Rate-1 Fully Local Somewhere Extractable Hashing from DDH

Pedro Branco[1(✉)], Nico Döttling[2], Akshayaram Srinivasan[3], and Riccardo Zanotto[2,4]

[1] Max-Planck Institute for Security and Privacy, Bochum, Germany
pedrodemelobranco@gmail.com
[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
[3] University of Toronto, Toronto, Canada
[4] Saarbrücken Graduate School of Computer Science, Saarbrücken, Germany

Abstract. Somewhere statistically binding (SSB) hashing allows us to sample a special hashing key such that the digest statistically binds the input at m secret locations. This hash function is said to be somewhere extractable (SE) if there is an additional trapdoor that allows the extraction of the input bits at the m locations from the digest.

Devadas, Goyal, Kalai, and Vaikuntanathan (FOCS 2022) introduced a variant of somewhere extractable hashing called rate-1 fully local SE hash functions. The rate-1 requirement states that the size of the digest is $m + \mathsf{poly}(\lambda)$ (where λ is the security parameter). The fully local property requires that for any index i, there is a "very short" opening showing that i-th bit of the hashed input is equal to b for some $b \in \{0, 1\}$. The size of this opening is required to be independent of m and in particular, this means that its size is independent of the size of the digest. Devadas et al. gave such a construction from Learning with Errors (LWE).

In this work, we give a construction of a rate-1 fully local somewhere extractable hash function from Decisional Diffie-Hellman (DDH) and BARGs. Under the same assumptions, we give constructions of rate-1 BARG and RAM SNARG with partial input soundness whose proof sizes are only matched by prior constructions based on LWE.

1 Introduction

Keyed hash functions are fundamental building blocks in cryptography. They consist of two algorithms $(\mathsf{Setup}, \mathsf{Eval})$. Setup is a PPT algorithm that takes in the security parameter 1^λ and outputs a hashing key hk. Eval is a deterministic algorithm that takes in the hashing key hk and an input x and outputs a short digest h of the input. A key property that many applications require is collision resistance. This guarantees that no PPT adversary $\mathcal{A}$ on input the hashing key hk (sampled using the Setup algorithm) can find two different inputs x, x' such that $\mathsf{Eval}(\mathsf{hk}, x) = \mathsf{Eval}(\mathsf{hk}, x')$. However, for many applications collision-resistance is not sufficient and one requires more advanced properties from the hash function.

Q. Tang and V. Teague (Eds.): PKC 2024, LNCS 14603, pp. 356–386, 2024.
https://doi.org/10.1007/978-3-031-57725-3_12

Somewhere Statistically Binding and Extractability. Somewhere statistically binding (SSB) hash functions [13,18] enhance collision resistance with stronger requirements. This family of hash function again consists of a pair of algorithms $(\mathsf{Setup}, \mathsf{Eval})$ where the Setup has a different syntax. Here, Setup takes in 1^λ and an index $i \in [n]$ (where n is the length of the input to the hash function) and outputs the hashing key hk. We require this hash function to satisfy two properties. The first property is hiding, which requires that the hashing key hk hides the location i from computationally bounded adversaries. The second property is statistical binding, which requires that the digest statistically binds to the location i. This means that any unbounded adversary should not be able to produce two inputs x and x' that differ at location i and hash to the same digest w.r.t. a hashing key hk that is sampled using $\mathsf{Setup}(1^\lambda, i)$.

An SSB hash function is said to be somewhere extractable (SE) if Setup outputs a trapdoor td along with the hashing key hk. There exists an extraction algorithm $\mathsf{Extract}$ that takes the digest h and td and outputs x_i.

SE and SSB hash functions are usually augmented with two other algorithms $(\mathsf{Open}, \mathsf{Verify})$. The Open algorithm takes in the hk, input x and a location $j \in [n]$ and outputs an opening ρ. The Verify algorithm takes in the digest h, the index j, the bit x_j, and an opening ρ and either accepts or rejects the opening. For efficiency purposes, we require the size of the opening to be much smaller than the length of the input x. SSB and SE hash functions can be naturally extended to the setting where the hash key hk binds to a subset $I \subseteq [n]$. The hiding requirement is modified to guarantee that for any two subsets I and I' of the same size, the hash keys generated w.r.t. to I and I' are computationally indistinguishable.

SSB hash functions are used in constructing very low communication MPC protocols [13], iO for Turing machines and RAM programs [1,12,17], and laconic oblivious transfer [5,10]. Somewhere extractable hash functions are used in the recent constructions of Batch Arguments from NP and Succinct Non-Interactive Arguments for deterministic polynomial-time computation [7,8,14,16,21].

Rate-1 Fully Local Somewhere Extractability. In recent work, Devadas, Goyal, Kalai, and Vaikuntanathan [9] introduced another variant of somewhere extractability called rate-1 fully local somewhere extractable hash functions. The rate-1 property requires that the size of the digest is $m + \mathsf{poly}(\lambda)$ where m is the size of the binding set I used in generating the hash key hk. Since the digest has to bind to m locations, its size must be at least m. The above requirement states that the size of the digest incurs a fixed additive polynomial overhead in λ when compared to the lower bound. The fully local opening requirement states that the size of the opening ρ to any position is a fixed polynomial in λ and is independent of m. This, in particular, means that the size of the opening is independent of the size of the digest. In the same work, they gave a construction of rate-1 fully local SE hash functions from Learning with Errors [20].

1.1 Our Results

In this work, we give a construction of a rate-1 fully local SE hash function assuming the hardness of Decisional Diffie-Hellman (DDH) and the existence of somewhere extractable Batch Arguments (seBARGs) (see Definition 3). Formally,

Informal Theorem 1. *Assuming the hardness of DDH and a somewhere extractable BARG, there exists a rate-1 fully local SE hash function.*

The works of Waters and Wu [21] and Choudhuri et al. [6] gave constructions of somewhere extractable BARGs from k-Lin and sub-exponential DDH respectively. As a corollary, we get:

Corollary 1. *Assuming either sub-exponential hardness of DDH or polynomial hardness of DDH and* k-Lin*, there exists a rate-1 fully local SE hash function.*

Application-1: Rate-1 BARG. As a direct corollary of the work of Devadas et al. [9], we get a construction of rate-1 BARG.

Corollary 2. *Assuming the hardness of DDH and a somewhere extractable BARG, there exists a construction of a BARG for* NP *where the proof size is* $m + \mathsf{poly}(\log k, \lambda)$*. Here,* m *is the size of a single witness and* k *is the batch size.*

The prior construction of BARG for NP based on the same assumptions due to Paneth and Pass [19] has a proof size of $m + o(m) \cdot \mathsf{poly}(\log k, \lambda)$.[1] The only known construction of BARG that achieves the above proof size is due to Devadas et al. [9] but their work relies on the LWE assumption.

Application-2: RAM SNARG with Partial Input Soundness. A RAM SNARG [4,8] allows a verifier to verify the correctness of a RAM program with read-only access to a large database D that runs in time T and uses space S. The verifier is given a short digest h of the database and a proof π whose size is $\mathsf{poly}(\lambda, \log T, S)$. The traditional soundness for RAM SNARG requires the adversary to "commit" to the entire database. Recent work of Kalai et al. [15] considered a stronger soundness requirement called partial input soundness. This guarantees that if the memory is digested using a SE hash function that is extractable on a set of coordinates I, and if the RAM computation only reads coordinates in I, then soundness holds. In particular, this doesn't require the adversary to commit to (or, in other words, exhibit knowledge of) the entire database beforehand. Plugging in our rate-1 fully local SE hash function into the RAM SNARG construction given in Kalai et al. [15], we obtain the following corollary:

[1] We note that this work only requires a rate-1 SE hash function (without the fully local opening) property in addition to somewhere-extractable BARG. The work of Kalai et al. [15] gave a construction of such a SE hash function from rate-1 OT. Rate-1 OT can be instantiated from DDH/QR/LWE [11].

Corollary 3. *Assuming the hardness of DDH and a somewhere extractable BARG, there exists a construction of a RAM SNARG with partial input soundness where the size of the database digest is* $m + \mathsf{poly}(\lambda)$ *and size of the proof is* $O(S) + \mathsf{poly}(\lambda, \log T)$*. Here,* m *is the size of the index* I *in the partial input soundness.*

The above parameters were previously known only from LWE [9].

1.2 Technical Outline

We will now give an overview of our construction.

Rate-1 SEH from DDH. Our starting observation is that the DDH-based trapdoor hash construction of [11] in fact already gives us a rate-1 somewhere extractable hash function. We will very briefly outline this construction, since our construction uses specific properties of it. Specifically, let $\mathbb{G}$ be a cyclic group of prime order p generated by a generator g. The setup algorithm, on input a set $I = \{i_1, \ldots, i_m\} \subseteq [N]$ first chooses $a_1, \ldots, a_m$ uniformly random from $\mathbb{Z}_p$ and sets $h_0 = g$ and $h_k = g^{a_k}$ for $k = 1, \ldots, m$. Next, it chooses $r_1, \ldots, r_N \in \mathbb{Z}_p$ uniformly at random and sets $M_{k,j} = h_k^{r_j} \cdot g^{\delta_{j,i_k}}$, where $\delta_{i,j} = 1$ if $i = j$ and otherwise 0. The hashing key consists of the matrix $\mathbf{M} = (M_{k,j})_{k,j}$, whereas the trapdoors are given by $a_1, \ldots, a_m$.

Hashing proceeds as follows. Given a vector $\mathbf{x} = (x_1, \ldots, x_N)$, we compute $c_0, c_1, \ldots, c_m$ via $c_k = \prod_{j=1}^N M_{k,j}^{x_j}$. Note now that $c_1, \ldots, c_m$ is a batch ElGamal encryption of $x_{i_1}, \ldots, x_{i_m}$ with ciphertext header c_0, that is it holds that $g^{x_{i_k}} = c_k \cdot c_0^{-a_k}$ for $k = 1, \ldots, m$. This ciphertext is now *compressed* via the distributed discrete logarithm technique [2]. In a nutshell, there is an efficiently computable keyed function $f_K : \mathbb{G} \to \{0, 1\}$ such that we can efficiently find a key K such that it holds $f_K(c_k) = f_K(c_0^{a_k}) \oplus x_{i_k}$ for $k = 1, \ldots, m$. Importantly, to find such a key we do not need to know the a_k. Now, given such a key K, we compute $v_1, \ldots, v_m$ via $v_i = f_K(c_i)$. We set the hash value to be $v = (K, c_0, v_1, \ldots, v_m)$. Note that since the $v_1, \ldots, v_m$ are bits, such a hash value is of size $m + \mathsf{poly}(\lambda)$ bits.

Clearly, given a_k we can recover x_{i_k} from $K, c_0, v_1, \ldots, v_m$ via $x_{i_k} = f_K(c_0^{a_k}) \oplus v_k$ using the property of f_K detailed above.

The only security requirement we make for trapdoor hash functions is that they are *index hiding*, that is the hashing key, in this case the matrix $\mathbf{M}$, hides the index set $I = \{i_1, \ldots, i_m\}$. For this construction, this follows immediately from the IND-CPA security of batch ElGamal encryption, as for each $j = 1, \ldots, N$ it holds that $M_{1,j}, \ldots, M_{m,j}$ is a batch ElGamal ciphertext with header $M_{0,j}$.

There are two dilemmata with this construction however: first, the hashing key is non-compact, that is the size of the hashing key scales with the size of the database. Second, this construction does not support local opening.

While we do not know how to solve the first issue, we observe that this issue does not affect any of the applications of fully local somewhere extractable hash functions as long as there is a *succinct verification key* which can be used to check the validity of openings. We will therefore compute a verification key by computing a (non-rate 1) fully local somewhere extractable hash of the hashing key.

Full Locality. To address the second issue, we will take a similar avenue as [9]. Specifically, we will compute a second, non-rate 1 somewhere extractable hash h_x of the input x and prove consistency between the two hashes v and h_x. To facilitate this, we will use specific properties of how v is computed. Indeed, observe that each c_i is just a product of group elements $h_i^{x_1}, \dots, h_i^{x_N}$. Recall that our goal is to make the size of the opening (essentially) independent of *both* N and m. Hence, the statement we are trying to prove cannot directly be proven with a BARG, as the product involves N terms. However, following an idea from [9], we can compute each c_i via a succinct sequence of local operations, each only involving two group elements. This is done via a binary multiplication tree. For the sake of simplicity, let N now be a power of two, i.e. $N = 2^T$. We define $z_{i,j}^{(0)} = M_{i,j}^{x_j}$ for $i \in [m]$ and $j \in [N]$. We can now recursively define the $z_{i,j}^{(t)}$ for $t = 1, \dots, T$ via

$$z_{i,j}^{(t)} = z_{i,2j-1}^{(t-1)} \cdot z_{i,2j}^{(t)}. \tag{1}$$

Here, we just set $z_{i,j}^{(t)}$ to undefined if either $z_{i,2j-1}^{(t-1)}$ or $z_{i,2j}^{(t-1)}$ is undefined (i.e. $2j-1$ or $2j$ is out of bounds). Now note that it holds routinely that $z_{i,1}^{(T)} = c_i$ via the recursive definition of the $z_{i,j}^{(t)}$.

The idea to prove consistency between v and h_x now comprises of 3 parts.

1. Prove for all i, j that $z_{i,j}^{(0)} = M_{i,j}^{x_j}$.
2. Prove for all i, j and all t that Eq. (1) holds.
3. Prove for all i that $v_i = f_K(z_{i,1}^{(T)})$.

Since all three items are local statements, we will enforce their validity using BARGs. To facilitate this, we will convert all statements into *index statements*. For item 1, the vector $\mathbf{x}$ is already implicitly given via the hash value h_x. As mentioned above, we will have an additional verification key which consists of an SEH hash $h_{\mathbf{M}}$ committing to the matrix $\mathbf{M}$. Moreover, for all $t = 1, \dots, T$ let $z^{(t)} = (z_{i,j}^{(t)})_{i,j}$ and let $h^{(t)}$ be an SEH hash of $z^{(t)}$.

