf{imp}} = \mathsf{Del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j\in[i-1]}))$.
4. Sets $\mathsf{pk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$.

Finally it outputs $\mathsf{pk} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_K), \mathsf{sk} = \mathsf{sk}_0$.

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

Language $\mathcal{L}_i^0$

Hardwired: hk.

Instance: $x = (j, h, \tilde{C}_y)$.

Witness: $\omega = (b_j, b_{j_0}, b_{j_1}, \rho_j, \rho_{j_0}, \rho_{j_1})$.

Membership: 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 (where $2\ell + 1 \leq j \leq N$). ω is a valid witness for $x \in \mathcal{L}_i^0$ if all of the followings are satisfied:

- For all $\alpha \in \{j, j_0, j_1\}$, $\mathsf{H.Verify}(\mathsf{hk}, h, \alpha, b_\alpha, \rho_\alpha) = 1$.
- $c(b_{j_0}, b_{j_1}) = b_j$.
- If $j = N$, then $b_j = 1$.

Figure 2: Description of language $\mathcal{L}_i^0$.

Language $\mathcal{L}_i^1$

Hardwired: pk_{i-1}.

Instance: x.

Witness: ω.

Membership: ω is a valid witness for $x \in \mathcal{L}_i^1$ if all of the following are satisfied:
If $i = 1$ let $x = (j, \mathsf{id}, h)$, $\omega = (\sigma, b, \rho)$, and check:

- $\mathsf{H.Verify}(\mathsf{hk}, h, j, b, \rho) = 1$,
- $\mathsf{Sig.Verify}(\mathsf{pk}_{i-1}, (\mathsf{id}, b), \sigma) = 1$ for $1 \leq j \leq \ell$,
- $\mathsf{Sig.Verify}(\mathsf{pk}_{i-1}, (\mathsf{id}, 1 - b), \sigma) = 1$ for $\ell + 1 \leq j \leq 2\ell$.

If $i \geq 2$, let $x = (j, h, \mathsf{C})$, $\omega = (\sigma, b, \rho)$, and $\sigma = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Additionally parse pk_{i-1} to find $(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}})$. Then check:

- $\mathsf{H.Verify}(\mathsf{hk}, h, j, b, \rho) = 1$,
- $\mathsf{Del.Verify}(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, b, h, \mathsf{C}), \mathsf{del}.\pi) = 1$, for $1 \leq j \leq \ell$.
- $\mathsf{Del.Verify}(\mathsf{del.crs}_{i-1}, h_{i-1}^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, 1 - b, h, \mathsf{C}), \mathsf{del}.\pi) = 1$ for $\ell + 1 \leq j \leq 2\ell$.

Figure 3: Description of the language $\mathcal{L}_i^1$.

RAM Machine $\mathcal{R}_i$

Explicit Input: $\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, \mathsf{C}$.

Implicit Input: $\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j\in[i-1]}$.

Output: $\mathcal{R}_i$ follows these steps:

1. Decomposes C to its children $(\mathsf{C}_j)_{j\in[\ell]}$ and a circuit C.
2. Sets monotone circuit $\tilde{C}_y = \mathcal{T}(C, y)$ following from Fig. 1.
3. For $j \in [\ell]$, sets $x_j = (j, h, \mathsf{C}_j)$, and $x_{j+\ell} = (j+\ell, h, \mathsf{C}_j)$.
4. For $j \in \{2\ell+1, \ldots, N\}$, sets x_j as $(j, h, \tilde{C}_y)$.
5. Accepts if and only if
   - $\mathsf{BARG.Verify}(\mathsf{barg.crs}_i^0, (x_j)_{j\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1$,
   - $\mathsf{BARG.Verify}(\mathsf{barg.crs}_i^1, (x_j)_{j\in[2\ell]}, \mathsf{barg}.\pi^1) = 1$.

Figure 4: Description of RAM Machine $\mathcal{R}_i$.

$\mathsf{Eval}(\mathsf{pk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C) \to \sigma$. The evaluation algorithm for the t-th hop follows these steps:

1. If C_i is empty for all i, then we assume that the circuit C is a labelled circuit where C' is the circuit and $(\mathsf{id}_i)_{i\in[\ell]}$ are labels for the input wires. It computes $y = C'(b_1, \ldots, b_\ell)$ and generates the monotone circuit $\tilde{C}_y = \mathcal{T}(C', y)$ using Fig. 1, where $\tilde{C}_y$ has a total number of N wires. It sets $b_i = 1 - b_{i-\ell}$ for all $i \in \{\ell+1, \ldots, 2\ell\}$. It then computes $\tilde{C}_y$ gate by gate, to find b_i (for all $i \in \{2\ell+1, \ldots, N\}$) as the value of the i-th wire of circuit $\tilde{C}_y$, and computes $h = \mathsf{H.Hash}(\mathsf{hk}, (b_1, \ldots, b_N))$.
2. For all $i \in [N]$, it computes the hash openings as $\rho_i = \mathsf{H.Open}(\mathsf{hk}, (b_1, \ldots, b_N), i)$.
3. It assigns an instance and a witness to each input wire. For all $i \in [\ell]$, if C_i is empty, then
   - $x_i = (i, \mathsf{id}_i, h)$, $\omega_i = (\sigma_i, b_i, \rho_i)$, and $x_{i+\ell} = (i+\ell, \mathsf{id}_i, h)$, $\omega_{i+\ell} = (\sigma_i, 1-b_i, \rho_{i+\ell})$.

   Otherwise,
   - $x_i = (i, h, \mathsf{C}_i)$, $\omega_i = (\sigma_i, b_i, \rho_i)$, and $x_{i+\ell} = (i+\ell, h, \mathsf{C}_i)$, $\omega_{i+\ell} = (\sigma_i, 1-b_i, \rho_{i+\ell})$.
4. It assigns an instance and a witness for each internal wire. For every $i \in \{2\ell+1, \ldots, N\}$, find a gate s.t. its output is wire i and let wires i_0 and i_1 be the inputs to such a gate. Then let
   - $x_i = (i, h, \tilde{C}_y)$, and $\omega_i = (b_i, b_{i_0}, b_{i_1}, \rho_j, \rho_{i_0}, \rho_{i_1})$.
5. It computes BARG proofs for $\mathcal{L}_i^1$ and $\mathcal{L}_i^2$ as follows:
   - $\mathsf{barg}.\pi^0 \leftarrow \mathsf{BARG.Prove}(\mathsf{barg.crs}_t^0, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, (\omega_i)_{i\in\{2\ell+1,\ldots,N\}})$.
   - $\mathsf{barg}.\pi^1 \leftarrow \mathsf{BARG.Prove}(\mathsf{barg.crs}_t^1, (x_i)_{i\in[2\ell]}, (\omega_i)_{i\in[2\ell]})$.
6. Compute C by composing $(\mathsf{C}_i)_{i\in[\ell]}$ and C.
7. It generates a RAM delegation proof:
   - $\mathsf{del}.\pi \leftarrow \mathsf{Del.Prove}(\mathsf{del.crs}_t, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, \mathsf{C}), (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, (\mathsf{pk}_j)_{j\in[t-1]}))$.
8. It outputs signature σ as $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, \mathsf{C}) \to \{0,1\}$. If C is empty then it outputs whatever $\mathsf{Sig.Verify}(\mathsf{pk}_0, (\mathsf{id}, y), \sigma)$ outputs. Otherwise it parses $\sigma = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$ and $\mathsf{pk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{del.crs}_t, h_t^{\mathsf{imp}}, \mathsf{hk})$ and outputs $\mathsf{Del.Verify}(\mathsf{del.crs}_t, h_t^{\mathsf{imp}}, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, \mathsf{C}), \mathsf{del}.\pi)$.

Remark 4.2. We apply a universal security parameter $\lambda' = (\lambda + 4dK)^{O(1)}$ for the design of the setup algorithm above. In fact for the above design where $\mathsf{pk} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$ and one may apply a tighter security parameter as $\lambda' = (\lambda + 4d \cdot (K - i))^{O(1)}$ when generating pk_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. Next, we analyze the efficiency of the above design.

Lemma 4.3. Assume that $\mathsf{BARG} = (\mathsf{BARG.Setup}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify}, \mathsf{BARG.Extract})$ 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 $\sigma_t \leftarrow \mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i \in [\ell]}, C)$.

Claim 4.4. 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 4.4 implies that $|\sigma| \le \mathsf{poly}(\lambda', K) \le \mathsf{poly}(\lambda, d, K)$. Next, we analyze the verification key size and verifier running time. For $t \in [K]$, $\mathsf{pk}_t = (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. By succinctness of rate-1 BARG barg, RAM delegation del, $|\mathsf{pk}_t|$ and setup running time for the t-th hop is bounded by $\mathsf{poly}(\lambda')$. Overall verification key size and setup running time is at most $\mathsf{poly}(\lambda', K) = \mathsf{poly}(\lambda, d, K)$.

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 $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, \mathsf{C})$ 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 4.4, 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 $\mathsf{poly}(n, |\mathcal{C}_{\ell,d,\mathsf{s}_C}|)$. Thus, verifier's running time is $\mathsf{poly}(\lambda', K, |V|, |\mathcal{C}_{\ell,d,\mathsf{s}_C}|)$. □

4.3 Unforgeability

Theorem 4.5. Assume that BARG satisfies sub-exponential index hiding and somewhere argument of knowledge, Del satisfies sub-exponential soundness, digital signature Sig satisfies sub-exponential unforgeability, and H satisfies sub-exponentially secure collision-resistance property, then our construction satisfies adaptive 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. Challenger first sets hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and generates single-hop homomorphic signature key $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Sig.Setup}(1^{\lambda'})$.

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

3. For all $i \in [K]$, challenger sets $\mathsf{pk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. Challenger outputs $\mathsf{pk} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_K), \mathsf{sk} = \mathsf{sk}_0$.