In our full construction of rate-1 fully local SEH, the hash value will consist of v, h_x, $h^{(1)}, \dots, h^{(T)}$ as well as $T+2$ BARGs (1 for item 1, T for item 2, and 1 for item 3). As the size of each BARG is independent of m and N, the total size of the hash value is still dominated by v and thus comes down to $m + T \cdot \mathsf{poly}(\lambda) = m + \mathsf{poly}(\lambda)$ (as $T = \log(N) = O(\lambda)$)).

Finally, a local opening in this construction simply consists of a local opening of h_x.

Proving Security. We will now provide a high level discussion on how we establish the somewhere extractability property of our construction. Hence, assume we had a PPT adversary $\mathcal{A}$ who succeeds in providing a valid local opening for a position $i^* \in I$ such that the opened value differs from the value extracted using the trapdoor $a_1, \dots, a_m$.

We will make use of the somewhere extractability properties of the hashes h_x, $h^{(1)}, \dots, h^{(T)}$ and $h_{\mathbf{M}}$. Specifically, it will suffice to make each of these hashes extractable at a constant number of locations. Hence the sizes of these hashes will still be $\mathsf{poly}(\lambda)$, and in particular independent of m and N.

As $|I| = m$, a security reduction can guess the index $j^* \in [m]$ such that $i^* = i_{j^*}$ with polynomial probability $1/m$, and produce a random output if the guess was wrong. The reduction will make h_x extractable at position i^*, and each $h^{(t)}$ extractable at locations $(0, j^{(t)})$ and $(j^*, j^{(t)})$, where the $j^{(t)}$ are on the root-to-leaf path to i^*. Due to the index hiding properties of the underlying SEH this modification is not noticed by the adversary.

Hence, the reduction will now be able to extract $z^{(t)}_{0,j^{(t)}}$ and $z^{(t)}_{j^*,j^{(t)}}$ for each $t = 1, \dots, T$. Our critical observation is now the following: If $z^{(t)}_{0,j^{(t)}}$ and $z^{(t)}_{j^*,j^{(t)}}$ were correctly computed, then they form an ElGamal ciphertext of x_{i^*} under the secret key a_{j^*}, that is it would hold that

$$z^{(t)}_{j^*,j^{(t)}} = (z^{(t)}_{0,j^{(t)}})^{a_{j^*}} \cdot g^{x_{i^*}}.$$

This follows via the definition of $\mathbf{M}$ and the $z^{(0)}_{i,j}$. Namely, as $M_{0,j} = g^{r_j}$, $M_{j^*,j} = h_{j^*}^{r_j} \cdot g^{\delta_{j,i^*}}$ and $z^{(0)}_{0,j} = M^{x_j}_{0,j}$, $z^{(0)}_{j^*,j} = M^{x_j}_{j^*,j}$, it holds that $(z^{(0)}_{0,j}, z^{(0)}_{j^*,j})$ is an ElGamal encryption of x_{i^*} for $j = i^*$, and otherwise an encryption of 0.

Furthermore, the above property is efficiently testable given the trapdoor a_{j^*}, that is for $t = 1, \dots, T$ the reduction can compute $X^{(t)} = z^{(t)}_{j^*,j^{(t)}} \cdot (z^{(t)}_{0,j^{(t)}})^{-a_{j^*}}$. Now, critically, if the opening provided by $\mathcal{A}$ opens to something different from x_{i^*}, then there must be an index $t^* \in [T]$ for which $X^{(t^*)}$ differs from $g^{x_{i^*}}$. The reduction can guess the smallest such index t^* with polynomial probability $1/T$.

If $t^* = 0$, we will routinely obtain a contradiction against the soundness of the BARG establishing item 1 above, whereas if $t^* = T$ we will obtain a contradiction against the soundness of the BARG establishing item 3. The challenging situation occurs if t^* lies in between 0 and T. To deal with this case, we make $h^{(t^*-1)}$ extractable at both children of $j^{(t^*)}$, which is not detectable as the underlying SEH is index hiding. Now, if the ElGamal ciphertext of one of the children is an encryption of 0 (which we can efficiently test), we immediately get a contradiction to the soundness of the BARG in item 2 for $t = t^*$ *as we know by the minimality of* t^* that the ElGamal ciphertext at the other child of $j^{(t^*)}$ is an encryption of x_{i^*}.

If the extracted ciphertext encrypts a non-zero value, we make $h^{(t^*-2)}$ extractable at both children of this node, which is again undetectable by the index hiding property. If both extracted ciphertexts encrypt 0, we again get a contradiction to the soundness of the corresponding BARG. Otherwise, we can guess with probability 1/2 which one of the two children yields a non-zero ciphertext. We will maintain this invariant in the remaining hybrids: for one of the two children, the extracted ciphertext must decrypt to a non-zero value, unless the soundness of the corresponding BARG is violated. We can hence "push" this inconsistency all the way down to the leaf layer of the tree, and eventually get a contradiction to the soundness of the BARG in item 1.

To see that the reduction has polynomial advantage, note that the overall success probability against the BARG in item 2 comes down to

$$\epsilon' = \frac{1}{m \cdot T \cdot 2^T} \cdot \epsilon = \frac{1}{m \cdot T \cdot N} \cdot \epsilon,$$

where ϵ is the success probability of $\mathcal{A}$. Noting that ϵ' is also polynomial, we conclude this outline.

2 Preliminaries

In the following, let $\mathcal{G}$ be a (prime-order) *group generator*, that is, $\mathcal{G}$ is an algorithm that takes as an input a security parameter 1^λ and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, p is the order of the group which is always a prime number unless differently specified, and g is a generator of the group. In the following we state the decisional version of the Diffie-Hellman (DDH) assumption.

Definition 1 (Decisional Diffie-Hellman Assumption). *Let* $(\mathbb{G}, p, g) \leftarrow\$ \mathcal{G}(1^\lambda)$. *We say that the DDH assumption holds (with respect to* $\mathcal{G}$*) if for any PPT adversary* $\mathcal{A}$

$$\left|\Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), (g^a, g^b, g^{ab}))] - \Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), (g^a, g^b, g^c))]\right| \leq \mathsf{negl}(\lambda)$$

where $a, b, c \leftarrow\$ \mathbb{Z}_p$.

We additionally recall a shrinking procedure which compresses a DDH-based ciphertext into a rate-1 ciphertext.

Lemma 1 ([3,11]). *There exists a correct pair of algorithms* $\mathsf{Shrink}, \mathsf{ShrinkDec}$ *such that given*

- $h_1 = g^{x_1}, \ldots, h_n = g^{x_n}$
- $c_0 = g^t$ *and* $c_i = h_i^t \cdot g^{m_i}$, *where* $m_1, \ldots, m_n$ *is a message and* $m_i \in \{0, 1\}$

it outputs

- $\mathsf{Shrink}(c_0, (c_1, \ldots, c_n)) = \mathsf{ct} = (K, d_0, (d_1, \ldots, d_n))$, *where the components are given by* $d_i = \mathsf{ShrinkComp}(K, c_i)$ *for* $i \in [n]$.
- $\mathsf{ShrinkDec}((x_1, \ldots, x_n), \mathsf{ct}) = (m_1, \ldots, m_n)$.

Moreover, $\mathsf{ShrinkDec}((x_1, \ldots, x_n), \mathsf{ct})$ *fails only with negligible probability in* λ, *and* $\mathsf{ShrinkComp}(K, c_i)$ *runs in expected polynomial time.*

In particular, the construction uses a pseudo-random function $\mathsf{PRF} : \{0, 1\}^\lambda \times \mathbb{G} \to \{0, 1\}^\tau$ *with output size* $\tau = \log(2n)$, *and* $\mathsf{ShrinkComp}(K, c_i)$ *computes the least* δ_i *such that* $\mathsf{PRF}(K, c_i \cdot g^{\delta_i}) = 0^\tau$ *and outputs* $\delta_i \mod 2$.

The compressing key K *is chosen such that* $\mathsf{PRF}(K, c_i/g) \neq 0$, *and that we have a bound* $\delta_i < D$, *where* $D = O(n\lambda)$.

2.1 Somewhere Extractable Hash Families

Definition 2 (Somewhere Extractable Hash). *A somewhere extractable hash family* SEH *consists of the following polynomial time algorithms:*

- $\mathsf{Gen}(1^\lambda, N, i^*) \to (\mathsf{hk}, \mathsf{td})$. *A probabilistic setup algorithm that takes as input the security parameter* 1^λ*, the message length* N*, and an index* $i^* \in [N]$*. It outputs a hashing key* hk *and a trapdoor* td*.*
- $\mathsf{Hash}(\mathsf{hk}, x) \to v$*. A deterministic algorithm that takes as input a hashing key* hk *and a message* $x \in \{0,1\}^N$*, and outputs a hash value* $v \in \{0,1\}^{\ell_{hash}}$*.*
- $\mathsf{Open}(\mathsf{hk}, x, j) \to (b, \rho)$*. A deterministic algorithm that takes as input a hashing key* hk*, a message* x *and an index* $j \in [N]$*. It outputs a bit* $b \in \{0,1\}$ *and an opening* $\rho \in \{0,1\}^{\ell_{open}}$*.*
- $\mathsf{Verify}(\mathsf{hk}, v, j, b, \rho) \to \{0,1\}$*. A deterministic algorithm that takes as input a hashing key* hk*, a hash value* v*, an index* $i \in [N]$*, a bit* b *and an opening* ρ*, and it outputs 1 (accept) or 0 (reject).*
- $\mathsf{Extract}(\mathsf{td}, v) \to u$*. A deterministic algorithm that takes as input the trapdoor* td *and a hash value* v*, and it outputs a bit* $u \in \{0,1\}$*.*

It is required to satisfy the following properties:

Efficiency. The size of the hashing key $|\mathsf{hk}|$, the size of the hash ℓ_{hash}, the size of the opening ℓ_{open} and the running time of Verify are all bounded by $\mathsf{poly}(\lambda, \log N)$.

Opening Completeness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any λ, any $N \leq 2^\lambda$, any $i^* \in [N]$, any $j \in [N]$ and any $x \in \{0,1\}^N$,

$$\Pr\left[\begin{array}{l} b = x_j \\ \wedge\ \mathsf{Verify}(\mathsf{hk}, v, j, b, \rho) = 1 \end{array} : \begin{array}{l} (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i^*), \\ v = \mathsf{Hash}(\mathsf{hk}, x), \\ (b, \rho) = \mathsf{Open}(\mathsf{hk}, x, j) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

Index Hiding. For any poly-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathrm{HIDE}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Experiment $\mathrm{HIDE}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda)$
$(1^N, i_0^*, i_1^*) \leftarrow \mathcal{A}_1(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i_b^*)$
$b' \leftarrow \mathcal{A}_2(\mathsf{hk})$
return $b' = b$

Somewhere Statistically (Resp. Computational) Binding w.r.t. Opening. For any all-powerful (resp. poly-time) adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathrm{OPEN}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Experiment $\mathrm{OPEN}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda)$
$(1^N, i^*) \leftarrow \mathcal{A}_1(1^\lambda)$
$(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i^*)$
$(v, j, b, \rho) \leftarrow \mathcal{A}_2(\mathsf{hk})$
$u = \mathsf{Extract}(\mathsf{td}, v)$
return $u \neq b \wedge \mathsf{Verify}(\mathsf{hk}, v, j, b, \rho)$

Remark 1 ([9,15]). Notice that we can easily convert any such SEH family into one that is extractable on m indices $i_1, \ldots, i_m$ by running each algorithm m times and concatenating the outputs.

Under this transformation, the sizes of ℓ_{hash}, ℓ_{open} and the efficiency of the Verify will be $|I| \cdot \mathsf{poly}(\lambda, \log N)$.

We will use the shorthand notation $\mathsf{Gen}(1^\lambda, N, I)$ to denote this construction, in which case $\mathsf{Extract}(\mathsf{td}, v)$ will output m bits $(u_i)_{i \in I}$.

Theorem 2 ([13]). *Assuming any FHE scheme, there exists a* SEH *family.*

Theorem 3 ([15]). *Assuming any rate-1 string OT with verifiable correctness, there exists a* SEH *family.*

Corollary 4. *There exists a* SEH *family from any of the* $\{\mathsf{DDH}$, $O(1)$-LIN, QR, DCR, $\mathsf{LWE}\}$ *assumptions.*

2.2 Somewhere Extractable Batch Arguments

We recall the notion of batch arguments (BARGs), which is an argument system to succinctly prove that, given a language $\mathcal{L}$, multiple instances $x_1, \ldots, x_k$ all have witnesses $w_1, \ldots, w_k$, with a complexity less than $\sum |w_i|$.

In particular, let $\mathsf{BatchCSAT}$ be the following language:

$$\mathsf{BatchCSAT} = \{(C, x_1, \ldots, x_k) \ : \ \exists w_1, \ldots, w_k \text{ s.t. } \forall i \in [k],\, C(x_i, w_i) = 1\},$$

where $C : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}$ is a boolean circuit that checks a relation with instance size n and witness size m.

Definition 3. *A somewhere extractable batch argument* seBARG *for* $\mathsf{BatchCSAT}$ *consists of the following polynomial time algorithms:*

– $\mathsf{Gen}(1^\lambda, k, 1^s, i^*) \rightarrow (\mathsf{crs}, \mathsf{td})$. *Given the number of instances k, an index i^* and a circuit size s, it outputs a* crs *and a trapdoor* td.

– $\mathsf{P}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \{w_i\}_{i\in[k]}) \rightarrow \pi$. *Given a* crs*, a circuit* C*,* k *statements* $x_1, \ldots, x_k \in \{0,1\}^n$ *and* k *witnesses* $w_1, \ldots, w_k \in \{0,1\}^m$*, it generates a proof* π*.*
– $\mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) \rightarrow \{0,1\}$. *Given a* crs*, a circuit* C*,* k *statements* $\{x_i\}_{i\in[k]}$ *and a proof* π*, it outputs a bit* b*.*
– $\mathsf{Extract}(\mathsf{td}, C, \{x_i\}_{i\in[k]}, \pi) \rightarrow w^*$. *Given a trapdoor* td*, a circuit* C*,* k *statements* $\{x_i\}_{i\in[k]}$ *and a proof* π*, it outputs a witness* w^* *for instance* i^**.*

L-succinctness. The crs and the proof π have length at most $L(k,\lambda)\cdot\mathsf{poly}(s)$, and the verifier runs in time $L(k,\lambda)\cdot\mathsf{poly}[s] + k\cdot\mathsf{poly}(n,\lambda)$.

Completeness. For all $\lambda \in \mathbb{N}$, all $k, n \in \mathsf{poly}(\lambda)$, all circuits $C : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}$ at size most s and all $(x_1, \ldots, x_k)$ and $(w_1, \ldots, w_k)$ such that $C(x_i, w_i) = 1$ we have that

$$\Pr\left[1 \leftarrow \mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, k, 1^s, i^*) \\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \{w_i\}_{i\in[k]})\end{array}\right] = 1.$$

Index Hiding. For all $\lambda \in \mathbb{N}$, all $k, n \in \mathsf{poly}(\lambda)$, all PPT adversaries $\mathcal{A}$ and all indices $i_0, i_1 \in [k]$ we have that

$$\Pr\left[b \leftarrow \mathcal{A}(\mathsf{crs}) : \begin{array}{c} b \leftarrow\$ \{0,1\} \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, k, 1^s, i_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Somewhere Argument of Knowledge. For all $\lambda \in \mathbb{N}$ there exists a PPT extractor Ext such that for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any polynomials $k, n = \mathsf{poly}(\lambda)$, and any index $i^* \in [k]$ we have that

$$\Pr\left[\begin{array}{c} 1 \leftarrow \mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) \\ \wedge \\ C(x_{i^*}, w^*) \neq 1 \end{array} : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, k, 1^s, i^*) \\ (C, \{x_i\}_{i\in[k]}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w^* \leftarrow \mathsf{Ext}(\mathsf{td}, C, \{x_i\}_{i\in[k]}, \pi)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

We remark that this notion is equivalent to the most common soundness notion of semi-adaptive soundness [15].

Index seBARG*s.* We say that a seBARG scheme is an index seBARG if the instances $x_1, \ldots, x_k$ are all of the form $x_i = (x, i)$ with a common x; however, in the L-succinctness property we require that the verification algorithm runs in time $L(k,\lambda)\cdot\mathsf{poly}(s)$, since it doesn't have to read all the instances anymore.

Lemma 2 **([15]).** *Assume the existence of*

– *An* L*-succinct index* BARG *proof system for* BatchCSAT
– *A* SEH *family with statistical binding as in Definition 2*

Then there exists an L*-succinct index* seBARG *proof system.*

Lemma 3 ([6,8,21]). *There exists an index* seBARG *with proof size and verifier running time of* $\mathsf{poly}(\lambda, \log k, |C|)$ *from* $\{\mathsf{DDH}, k\text{-}\mathsf{LIN}, \mathsf{LWE}\}$ *assumptions.*

Remark 2 ([9,15]). As with the SEH hash families, we can easily make the seBARG extractable on a subset $I \subset [k]$ of indices by running all the algorithms in parallel, incurring in a multiplicative factor of $|I|$ increase of all running times and sizes.

In our construction of a flSEH we will then be using the following syntax and efficiency properties of an index seBARG.

Fix an index language $\mathcal{L}$ given by a relation $\mathcal{R}(x, i, w_i)$, where x represents the common part of the statement of the index seBARG. All the algorithms will then implicitly build the circuit C from the relation $\mathcal{R}$ and the value x for the common part of the instances.

- $\mathsf{Gen}(1^\lambda, k, I) \to (\mathsf{crs}, \mathsf{td})$. Given the number of instances k, and the extraction set $I \subset [k]$, it outputs a crs and a trapdoor td.
- $\mathsf{P}(\mathsf{crs}, x, \{w_i\}_{i\in[k]}) \to \pi$. Given a crs, a common statement x and k witnesses $w_1, \ldots, w_k \in \{0,1\}^m$, it generates a proof π.
- $\mathsf{V}(\mathsf{crs}, x, \pi) \to \{0,1\}$. Given a crs, a common statement x and a proof π, it outputs a bit b.
- $\mathsf{Extract}(\mathsf{td}, x, \pi) \to (w_i^*)_{i\in[k]}$. Given a trapdoor td, a common statement x and a proof π, it outputs witnesses w_i^* for all indices $i \in [k]$.

Efficiency. We require a (multi-extractable) index seBARG to have proofs of size $|\pi| = |I| \cdot \mathsf{poly}(\lambda, \log k, |x|, m)$.

Remark 3 (On large CRS). We remark that we do not impose any restrictions in the size of the crs, as it is done in previous works. The only restriction that we require is that the verifier runs in time logarithmically in k given RAM access to the crs. This is enough for most applications of seBARG as it is noted in [9].

3 Fully Local SEH from DDH

3.1 Definition

A Fully-Local Somewhere Extractable Hash family (flSEH) is a strengthening of the SEH hash family introduced by [9,15], where the verification running time is required to be independent of the hash size (i.e. the number of binding positions).

In order to do so, we need to split the output of Hash into a *long value* and a *short digest*, and similarly split the key output by Gen into a *hashing key* and a (short) *verification key.*

The full syntax and properties are described below.

Definition 4 (Fully Local Somewhere Extractable Hash). *The syntax for a fully-local SEH hash family is the following:*

– $\mathsf{Gen}(1^\lambda, N, I) \to (\mathsf{hk}, \mathsf{vk}, \mathsf{td})$. *This is a probabilistic algorithm that takes as input the security parameter* 1^λ*, the message length* N*, and a set of indices* $I \subset [N]$*. It outputs a (long) hashing key* hk*, a (short) verification key* vk *and a trapdoor* td*.*
– $\mathsf{Hash}(\mathsf{hk}, x) \to (v, \mathsf{rt})$. *This is a deterministic algorithm that takes as input a hashing key* hk *and a message* $x \in \{0,1\}^N$*, and outputs a (long) hash value* v *and a (short) digest* rt*.*
– $\mathsf{Open}(\mathsf{hk}, x, i) \to (b, \rho)$. *This is a deterministic algorithm that takes as input a hashing key* hk*, a message* x *and an index* i*. It outputs a bit* $b \in \{0,1\}$ *and an opening* ρ*.*
– $\mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, i, b, \rho) \to \{0,1\}$. *This is a deterministic algorithm that takes as input the verification key* vk*, the short digest* rt*, an index* i*, a bit* b *and an opening* ρ*. It verifies the validity of the opening* (b, ρ) *against* rt*.*
– $\mathsf{Validate}(\mathsf{vk}, v, \mathsf{rt}) \to \{0,1\}$. *This is a deterministic algorithm that takes as input the verification key* vk*, a hash value* v *and a digest* rt*. It checks the consistency of* v *and* rt*.*
– $\mathsf{Extract}(\mathsf{td}, v) \to u$. *This is a deterministic algorithm that takes as input the trapdoor* td *and a hash value* v*, and it outputs an extracted message* $u \in \{0,1\}^{|I|}$*.*

It is required to satisfy the following properties:

Efficiency. The running time of Verify is $\mathsf{poly}(\lambda, \log N)$. Moreover, we say that a flSEH is *rate-1* if the length of the hash value v is $|I| + \mathsf{poly}(\lambda)$.

Opening Completeness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any λ, any $N \leq 2^\lambda$, any $I \subset [N]$, any $j \in [N]$ and any $x \in \{0,1\}^N$,

$$\Pr\left[\begin{array}{l} b = x_j \\ \wedge\ \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j, b, \rho) = 1 \end{array} : \begin{array}{l} (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, I), \\ (v, \mathsf{rt}) = \mathsf{Hash}(\mathsf{hk}, x), \\ (b, \rho) = \mathsf{Open}(\mathsf{hk}, x, j) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

Index Hiding. For any polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathrm{HIDE}^{\mathcal{A}_1, \mathcal{A}_2}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

$$\boxed{\begin{array}{l} \text{Experiment } \ \mathrm{HIDE}^{\mathcal{A}_1, \mathcal{A}_2}(1^\lambda) \\ \hline (1^N, I_0, I_1) \leftarrow \mathcal{A}_1(1^\lambda) \\ b \leftarrow\!\$\, \{0,1\} \\ (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, I_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{hk}, \mathsf{vk}) \\ \textbf{return } |I_0| = |I_1| \wedge b' = b \end{array}}$$

Somewhere Extractability w.r.t Opening. For any polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathrm{OPEN}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Experiment $\mathrm{OPEN}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda)$
$(1^N, I) \leftarrow \mathcal{A}_1(1^\lambda)$
$(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, I)$
$(v, \mathsf{rt}, (b_j)_{j\in I}, (\rho_j)_{j\in I}) \leftarrow \mathcal{A}_2(\mathsf{hk}, \mathsf{vk})$
$(x_j)_{j\in I} = \mathsf{Extract}(\mathsf{td}, v)$
return $\mathsf{Validate}(\mathsf{vk}, v, \mathsf{rt}) \wedge \left(\bigvee_{j\in I} x_j \neq b_j \wedge \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j, b_j, \rho_j)\right)$

3.2 Construction

Our construction of a fully local SEH is, at its core, based on the DDH-based construction of trapdoor hash functions due to [11].

Fix a generator $g \in \mathbb{G}$ of a group $\mathbb{G}$ of prime order p; let $P = \lceil \log p \rceil$ be the bitlength of elements in $\mathbb{G}$.

For our purposes, we will need to open up the distributed discrete logarithm compression mechanism due to [3]; in particular, let PRF be apseudo-random function and $\mathsf{Shrink} : \mathbb{G} \to \{0, 1\}$ the related compression function for the group $(\mathbb{G}, g)$, as described in Lemma 1.

Additional Ingredients. Our construction further requires as additional components a (non rate-1) somewhere extractable hash family SEH, and an index somewhere extractable batch argument system seBARG for NP. We will use seBARG with the following index languages.

– Let seBARG_0 be a BARG for the index language $\mathcal{L}_0$ defined by the relation

$$\mathcal{R}_0((\mathsf{hk}_{\mathbf{M}}, \mathsf{hk}_x, \mathsf{hk}_z, h_{\mathbf{M}}, h_x, h_z), (i, j), (M_{i,j}, x_j, z_{i,j}, \rho^M_{i,j}, \rho^x_j, \rho^z_{i,j}))$$

  that outputs 1 if and only if
  - $\mathsf{SEH.Verify}(\mathsf{hk}_{\mathbf{M}}, h_{\mathbf{M}}, (i, j), M_{i,j}, \rho^M_{i,j}) = 1$
  - $\mathsf{SEH.Verify}(\mathsf{hk}_x, h_x, j, x_j, \rho^x_j) = 1$
  - $\mathsf{SEH.Verify}(\mathsf{hk}_z, h_z, (i, j), z_{i,j}, \rho^z_{i,j}) = 1$
  - $z_{i,j} = M_{i,j}^{x_j}$

  In essence, this language ensures that group elements $z_{i,j}$ committed to in the hash value h_z are well-formed exponentiations of $M_{i,j}$ (committed to in $h_{\mathbf{M}}$) with x_j (committed to in h_x).
– Let seBARG_{mult} be a BARG for the language $\mathcal{L}_{mult}$ defined by the relation

$$\mathcal{R}_{mult}((\mathsf{hk}_1, \mathsf{hk}_2, h_1, h_2), (i, j), (z, z_l, z_r, \rho_z, \rho_{zl}, \rho_{zr}))$$

  that checks the following statements

  - $\mathsf{SEH.Verify}(\mathsf{hk}_1, h_1, (i, j), z, \rho_z) = 1$
  - $\mathsf{SEH.Verify}(\mathsf{hk}_2, h_2, (i, 2j-1), z_l, \rho_{zl}) = 1$
  - $\mathsf{SEH.Verify}(\mathsf{hk}_2, h_2, (i, 2j), z_r, \rho_{zr}) = 1$
  - $z = z_l \cdot z_r$

  This language ensures that the intermediate values $z^{(t)}$ are correctly computed in a binary tree structure.
- Let seBARG_{fin} be a BARG for the language $\mathcal{L}_{fin}$ defined by the relation

$$\mathcal{R}_{fin}((K, \mathsf{hk}_\kappa, \mathsf{hk}_{\mathbf{v}}, \mathsf{hk}_z, h_\kappa, h_{\mathbf{v}}, h_z), (i, j), (v_i, z_i, \kappa_i, \rho_i^v, \rho_i^z, \rho_i^\kappa))$$

  that checks all the following
  - $\mathsf{SEH.Verify}(\mathsf{hk}_{\mathbf{v}}, h_{\mathbf{v}}, i, v_i, \rho_i^v) = 1$
  - $\mathsf{SEH.Verify}(\mathsf{hk}_z, h_z, i, z_i, \rho_i^z) = 1$
  - $\mathsf{SEH.Verify}(\mathsf{hk}_\kappa, h_\kappa, i, \kappa_i, \rho_i^\kappa) = 1$
  - $v_i = \kappa_i \mod 2$
  - If $j < \kappa_i + 2$ check if $\mathsf{PRF}(K, z_i \cdot g^{j-2}) \neq 0$
  - If $j = \kappa_i + 2$, check that $\mathsf{PRF}(K, z_i \cdot g^{\kappa_i}) = 0$.

  This language checks that the final hash value $\mathbf{v}$ is correctly computed from compressing the last values $z^{(T)}$.

Construction. We now present the full construction.