4. The attacker $\mathcal{A}$ outputs a sequence of messages $(\beta_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$, and the challenger computes and outputs $\sigma_{\mathsf{id}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, (\mathsf{id}, \beta_{\mathsf{id}}))$ for all $\mathsf{id} \in \mathcal{I}$.

5. The attacker $\mathcal{A}$ outputs a structured circuit C taking indexes $(\mathsf{id}_i)_{i \in [M]}$ as input. Let y^* be the output of C on input $(\beta_{\mathsf{id}_i})_{\mathsf{id}_i \in [M]}$. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}) = 1$ and $y^* \neq y$.

Let root denote the root node of G (G is the structure of C). Let $(\mathsf{C}_j)_{j \in [\ell]}$ be the decomposed circuit C. Set circuit $\tilde{C}_{y^*}$ as the monotone circuit of circuit C_{root} ($\tilde{C}_{y^*}$ follows Fig. 1). Let b_j^* be the output of C_j, and let $b_{j+\ell}^*$ be $1 - b_j^*$. Let $(b_{2\ell+1}^*, \ldots, b_N^*)$ be the value of the internal wires of circuit $\tilde{C}_{y^*}$ taking $(b_1^*, \ldots, b_{2\ell}^*)$ as input. Parse the signature by $\mathcal{A}$ as $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. For $j \in [\ell]$, set $x_j = (j, h, \mathsf{C}_j)$, and $x_{j+\ell} = (j + \ell, h, \mathsf{C}_{j+\ell})$. 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 $\mathsf{BARG.Extract}(\mathsf{barg.td}_t^0, (x_i)_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0)$. For $b \in \{0, 1\}$, parse ω_{j_b} as $(b_{j_b}, \ldots)$. For adversary $\mathcal{A}$ in Hybrid (j_0, j_1), let $\mathsf{Adv}_{\mathcal{A}}^{j_0, j_1, b}$ denote the following:

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

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 in such hybrid as $\mathsf{Adv}_{\mathcal{A}}$:

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

We also note that t is set as the current hop that $\mathcal{A}$ is attacking.

Lemma 4.6. 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 $\epsilon(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{N,N,0} \geq \epsilon(\lambda)$. 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 $3^{d-2} \cdot \mathsf{Adv}_{\mathcal{A}}^{j^*, j^*, 0} \geq \mathsf{Adv}_{\mathcal{A}}^{N,N,0}$ for every $\lambda \in \mathbb{N}$.

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

Claim 4.7. Assume that the BARG satisfies sub-exponentially secure index hiding, then for any PPT adversary $\mathcal{A}$, there exists a strongly-negligible function $\mathsf{negl}'(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j',0} \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda')$ for every $\lambda' \in \mathbb{N}$.

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

$\mathcal{B}$ 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 $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$ for language $\mathcal{L}^0_i$. $\mathcal{B}$ sets up $(\mathsf{barg.crs}^0_{t_0}, \mathsf{barg.td}^0_{t_0}) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, j-2\ell)$ for $\mathcal{L}^0_t$. $\mathcal{B}$ 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 $(\mathsf{barg.crs}^1_i, \mathsf{barg.td}^1_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, $h^{\mathsf{imp}}_i = \mathsf{Del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$, and $\mathsf{pk}_i = (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. It outputs pk as $(\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$ and sets sk as sk_0. $\mathcal{A}$ then outputs $(\beta_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$ and $\mathcal{B}$ returns $\mathsf{Sig.Sign}(\mathsf{sk}_0, \mathsf{id}, \beta_{\mathsf{id}})$ for $\mathsf{id} \in \mathcal{I}$. $\mathcal{A}$ outputs C and $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. 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 $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}) = 1$ 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 $\mathsf{barg.crs}^0_{t_1} \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, j-2\ell)$ and otherwise for $\beta = 1$, challenger returns $\mathsf{barg.crs}^0_{t_1} \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, j'-2\ell)$. Thus for $\beta = 0$, the above experiment corresponds to Hybrid (j, j) and otherwise it corresponds to Hybrid (j, j'). Assume towards contradiction that $|\mathsf{Adv}^{j,j',0}_{\mathcal{A}} - \mathsf{Adv}^{j,j,0}_{\mathcal{A}}|$ is not strongly negligible. Then, the above reduction algorithm $\mathcal{B}$ breaks the sub-exponentially secure index-hiding property of BARG. $\square$

Claim 4.8. 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 strongly negligible function $\mathsf{negl}'(\cdot)$ such that for every $\lambda' \in \mathbb{N}$:

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

Proof. Consider for Hybrid (j, j'), adversary $\mathcal{A}$ outputs C and $(h, \mathsf{barg}.\pi, \mathsf{del}.\pi)$ in step 5. 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 $\mathcal{L}^0_t$. Assume towards contradiction that with non strongly negligible probability, it satisfies that $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}) = 1$ and $\mathsf{BARG.Verify}(\mathsf{barg.crs}^0_t, (x_i)_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0) \neq 1$.

We design a simple reduction algorithm $\mathcal{B}$ that breaks the soundness property of RAM delegation scheme Del. $\mathcal{B}$ 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 $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, \{N-2\ell, N-2\ell\})$ and $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$. $\mathcal{B}$ sets $(\mathsf{barg.crs}^0_t, \mathsf{barg.td}^0_t) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (j-2\ell, j'-2\ell))$. $\mathcal{B}$ 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 $(\mathsf{barg.crs}^1_i, \mathsf{barg.td}^1_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$, $h^{\mathsf{imp}}_i \leftarrow \mathsf{Del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$, $\mathsf{pk}_i = (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. $\mathcal{B}$ outputs pk as $(\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$. $\mathcal{A}$ then outputs $(\beta_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$ and $\mathcal{B}$ returns $\mathsf{Sig.Sign}(\mathsf{sk}_0, \mathsf{id}, \beta_{\mathsf{id}})$ for $\mathsf{id} \in \mathcal{I}$. $\mathcal{A}$ outputs C, y^*, and $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Upon the output by $\mathcal{A}$, $\mathcal{B}$ outputs $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, \mathsf{C})$, $(\mathsf{barg.crs}^0_t, \mathsf{barg.crs}^1_t)$, and $\mathsf{del}.\pi$.

By the above assumption where it holds that $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}) = 1$, we have $\mathsf{Del.Verify}(\mathsf{del.crs}_t, h^{\mathsf{imp}}_t, (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, y, h, \mathsf{C}), \mathsf{del}.\pi) = 1$. By the assumption $\mathsf{BARG.Verify}(\mathsf{barg.crs}^0_t, (x^0_i)_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0) \neq 1$, hence $\mathcal{B}$ breaks the RAM delegation soundness with an advantage of $\epsilon(\lambda')$. Our claim holds by the reduction and by Claim 4.7. $\square$

Claim 4.9. Assume that the seBARG BARG is a sub-exponentially secure somewhere argument of knowledge, then for any PPT adversary $\mathcal{A}$ in Hybrid (j, j') and $(\omega_j, \omega_{j'}) = \mathsf{BARG.Extract}(\mathsf{barg.td}^0_t, (x^0_i)_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0)$, there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such for every $\lambda' \in \mathbb{N}$:

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

Proof. Assume towards contradiction that with some non strongly negligible probability, it holds that $\mathsf{BARG.Verify}(\mathsf{barg.crs}^0_t, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1$ and at least one of $\omega_j, \omega_{j'}$ is not a valid witness for $\mathcal{L}^0_t$.

There exists a reduction algorithm $\mathcal{B}$ that breaks the somewhere argument of knowledge property of seBARG scheme BARG. $\mathcal{B}$ sets $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and $(\mathsf{sk}_0, \mathsf{vk}_0) \leftarrow \mathsf{Sig.Setup}(1^{\lambda'})$. For all $i \in \{1,\ldots,K\}\backslash\{t\}$, $\mathcal{B}$ generates $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$. $\mathcal{B}$ queries the BARG challenger with $(1^{\lambda'}, N-2\ell, (j-2\ell, j'-2\ell))$ and the challenger outputs $\mathsf{barg.crs}^0_t$. For all $i \in \{1,\ldots,K\}$, $\mathcal{B}$ generates $\mathsf{barg.crs}^1_i \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, $h^{\mathsf{imp}}_i = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$, and sets pk_i as $(\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. $\mathcal{B}$ outputs pk as $(\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$. $\mathcal{A}$ then outputs $(\beta_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}}$ and $\mathcal{B}$ returns $\mathsf{Sig.Sign}(\mathsf{sk}_0, \mathsf{id}, (\beta_{\mathsf{id}}))$ for $\mathsf{id} \in \mathcal{I}$. $\mathcal{A}$ outputs C with $(h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. $\mathcal{B}$ outputs $(x_i)_{i\in\{2\ell+1,\ldots,N\}}$ and $\mathsf{barg}.\pi^0$.