$\mathsf{Gen}(1^\lambda, N, I)$:

- Let $m = |I|$ and $I = \{i_1, \ldots, i_m\}$.
- Let $T = \lceil \log N \rceil$; assume that actually $N = 2^T$, if need be by padding.
- Randomly sample $a_1, \ldots, a_m$ from $\mathbb{Z}_p$, compute $h_k = g^{a_k}$, and set $\mathsf{td} = (a_1, \ldots, a_m)$.
- Randomly sample $r_1, \ldots, r_N$ from $\mathbb{Z}_p$ and compute a matrix $\mathbf{M} \in \mathbb{G}^{(1+m)\times N}$ with $M_{0,j} = g^{r_j}$, and $M_{k,j} = h_k^{r_j} \cdot g^{\delta_{j,i_k}}$ for $k = 1, \ldots, m$, i.e.

$$\mathbf{M} = \begin{pmatrix} g^{r_1} & g^{r_2} & \ldots & \ldots & g^{r_N} \\ h_1^{r_1} & \ldots & h_1^{r_{i_1}} g & \ldots & h_1^{r_N} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ h_m^{r_1} & \ldots & \ldots & h_m^{r_{i_m}} g & h_m^{r_N} \end{pmatrix}$$

- Compute $(\mathsf{hk}_x, *) = \mathsf{SEH.Gen}(1^\lambda, N, \emptyset)$
- Compute $(\mathsf{hk}_{\mathbf{M}}, *) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N, \emptyset)$
- For all $t = 0, \ldots, T$ compute $(\mathsf{hk}^{(t)}, *) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N/2^t, \emptyset)$
- Compute $(\mathsf{hk}_{\mathbf{v}}, *) = \mathsf{SEH.Gen}(1^\lambda, m, \emptyset)$
- Compute $(\mathsf{hk}_\kappa, *) = \mathsf{SEH.Gen}(1^\lambda, m, \emptyset)$
- Run $(\mathsf{crs}_0, *) = \mathsf{seBARG}_0.\mathsf{Gen}(1^\lambda, (m+1) \cdot N, \emptyset)$
- For all $t = 1, \ldots, T$, run $(\mathsf{crs}_t, *) = \mathsf{seBARG}_{mult}.\mathsf{Gen}(1^\lambda, (m+1) \cdot N/2^t, \emptyset)$
- Run $(\mathsf{crs}_{fin}, *) = \mathsf{seBARG}_{fin}.\mathsf{Gen}(1^\lambda, m, \emptyset)$
- Compute $h_{\mathbf{M}} = \mathsf{SEH.Hash}(\mathsf{hk}_{\mathbf{M}}, \mathbf{M})$.
- Set $\mathsf{vk} = \left(\mathsf{hk}_x, \mathsf{hk}_{\mathbf{M}}, \left\{\mathsf{hk}^{(t)}\right\}_{t\in[T]}, \mathsf{hk}_{\mathbf{v}}, \mathsf{hk}_\kappa, \left\{\mathsf{crs}_t\right\}_{t\in[T]}, \mathsf{crs}_{fin}, h_{\mathbf{M}}\right)$.
- Set $\mathsf{hk} = (\mathbf{M}, \mathsf{vk})$ and output hk, vk and td.

$\mathsf{Hash}(\mathsf{hk}, x)$:

- Parse $\mathsf{hk} = (\mathbf{M}, \mathsf{vk})$ and
  $\mathsf{vk} = \left(\mathsf{hk}_x, \mathsf{hk}_{\mathbf{M}}, \left\{\mathsf{hk}^{(t)}\right\}_{t=0,\ldots,T}, \mathsf{hk}_{\mathbf{v}}, \left\{\mathsf{crs}_t\right\}_{t=0,\ldots,T}, \mathsf{crs}_{fin}, h_{\mathbf{M}}\right)$.
- Compute $c_k = \prod_{j=1}^{N} M_{k,j}^{x_j}$ for all $k = 0, \ldots, m$.
- Compute $z_{i,j}^{(0)} = M_{i,j}^{x_j}$.
- Recursively compute $z_{i,j}^{(t+1)} = z_{i,2j-1}^{(t)} \cdot z_{i,2j}^{(t)}$, from $t = 0$ up until T. In particular, $z_i^{(T)}$ will only have one component, and $z_{i,1}^{(T)} = c_i$.
- Choose $K \leftarrow\$ \{0,1\}^\lambda$ uniformly at random and for $k = 1, \ldots, m$ proceed as follows
  - Compute the smallest $\kappa_k \in [0, D]$ such that $\mathsf{PRF}(K, c_i \cdot g^{\kappa_k}) = 0$, where D is the bound needed for the compression function.
  - If no such κ_k exists or if $\mathsf{PRF}(K, c_i/g) = 0$, resample $K \leftarrow\$ \{0,1\}^\lambda$ and retry until both conditions are met.
  - Set $v_k = \kappa_k \mod 2$.
- Set $v = (K, c_0, \mathbf{v})$
- Compute $h_\kappa = \mathsf{SEH.Hash}(\mathsf{hk}_\kappa, \kappa)$.
- Compute $h_{\mathbf{v}} = \mathsf{SEH.Hash}(\mathsf{hk}_{\mathbf{v}}, \mathbf{v})$.
- Compute $h_x = \mathsf{SEH.Hash}(\mathsf{hk}_x, x)$.
- For all $t = 0, \ldots, T$, compute $h^{(t)} = \mathsf{SEH.Hash}(\mathsf{hk}^{(t)}, z^{(t)})$.
- For all i, j compute the openings
  - $\rho_j^x = \mathsf{SEH.Open}(\mathsf{hk}_x, x, j)$
  - $\rho_{i,j}^z = \mathsf{SEH.Open}(\mathsf{hk}^{(0)}, z^{(0)}, (i,j))$
  - $\rho_{i,j}^M = \mathsf{SEH.Open}(\mathsf{hk}_{\mathbf{M}}, M, (i,j))$
- Given the witnesses $w_{i,j} = (M_{i,j}, x_j, z_{i,j}^{(0)}, \rho_{i,j}^M, \rho_j^x, \rho_{i,j}^z)$, compute

$$\pi_0 = \mathsf{seBARG}_0.\mathsf{P}\left(\mathsf{crs}_0, (\mathsf{hk}_{\mathbf{M}}, \mathsf{hk}_x, \mathsf{hk}^{(0)}, h_{\mathbf{M}}, h_x, h^{(0)}), \{w_{i,j}\}_{i,j}\right).$$

- For all $t = 1, \ldots, T$
  - For all i, j compute the openings
    * $\rho_{i,j}^z = \mathsf{SEH.Open}(\mathsf{hk}^{(t)}, z^{(t)}, (i,j))$
    * $\rho_{i,j}^{zl} = \mathsf{SEH.Open}(\mathsf{hk}^{(t-1)}, z^{(t-1)}, (i, 2j-1))$
    * $\rho_{i,j}^{zr} = \mathsf{SEH.Open}(\mathsf{hk}^{(t-1)}, z^{(t-1)}, (i, 2j))$
  - Using the witnesses $w_{i,j} = (z_{i,j}^{(t)}, z_{i,2j-1}^{(t-1)}, z_{i,2j}^{(t-1)}, \rho_{i,j}^z, \rho_{i,j}^{zl}, \rho_{i,j}^{zr})$, compute

$$\pi_t = \mathsf{seBARG}_{mult}.\mathsf{P}\left(\mathsf{crs}_t, (\mathsf{hk}^{(t)}, \mathsf{hk}^{(t-1)}, h^{(t)}, h^{(t-1)}), \{w_{i,j}\}_{i,j}\right).$$

- For all $i = 1, \ldots, m$ compute the openings
  - $\rho_i^z = \mathsf{SEH.Open}(\mathsf{hk}^{(T)}, z^{(T)}, i)$
  - $\rho_i^\kappa = \mathsf{SEH.Open}(\mathsf{hk}_\kappa, \kappa, i)$
  - $\rho_i^v = \mathsf{SEH.Open}(\mathsf{hk}_{\mathbf{v}}, \mathbf{v}, i)$

– From the witnesses $w_{i,j} = (v_i, z_{i,1}^{(T)}, \kappa_i, \rho_i^v, \rho_i^z, \rho_i^\kappa)$, compute

$$\pi_{fin} = \mathsf{seBARG}_{fin}.\mathsf{P}\left(\mathsf{crs}_{fin}, (K, \mathsf{hk}_\kappa, \mathsf{hk}_\mathbf{v}, \mathsf{hk}^{(T)}, h_\kappa, h_\mathbf{v}, h^{(T)}), \{w_{i,j}\}_{i,j}\right),$$

where $i = 1, \ldots, m$ and $j = 1, \ldots, D$.

– Set $\mathsf{rt} = \left(h_x, \left(h^{(t)}, \pi_t\right)_{t=0,\ldots,T}, c_0, h_\mathbf{v}, K, h_\kappa, \pi_{fin}\right)$.
– Output (v, rt).

$\mathsf{Open}(\mathsf{hk}, x, i)$:

– Parse $\mathsf{hk} = (M, \mathsf{vk})$ and
$\mathsf{vk} = \left(\mathsf{hk}_x, \mathsf{hk}_\mathbf{M}, \left\{\mathsf{hk}^{(t)}\right\}_{t=0,\ldots,T}, \mathsf{hk}_\mathbf{v}, \mathsf{hk}_\kappa, \left\{\mathsf{crs}_t\right\}_{t=0,\ldots,T}, \mathsf{crs}_{fin}, h_\mathbf{M}\right)$.
– Output $\mathsf{SEH.Open}(\mathsf{hk}_x, x, i)$

$\mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, i, b, \rho)$:

– Parse $\mathsf{vk} = \left(\mathsf{hk}_x, \mathsf{hk}_\mathbf{M}, \left\{\mathsf{hk}^{(t)}\right\}_{t=0,\ldots,T}, \mathsf{hk}_\mathbf{v}, \mathsf{hk}_\kappa, \left\{\mathsf{crs}_t\right\}_{t=0,\ldots,T}, \mathsf{crs}_{fin}, h_\mathbf{M}\right)$.
– Parse $\mathsf{rt} = \left(h_x, \left(h^{(t)}, \pi_t\right)_{t=0,\ldots,T}, c_0, h_\mathbf{v}, K, h_\kappa, \pi_{fin}\right)$.
– Check that $\mathsf{seBARG}_0.\mathsf{V}\left(\mathsf{crs}_0, (\mathsf{hk}_\mathbf{M}, \mathsf{hk}_x, \mathsf{hk}^{(0)}, h_\mathbf{M}, h_x, h^{(0)}), \pi_0\right) = 1$.
– Check that $\mathsf{seBARG}_{mult}.\mathsf{V}\left(\mathsf{crs}_t, (\mathsf{hk}^{(t)}, \mathsf{hk}^{(t-1)}, h^{(t)}, h^{(t-1)}), \pi_t\right) = 1$ for all $t = 1, \ldots, T$.
– Check that $\mathsf{seBARG}_{fin}.\mathsf{V}\left(\mathsf{crs}_{fin}, (K, \mathsf{hk}_\kappa, \mathsf{hk}_\mathbf{v}, \mathsf{hk}^{(T)}, h_\kappa, h_\mathbf{v}, h^{(T)}), \pi_{fin}\right) = 1$.
– Check that $\mathsf{SEH.Verify}(\mathsf{hk}_x, h_x, i, b, \rho) = 1$.
– Output 1 if and only if all checks pass.

$\mathsf{Validate}(\mathsf{vk}, v, \mathsf{rt})$:

– Parse $\mathsf{vk} = \left(\mathsf{hk}_x, \mathsf{hk}_\mathbf{M}, \left\{\mathsf{hk}^{(t)}\right\}_{t=0,\ldots,T}, \mathsf{hk}_\mathbf{v}, \mathsf{hk}_\kappa, \left\{\mathsf{crs}_t\right\}_{t=0,\ldots,T}, \mathsf{crs}_{fin}, h_\mathbf{M}\right)$.
– Parse $\mathsf{rt} = \left(h_x, \left(h^{(t)}, \pi_t\right)_{t=0,\ldots,T}, c_\mathsf{rt}, h_\mathbf{v}, K_\mathsf{rt}, h_\kappa, \pi_{fin}\right)$.
– Parse $v = (K_v, c_v, \mathbf{v})$.
– Check that $c_v = c_\mathsf{rt}$ and $K_v = K_\mathsf{rt}$.
– Check that $\mathsf{SEH.Hash}(\mathsf{hk}_\mathbf{v}, \mathbf{v}) = h_\mathbf{v}$.

$\mathsf{Extract}(\mathsf{td}, v)$:

– Output $\mathsf{ShrinkDec}(\mathsf{td}, v)$.

3.3 Security Analysis

Lemma 4. *The construction in Sect. 3.2 is efficient and rate-1; in particular,* $|\mathsf{vk}|, |\mathsf{rt}|$ *and the running time of* Verify *are bounded by* $\mathsf{poly}(\lambda, \log N, \log |I|)$.

Proof. By the efficiency of the underlying SEH scheme, all the hashing keys $\mathsf{hk}_x, \mathsf{hk}_{\mathbf{M}}, \mathsf{hk}^{(t)}, \mathsf{hk}_{\mathbf{v}}, \mathsf{hk}_\kappa$ and all the openings that will be used as witnesses in the seBARGs for the languages $\mathcal{L}_0, \mathcal{L}_{mult}, \mathcal{L}_{fin}$ are of size $\mathsf{poly}(\lambda, \log(mNP))$, since our message is an $(m+1) \times N$ matrix of group elements.

This means that the circuit sizes for the seBARGs will be of size $\mathsf{poly}(\lambda, \log m, \log N)$, given also the efficiency of the algorithm $\mathsf{SEH.Verify}$. Since we have $k = (m+1) \times N$ instances, by the succinctness of the index seBARG we get that the size of all the $\mathsf{seBARG.crs}$ and proofs $\mathsf{seBARG}.\pi$, as well as the running time of $\mathsf{seBARG.V}$, are bounded by $\mathsf{poly}(\lambda, \log m, \log N)$.

Thus, given that we only have $\log N$ many of $\mathsf{hk}^{(t)}, \pi_t$, we get that $|\mathsf{vk}|, |\mathsf{rt}|$ and the running time of Verify are bounded by $\mathsf{poly}(\lambda, \log N, \log |I|)$.

Finally, by construction we have that $|v| = |I| + \mathsf{poly}(\lambda)$, i.e. our construction is rate-1.

Lemma 5. *Assume that the DDH assumption holds in the group $\mathbb{G}$. Then the construction in Sect. 3.2 satisfies the index-hiding property.*

Proof. We can easily see that by repeated application of the DDH assumption the matrices outputted by the Gen algorithm are pseudorandom. For simplicity we can consider the 2-row matrices.

If (g^a, g^b, g^c) is a DDH challenge, where c is either ab or random, we see that

$$\begin{pmatrix} g^{r_1} & \dots & g^{a} & \dots & g^{r_N} \\ g^{br_1} & \dots & g^{c+1} & \dots & g^{br_N} \end{pmatrix}$$

follows the distribution of Gen in the case that $c = ab$, and is random at the i-th column if c is random.

Lemma 6. *Assume that* SEH *is a somewhere extractable hash function,* seBARG_0 *is a somewhere extractable BARG for the language $\mathcal{L}_0$,* seBARG_{mult} *is a somewhere extractable BARG for the language $\mathcal{L}_{mult}$ and* seBARG_{fin} *is a somewhere extractable BARG for the language $\mathcal{L}_{fin}$, where $\mathcal{L}_0, \mathcal{L}_{mult}$ and $\mathcal{L}_{fin}$ are defined in Sect. 3.2. Then the scheme constructed in Sect. 3.2 satisfies the opening completeness property.*

Proof. This follows directly from the completeness of the underlying SEH family and index seBARG system.

Theorem 4. *Assume that* SEH *is a somewhere extractable hash function,* seBARG_0 *is a somewhere extractable BARG for the language $\mathcal{L}_0$,* seBARG_{mult} *is a somewhere extractable BARG for the language $\mathcal{L}_{mult}$ and* seBARG_{fin} *is a somewhere extractable BARG for the language $\mathcal{L}_{fin}$, where $\mathcal{L}_0, \mathcal{L}_{mult}$ and $\mathcal{L}_{fin}$ are defined in Sect. 3.2. Then the scheme constructed in Sect. 3.2 is somewhere binding with respect to opening.*

Proof. Assume towards contradiction that there exists an a PPT adversary $\mathcal{A}$ with non-negligible success probability ϵ against the somewhere binding w.r.t. opening experiment. We will proceed in a sequence of hybrids to establish this contradiction.

Experiment Exp_0. Let Exp_0 be the real experiment, given as follows.

$\mathsf{Exp}_0(\mathcal{A})$
- $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, I)$
- $(v, \mathsf{rt}, (b_j), (\rho_j)) \leftarrow \mathcal{A}(\mathsf{hk}, \mathsf{vk})$
- $(\hat{b}_j) = \mathsf{Extract}(\mathsf{td}, v)$
- Output 1 if $\mathsf{Validate}(\mathsf{vk}, v, \mathsf{rt}) = 1$ and there exists a $j^* \in [m]$ such that $b_{j^*} \neq \hat{b}_{j^*}$ and $\mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j^*, b_{j^*}, \rho_{j^*}) = 1$, otherwise output 0.

By our assumption on $\mathcal{A}$ it holds that $\Pr[\mathsf{Exp}_0(\mathcal{A})] > \epsilon$.

Denote by E_{val} the event that in the experiment we have $\mathsf{Validate}(\mathsf{vk}, v, \mathsf{rt}) = 1$, and E_{cheat} the event that $\bigvee_{j \in I} \hat{b}_j \neq b_j \wedge \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j, b_j, \rho_j) = 1$.

Then

$$\begin{aligned}\Pr\left[\mathsf{Exp}_0(\mathcal{A})\right] = \Pr\left[E_{val} \cap E_{cheat}\right] &= \Pr\left[E_{cheat} \mid E_{val}\right] \cdot \Pr\left[E_{val}\right] \leq \\ &\leq \Pr\left[E_{cheat} \mid E_{val}\right]\end{aligned}$$

In order to show that the hypothesis $\Pr\left[E_{cheat} \mid E_{val}\right] > \epsilon$ leads to a contradiction, we will then implicitly condition on E_{val} in all the next experiments; in particular, we assume that $\mathsf{SEH.Hash}(\mathsf{hk}_{\mathbf{v}}, \mathbf{v}) = h_{\mathbf{v}}$ and that the decryption headers K, c_0 in v are the correct ones w.r.t. the digest rt.

Experiment Exp_1. In the second experiment Exp_1 we will change the success condition of the adversary. Specifically, the experiment guesses the index $j^* \leftarrow\$ [m]$ uniformly random in the very beginning, and outputs 0 if the mismatch between the extracted value and the opened value does not occur at index j^*. Exp_1 is given as follows.

$\mathsf{Exp}_1(\mathcal{A})$
- $j^* \leftarrow\$ [m]$
- $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, I)$
- $(v, \mathsf{rt}, (b_j), (\rho_j)) \leftarrow \mathcal{A}(\mathsf{hk}, \mathsf{vk})$
- $(\hat{b}_j) = \mathsf{Extract}(\mathsf{td}, v)$
- Output 1 if $b_{j^*} \neq \hat{b}_{j^*}$ and $\mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j^*, b_{j^*}, \rho_{j^*}) = 1$, otherwise output 0.

Define S be the set of indices i for which $b_i \neq \hat{b}_i$. Conditioned on $j^* \in S$, $\mathsf{Exp}_0(\mathcal{A})$ and $\mathsf{Exp}_1(\mathcal{A})$ are identically distributed. Hence it holds that

$$\begin{aligned}\Pr[\mathsf{Exp}_1(\mathcal{A}) = 1] &= \underbrace{\Pr[\mathsf{Exp}_1(\mathcal{A}) = 1 \text{ and } j^* \in S]}_{=\Pr[\mathsf{Exp}_0(\mathcal{A})=1 \text{ and } j^* \in S]} + \underbrace{\Pr[\mathsf{Exp}_1(\mathcal{A}) = 1 \text{ and } j^* \notin S]}_{=0} \\ &= \Pr[j^* \in S | \mathsf{Exp}_0(\mathcal{A}) = 1] \cdot \underbrace{\Pr[\mathsf{Exp}_0(\mathcal{A}) = 1]}_{>\epsilon} \\ &> \Pr[j^* \in S | \mathsf{Exp}_0(\mathcal{A}) = 1] \cdot \epsilon \\ &\geq \epsilon / m,\end{aligned}$$

where the last inequality holds as S is non-empty conditioned on $\mathsf{Exp}_0(\mathcal{A}) = 1$ and j^* is independent of Exp_0.

Experiment Exp_2. In experiment Exp_2 we will modify the hashing keys hk_x, $\mathsf{hk}_{\mathbf{M}}$, $\mathsf{hk}^{(t)}$, $\mathsf{hk}_{\mathbf{v}}$ and hk_κ to be extractable on the root-to-leaf path corresponding to j^*, both for the "header" row and for the "payload" row.

Specifically, we modify the Gen algorithm such that hk_x, $\mathsf{hk}_{\mathbf{M}}$, $\mathsf{hk}^{(t)}$, hk_κ and $\mathsf{hk}_{\mathbf{v}}$ are generated as follows depending on j^*. Let $I = \{i_1, \ldots, i_m\}$ and define $i^* = i_{j^*}$ and $i^*_t = \lceil i^*/2^t \rceil$ for $t = 0, \ldots, T$.

- Compute $(\mathsf{hk}_x, \mathsf{td}_x) = \mathsf{SEH.Gen}(1^\lambda, N, \{i^*\})$
- Compute $(\mathsf{hk}_{\mathbf{M}}, \mathsf{td}_{\mathbf{M}}) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N, \{(0, i^*), (j^*, i^*)\})$
- Compute $(\mathsf{hk}^{(t)}, \mathsf{td}^{(t)}) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N/2^t, \{(0, i^*_t), (j^*, i^*_t)\})$ for all $t = 0, \ldots, T$
- Compute $(\mathsf{hk}_\kappa, \mathsf{td}_\kappa) = \mathsf{SEH.Gen}(1^\lambda, m, \{j^*\})$
- Compute $(\mathsf{hk}_{\mathbf{v}}, \mathsf{td}_{\mathbf{v}}) = \mathsf{SEH.Gen}(1^\lambda, m, \{j^*\})$

Computational indistinguishability between Exp_1 and Exp_2 follows routinely via a simple hybrid argument from the index-hiding property of SEH. Hence we have that

$$\Pr[\mathsf{Exp}_2(\mathcal{A}) = 1] \geq \Pr[\mathsf{Exp}_1(\mathcal{A}) = 1] - \mathsf{negl}(\lambda) \geq \epsilon/m - \mathsf{negl}(\lambda).$$

Experiment Exp_3. In this experiment we will extract M_{0,i^*} and M_{j^*,i^*} from $h_{\mathbf{M}}$, x_{i^*} from h_x, $z^{(t)}_{0,i^*_t}$ and $z^{(t)}_{j^*,i^*_t}$ from each $h^{(t)}$, κ_{j^*} from h_κ and v_{j^*} from $h_{\mathbf{v}}$, i.e.

- $M_{0,i^*} = \mathsf{SEH.Extract}(\mathsf{td}_{\mathbf{M}}, h_{\mathbf{M}}, (0, i^*))$
- $M_{j^*,i^*} = \mathsf{SEH.Extract}(\mathsf{td}_{\mathbf{M}}, h_{\mathbf{M}}, (j^*, i^*))$
- $x_{i^*} = \mathsf{SEH.Extract}(\mathsf{td}_x, h_x, i^*)$
- $z^{(t)}_{0,i^*_t} = \mathsf{SEH.Extract}(\mathsf{td}^{(t)}, h^{(t)}, (0, i^*_t))$
- $z^{(t)}_{j^*,i^*_t} = \mathsf{SEH.Extract}(\mathsf{td}^{(t)}, h^{(t)}, (j^*, i^*_t))$.
- $\kappa_{j^*} = \mathsf{SEH.Extract}(\mathsf{td}_\kappa, h_\kappa, j^*)$
- $v_{j^*} = \mathsf{SEH.Extract}(\mathsf{td}_{\mathbf{v}}, h_{\mathbf{v}}, j^*)$

Note that this modification does not affect the outcome of the experiment, hence it is merely syntactical, that is

$$\Pr[\mathsf{Exp}_3(\mathcal{A}) = 1] = \Pr[\mathsf{Exp}_2(\mathcal{A}) = 1] - \mathsf{negl}(\lambda) \geq \epsilon/m - \mathsf{negl}(\lambda).$$

We will now define events E_0, E_t for $t \in [T]$ and E_{fin} via

$$\begin{aligned} E_0 = 1 &:\Leftrightarrow \left(z^{(0)}_{0,i^*} \neq M^{x_{i^*}}_{0,i^*} \text{ or } z^{(0)}_{j^*,i^*} \neq M^{x_{i^*}}_{j^*,i^*} \right) \\ E_t = 1 &:\Leftrightarrow z^{(t)}_{j^*,i^*_t} \neq (z^{(t)}_{0,i^*_t})^{a_{j^*}} \cdot g^{x_{i^*}} \\ E_{fin} = 1 &:\Leftrightarrow \left(v_{j^*} \neq \mathsf{ShrinkComp}(K, z^{(T)}_{j^*,1}) \text{ or } \mathsf{PRF}(K, z^{(T)}_{j^*,1}/g) = 0 \right) \end{aligned}$$

where $\mathsf{td} = (a_1, \ldots a_m)$ is the trapdoor of the matrix $\mathbf{M}$. Now note that if none of the events E_0, E_t for some $t \in [T]$ or E_{fin} hold, then it *must hold* that $b_{j^*} = \hat{b}_{j^*}$.

Consequently, if Exp_3 outputs 1, then *at least one of these events must hold*, and therefore

$$\begin{aligned}
\epsilon/m - \mathsf{negl}(\lambda) &\le \Pr[(E_0 \vee E_{fin} \vee \exists t \in [T] \text{ s.t. } E_t) \text{ and } \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j^*, b_{j^*}, \rho_{j^*}) = 1] \\
&\le \Pr[E_0 \text{ and } \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j^*, b_{j^*}, \rho_{j^*}) = 1] \\
&\quad + \Pr[E_{fin} \text{ and } \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j^*, b_{j^*}, \rho_{j^*}) = 1] \\
&\quad + \Pr[\exists t \in [T] \text{ s.t. } E_t \text{ and } \mathsf{Verify}(\mathsf{vk}, \mathsf{rt}, j^*, b_{j^*}, \rho_{j^*}) = 1] \\
&\le \Pr[E_0 \text{ and } \mathsf{seBARG}_0.\mathsf{V}\left(\mathsf{crs}_0, (\mathsf{hk}_{\mathbf{M}}, \mathsf{hk}_x, \mathsf{hk}^{(0)}, h_{\mathbf{M}}, h_x, h^{(0)}), \pi_0\right) = 1] \\
&\quad + \Pr[E_{fin} \text{ and } \mathsf{seBARG}_{fin}.\mathsf{V}\left(\mathsf{crs}_{fin}, (K, \mathsf{hk}_\kappa, \mathsf{hk}_{\mathbf{v}}, \mathsf{hk}^{(T)}, h_\kappa, h_{\mathbf{v}}, h^{(T)}), \pi_{fin}\right) = 1] \\
&\quad + \Pr[\exists t \in [T] \text{ s.t. } E_t \text{ and } \mathsf{seBARG}_{mult}.\mathsf{V}\left(\mathsf{crs}_t, (\mathsf{hk}^{(t)}, \mathsf{hk}^{(t-1)}, h^{(t)}, h^{(t-1)}), \pi_t\right) = 1]
\end{aligned}$$

where the first inequality follows by the union bound,

That is, one of these three events must have non-negligible probability of occurrence. Hence we will now distinguish 3 cases.