By the contradictory assumption, $\mathsf{BARG.Verify}(\mathsf{barg.crs}^0_t, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1$ and either ω_j or $\omega_{j'}$ is not a valid witness for $\mathcal{L}^0_t$, 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, $\mathcal{B}$ breaks the sub-exponentially secure somewhere argument of knowledge property of BARG. Our claim holds by the above reduction and Claim 4.8. □

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 placed at level i of $\tilde{C}_{y^*}$ where $3 \le i \le d$, 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 $(\omega_j, \omega_{j_\alpha}) = \mathsf{BARG.Extract}(\mathsf{barg.td}^0_t, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$. 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 4.10. Assume that the Hash Tree H satisfies sub-exponential collision-resistance, then for $\alpha \in \{0,1\}$ and PPT adversary $\mathcal{A}$ in Hybrid (j, j_α), there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such that for every $\lambda' \in \mathbb{N}$:

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

Proof. Assume towards contradiction that with non strongly negligible probability, 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. $\mathcal{B}$ 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 $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$. $\mathcal{B}$ additionally sets $\mathsf{barg.crs}^0_t$ as $\mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (j-2\ell, j'-2\ell))$. For all $i \in \{1,\ldots,K\}$, $\mathcal{B}$ generates $\mathsf{barg.crs}^1 \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, $h^{\mathsf{imp}}_i = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$, and sets $\mathsf{pk}_i = (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. $\mathcal{B}$ outputs pk as $(\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$. $\mathcal{A}$ then outputs $(\beta_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}}$ and $\mathcal{B}$ returns $\mathsf{Sig.Sign}(\mathsf{sk}_0, \mathsf{id}, \beta_{\mathsf{id}})$ for $\mathsf{id} \in \mathcal{I}$. $\mathcal{A}$ outputs C, y^*, and $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. $\mathcal{B}$ extracts $(\omega_j, \omega_{j_\alpha})$ using $\mathsf{BARG.Extract}(\mathsf{barg.td}^0_t, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$ 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 $\mathcal{L}_t$, hence it holds 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 4.9, our claim holds. □

Claim 4.11. For any PPT adversary $\mathcal{A}$, there exists some $\alpha \in \{0,1\}$, a strongly negligible function $\mathsf{negl}'(\cdot)$

such that for every $\lambda' \in \mathbb{N}$

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

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 $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda)$ where $\epsilon(\cdot)$ is non-negligible, the claim follows immediately from Claim 4.10. □

Claim 4.12. For any PPT adversary $\mathcal{A}$, if there exists a non-negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda)$ for any $\lambda \in \mathbb{N}$, 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 4.11, we obtain:

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

We omit the proof for the above as it is almost the same as the proof of Claim 4.7. Next, $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda)$ implies that $\frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{2} - \mathsf{negl}'(\lambda') \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0}}{3}$. □

By a simple inductive proof using Claim 4.12, 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. Challenger first sets hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and generates single-hop homomorphic signature key $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Sig.Setup}(1^{\lambda'})$.
2. For all $i \in \{1,\ldots,K\} \setminus \{t\}$, the challenger first generates BARG parameters $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$ and $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$. Then it sets $(\mathsf{barg.crs}_t^0, \mathsf{barg.td}_t^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (j^*, j^*))$ and $(\mathsf{barg.crs}_t^1, \mathsf{barg.td}_t^1) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, j_\alpha^*)$. For all $i \in [K]$, challenger sets $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$ and $h_i^{\mathsf{imp}} = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j\in[i-1]})))$.
3. For all $i \in [K]$, challenger sets $\mathsf{pk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. Challenger outputs $\mathsf{pk} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_K), \mathsf{sk} = \mathsf{sk}'$. For all $i \in [\ell]$, the challenger computes and outputs $\sigma_i \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, (i, \beta_i))$.
4. The attacker $\mathcal{A}$ outputs a sequence of messages $(\beta_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}}$, and the challenger computes and outputs $\sigma_{\mathsf{id}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, (\mathsf{id}, \beta_{\mathsf{id}}))$ for all $\mathsf{id} \in \mathcal{I}$.
5. The attacker $\mathcal{A}$ outputs a structured circuit C, a message y^*, and σ^*. Let y be the actual output of the structured circuits C taking $(\beta_{\mathsf{id}_i})_{i\in[M]}$ as input. $\mathcal{A}$ wins if and only if $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}^*) = 1$ and $y^* \neq y$.

Let $(\omega_j^*, \cdot) \leftarrow \mathsf{BARG.Extract}(\mathsf{barg.td}_t^0, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$ and parse $\omega_j^* = (b_{j^*}, \ldots)$. Additionally let $\omega_{j_\alpha^*} = \mathsf{BARG.Extract}(\mathsf{barg.td}_t^1, (x_i)_{i\in\{1,\ldots,2\ell\}}, \mathsf{barg}.\pi^1)$ where $\omega_{j_\alpha^*} = (\sigma, b_{j_\alpha^*}, \rho_{j_\alpha^*})$.

Lemma 4.13. 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 $\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \epsilon(\lambda)/3^{d-2}$ for any $\lambda \in \mathbb{N}$ where $\epsilon(\cdot)$ is some non-negligible function, 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

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

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

Claim 4.14. Assume that the BARG satisfies sub-exponentially secure index hiding, then for any PPT adversary $\mathcal{A}$, there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

$$\Pr_{\mathsf{hyb}_{j^*,j^*_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}) = 1\\ \wedge\, b_{j^*} > b^*_{j^*}\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} - \mathsf{negl}'(\lambda').$$

Proof. We omit the proof as it is nearly identical to the proof of Claim 4.7. ☐

Claim 4.15. Assume that RAM delegation scheme Del satisfies sub-exponential soundness, then for any PPT adversary $\mathcal{A}$, there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

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

Proof. The proof is nearly identical to the proof of Claim 4.8. ☐

Claim 4.16. Assume that seBARG scheme BARG satisfies sub-exponentially secure somewhere argument of knowledge property, then for any PPT adverary $\mathcal{A}$, there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

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

Proof. The proof is nearly identical to the proof of Claim 4.9. ☐

Claim 4.17. 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 strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

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

Proof. (Omitted) Follows by the proof of Claim 4.10. ☐

Now, since we are assuming that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \epsilon(\lambda)/3^{d-2}$ where $\epsilon(\cdot)$ is non-negligible. By the above claim, we conclude that our lemma holds. There exists some $\alpha \in \{0,1\}$, such that

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

□

Recall that we define $\mathsf{Adv}_{\mathcal{A}}$ as the overall winning probability of attacker $\mathcal{A}$, where

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

Lemma 4.18. 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 $\epsilon(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda)$ for any $\lambda \in \mathbb{N}$. Then 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 4.8 and 4.9):

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

Thus by Lemma 4.6,

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

Recall that we assume $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda)$. Our lemma immediately follows. □

Recall that $\omega_{j^*_\alpha} = (\sigma, b_{j^*_\alpha}, \rho_{j^*_\alpha})$. Note that in Lemma 4.13, $\omega_{j^*_\alpha}$ 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 4.13 and Lemma 4.18, 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 $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^d}$ if BARG trapdoors are set at proper indexes. Furthermore, $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda)$ implies a polynomial time reduction algorithm which breaks the unforgeability of sig with advantage that is not strongly-negligible in λ'. We give a formal reduction as follows:

Lemma 4.19. Assume that public key signature scheme Sig satisfies sub-exponentially secure unforgeability, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, it holds that $\mathsf{Adv}_{\mathcal{A}} \leq \mathsf{negl}(\lambda)$.

Proof. Reduction algorithm $\mathcal{B}$ sets $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and queries the digital signature challenger and the challenger returns vk_0. For all $i \in [K]$, $\mathcal{B}$ randomly samples $r_i \in [2\ell]$, $\mathcal{B}$ then generates $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$, $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, r_i)$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, $h_i^{\mathsf{imp}} = \mathsf{Del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j \in [i-1]}))$, and then lets $\mathsf{pk}_i = (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. It outputs pk as $(\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$. $\mathcal{A}$ then outputs $(\beta_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$ and $\mathcal{B}$ sends it to the digital signature challenger and the challenger returns signatures of messages $(\mathsf{id}, \beta_{\mathsf{id}})$ for $\mathsf{id} \in \mathcal{I}$. $\mathcal{A}$ outputs C^*, y^*, and $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Upon the output by $\mathcal{A}$, $\mathcal{B}$ computes instances $(x_j^i)_{j \in [2\ell]}$ and extracts the proof at level $i-1$ as $\mathsf{barg}.\pi_{i-1}^1 = \mathsf{BARG.Extract}(\mathsf{barg.td}_i^1, (x_j^i)_{j \in [2\ell]}, \mathsf{barg}.\pi_i^1)$ for all $i \in \{2, \ldots, \ell\}$. $\mathcal{B}$ extracts witness (b', σ', ρ') using $\mathsf{BARG.Extract}(\mathsf{barg.td}_1^1, (x_j^1)_{j \in [2\ell]}, \mathsf{barg}.\pi_1^1)$. $\mathcal{B}$ then outputs the extracted signature

σ', the message which includes the corresponding index id and the bit b for $1 \leq r_i \leq \ell$ and $(1-b)$ for $\ell+1 \leq r_i \leq 2\ell$.

$\mathcal{B}$'s advantage of outputting a forgery and attacking the digital signature challenger is at least $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^{d\cdot K}\cdot(2\ell)^K}$. Since ℓ represents the maximum number of input bits of circuit class $\mathcal{C}$ and d represents the maximum depth of $\mathcal{C}$, we assume without loss of generality that $\ell \leq 2^d$. Since $\mathcal{B}$ has at most $\mathsf{negl}'(\lambda')$ advantage against the digital signature challenger, $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^{d\cdot K}\cdot(2\ell)^K} \leq \mathsf{negl}'(\lambda')$, which implies that $\mathsf{Adv}_{\mathcal{A}} \leq \mathsf{negl}(\lambda)$. □

Combining the above lemmas, we conclude that our design satisfies adaptive unforgeability. □

Theorem 4.20 (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 index language with unbounded witness length and with additive overhead $\sigma(\lambda,m) = \frac{3m}{\lambda} + \mathsf{poly}(\lambda)$.

Corollary 4.21. Assuming either sub-exponential LWE, k-LIN over pairing groups, or DDH, there exists a multi-hop homomorphic signature scheme.

Proof. By Theorem 4.20 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 Lemma 4.3 and Theorem 4.5, our corollary follows. □

4.4 Further Optimization

Fast verification. As mentioned in the technical overview on can consider the verification with pre-processing where the verifier algorithm is split to a $\mathsf{PreVerify}$ and an $\mathsf{OnlineVerify}$ such that the pre-verification computes a short digest of the large input C and the fast online verification algorithm only takes the short digest as input and verifies the signature. This implies an online verification algorithm that only grow with the depth of the computation instead of the entire size of the computation, namely online verifier grow with $\mathsf{poly}(\lambda, d, K)$. More on how offline and online verification works can be found in Appendix A.6 where we show how to make our construction of single-hop homomorphic signature to be context hiding by taking advantage of the fast verification.

Remark 4.22. The verification first transforms C that is the circuit associated with the root of $G \in \mathsf{C}$ to a monotone circuit $\tilde{C}_y$ and then verifies the statement. Thus the online verification would require a digest of the statements where the circuit C is transformed to a monotone one. However, our monotone circuit transformation $\tilde{C}_y = \mathcal{T}(C,y)$ takes the output $C(m) = y$ as input. Hence this seems to cause an issue because the input m (and hence the output y) is not available when one is digesting C in the offline/pre-verification.

We observe that one can get around this issue by taking advantage of the structure of transformation $\mathcal{T}$. Namely, the transformation from C to C_y using output y, only affects the last layer of C_y. Moreover, the transformation from C_y to a monotone circuit $\tilde{C}_y$ works layer by layer (transforms each layer independently). Therefore, the offline verification can compute a partial digest of $\tilde{C}_y$ and let the online verification compute the complementary digest given a short partial circuit (i.e. the last layer of C_y) which would be a fast operation.

More general computation. In our construction we assume the graph G is a tree, which already covers a large class of computation especially because circuit C_v can be any general circuit, namely it can copy its inputs. However, one might wonder about a more general case in which G is a DAG. More specifically, consider a real world scenario that a user homomorphically evaluates a signature σ for some b and several other users later on want to compute signatures σ_i for b_i using b as their input. Moreover, another user wants to compute a signature σ^* on some b^* that is computed over b_i values as input. In this case the tree representation on G leads to large (even exponential if it happen in several hops) on the size of C. Hence, by considering G to be a DAG one can capture more real world application.

While at a first glance this seems to be a more convoluted task as one have to prove the consistency of the reused wires across different users, we observe out that our construction indeed captures this more general problem if we consider that every intermediate circuit is also a labelled circuit. This is due to the fact that our proof is based on a guessing strategy and an induction over the layers (depth/hops), hence one does not need to explicitly enforce the consistency of such values across the structured circuit. For ease of exposition, we provide our construction where the computation graph is viewed as a tree.

5 Towards Context Hiding Homomorphic Signatures

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. Hence below we only definethe context hiding property.

Definition 5.1 (Context Hiding)**.** A multi-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 $\lambda, \ell, d, \mathsf{s}_C \in \mathbb{N}$, the following holds:

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

where C is the composition of $(\mathsf{C}_i)_{i\in[\ell]}$ and C.

Construction. In addition to the notations and parameters described in Construction 4.2, we let $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a rate-1 NIZK and $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE system. We present our construction of multi-hop homomorphic signature with context hiding as follows:

$\mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C) \to (\mathsf{pk}, \mathsf{sk})$. The setup algorithm is identical to the setup algorithm of the design without context hiding in Construction 4.2, except that it samples $\mathsf{pke.pk} \leftarrow \mathsf{PKE.Gen}(1^{\lambda'})$ and include $\mathsf{pke.pk}$ in every pk_i, and for every $i \in [K]$:

1. samples a NIZK CRS as $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{NIZK.Setup}(1^{\lambda'})$ for language $\mathcal{L}_i^2$ (Fig. 5),
2. includes each $\mathsf{nizk.crs}_i$ as part of verification key pk_i,
3. sets hash value as $h_i^{\mathsf{imp}} = \mathsf{del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j\in[t-1]}, \mathsf{pke.pk}, \mathsf{nizk.crs}_i))$, and
4. slightly modifies the description of RAM machine $\mathcal{R}_i$.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{id}, b) \to \sigma$**.** Same in Construction 4.2.

$\mathsf{Eval}(\mathsf{pk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C) \to \sigma$. The evaluation algorithm is the same as Eval in Section 4, except for Items 7 and 8 where the evaluator additionally generates and compute the RAM proof for the NIZK verification as follows:

7. It samples randomness r, encrypts h as $\mathsf{ct}_h = \mathsf{PKE.Enc}(\mathsf{pke.pk}, h; r)$, and computes a NIZK proof
   - $\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}), (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h))$.

   Then, it generates a RAM delegation proof
   - $\mathsf{del}.\pi \leftarrow \mathsf{Del.Prove}(\mathsf{del.crs}_t, (\mathsf{nizk}.\pi, y, \mathsf{ct}_h, \mathsf{C}), (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, (\mathsf{pk}_j)_{j\in[t-1]}, \mathsf{pke.pk}, \mathsf{nizk.crs}_t))$.
8. It outputs signature σ as $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$.

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, \mathsf{C}) \to \{0,1\}$. The verification algorithm is similar to Construction 4.2, except that the RAM delegation verifier takes as input $\mathsf{nizk}.\pi, \mathsf{ct}_h$ instead of $\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, h$. It outputs

$$\mathsf{del.Verify}(\mathsf{del.crs}_t, h_t^{\mathsf{imp}}, (\mathsf{nizk}.\pi, y, \mathsf{ct}_h, \mathsf{C}), \mathsf{del}.\pi).$$

Language $\mathcal{L}_i^2$

Hardwired: $\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j\in[i-1]}, \mathsf{pke.pk}$

Instance: $x = (y, \mathsf{ct}_h, \mathsf{C})$.

Witness: $\omega = (\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h)$.

Membership: ω is a valid witness for $x \in \mathcal{L}_i^2$ if all of the following are satisfied:

1. $\mathsf{PKE.Enc}(\mathsf{pke.pk}, r, h) = \mathsf{ct}_h$.
2. $\mathsf{BARG.Verify}(\mathsf{barg.crs}_i^0, (x_j)_{j\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1$,
3. $\mathsf{BARG.Verify}(\mathsf{barg.crs}_i^1, (x_j)_{j\in[2\ell]}, \mathsf{barg}.\pi^1) = 1$.

where $(x_j)_{j\in[N]}$ is defined as follows — Decompose C to its children $(\mathsf{C}_j)_{j\in[\ell]}$ and a circuit C, and Set $\tilde{C}_y$ as $\tilde{C}_y = \mathcal{T}(C, y)$ following from Fig. 1. Then,

- For $j \in [\ell]$, let $x_j = (j, h, \mathsf{C}_j)$, and $x_{j+\ell} = (j+\ell, h, \mathsf{C}_j)$.
- For $j \in \{2\ell+1, \ldots, N\}$, let $x_j = (j, h, \tilde{C}_y)$.

Figure 5: Description of language $\mathcal{L}_i^2$.

RAM Machine $\mathcal{R}_i$

Explicit Input: $\mathsf{nizk}.\pi, y, \mathsf{ct}_h, \mathsf{C}$.

Implicit Input: $\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, (\mathsf{pk}_j)_{j\in[i-1]}, \mathsf{pke.pk}, \mathsf{nizk.crs}_i$.

Output: If $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}_i, (y, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi) = 1$, then $\mathcal{R}_i$ accepts. Otherwise, it rejects.

Figure 6: Description of RAM Machine $\mathcal{R}_i$.

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. The only difference between this design and the design of Section 4 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 efficiency of the above design satisfies the requirement as it directly follows from the proof of Lemma 4.3.

Unforgeability. The proof of unforgeability is a direct extension of the one described in Section 4, except we need additional hybrids to rely on NIZK extractability. We provide it formally below.

Theorem 5.2. 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, digital signature Sig satisfies sub-exponential unforgeability, and H satisfies sub-exponentially secure collision-resistance property, then our construction satisfies adaptive 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 $2\ell + 1 \leq j_0, j_1 \leq N$.

Hybrid (j_0, j_1)

1. Challenger first sets hash key $\mathsf{hk} \leftarrow \mathsf{H.Setup}(1^{\lambda'})$ and generates single-hop homomorphic signature key $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Sig.Setup}(1^{\lambda'})$.
2. For all $i \in \{1, \ldots, K\} \setminus \{t\}$, it generates $(\mathsf{barg.crs}^0_i, \mathsf{barg.td}^0_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$. It generates $(\mathsf{barg.crs}^0_t, \mathsf{barg.td}^0_t) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (j_0-2\ell, j_1-2\ell))$. Next for all $i \in \{1, \ldots, K\}$, it sets $(\mathsf{barg.crs}^1_i, \mathsf{barg.td}^1_i) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$, $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{NIZK.Setup}(1^{\lambda'})$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, and $h^{\mathsf{imp}}_i = \mathsf{Del.Digest}(\mathsf{hk}, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i))$.
3. For all $i \in [K]$, challenger sets $\mathsf{pk}_i = (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{del.crs}_i, h^{\mathsf{imp}}_i, \mathsf{hk})$. Challenger outputs $\mathsf{pk} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_K), \mathsf{sk} = \mathsf{sk}_0$.
4. The attacker $\mathcal{A}$ outputs a sequence of messages $(\beta_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$. For all $\mathsf{id} \in \mathcal{I}$, the challenger computes and outputs $\sigma_{\mathsf{id}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, (\mathsf{id}, \beta_{\mathsf{id}}))$.
5. The attacker $\mathcal{A}$ outputs C with indexes $(\mathsf{id}_i)_{i \in [M]}$, and σ^*. Let y be the actual output of the structured circuits C taking $(\beta_{\mathsf{id}_i})_{i \in [M]}$ as input. Then $\mathcal{A}$ wins if and only if it holds that $\mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}) = 1$ and $y^* \neq y$.