1. Assume that

$$\Pr[E_0 \text{ and } \mathsf{seBARG}_0.\mathsf{V}\left(\mathsf{crs}_0, (\mathsf{hk}_{\mathbf{M}}, \mathsf{hk}_x, \mathsf{hk}^{(0)}, h_{\mathbf{M}}, h_x, h^{(0)}), \pi_0\right) = 1] > \epsilon_0$$

for a non-negligible ϵ_0.
Define an experiment $\mathsf{Exp}_{3,0,1}$ which is identical to Exp_3, but outputs 1 if and only if E_0 and $\mathsf{seBARG}_0.\mathsf{V}\left(\mathsf{crs}_0, (\mathsf{hk}_{\mathbf{M}}, \mathsf{hk}_x, \mathsf{hk}^{(0)}, h_{\mathbf{M}}, h_x, h^{(0)}), \pi_0\right) = 1$ holds. Clearly, by our assumption it holds that $\Pr[\mathsf{Exp}_{3,0,1} = 1] > \epsilon_0$. In the next experiment will make seBARG_0 extractable at positions $(0, i^*)$ and (j^*, i^*). Specifically, define an experiment $\mathsf{Exp}_{3,0,2}$ which is identical to $\mathsf{Exp}_{3,0,1}$ except that we compute crs_0 via
   - $(\mathsf{crs}_0, \mathsf{td}_0^*) = \mathsf{seBARG}_0.\mathsf{Gen}(1^\lambda, (m+1) \cdot N, \{(0, i^*), (j^*, i^*)\})$

It follows routinely from the index-hiding property of seBARG_0 that $\mathsf{Exp}_{3,0,1}$ and $\mathsf{Exp}_{3,0,2}$ are computationally indistinguishable, that is it holds that

$$\Pr[\mathsf{Exp}_{3,0,2} = 1] \ge \Pr[\mathsf{Exp}_{3,0,1} = 1] - \mathsf{negl}(\lambda) \ge \epsilon_0 - \mathsf{negl}(\lambda).$$

Now we immediately get a contradiction against the somewhere argument of knowledge/somewhere soundness property of seBARG_0, as either the statement $z^{(0)}_{0,i^*} = M^{x_{i^*}}_{0,i^*}$ or the statement $z^{(0)}_{j^*,i^*} = M^{x_{i^*}}_{j^*,i^*}$ is false, and the keys hk_x, $\mathsf{hk}_{\mathbf{M}}$ and $\mathsf{hk}^{(0)}$ are statistically binding to the corresponding positions.
2. Assume that

$$\Pr[E_{fin} \text{ and } \mathsf{seBARG}_{fin}.\mathsf{V}\left(\mathsf{crs}_{fin}, (K, \mathsf{hk}_\kappa, \mathsf{hk}_{\mathbf{v}}, \mathsf{hk}^{(T)}, h_\kappa, h_{\mathbf{v}}, h^{(T)}), \pi_{fin}\right) = 1] > \epsilon_{fin}$$

for a non-negligible ϵ_{fin}.
We modify Exp_3 into an experiment $\mathsf{Exp}_{3,fin,1}$ which outputs 1 if and only if E_{fin} and $\mathsf{seBARG}_{fin}.\mathsf{V}\left(\mathsf{crs}_{fin}, (K, \mathsf{hk}_\kappa, \mathsf{hk}_{\mathbf{v}}, \mathsf{hk}^{(T)}, h_\kappa, h_{\mathbf{v}}, h^{(T)}), \pi_{fin}\right) = 1$ hold. Again, by our assumption it holds immediately that $\Pr[\mathsf{Exp}_{3,fin,1} = 1] > \epsilon_{fin}$.
We also define events O_κ such that $O_\kappa = 1$ if and only if $\kappa < \kappa_{j^*}$ such that

$\mathsf{PRF}(K, z^{(T)}_{j^*,1} \cdot g^{\kappa}) = 0$.
Notice that

$$\Pr[\mathsf{Exp}_{3,fin,1} = 1] = \Pr[\mathsf{Exp}_{3,fin,1} = 1 \text{ and } \exists \kappa, O_\kappa = 1] + \\ + \Pr[\mathsf{Exp}_{3,fin,1} = 1 \text{ and } \forall \kappa, O_\kappa \neq 1]$$

We now define an experiment $\mathsf{Exp}_{3,fin,2}$ where we first make a guess $\kappa^* \in [0, \kappa_{j^*}]$ and then output 1 if also event $O_{\kappa^*} = 1$, i.e. if $\mathsf{PRF}(K, z^{(T)}_{j^*,1} \cdot g^{\kappa^*}) = 0$. Since our guess is independent from the experiment, we get that

$$\Pr[\mathsf{Exp}_{3,fin,2} = 1] \geq \Pr[\mathsf{Exp}_{3,fin,1} \text{ and } \exists \kappa, O_\kappa = 1]/D,$$

where $D = O(m\lambda)$.
We then define experiment $\mathsf{Exp}_{3,fin,3}$, where we make seBARG_{fin} extractable at index (j^*, κ^*). That is, experiment $\mathsf{Exp}_{3,fin,3}$ is identical to experiment $\mathsf{Exp}_{3,fin,2}$ except that we compute crs_{fin} via

– $(\mathsf{crs}_{fin}, \mathsf{td}^*_{fin}) = \mathsf{seBARG}_{fin}.\mathsf{Gen}(1^\lambda, m, \{(j^*, \kappa^*)\})$.

Indistinguishability of $\mathsf{Exp}_{3,fin,3}$ and $\mathsf{Exp}_{3,fin,2}$ follows from index-hiding of seBARG_{fin}. Moreover, since $\mathcal{L}_{fin}$ checks that $\mathsf{PRF}(K, z^{(T)}_{j^*,1} \cdot g^{\kappa^*}) \neq 0$, and we can extract a witness for the event O_{κ^*}, i.e. $\mathsf{PRF}(K, z^{(T)}_{j^*,1} \cdot g^{\kappa^*}) = 0$, we get that $\Pr[\mathsf{Exp}_{3,fin,3} = 1] \leq \mathsf{negl}(\lambda)$ by the soundness of seBARG_{fin}.
This means that $\Pr[\mathsf{Exp}_{3,fin,1} \text{ and } \exists \kappa, O_\kappa = 1] \leq D \cdot \Pr[\mathsf{Exp}_{3,fin,2} = 1] \leq \mathsf{negl}(\lambda)$, and thus $\Pr[\mathsf{Exp}_{3,fin,1} = 1 \text{ and } \forall \kappa, O_\kappa \neq 1] \geq \epsilon_{fin} - \mathsf{negl}(\lambda)$.
Now we deal with the second part of the probability, $\Pr[\mathsf{Exp}_{3,fin,1} = 1 \text{ and } \forall \kappa, O_\kappa \neq 1]$. We define experiment $\mathsf{Exp}_{3,fin,4}$, which is identical to experiment $\mathsf{Exp}_{3,fin,1}$ except that we compute crs_{fin} via

– $(\mathsf{crs}_{fin}, \mathsf{td}^*_{fin}) = \mathsf{seBARG}_{fin}.\mathsf{Gen}(1^\lambda, m, \{(j^*, 0)\})$.

Computational indistinguishability of $\mathsf{Exp}_{3,fin,4}$ and $\mathsf{Exp}_{3,fin,1}$ follows again routinely from the index-hiding property of seBARG_{fin}. Consequently, it holds that

$$\Pr[\mathsf{Exp}_{3,fin,4} = 1 \text{ and } \forall \kappa, O_\kappa \neq 1] \geq \epsilon_{fin} - \mathsf{negl}(\lambda).$$

Notice now that given that all events O_κ are false, the computation $\mathsf{ShrinkComp}(K, z^{(T)}_{j^*,1})$ is correct. This means that the extracted witness, conditioned on the event E_{fin}, is not valid for the language $\mathcal{L}_{fin}$, thus breaking the somewhere argument of knowledge/somewhere soundness property of seBARG_{fin}, which is a contradiction.

3. Finally assume that

$$\Pr[\exists t \in [T] \text{ s.t. } E_t \text{ and } \mathsf{seBARG}_{mult}.\mathsf{V}\left(\mathsf{crs}_t, (\mathsf{hk}^{(t)}, \mathsf{hk}^{(t-1)}, h^{(t)}, h^{(t-1)}), \pi_t\right) = 1] > \epsilon'$$

for a non-negligible ϵ'. Now, let $\mathsf{Exp}'_{3,1}$ be identical to Exp_3, except that the experiment outputs 1 if and only if there exists a $t \in [T]$ s.t. E_t holds and $\mathsf{seBARG}_{mult}.\mathsf{V}\left(\mathsf{crs}_t, (\mathsf{hk}^{(t)}, \mathsf{hk}^{(t-1)}, h^{(t)}, h^{(t-1)}), \pi_t\right) = 1$. Clearly, by our assumption it holds that $\Pr[\mathsf{Exp}'_{3,1} = 1] > \epsilon'$.

In the next experiment $\mathsf{Exp}'_{3,2}$ we guess an index $t^* \leftarrow\$ [T]$ such that t^* is the smallest t for which E_t holds. Specifically, $\mathsf{Exp}'_{3,2}$ outputs 0 if the guess t^* was wrong. Via the essentially same reasoning as in the step between Exp_0 and Exp_1 it holds that

$$\Pr[\mathsf{Exp}'_{3,2} = 1] \geq \Pr[\mathsf{Exp}'_{3,1} = 1]/T > \epsilon'/T.$$

In the next experiment, we make $\mathsf{hk}^{(t^*-1)}$ *also* extractable at the other child node of i_t^*, that is let

$$\bar{i}^*_{t^*-1} = \begin{cases} 2i_t^* - 1 & \text{if } i^*_{t^*-1} = 2i_t^* \\ 2i_t^* & \text{otherwise} \end{cases}.$$

Thus, in $\mathsf{Exp}'_{3,3}$ we will compute $\mathsf{hk}^{(t^*-1)}$ via

– $(\mathsf{hk}^{(t^*-1)}, \mathsf{td}^{(t^*-1)}) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N/2^t, \{(0, i^*_{t^*-1}), (j^*, i^*_{t^*-1}), (0, \bar{i}^*_{t^*-1}), (j^*, \bar{i}^*_{t^*-1})\})$

Computational indistinguishability of $\mathsf{Exp}'_{3,2}$ and $\mathsf{Exp}'_{3,3}$ follows from the index-hiding property of SEH. Thus we have

$$\Pr[\mathsf{Exp}'_{3,3} = 1] \geq \Pr[\mathsf{Exp}'_{3,2} = 1] - \mathsf{negl}(\lambda) > \epsilon'/T - \mathsf{negl}(\lambda).$$

Note that by Remark 2, our notion of being able to extract at several points is essentially for notational convenience; we have a fresh key (and hash value) for each extraction slot, thus we can introduce a new extraction slots while maintaining the ability to extract at previously planted extraction slots.

In the next hybrid $\mathsf{Exp}'_{3,4}$ we extract $h^{(t^*-1)}$ at $(0, \bar{i}^*_{t^*-1})$ and $(j^*, \bar{i}^*_{t^*-1})$, that is we compute

– $z^{(t^*-1)}_{0,\bar{i}^*_{t^*-1}} = \mathsf{SEH.Extract}(\mathsf{td}^{(t^*-1)}, h^{(t^*-1)}, (0, \bar{i}^*_{t^*-1}))$

– $z^{(t^*-1)}_{j^*,\bar{i}^*_{t^*-1}} = \mathsf{SEH.Extract}(\mathsf{td}^{(t^*-1)}, h^{(t^*-1)}, (j^*, \bar{i}^*_{t^*-1}))$

Notice that this modification has no effect on the output of the experiment. Moreover, in $\mathsf{Exp}'_{3,4}$ we also make seBARG_{mult} extractable at positions $(0, i^*_{t^*})$ and $(j^*, i^*_{t^*})$, that is, we will now generate $\mathsf{crs}^{(t^*)}$ via

– $(\mathsf{crs}^{(t^*)}, \hat{\mathsf{td}}^{(t^*)}) \leftarrow \mathsf{seBARG}_{mult}.\mathsf{Gen}(1^\lambda, (m+1) \cdot N/2^{t^*}, \{(0, i^*_{t^*}), (j^*, i^*_{t^*})\})$.

By the index-hiding property of seBARG_{mult}, $\mathsf{Exp}'_{3,3}$ and $\mathsf{Exp}'_{3,4}$ are computationally indistinguishable, that is

$$\Pr[\mathsf{Exp}'_{3,4} = 1] \geq \Pr[\mathsf{Exp}'_{3,3} = 1] - \mathsf{negl}(\lambda) > \epsilon'/T - \mathsf{negl}(\lambda).$$

In $\mathsf{Exp}'_{3,5}$ we will introduce an additional condition which causes the experiment to output 0. Specifically, let F_{t^*} be the event that $(z^{(t^*-1)}_{0,\bar{i}^*_{t^*-1}}, z^{(t^*-1)}_{j^*,\bar{i}^*_{t^*-1}})$ is an encryption of 0, that is $F_{t^*} = 1$ if and only if

$$z^{(t^*-1)}_{j^*,\bar{i}^*_{t^*-1}} = (z^{(t^*-1)}_{0,\bar{i}^*_{t^*-1}})^{a_{j^*}}.$$

$\mathsf{Exp}'_{3,5}$ is identical to $\mathsf{Exp}'_{3,4}$, except that it outputs 0 if $F_{t^*} = 1$. Note that the event F_{t^*} can be efficiently tested for given a_{j^*}. We can appeal to the extractability property of seBARG_{mult} to argue that $\Pr[F_{t^*} = 1] \leq \mathsf{negl}(\lambda)$. Otherwise, we would get a violation of the somewhere extractability/somewhere soundness of seBARG_{mult}. Specifically, assume that F_{t^*} holds, i.e.

$$z^{(t^*-1)}_{j^*,\bar{i}^*_{t^*-1}} = (z^{(t^*-1)}_{0,\bar{i}^*_{t^*-1}})^{a_{j^*}}. \tag{2}$$