Let root denote the root node of G (G is the structure of C). Let $(\mathsf{C}_j)_{j \in [\ell]}$ be the decomposed circuit C. Set circuit $\tilde{C}_{y^*}$ as the monotone circuit of circuit C_{root} ($\tilde{C}_{y^*}$ follows Fig. 1). Let b^*_j be the output of C_j, and let $b^*_{j+\ell}$ be $1 - b^*_j$. Let $(b^*_{2\ell+1}, \ldots, b^*_N)$ be the value of the internal wires of circuit $\tilde{C}_{y^*}$ taking $(b^*_1, \ldots, b^*_{2\ell})$ as input. Parse the signature by $\mathcal{A}$ as $\sigma^* = (h, \mathsf{del}.\pi, \mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1)$. Set circuit $\tilde{C}_{y^*}$ as the monotone circuit of circuit C_{root} (using Fig. 1). Let $(b^*_{2\ell+1}, \ldots, b^*_N)$ be the value of the internal 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 $x_j = (j, h, \mathsf{C})$ and $x_{j+\ell} = (j+\ell, h, \mathsf{C})$. For $j \in \{2\ell+1, \ldots, N\}$, set x_j as $(j, h, \tilde{C}_{y^*})$. Let the output of $\mathsf{NIZK}.\mathcal{E}(\mathsf{nizk.td}_t, (\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{pke.pk}, y^*, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi)$ be $(\mathsf{barg}.\pi^0, \mathsf{barg}.\pi^1, r, h)$. Let $(\omega_{j_0}, \omega_{j_1})$ be $\mathsf{BARG.Extract}(\mathsf{barg.td}^0_t, (x_i)_{i \in \{2\ell+1, \ldots, N\}}, \mathsf{barg}.\pi^0)$. 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 $\mathsf{Adv}^{j_0,j_1,b}_{\mathcal{A}}$ denote the following:

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

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 $\mathsf{Adv}_{\mathcal{A}}$:

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

We also note that t is set as the current hop that $\mathcal{A}$ is attacking.

Lemma 5.3. 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 $\epsilon(\cdot)$ such that $\mathsf{Adv}^{N,N,0}_{\mathcal{A}} \geq \epsilon(\lambda)$ for some $\lambda \in \mathbb{N}$. 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 $\epsilon(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda)$, then there exists at least one node j_α on level $i-1$, such that $3 \cdot \mathsf{Adv}_{\mathcal{A}}^{j_\alpha,j_\alpha,0} \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0}$. The above implies our lemma by a simple induction. To prove it, consider these claims:

Claim 5.4. Assume that the somewhere extractable BARG BARG satisfies sub-exponentially secure index hiding, then for any PPT adversary $\mathcal{A}$, there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j',0} \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda')$ for all $\lambda' \in \mathbb{N}$.

Proof. Omitted as it is nearly identical to the proof of Claim 4.7. ◻

Claim 5.5. 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 strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, \mathsf{C}) = 1 \\ \wedge\, b_j > b_j^* \\ \wedge\, \mathsf{NIZK.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi) = 1\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda').$$

Proof. Omitted by proof of Claim 4.8. ◻

Claim 5.6. 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 strongly negligible function $\mathsf{negl}'(\cdot')$ such that for all $\lambda' \in \mathbb{N}$,

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, \mathsf{C}) = 1 \\ \wedge\, b_j > b_j^* \\ \wedge\, \mathsf{NIZK.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi) = 1 \\ \wedge\, \mathsf{BARG.Verify}(\mathsf{barg.crs}_t^0, (x_i)_{i \in \{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda').$$

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

There exists a reduction algorithm $\mathcal{B}$ that breaks the argument of knowledge property of NIZK scheme NIZK. $\mathcal{B}$ 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 $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N - 2\ell, (N - 2\ell, N - 2\ell))$ and $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{NIZK.Setup}(1^{\lambda'})$. $\mathcal{B}$ generates $(\mathsf{barg.crs}_t^0, \mathsf{barg.td}_t^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N - 2\ell, (j - 2\ell, j' - 2\ell))$. $\mathcal{B}$ queries the NIZK challenger with language $\mathcal{L}_t^2$ and the challenger replies with $\mathsf{nizk.crs}_t$. For all $i \in \{1, \ldots, K\}$, $\mathcal{B}$ generates $\mathsf{barg.crs}_i^1 \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, $h_i^{\mathsf{imp}} = \mathsf{Del.Digest}(\mathsf{hk}, (\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$, and sets pk_i as $(\mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1, \mathsf{nizk.crs}_i, \mathsf{del.crs}_i, h_i^{\mathsf{imp}}, \mathsf{hk})$. $\mathcal{B}$ outputs pk as $(\mathsf{pk}_0, \ldots, \mathsf{pk}_K)$ and sk as sk_0. Attacker $\mathcal{A}$ outputs $(\beta_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$ and receives the signatures $(\sigma_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}}$ from the signature. $\mathcal{A}$ then outputs C, y^*. Next, $\mathcal{A}$ outputs σ^* as $(\mathsf{ct}_h, \mathsf{del}.\pi, \mathsf{nizk}.\pi)$. $\mathcal{B}$ outputs $\mathsf{nizk}.\pi$.

By the contradictory assumption, $\mathsf{BARG.Verify}(\mathsf{barg.crs}_t^0, (x_i)_{i \in \{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 0$ and thus the extracted witness is not a valid witness for $\mathcal{L}_t^2$. Then $\mathcal{B}$ breaks the sub-exponentially secure argument of knowledge property of NIZK. Our claim holds by the above reduction with Claim 5.5. ◻

Claim 5.7. Assume that the seBARG BARG is sub-exponentially secure somewhere argument of knowledge, then for any PPT adversary $\mathcal{A}$ in Hybrid (j, j') where $(\omega_j, \omega_{j'}) = \mathsf{BARG.Extract}(\mathsf{barg.td}_t^0, (x_i^0)_{i \in \{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$, there exists a strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

$$\Pr_{\mathsf{hyb}_{j,j'}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, \mathsf{C}) = 1 \\ \wedge\, b_j > b_j^* \\ \wedge\, \mathsf{NIZK.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi) = 1 \\ \wedge\, \mathsf{BARG.Verify}(\mathsf{barg.crs}_t^0, (x_i)_{i \in \{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1 \\ \wedge\, \omega_j, \omega_{j'} \text{ are valid witnesses for } \mathcal{L}_t^0\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda').$$

Proof. Omitted by proof of Claim 4.9. $\square$

Claim 5.8. 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 strongly negligible function $\mathsf{negl}'(\cdot)$ such that for all $\lambda' \in \mathbb{N}$,

$$\Pr_{\mathsf{hyb}_{j,j_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, \mathsf{C}) = 1 \\ \wedge\, b_j > b_j^* \\ \wedge\, \mathsf{NIZK.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi) = 1 \\ \wedge\, \mathsf{BARG.Verify}(\mathsf{barg.crs}_t^0, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1 \\ \wedge\, \omega_j, \omega_{j_\alpha} \text{ are valid witnesses for } \mathcal{L}_t^0 \\ \wedge\, b_{j_\alpha} = b_\alpha\end{array}\right] \geq \mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda').$$

Proof. Omitted by proof of Claim 4.10. $\square$

Claim 5.9. For any PPT adversary $\mathcal{A}$, there exists a strongly-negligible function $\mathsf{negl}'(\cdot)$ and some $\alpha \in \{0,1\}$ such that for all $\lambda' \in \mathbb{N}$,

$$\Pr_{\mathsf{hyb}_{j,j_\alpha}}\left[\begin{array}{l}\mathsf{Verify}(\mathsf{pk}, y^*, \sigma^*, \mathsf{C}) = 1 \\ \wedge\, b_j > b_j^* \\ \wedge\, \mathsf{NIZK.Verify}(\mathsf{nizk.crs}_t, (\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}), \mathsf{nizk}.\pi) = 1 \\ \wedge\, \mathsf{BARG.Verify}(\mathsf{barg.crs}_t^0, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0) = 1 \\ \wedge\, \omega_j, \omega_{j_\alpha} \text{ are valid witnesses for } \mathcal{L}_t^0 \\ \wedge\, b_{j_\alpha} = b_\alpha \\ \wedge\, b_{j_\alpha} > b_{j_\alpha}^*\end{array}\right] \geq \frac{\mathsf{Adv}_{\mathcal{A}}^{j,j,0} - \mathsf{negl}'(\lambda')}{2}.$$

Proof. Omitted by proof of Claim 4.11. $\square$

Claim 5.10. For any PPT adversary $\mathcal{A}$, if there exists a non-negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}}^{j,j,0} \geq \epsilon(\lambda)$, 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 4.12. $\square$

Our lemma follows from a induction using Claim 5.10. $\square$

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 $(\mathsf{barg.crs}_i^0, \mathsf{barg.td}_i^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (N-2\ell, N-2\ell))$ and $(\mathsf{barg.crs}_i^1, \mathsf{barg.td}_i^1) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, 1)$. It sets $(\mathsf{barg.crs}_t^0, \mathsf{barg.td}_t^0) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, N-2\ell, (j^*, j^*))$ and $(\mathsf{barg.crs}_t^1, \mathsf{barg.td}_t^1) \leftarrow \mathsf{BARG.Setup}(1^{\lambda'}, 2\ell, j_\alpha^*)$. Then for all $i \in [K]$, challenger sets $(\mathsf{nizk.crs}_i, \mathsf{nizk.td}_i) \leftarrow \mathsf{NIZK.Setup}(1^{\lambda'})$, $\mathsf{del.crs}_i \leftarrow \mathsf{Del.Setup}(1^{\lambda'}, T)$, $h_i^{\mathsf{imp}} = \mathsf{H.Hash}(\mathsf{hk}, (\mathsf{nizk.crs}_i, \mathsf{pke.pk}, \mathsf{barg.crs}_i^0, \mathsf{barg.crs}_i^1))$.

Let $(\omega_j^*, \cdot)$ be $\mathsf{BARG.Extract}(\mathsf{barg.td}_t^0, (x_i)_{i\in\{2\ell+1,\ldots,N\}}, \mathsf{barg}.\pi^0)$ and parse ω_j^* as $(b_{j^*}, \ldots)$. Also, let $\omega_{j_\alpha^*}$ be $\mathsf{BARG.Extract}(\mathsf{barg.td}_t^1, (x_i)_{i\in\{1,\ldots,2\ell\}}, \mathsf{barg}.\pi^1)$ where $\omega_{j_\alpha^*} = (\sigma, b_{j_\alpha^*}, \rho_{j_\alpha^*})$.