We will argue that this implies that either

$$z^{(t^*)}_{0,i^*_{t^*}} \neq z^{(t^*-1)}_{0,i^*_{t^*-1}} \cdot z^{(t^*-1)}_{0,\bar{i}^*_{t^*-1}}$$

or

$$z^{(t^*)}_{j^*,i^*_{t^*}} \neq z^{(t^*-1)}_{j^*,i^*_{t^*-1}} \cdot z^{(t^*-1)}_{j^*,\bar{i}^*_{t^*-1}},$$

which routinely implies a contradiction to the somewhere soundness of seBARG_{mult}. To see this, assume that both

$$z^{(t^*)}_{0,i^*_{t^*}} = z^{(t^*-1)}_{0,i^*_{t^*-1}} \cdot z^{(t^*-1)}_{0,\bar{i}^*_{t^*-1}}, \tag{3}$$

$$z^{(t^*)}_{j^*,i^*_{t^*}} = z^{(t^*-1)}_{j^*,i^*_{t^*-1}} \cdot z^{(t^*-1)}_{j^*,\bar{i}^*_{t^*-1}}. \tag{4}$$

Recall now that t^* is the smallest t for which $z^{(t)}_{j^*,i^*_t} \neq (z^{(t)}_{0,i^*_t})^{a_{j^*}} \cdot g^{x_{i^*}}$, hence it holds that

$$z^{(t^*-1)}_{j^*,i^*_{t^*-1}} = (z^{(t^*-1)}_{0,i^*_{t^*-1}})^{a_{j^*}} \cdot g^{x_{i^*}} \tag{5}$$

Thus, by exponentiating (3) and (4) by a_{j^*} and combining (2) and (5) we can conclude that

$$z^{(t^*)}_{j^*,i^*_{t^*}} = (z^{(t^*)}_{0,i^*_{t^*}})^{a_{j^*}} \cdot g^{x_{i^*}},$$

but this means that E_{t^*} *does not hold*, i.e. it is a contradiction to t^* be the smallest t for which E_t holds. Hence we conclude that

$$\Pr[\mathsf{Exp}'_{3,5} = 1] \geq \Pr[\mathsf{Exp}'_{3,4} = 1] - \mathsf{negl}(\lambda) > \epsilon'/T - \mathsf{negl}(\lambda).$$

Now, to simplify notation define $\tilde{i} = \bar{i}^*_{t^*-1}$. In experiment $\mathsf{Exp}'_{3,5}$ we have the guarantee that if the experiment outputs 1 (which happens with non-negligible probability $\epsilon'/T - \mathsf{negl}(\lambda)$), then we have the equation $z^{(t^*-1)}_{j^*,\tilde{i}} = (z^{(t^*-1)}_{0,\tilde{i}})^{a_{j^*}} \cdot g^{\tau}$ for a non-zero τ.

In the following hybrids, we will consider a path $\tilde{i}_{t^*-1}, \ldots, \tilde{i}_0$ from $\tilde{i}_{t^*-1} = \tilde{i}$ to a leaf node $\tilde{i}_0$ and establish the invariant that all ciphertexts $(z^{(k)}_{0,\tilde{i}_k}, z^{(k)}_{j^*,\tilde{i}_k})$ encrypt non-zero values, while maintaining non-negligible probabilities for the experiments to output 1. We will achieve this using the somewhere extractability of SEH and seBARG_{mult}. Eventually, once we reached a leaf-node we will arrive at a contradiction against the soundness of seBARG_0. We will thus consider a sequence of experiments $\mathsf{Exp}''_{k,0}, \mathsf{Exp}''_{k,1}, \mathsf{Exp}''_{k,2}, \mathsf{Exp}''_{k,3}, \mathsf{Exp}''_{k,4}$ for $k = t^* - 1, \ldots, 0$. We chain them by defining $\mathsf{Exp}''_{t^*,0} = \mathsf{Exp}'_{3,5}$ and $\mathsf{Exp}''_{k-1,0} = \mathsf{Exp}''_{k,4}$.

The experiment $\mathsf{Exp}''_{k,1}$ is identical to the experiment $\mathsf{Exp}''_{k,0}$, except that we make $\mathsf{hk}^{(k-1)}$ extractable at the children nodes of $(0, \tilde{i}_k)$ and $(j^*, \tilde{i}_k)$, i.e. at positions $(0, 2\tilde{i}_k - 1)$, $(0, 2\tilde{i}_k)$, $(j^*, 2\tilde{i}_k - 1)$, $(j^*, 2\tilde{i}_k)$. In particular, we generate $\mathsf{hk}^{(k-1)}$ via

- $(\mathsf{hk}^{(k-1)}, \mathsf{td}^{(k-1)}) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N/2^{k-1}, \{(0, 2\tilde{i}_k - 1), (0, 2\tilde{i}_k), (j^*, 2\tilde{i}_k - 1), (j^*, 2\tilde{i}_k)\})$.

Computational indistinguishability of $\mathsf{Exp}''_{k,1}$ and its preceding experiment follows from the index-hiding property of SEH.

In experiment $\mathsf{Exp}''_{k,2}$, we make seBARG_{mult} extractable at positions $(0, \tilde{i}_k)$ and $(j^*, \tilde{i}_k)$.

- $(\mathsf{crs}^{(k-1)}, \hat{\mathsf{td}}^{(k-1)}) = \mathsf{seBARG}_{mult}.\mathsf{Gen}(1^\lambda, (m+1) \cdot N/2^t, \{(0, \tilde{i}_k), (j^*, \tilde{i}_k)\})$.

Computational indistinguishability follows from the index-hiding property of seBARG_{mult}.

In experiment $\mathsf{Exp}''_{k,3}$, we extract both ciphertexts at the children nodes of $\tilde{i}_k$, that is we compute

- $z^{(k-1)}_{0,2\tilde{i}_k-1} = \mathsf{SEH.Extract}(\mathsf{td}^{(k-1)}, h^{(k-1)}, (0, 2\tilde{i}_k - 1))$
- $z^{(k-1)}_{j^*,2\tilde{i}_k-1} = \mathsf{SEH.Extract}(\mathsf{td}^{(k-1)}, h^{(k-1)}, (j^*, 2\tilde{i}_k - 1))$
- $z^{(k-1)}_{0,2\tilde{i}_k} = \mathsf{SEH.Extract}(\mathsf{td}^{(k-1)}, h^{(k-1)}, (0, 2\tilde{i}_k))$
- $z^{(k-1)}_{j^*,2\tilde{i}_k} = \mathsf{SEH.Extract}(\mathsf{td}^{(k-1)}, h^{(k-1)}, (j^*, 2\tilde{i}_k))$

Furthermore, let F_k be the event that both $(z^{(k-1)}_{0,2\tilde{i}_k-1}, z^{(k-1)}_{j^*,2\tilde{i}_k-1})$ and $(z^{(k-1)}_{0,2\tilde{i}_k}, z^{(k-1)}_{j^*,2\tilde{i}_k})$ are encryptions of 0, that is it holds that both

$$z^{(k-1)}_{j^*,2\tilde{i}_k-1} = (z^{(k-1)}_{0,2\tilde{i}_k-1})^{a_{j^*}},$$
$$z^{(k-1)}_{j^*,2\tilde{i}_k} = (z^{(k-1)}_{0,2\tilde{i}_k})^{a_{j^*}}.$$

Note that we can efficiently test for this event given a_{j^*}.

In $\mathsf{Exp}''_{k,3}$ we add the additional condition that the experiment outputs 0 if the event F_k holds.

We will now argue that given that seBARG_{mult} is somewhere extractable/somewhere sound, the event F_k happens only with negligible probability.

Given that F_k happens, we claim it must hold that either

$$z^{(k)}_{0,\tilde{i}_k} \neq z^{(k-1)}_{0,2\tilde{i}_k-1} \cdot z^{(k-1)}_{0,2\tilde{i}_k}$$

or

$$z^{(k)}_{j^*,\tilde{i}_k} \neq z^{(k-1)}_{j^*,2\tilde{i}_k-1} \cdot z^{(k-1)}_{j^*,2\tilde{i}_k}$$

which routinely leads to a contradiction to the somewhere extractability/somewhere soundness of seBARG_{mult}. Otherwise, if both equations

$$z^{(k)}_{0,\tilde{i}_k} = z^{(k-1)}_{0,2\tilde{i}_k-1} \cdot z^{(k-1)}_{0,2\tilde{i}_k},$$
$$z^{(k)}_{j^*,\tilde{i}_k} = z^{(k-1)}_{j^*,2\tilde{i}_k-1} \cdot z^{(k-1)}_{j^*,2\tilde{i}_k}$$

hold, then given the equations for the event F_k, this implies that

$$z^{(k)}_{j^*,\tilde{i}_k} = (z^{(k)}_{0,\tilde{i}_k})^{a_{j^*}},$$

i.e. $(z^{(k)}_{0,\tilde{i}_k}, z^{(k)}_{j^*,\tilde{i}_k})$ is an encryption of 0. But this violates our invariant that $(z^{(k)}_{0,\tilde{i}_k}, z^{(k)}_{j^*,\tilde{i}_k})$ is an encryption of a non-zero value. Hence the claim follows, and $\mathsf{Exp}''_{k,3}$ is computationally indistinguishable from $\mathsf{Exp}''_{k,2}$.
In $\mathsf{Exp}''_{k,4}$, we guess a random bit $\beta_{k-1} \leftarrow\$ \{0,1\}$ uniformly at random at the beginning of the experiment and set $\tilde{i}_{k-1} = 2\tilde{i}_k - 1$ if $\beta_{k-1} = 0$ and $\tilde{i}_{k-1} = 2\tilde{i}_k$ if $\beta_{k-1} = 1$. Let G_{k-1} be the event that $(z^{(k-1)}_{0,\tilde{i}_{k-1}}, z^{(k-1)}_{j^*,\tilde{i}_{k-1}})$ is an encryption of 0, i.e. $G_{k-1} = 1$ if and only if

$$z^{(k-1)}_{j^*,\tilde{i}_{k-1}} = (z^{(k-1)}_{0,\tilde{i}_{k-1}})^{a_{j^*}}.$$

Now, in $\mathsf{Exp}''_{k,4}$ we add the additional condition that the experiment outputs 0 if the event G_{k-1} holds. Since the bit β_{k-1} is chosen uniformly at random and we have the promise (from experiment $\mathsf{Exp}''_{k,3}$) that either $(z^{(k-1)}_{0,2\tilde{i}_k-1}, z^{(k-1)}_{j^*,2\tilde{i}_k-1})$ or $(z^{(k-1)}_{0,2\tilde{i}_k}, z^{(k-1)}_{j^*,2\tilde{i}_k})$ is an encryption of a non-zero value, we get that the event G_{k-1} has probability at least 1/2, and therefore

$$\Pr[\mathsf{Exp}''_{k,4} = 1] \geq \Pr[\mathsf{Exp}''_{k,3} = 1]/2.$$

In particular, we have that

$$\Pr[\mathsf{Exp}''_{k,4} = 1] \geq \Pr[\mathsf{Exp}''_{k,0} = 1]/2 - \mathsf{negl}(\lambda),$$

and given that $\Pr[\mathsf{Exp}''_{k,0} = 1] = \Pr[\mathsf{Exp}''_{k+1,4} = 1]$, this implies that for the final experiment $\mathsf{Exp}''_{0,4}$ in this sequence it holds that

$$\Pr[\mathsf{Exp}''_{0,4} = 1] \geq \Pr[\mathsf{Exp}''_{t^*,1} = 1]/2^{t^*} \geq \Pr[\mathsf{Exp}''_{t^*,1} = 1]/2^T \geq \epsilon'/(2^T \cdot T) - \mathsf{negl}(\lambda),$$

which is non-negligible as ϵ' is non-negligible and $T = O(\log(\lambda))$.
In the final two experiments we will proceed analogously to the first case above, namely, we will make hk_x and $\mathsf{hk}_{\mathbf{M}}$ extractable at positions corresponding to $\tilde{i}_0$ and establish a contradiction to the somewhere extractability/somewhere soundness of seBARG_0.
That is, in Exp'''_0 we switch hk_x to be extractable at position $\tilde{i}_0$ and $\mathsf{hk}_{\mathbf{M}}$ to be extractable at positions $(0, \tilde{i}_0)$ and $(j^*, \tilde{i}_0)$, formally we compute

- $(\mathsf{hk}_x, \mathsf{td}_x) = \mathsf{SEH.Gen}(1^\lambda, N, \{i^*, \tilde{i}_0\})$
- $(\mathsf{hk}_{\mathbf{M}}, \mathsf{td}_{\mathbf{M}}) = \mathsf{SEH.Gen}(1^\lambda, (m+1) \cdot N, \{(0, i^*), (j^*, i^*), (0, \tilde{i}_0), (j^*, \tilde{i}_0)\})$

Computational indistinguishability of $\mathsf{Exp}''_{0,4}$ and Exp'''_0 follows routinely from the index-hiding property of SEH.
In experiment Exp'''_1, we switch crs_0 to be extractable at positions $(0, \tilde{i}_0)$ and $(j^*, \tilde{i}_0)$, that is we set

- $(\mathsf{crs}_0, \mathsf{td}_0) = \mathsf{seBARG}_0.\mathsf{Gen}(1^\lambda, (m+1) \cdot N, \{(0, \tilde{i}_0), (j^*, \tilde{i}_0)\})$.

Computational indistinguishability again follows routinely from the index-hiding property of seBARG_0.

We can now finally show a contradiction to the somewhere extractability/somewhere soundness property of seBARG_0.

Note that by our invariant $(z^{(0)}_{0,\tilde{i}_0}, z^{(0)}_{j^*,\tilde{i}_0})$ is an encryption of a non-zero value (conditioned on $\mathsf{Exp}_1''' = 1$). At the same time it holds that

$$M^{x_{\tilde{i}_0}}_{0,\tilde{i}_0} = g^{r_{\tilde{i}_0} \cdot x_{\tilde{i}_0}}$$
$$M^{x_{\tilde{i}_0}}_{j^*,\tilde{i}_0} = g^{a^*_j \cdot r_{\tilde{i}_0} \cdot x_{\tilde{i}_0}},$$

that is $(M^{x_{\tilde{i}_0}}_{0,\tilde{i}_0}, M^{x_{\tilde{i}_0}}_{j^*,\tilde{i}_0})$ is an encryption of 0. But this means that either

$$z^{(0)}_{0,\tilde{i}_0} \neq M^{x_{\tilde{i}_0}}_{0,\tilde{i}_0}$$

or

$$z^{(0)}_{j^*,\tilde{i}_0} \neq M^{x_{\tilde{i}_0}}_{j^*,\tilde{i}_0},$$

which routinely leads to a contradiction to the somewhere extractability of seBARG_0.