Lemma 5.11. 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 $\mathsf{Adv}_{\mathcal{A}}^{j^*,j^*,0} \geq \epsilon(\lambda)$ for any $\lambda \in \mathbb{N}$ where $\epsilon(\cdot)$ is some non-negligible function, 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

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

Proof. We omit the proof here as the proof proceeds identically to the proof of Lemma 4.13, except that we rely on the argument of knowledge property of NIZK, as what we did in Claim 5.6. ☐

Lemma 5.12. 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 an PPT adversary $\mathcal{A}$ and a non-negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda)$. 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. Omitted by proof of Lemma 4.18. ☐

Combining Lemma 5.11 and Lemma 5.12, we conclude: for any PPT adversary $\mathcal{A}$ that breaks the unforgeability of our scheme with advantage $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda)$ 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 $\frac{\mathsf{Adv}_{\mathcal{A}}}{3^d}$ (the proof is omitted here by proof of Lemma 4.19). With a simple induction, $\mathsf{Adv}_{\mathcal{A}} \geq \epsilon(\lambda)$ 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 not "strongly negligible" in λ'. We conclude that our theorem follows. ☐

Context Hiding. Finally, we prove context hiding of our construction. Formally, we show the following.

Theorem 5.13. Assume that NIZK satisfies zero knowledge property and PKE satisfies semantic security then our construction satisfies context hiding.

Proof. We first define our simulated evaluator $\mathcal{S}$ as the following: $\mathcal{S}$ first sets up $(\mathsf{pk}, \mathsf{sk})$ following the algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except that $\mathsf{nizk.crs}_i$ is set using $\mathsf{NIZK}.S(1^{\lambda'})$ for all $i \in [K]$. To generate simulated signatures, $\mathcal{S}$ takes as input $(t, (b_i, \sigma_i)_{i\in[\ell]}, \mathsf{C})$. 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 for $i \in [N]$. 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

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}^0_i, \mathsf{barg.crs}^1_i, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

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

Hybrid 0 It is the actual context-hiding experiment by definition.

1. The challenger tosses a coin $b \in \{0, 1\}$. For $b = 0$, the challenger sets up $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$.
2. For $b = 1$, the challenger sets up $(\mathsf{pk}, \mathsf{sk})$ following the algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except that it sets $\mathsf{nizk.crs}_i$ using $\mathsf{NIZK}.S(1^{\lambda'})$ for all $i \in [K]$.

3. The attacker $\mathcal{A}$ outputs $(t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$.

4. For $b = 0$, the challenger computes

$$\sigma \leftarrow \mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C).$$

5. For $b = 1$, the challenger computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$ 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

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

It sets σ accordingly.

6. Challenger outputs σ. $\mathcal{A}$ outputs b' and wins if and only if $b' = b$.

Hybrid 1 Instead of setting NIZK CRS using the actual setup algorithm, challenger sets NIZK CRS using simulator in hybrid 1.

1. The challenger tosses a coin $b \in \{0, 1\}$. For $b = 0$, the challenger sets up $(\mathsf{pk}, \mathsf{sk})$ following the algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except that it sets $\mathsf{nizk.crs}_i$ using $\mathsf{NIZK}.S(1^{\lambda'})$ for all $i \in [K]$.

4. For $b = 0$, the challenger computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$ step by step, except for step 7. At step 7, the challenger instead generates

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

The challenger sets σ accordingly.

Hybrid 2 Instead of encrypting hash value h, the challenger encrypts an all-zero string.

4. For $b = 0$, the challenger computes σ following from $\mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$ 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

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

It sets σ accordingly.

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

Lemma 5.14. Assume that NIZK satisfies zero-knowledge, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\mathsf{Adv}_{\mathcal{A}}^0 - \mathsf{Adv}_{\mathcal{A}}^1| \leq \mathsf{negl}(\lambda')$ for all $\lambda' \in \mathbb{N}$.

Proof. We set internal hybrid $(0, i^*)$ for $0 \leq i^* \leq K$, the hybrid is exactly the same as hybrid 0 and hybrid 1, except for step $1, 4$:

Hybrid $(0, i^*)$

1. The challenger tosses a coin $b \in \{0,1\}$. For $b = 0$, the challenger sets up $(\mathsf{pk}, \mathsf{sk})$ following the algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except that it sets $\mathsf{nizk.crs}_i$ using $\mathsf{NIZK}.S(1^{\lambda'})$ for all $i \in [i^*]$.

4. For $b = 0$ and $t > i^*$, the challenger computes $\sigma \leftarrow \mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$. For $b = 0$ and $t \leq i^*$, the challenger sets σ using $\mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$ step by step, except for step 7. At step 7, the challenger instead generates

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

The challenger sets σ accordingly.

We note that hybrid $(0, 0)$ is exactly the same as hybrid 0 and hybrid $(0, K)$ is equivalent to hybrid 1. To prove the lemma, we simply prove that $|\mathsf{Adv}_{\mathcal{A}}^{0,i^*} - \mathsf{Adv}_{\mathcal{A}}^{0,i^*+1}| \leq \mathsf{negl}(\lambda')$ using a reduction algorithm $\mathcal{B}$ that breaks the zero-knowledge property of NIZK.

$\mathcal{B}$ samples $b \in \{0,1\}$. If $b = 0$, $\mathcal{B}$ sets up $(\mathsf{pk}, \mathsf{sk})$ following the algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except that it sets $\mathsf{nizk.crs}_i$ using $\mathsf{NIZK}.S(1^{\lambda'})$ for all $i \in [i^*]$. $\mathcal{B}$ sends $\mathcal{L}^2_{i^*+1}$ to the zero-knowledge challenger, and $\mathcal{B}$ sets the challenger's output as $\mathsf{nizk.crs}_{i^*+1}$. If $b = 1$, $\mathcal{B}$ also sets $(\mathsf{pk}, \mathsf{sk})$ using algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except $\mathsf{nizk.crs}_i \leftarrow \mathsf{NIZK}.S(1^{\lambda'})$ for all $i \in [K]$. $\mathcal{A}$ outputs $t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C$.

Next if $b = 0$, $\mathcal{B}$ proceeds as the challenger of hybrid $(1, i^*)$ and $(1, i^* + 1)$ in step 4: For $t > i^* + 1$, $\mathcal{B}$ computes $\sigma \leftarrow \mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$. For $t < i^* + 1$, $\mathcal{B}$ computes σ using $\mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$, except for step 7, where $\mathcal{B}$ instead generates

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

Otherwise for $t = i^* + 1$, $\mathcal{B}$ queries the NIZK challenger using $\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}$ and the challenger outputs $\mathsf{nizk}.\pi$. $\mathcal{B}$ then sets σ accordingly.

Otherwise if $b = 1$, $\mathcal{B}$ simply proceeds as the challenger of step 5. $\mathcal{B}$ sends the signature σ to $\mathcal{A}$ and $\mathcal{A}$ responses with b'. $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise.

Assume towards contradiction that $|\mathsf{Adv}_{\mathcal{A}}^{0,i^*} - \mathsf{Adv}_{\mathcal{A}}^{0,i^*+1}| \geq \epsilon(\lambda')$ for some non-negligible function $\epsilon(\cdot)$. $\mathcal{B}$ then breaks the zero-knowledge property of the underlying NIZK with non-negligible advantage. Thus $|\mathsf{Adv}_{\mathcal{A}}^{0,i^*} - \mathsf{Adv}_{\mathcal{A}}^{0,i^*+1}| \leq \mathsf{negl}(\lambda')$, which implies that $|\mathsf{Adv}_{\mathcal{A}}^{0} - \mathsf{Adv}_{\mathcal{A}}^{1}| \leq \mathsf{negl}(\lambda')$. ☐

Lemma 5.15. Assume that PKE satisfies semantic-security, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\mathsf{Adv}_{\mathcal{A}}^{1} - \mathsf{Adv}_{\mathcal{A}}^{2}| \leq \mathsf{negl}(\lambda')$ for all $\lambda' \in \mathbb{N}$.

Proof. We prove the lemma using a reduction algorithm $\mathcal{B}$ that breaks the semantic security of PKE.

$\mathcal{B}$ samples $b \in \{0,1\}$. For either $b = 0$ or $b = 1$, $\mathcal{B}$ sets up $(\mathsf{pk}, \mathsf{sk})$ following thee algorithm $\mathsf{Setup}(1^{\lambda'}, 1^K, \mathcal{C}_\ell)$, except that it sets $\mathsf{nizk.crs}_i$ using $\mathsf{NIZK}.S(1^{\lambda'}, \mathcal{L}_i^2)$ for all $i \in [K]$. $\mathcal{A}$ outputs $t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C$.

Next if $b = 0$, $\mathcal{B}$ computes σ using $\mathsf{Eval}(\mathsf{vk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C)$, except for step 7. At step 7, $\mathcal{B}$ queries PKE challenger using hash value h and all zero string 0, and the challenger responses with ct_h. $\mathcal{B}$ then generates

$$\mathsf{nizk}.\pi \leftarrow \mathsf{NIZK}.\mathcal{S}(\mathsf{barg.crs}_t^0, \mathsf{barg.crs}_t^1, \mathsf{pke.pk}, y, \mathsf{ct}_h, \mathsf{C}).$$

Otherwise if $b = 1$, $\mathcal{B}$ simply proceeds as the challenger of step 5.

We note that if the challenger outputs encryption of h, the above experiment corresponds to hybrid 1, and if challenger encrypts 0, it corresponds to hybrid 2. 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 2, 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 5.16. 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 5.2 and Theorem 5.13, our corollary follows. ☐

6 Extending to Multi-Key Homomorphism

Structured circuit C. We use similar structured circuits as in Section 4 except that we additionally require any circuit C_v with n_{in} inputs such that v is a leaf in graph G is associated with $(\mathsf{vk}_i)_{i\in[n_{\mathsf{in}}]}$.

Definition. Here we define multi-hop multi-key homomorphic signature scheme by describing what changes compared to the single-key setting.

Syntax. The syntax is similar to that of Section 4.1 except that there is an additional KeyGen algorithm as follows:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$. The key generation algorithm takes as input security parameter λ, and outputs verification/secret key $(\mathsf{vk}, \mathsf{sk})$.

Definition 6.1 (Multi-Hop Multi-Key Homomorphic Signature)**.** A multi-hop multi-key homomorphic signature scheme $\mathsf{MKHSig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ is required to satisfy the following properties:

Completeness. The completeness is defined similar to Definition 4.1 except that we require every verification key vk in any structured circuit C_i for $i \in [\ell']$ is honestly generated.

Efficiency. The efficiency is defined similar to Definition 4.1.

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

$$\Pr\left[\begin{array}{l} y^* \neq y \\ \wedge\ \mathsf{Verify}(\mathsf{vk}, y^*, \sigma^*, \mathsf{C}^*) = 1 \end{array} : \begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C) \\ \mathcal{J} \leftarrow \mathcal{A}(\mathsf{pk}) \\ \forall \hat{\mathsf{id}} \in \mathcal{J}, (\mathsf{sk}_{\hat{\mathsf{id}}}, \mathsf{vk}_{\hat{\mathsf{id}}}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathcal{I}, (b_{\mathsf{id}^*})_{\mathsf{id}^* \in \mathcal{I}}) \leftarrow \mathcal{A}((\mathsf{vk}_{\hat{\mathsf{id}}})_{\hat{\mathsf{id}} \in \mathcal{J}}) \\ \forall \mathsf{id}^* = (\hat{\mathsf{id}}, \mathsf{id}) \in \mathcal{I}, \sigma_{\mathsf{id}^*} \leftarrow \mathsf{Sign}(\mathsf{sk}_{\hat{\mathsf{id}}}, \mathsf{id}, b_i) \\ (\mathcal{J}') \leftarrow \mathcal{A}((\sigma_{\mathsf{id}^*})_{\mathsf{id}^* \in \mathcal{I}}), \mathcal{J}' \subseteq \mathcal{J}, \\ (\mathsf{C}^*, y^*, \sigma^*) \leftarrow \mathcal{A}((\mathsf{sk}_{\hat{\mathsf{id}}})_{\hat{\mathsf{id}} \in \mathcal{J}'}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where y is the actual output of the structured circuit C^* given $(b_{\mathsf{id}^*})_{\mathsf{id}^* \in \mathcal{I}}$ and $\mathcal{A}$ is admissible if for every label $\mathsf{vk}_{\hat{\mathsf{id}}}$ in the structured circuit C^* it holds that $\hat{\mathsf{id}} \notin \mathcal{J}'$.

NOTE. We remark that in the above security experiment, we consider a slightly simplified experiment where the attacker indicates all keys it wants to corrupt at once. However, one could consider more general attackers which make signature queries as well as signing key corruption queries adaptively in an arbitrarily interleaved order. Our construction is secure under such a more general security experiment as well, but for simplicity, we consider all corruptions happen at once.

Construction. We slightly modify $\mathcal{L}_i^1$ as follows — If $i = 1$ then it parses $x = (j, \underline{\mathsf{vk}}, \mathsf{id}, h)$ and to run the signature verification it uses vk instead of pk_0. Now since $\mathcal{L}_i^1$ is slightly modified, we let $n_{\mathsf{bp},1,1} = \log 2\ell + 2\lambda + |\mathsf{vk}|$.

$\mathsf{Setup}(1^\lambda, 1^K, \ell, d, \mathsf{s}_C) \to \mathsf{pk}$. Similar to that of Construction 4.2, except that it doesn't sample the $(\mathsf{sk}', \mathsf{vk}')$.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$. It simply samples a signing-verification key pair as $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Sig.Setup}(1^{\lambda'})$.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{id}, b) \to \sigma$. Similar to that of Construction 4.2.

$\mathsf{Eval}(\mathsf{pk}, t, (b_i, \sigma_i, \mathsf{C}_i)_{i\in[\ell]}, C) \to \sigma$. Similar to that of Construction 4.2, except that in Item 1 if C_i is empty then in addition to $(\mathsf{id}_i)_{i\in[\ell]}$ it finds $(\mathsf{vk}_i)_{i\in[\ell]}$ in the labelled circuit C, and in Item 3 if C_i is empty for $i \in [\ell]$ it lets $x_i = (i, \mathsf{vk}_i, \mathsf{id}_i, h)$ and $x_{i+\ell} = (i + \ell, \mathsf{vk}_i, \mathsf{id}_i, h)$

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, \mathsf{C}) \to \{0, 1\}$. If C is empty then the verification outputs whatever $\mathsf{Sig.Verify}(\mathsf{pk}, (\mathsf{id}, y), \sigma)$ outputs, otherwise it is similar to the verifier in Construction 4.2.

Remark 6.2 (Multi-hop multi-key homomorphic signature *with context hiding*)**.** Similarly our construction of multi-hop homomorphic signature with context hiding (see Section 5) extends to the multi-key setting with a few minor modifications.

Theorem 6.3. If BARG is a sub-exponentially secure seBARG, Del is a sub-exponentially secure delegation scheme, Sig is a sub-exponentially secure digital signature scheme, and H is a sub-exponentially secure hash tree, then the above construction is a multi-hop multi-key-homomorphic signature scheme.

Corollary 6.4. Assuming sub-exponential security of either LWE, k-LIN over pairing groups, or DDH, there exists a multi-hop multi-key homomorphic signature scheme.

Proof of Theorem 6.3. The completeness and efficiency proof are similar to that of Construction 4.2. For Adaptive unforgeability, recall that the proof of Construction 4.2 follows an inductive approach by propagating $\mathcal{A}$'s advantage in forging a homomorphically evaluated signature to a forgery on a digital signature at the bottom level. The proof of the multi-key setting is very similar except that the reduction algorithm has to additionally simulate the corruption queries for $\mathcal{A}$. Hence we let the reduction algorithm guess a $\hat{\mathsf{id}}$ and then simulate the experiment for the adversary as follows — for the guesses $\hat{\mathsf{id}}$ the reduction algorithm queries the digital signature scheme for the verification key, and it generates the rest of the verification keys itself. Our reduction algorithm succeeds only if the guessing from root to leaf leads to a forgery on $\mathsf{vk}_{\hat{\mathsf{id}}}$ hence there is a $1/|\mathcal{J}|$ loss in the security of the scheme. The rest of the proof is similar to that of Construction 4.2. □

Acknowledgements. We thank Gaspard Anthoine, David Balbás, and Dario Fiore for helpful discussions regarding [ABF24], and helping us understand the current state of art in multi-key homomorphic signatures.

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. In *Annual International Cryptology Conference*, pages 317–351. Springer, 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.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Annual international cryptology conference*, pages 41–55. Springer, 2004.

[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.

[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 arguments of knowledge. *Unpublished manuscript (personal communication)*, 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.

[CGKS23] Matteo Campanelli, Chaya Ganesh, Hamidreza Khoshakhlagh, and Janno Siim. Impossibilities in succinct arguments: Black-box extraction and more. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *AFRICACRYPT 23: 14th International Conference on Cryptology in Africa*, volume 14064 of *Lecture Notes in Computer Science*, pages 465–489, July 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.

[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.

[DCB24] Trisha Datta, Binyi Chen, and Dan Boneh. Veritas: Verifying image transformations at scale. *Cryptology ePrint Archive*, 2024.

[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.

[DH76a] Whitfield Diffie and Martin E. Hellman. Multiuser cryptographic techniques. In *AFIPS National Computer Conference*, pages 109–112, 1976.

[DH76b] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[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.

[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.

[FP18] Dario Fiore and Elena Pagnin. Matrioska: a compiler for multi-key homomorphic signatures. In *International Conference on Security and Cryptography for Networks*, pages 43–62. Springer, 2018.

[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.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 505–522, January 2016.

[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.

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

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06*, 2006.

[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.

[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.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43, 1989.

[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.

[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.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, 1990.

[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.

[SFVA21] Somayeh Dolatnezhad Samarin, Dario Fiore, Daniele Venturi, and Morteza Amini. A compiler for multi-key homomorphic signatures for turing machines. *Theoretical Computer Science*, 889:145–170, 2021.

[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.

[Vad06] Salil P Vadhan. An unconditional study of computational zero knowledge. *SIAM Journal on Computing*, 36(4):1160–1214, 2006.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5*, pages 1–18. Springer, 2008.

[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 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.

A.1 Overview

At a high level, our idea is to rely on the template provided by Goyal [Goy24], and find an alternate proof strategy. By alternate, we mean that we no longer need to rely on even somewhere extractability, leave aside full extractability. At first, it might appear that the need for full extractability could be a fundamental bottleneck. However, 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 observed that by employing all-but-one (ABO) signatures [GVW19], the extraction issue can be bypassed.

All-but-one signatures, and why they aren't enough? 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 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 A.3.

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 $\mathsf{poly}(\lambda, \log |C|)$. We provide the full construction and proof in Appendix A.

A.2 SNARGs for Monotone-Policy BatchNP

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 A.1 (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 A.2 (SNARGs for Monotone-Policy BatchNP)**.** A monotone-policy batch argument $\mathsf{BARG} = (\mathsf{BARG.Gen}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify})$ 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 A.3 ([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 $|\pi| = \mathsf{poly}(\lambda, \mathsf{s}_{\mathcal{R}}, \log \mathsf{s}_{\tilde{C}})$.

A.3 All-But-One Signatures for Single-Bit Messages

A.3.1 Definition

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 A.4 (Punctурable (or All-but-one) Signature)**.** A puncturable (or All-but-one) signature $\mathsf{PSig} = (\mathsf{Setup}, \mathsf{Setup\text{-}Punc}, \mathsf{Sign}, \mathsf{Verify})$ 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 A.5 ([GVW19])**.** Assuming LWE or k-LIN over pairing groups for any constant $k \in \mathbb{N}$, there exists all-but-one signatures.

Theorem A.6 (Theorem A.9)**.** Assuming injective PRGs there exists all-but-one signatures for single-bit messages.

Definition A.7 (ABO signature for single-bit messages)**.** An all-but-one (ABO) signature for single-bit messages is defined the same as Definition A.4 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 that in the security game the adversary is allowed to make a single signing query before generating a forgery.

A.3.2 Construction

Construction A.8 (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 A.9. Assuming injective PRGs, the Construction A.8 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 A.10. 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 A.11. Assuming either LWE, DLIN, or DDH, single-bit puncturable signatures exist.

A.4 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 A.12 (Single-Hop Homomorphic Signature)**.** A single-hop homomorphic signature scheme $\mathsf{HSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ 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 A.13 (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:

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

A.5 Construction

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

Construction A.14. [Single-Hop Homomorphic Signatures for General Computation Circuits] Let $\mathsf{PSig} = (\mathsf{PSig.Setup}, \mathsf{PSig.Setup\text{-}Punc}, \mathsf{PSig.Sign}, \mathsf{PSig.Verify})$ be an all-but-one signature scheme for single-bit messages and $\mathsf{BARG} = (\mathsf{BARG.Gen}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify})$ be a monotone-policy BARG scheme. We construct a single-hop homomorphic signature for general computation circuits $\mathsf{HSig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda, 1^k, 1^{\mathsf{s}_C}) \rightarrow (\mathsf{pk}, \mathsf{sk})$**.** It samples $(\mathsf{sk}_{\mathsf{PSig},i}, \mathsf{vk}_{\mathsf{PSig},i}) \leftarrow \mathsf{PSig.Setup}(1^\lambda)$ 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 3.17, and $\mathsf{s}_\mathcal{R}$ is the size of the circuit for the relation in Item 3.) and lets $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PSig},i})_{i \in [k]}$, $\mathsf{pk} = ((\mathsf{vk}_{\mathsf{PSig},i})_{i \in [k]}, \mathsf{crs}_{\mathsf{BARG}})$.

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

$\mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i \in [k]}, C) \rightarrow \sigma$**.** 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 A.15 (Correctness.)**.** If BARG is complete PSig is correct, then the homomorphic signature from construction A.14 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 A.16 (Succinctness.)**.** If BARG is succinct, then the homomorphic signature from construction A.14 satisfies $|\sigma| \le \mathsf{poly}(\lambda, \log |C|)$.

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 A.17 (Selective Unforgeability.)**.** If BARG is non-adaptively sound and PSig satisfied punctured correctness, then the homomorphic signature from construction A.14 is selectively unforgeable.

Proof. Consider adversary $\mathcal{A}$ that breaks the unforgeability of the construction A.14. First, we construct the following hybrids:

- hyb_i. For $i \in \{0, \dots, k\}$ 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 $(m_1, \dots, m_k)$, 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 $(\sigma_1, \dots, \sigma_k)$ 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 in hyb_k it holds that $(\mathsf{sk}_{\mathsf{PSig},j}, \mathsf{vk}_{\mathsf{PSig},j})_{j\in[k]}$ 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 A.18. If PSig satisfies key-indistinguishability, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

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

Proof. Suppose towards the contradiction that $|\Pr[\mathsf{hyb}_{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_i(\mathcal{A}) = 1]| \ge \epsilon(\lambda)$ 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 $(\mathsf{s}_C, 1^k, m_1, \dots, m_k, C^*, y^*)$.
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. $\square$

Lemma A.19. 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:

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

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

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

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 A.18 and A.19 and the hybrid argument. ☐

Theorem A.20. The Construction A.14 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 A.15 to A.17. ☐

Corollary A.21. The Construction A.14 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 A.3, A.5 and A.20 and Corollary A.11. ☐

A.6 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 A.22 (Split-Verification for BARGs)**.** We say that a BARG scheme for $(\hat{x}_1, \dots, \hat{x}_k)$ where $\hat{x}_i = (x_i, x_{\mathsf{c}})$[7] for $i \in [k]$ satisfies split-verification if the Verify algorithm can be split into the following:

$\mathsf{PreVerify}(\mathsf{crs}, \mathbf{x} = (x_1, \dots, x_k)) \to \mathsf{dig}_{\mathbf{x}}$**.** A pre-verification algorithm that takes as input the crs and statements $\mathbf{x} = (x_1, \dots, x_k)$, 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).

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

Theorem A.23 ([CJJ21b])**.** Somewhere-extractable BARGs with split verification can be generically constructed from any somewhere-extractable BARG.

Proof Sketch. Consider statements $(\hat{x}_1, \dots, \hat{x}_k)$ where $\hat{x}_i = (x_i, x_{\mathsf{c}})$ and witnesses $(w_1, \dots, w_k)$ 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 $(x_1, \dots, x_k)$ and their corresponding local openings $(\mathsf{op}_1, \dots, \mathsf{op}_k)$. 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 $(x_1, \dots, x_k)$ 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 A.24 (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 $\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}_{\mathsf{dig}_0,\mathsf{dig}_1}, (x_1, \dots, x_k), (w_1, \dots, w_k))$ (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 $\pi'_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}', (\hat{x}_1, \dots, \hat{x}_k), (w_1, \dots, w_k))$ (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 A.25. [Single-Hop Homomorphic Signatures for General Circuits] We will show how to update the *Construction A.14* 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, $\mathsf{PSig} = (\mathsf{Setup}, \mathsf{Setup\text{-}Punc}, \mathsf{Sign}, \mathsf{Verify})$ be an all-but-one signature scheme for single-bit messages, and $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ 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:

$\mathsf{Setup}(1^\lambda, 1^k, 1^{s_C}.) \to (\mathsf{pk}, \mathsf{sk})$**.** Same as in Construction A.14 except that it additionally samples a $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, n_{\mathsf{NIZK}})$[8] and appends it to pk.

$\mathsf{Sign}(\mathsf{sk}, i, m_i) \to \sigma_i$**.** Same as in Construction A.14.

$\mathsf{Eval}(\mathsf{pk}, (m_i, \sigma_i)_{i\in[k]}, C) \to \sigma$**.** Same as in Construction A.14 except that after computing π_{BARG} it does the following:

1. Recompute the public info as $\mathsf{dig}_{\mathbf{x}} \leftarrow \mathsf{BARG.PreVerify}(\mathsf{crs}_{\mathsf{BARG}}, \tilde{C}_y, \mathbf{x} = (x_1, \dots, x_{2k}))$, 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 $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}}), \pi_{\mathsf{BARG}})$ for the following relation:
   - $\mathsf{OnlineVerify}(\mathsf{crs}_{\mathsf{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}}, \pi_{\mathsf{BARG}}) = 1$.
3. Output $\sigma = \pi_{\mathsf{NIZK}}$.

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

$\mathsf{Verify}(\mathsf{pk}, y, \sigma, C) \to 0/1$. It is the same as in Construction A.14 except that instead of running the BARG verification $0/1 \leftarrow \mathsf{BARG.Verify}(\mathsf{crs_{BARG}}, \mathcal{R}, \tilde{C}_y, (x_i)_{i\in[2k]}, \sigma)$, it first computes the digest as follows $\mathsf{dig}_{\mathbf{x}} \leftarrow \mathsf{BARG.PreVerify}(\mathsf{crs_{BARG}}, \tilde{C}_y, \mathbf{x} = (x_1, \ldots, x_{2k}))$, and then it runs the NIZK verification $0/1 \leftarrow \mathsf{NIZK.Verify}(\mathsf{crs_{NIZK}}, (\mathsf{crs_{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}}), \sigma)$.

Theorem A.26. The Construction A.25 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 A.20. 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 $\mathsf{crs_{NIZK}}$ 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 $\mathsf{td_{NIZK}}$, 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 A.27. If NIZK satisfies argument of knowledge, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

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

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

Lemma A.28. If NIZK satisfies argument of knowledge, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

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

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 $\mathsf{crs_{NIZK}}$. To generated a simulated evaluated signature, $\mathcal{S}$ just runs the NIZK simulator. Note that the statement for NIZK proof is $(\mathsf{crs_{BARG}}, \mathcal{R}, \mathsf{dig}_{\mathbf{x}})$ where $\mathsf{crs_{BARG}}$ 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 A.29. The Construction A.25 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 A.3, A.5 and A.26, Corollary A.11, and Remark 3.5. □

B 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.

B.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.

$\mathsf{AggVerify}(\mathsf{crs}, m, \tilde{C}, (\mathsf{vk}_1, \dots, \mathsf{vk}_k), \sigma) \to 0/1$. 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 B.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 $\tilde{C}(\mathsf{Verify}(\mathsf{vk}_1, m, \sigma_1), \dots, \mathsf{Verify}(\mathsf{vk}_k, m, \sigma_l)) = 1$ it holds that

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

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 $\mathsf{vk}_1, \dots, \mathsf{vk}_k$ 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 $m \in \{0,1\}^\lambda \setminus \{m^*\}$. 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 $(vk_1^*, \dots, \mathsf{vk}_k^*)$.
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 B.2 ([NWW23]). Assuming monotone-policy BARGs there exists monotone-policy aggregate signatures when the underlying digital signatures is instantiated using puncturable signatures.

B.2 Construction

Construction B.3 (Single-Hop Homomorphic Signatures for General Circuits)**.** Here we let $\mathsf{AggSig} = (\mathsf{AggSig.Setup}, \mathsf{AggSig.Sign}, \mathsf{AggSig.Verify}, \mathsf{AggSig.Aggregate}, \mathsf{AggSig.AggVerify})$ 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 3.17) 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 B.4. The Construction B.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 B.5. The Construction B.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 A.3, A.5, B.2 and B.4. ◻