This concludes the proof.

4 Applications

4.1 Rate-1 seBARGs

Rate-1. Finally, we define the rate-1 property. A seBARG is said to be rate-1 if the proof is of size $|\pi| = m + o(m) \cdot \mathsf{poly}(\lambda, \log k)$.

The following lemma states that rate-1 BARGs exist given an index BARGs and a rate-1 fully-local SEH.

Lemma 7 ([9]). *Assuming the existence of an index* seBARG *and a rate-1 fully-local* SEH*, there exists a rate-1* seBARG*.*

Instantiating the rate-1 flSEH with the construction from Sect. 3.2 and the BARG with one from Lemma 3, we obtain the following corollary.

Corollary 5. *There exists a rate-1 BARG from subexponential DDH or k-LIN where the proof has size $m + \mathsf{poly}(\lambda)$.*

Previously, this was known from the same assumptions by plugging the rate-1 SEH construction from [15] with the construction of [19] with proof size $m + \frac{3m}{\lambda} + \mathsf{poly}(\lambda)$.

4.2 Rate-1 BARGs with Short CRS

Our rate-1 BARG from Sect. 4.1 has a large CRS, that is, the size of the CRS grows with the number of instances. In this section, we show a generic transformation from rate-1 BARGs with large CRS to a rate-1 BARG with a compact CRS, that is, a CRS with size $\mathsf{poly}(\lambda)$ (independent of the number of instances).

In particular, we prove the following theorem.

Theorem 5. *Suppose* seBARG_0 *is a somewhere extractable BARG for language* $\mathcal{L}$ *with proof size* $m + \mathsf{poly}(\lambda, \log k)$ *and CRS size* $\mathsf{poly}(\lambda, k)$*, where* k *is the number of instances and* m *is the size of a witness for* $\mathcal{L}$*. Then there exists a somewhere extractable BARG* seBARG_1 *for* $\mathcal{L}$ *with proof size* $m + \mathsf{poly}(\lambda, \log k)$ *and CRS size* $\mathsf{poly}(\lambda, \log k)$*.*

Construction. We first sketch a construction of seBARG_1, which is based on a binary tree, where each node is a seBARG_0 proof that the two children are themselves valid seBARG_0 proofs, i.e. at each layer we use the BARG for just 2 statements.[2] Concretely, at the leaf level, let $\mathcal{L}_0 = \mathcal{L}$ be the base language for which we want a BARG. For each following layer $j \geq 1$, we define the language $\mathcal{L}_j$: a statement is a tuple $y_j = (x_1, \ldots, x_{2^j})$, a witness is a proof π, and the relation $\mathcal{R}_j$ is

$$\mathcal{R}_j(y_j, \pi) = \mathsf{seBARG}_0.\mathsf{V}(\mathsf{crs}_{j-1}, \mathcal{L}_{j-1}, \{(x_1, \ldots, x_{2^{j-1}}), (x_{1+2^{j-1}}, \ldots, x_{2^j})\}, \pi).$$

The algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{Vf}, \mathsf{Extract})$ for seBARG_1 are then given by the following description.

- $\mathsf{seBARG}_1.\mathsf{Gen}(1^\lambda, k, 1^s, i^*) \to (\mathsf{crs}, \mathsf{td})$.
  Let $K = \lceil \log k \rceil$, and let $i_{K-1} i_{K-2} \ldots i_1 i_0$ be the binary representation of i^*; denote by $\tilde{i}_j = \lfloor i^*/2^{j+1} \rfloor = i_{K-1} i_{K-2} \ldots i_{j+1}$.
  For each $j \in [K]$, run $(\mathsf{crs}_j, \mathsf{td}_j) = \mathsf{seBARG}_0.\mathsf{Gen}(1^\lambda, 2, 1^{s_j}, i_j)$, where $s_0 = s$, and s_{j+1} is an upper bound to the size of the verification circuit $\mathcal{R}_j$ at layer j.
  Return $\mathsf{crs} = \{\mathsf{crs}_j\}, \mathsf{td} = \{\mathsf{td}_j\}$.
- $\mathsf{seBARG}_1.\mathsf{P}(\mathsf{crs}, C, \{x_i\}_{i \in [k]}, \{w_i\}_{i \in [k]}) \to \pi$.
  Recursively compute proofs in the following way: in the first step, compute

  $$\pi_i^{(0)} = \mathsf{seBARG}_0.\mathsf{P}(\mathsf{crs}_0, \mathcal{L}, \{x_{2i}, x_{2i+1}\}, \{w_{2i}, w_{2i+1}\}).$$

  Now, for any $1 \leq j \leq K-1$ define $y_i^{(j)} = (x_{i \cdot 2^j}, \ldots, x_{(i+1) \cdot 2^j - 1})$.
  Then, recursively compute

  $$\pi_i^{(j)} = \mathsf{seBARG}_0.\mathsf{P}\left(\mathsf{crs}_j, \mathcal{L}_j, \left\{y_{2i}^{(j)}, y_{2i+1}^{(j)}\right\}, \left\{\pi_{2i}^{(j-1)}, \pi_{2i+1}^{(j-1)}\right\}\right).$$

  Output $\pi_0^{(K-1)}$ as the proof.

[2] This framework can also be trivially adapted to use a ℓ-ary tree, instead of a binary one. The resulting CRS has size $\log_\ell(k) \cdot \mathsf{poly}(\lambda, \ell)$.

– $\mathsf{seBARG}_1.\mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) \to \{0,1\}$.
 Recursively recompute the $y_i^{(j)}$s and output the result of

$$\mathsf{seBARG}_0.\mathsf{V}\left(\mathsf{crs}_{K-1}, \mathcal{L}_{K-1}, \left\{y_0^{(K-1)}, y_1^{(K-1)}\right\}, \pi\right).$$

– $\mathsf{seBARG}_1.\mathsf{Extract}(\mathsf{td}, C, \{x_i\}_{i\in[k]}, \pi) \to w^*$.
 Recursively extract the proofs until the last layer, and then extract the witness. In particular, recompute the $y_i^{(j)}$s, define $\pi^{(K-1)} = \pi$ and then recursively compute

$$\pi^{(j-1)} = \mathsf{seBARG}_0.\mathsf{Extract}\left(\mathsf{td}_j, \mathcal{L}_j, \left\{y_{2\tilde{i}_j}^{(j)}, y_{2\tilde{i}_j+1}^{(j)}\right\}, \pi^{(j)}\right).$$

 Finally, return

$$w^* = \mathsf{seBARG}_0.\mathsf{Extract}\left(\mathsf{td}_0, \mathcal{L}, \left\{x_{2\tilde{i}_0}, x_{2\tilde{i}_0+1}\right\}, \pi^{(0)}\right).$$

Properties. We sketch a proof for all the required properties of the resulting scheme seBARG_1.

CRS Succinctness. The CRS of seBARG_1 consists of $\log k$ many CRSs of seBARG_0 with a constant number of statements (in particular, 2). Thus, it is of size $\log k \cdot \mathsf{poly}(\lambda)$.

Rate. Since seBARG_0 is rate-1, we have that $|\pi_i^{(j)}| = |\pi_i^{(j-1)}| + \mathsf{poly}(\lambda)$. Thus, if m is the size of a witness for $\mathcal{L}$, the proof size of seBARG_1 is $m + \log k \cdot \mathsf{poly}(\lambda)$.

Index Hiding. This property follows directly from index hiding of seBARG_0, since the crs of seBARG_1 is the union of many independent crs of seBARG_0.

Somewhere Argument of Knowledge. The following lemma establishes that seBARG_1 is a somewhere argument of knowledge, given that seBARG_0 is a somewhere argument of knowledge.

Lemma 8. *Let* seBARG_0 *be a somewhere extractable argument of knowledge, then* seBARG_1 *given above is also a somewhere argument of knowledge.*

Proof. Let $\mathcal{A}$ be an adversary against the somewhere argument-of-knowledge property of seBARG_1. In particular, let i^* the extractable index, and π the proof given by $\mathcal{A}$. We denote by $w^* = \mathsf{seBARG}_1.\mathsf{Extract}(\mathsf{td}, C, \{x_i\}_{i\in[k]}, \pi)$ the extracted witness, and recall that the extraction algorithm also extracts witnesses $w_j = \pi^{(j)}$ for each layer. Consider then the following hybrids.

– Hybrid $\mathcal{H}_0$: This is the real experiment
– Hybrid $\mathcal{H}_k$ (for $k = 1, \ldots, K-1$): This is the same as hybrid $\mathcal{H}_{k-1}$, except that the experiment outputs 0 if the conditions $\mathcal{R}_j(y_{\tilde{i}_j}^{(j)}, w_j) \neq 1$ and $\mathsf{seBARG}_0.V\left(\mathsf{crs}_j, \left\{y_{2\tilde{i}_j}^{(j)}, y_{2\tilde{i}_j+1}^{(j)}\right\}, \pi^{(j)}\right) = 1$ hold, where $j = K-1-k$.

Note that the last experiment $\mathcal{H}_{K-1}$ aborts if $\mathcal{R}_0(x_{i^*}, w^*) \neq 1$. But since $x_{i^*} \notin \mathcal{L}_0 = \mathcal{L}$, this experiment always outputs 0, i.e. $\mathcal{A}$ has advantage 0 in this experiment.

It remains to show that experiments $\mathcal{H}_{k-1}$ and $\mathcal{H}_k$ are indistinguishable given that seBARG_0 is somewhere extractable. Concretely, if $|\Pr[\mathcal{H}_k = 1] - \Pr[\mathcal{H}_{k-1} = 1]| \geq \epsilon$ we can construct an adversary $\mathcal{A}'$ against the somewhere argument of knowledge property of seBARG_0 with advantage ϵ as follows. $\mathcal{A}'$ simulates $\mathcal{H}_{k-1}$ but only outputs the statements $y^{(j)}_{2\tilde{i}_j}$ and $y^{(j)}_{2\tilde{i}_j+1}$ as well as the proof $\pi^{(j)}$. If $y^{(j)}_{\tilde{i}_{j-1}} \in \mathcal{L}_j$ both experiments are identically distributed. Hence, it must holds that $y^{(j)}_{\tilde{i}_{j-1}} \notin \mathcal{L}_j$ but $\mathsf{seBARG}_0.V\left(\mathsf{crs}_j, \left\{y^{(j)}_{2\tilde{i}_j}, y^{(j)}_{2\tilde{i}_j+1}\right\}, \pi^{(j)}\right) = 1$ with probability at least ϵ. Hence $\mathcal{A}'$ breaks the somewhere argument of knowledge property of seBARG_0 with advantage ϵ, which concludes the proof.

4.3 RAM SNARGs with Partial Input Soundness

A RAM SNARG allows a verifier to verify that a RAM computation was well performed given just the hash of the input (or initial database) h and a proof π. Importantly, the verifier should run in time $\mathsf{poly}(\lambda, \log T)$ where T is the running time of the RAM computation.

Here, we are interested in RAM SNARGs that achieve a strong soundness property known as partial input soundness [15]. This guarantees that if the memory is digested using a SEH function that is extractable on a set of coordinates I, and if the RAM computation only reads coordinates in I, then soundness holds. We refer the reader to [9,15] for formal definitions.

It is known that a flexible RAM SNARG can be constructed from seBARGs and a fully-local SEH function.

Lemma 9 ([15]). *Assuming the existence of a* seBARG *and a fully-local* SEH*, there exists a RAM SNARG with partial input soundness.*

Let S be the size of a single intermediate state of the RAM computation. Then the RAM SNARG construction presented in [15] has proof size $S \cdot \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda, \log T, S)$, where $S \cdot \mathsf{poly}(\lambda)$ corresponds to the output of the (fully-local) SEH and $\mathsf{poly}(\lambda, \log T, S)$ corresponds to the size of the seBARG proof. Additionally, assume that only V positions are read from the initial memory $\mathbf{X}$. Then the hash value of $\mathbf{X}$ has size $V \cdot \mathsf{poly}(\lambda)$.

If we instantiate the underlying seBARG with a rate-1 BARG (from Corollary 5) and the fully-local SEH with a rate-1 scheme (as the one from Sect. 3.2), we obtain the following corollary.

Corollary 6. *There exists a RAM SNARG with partial input soundness from subexponential DDH or k-LIN assumptions with proof size* $\mathcal{O}(S) + \mathsf{poly}(\lambda)$ *and an hash value (of the initial database) of size* $V + \mathsf{poly}(\lambda)$*.*

Acknowledgements. Pedro Branco was partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038.

Nico Döttling and Riccardo Zanotto: Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Ananth, P., Lombardi, A.: Succinct garbling schemes from functional encryption through a local simulation paradigm. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 455–472. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-030-03810-6_17
2. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 509–539. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_19
3. Brakerski, Z., Branco, P., Döttling, N., Garg, S., Malavolta, G.: Constant ciphertext-rate non-committing encryption from standard assumptions. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 58–87. Springer, Heidelberg (Nov (2020). https://doi.org/10.1007/978-3-030-64375-1_3
4. Brakerski, Z., Holmgren, J., Kalai, Y.T.: Non-interactive delegation and batch NP verification from standard computational assumptions. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC, pp. 474–482. ACM Press (2017)
5. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 33–65. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-63715-0_2
6. Choudhuri, A.R., Garg, S., Jain, A., Jin, Z., Zhang, J.: Correlation intractability and SNARGs from sub-exponential DDH. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. LNCS, pp. 635–668. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38551-3_20
7. Choudhuri, A.R., Jain, A., Jin, Z.: Non-interactive batch arguments for NP from standard assumptions. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 394–423. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-84259-8_14
8. Choudhuri, A.R., Jain, A., Jin, Z.: SNARGs for $\mathcal{P}$ from LWE. In: 62nd FOCS, pp. 68–79. IEEE Computer Society Press (2022)
9. Devadas, L., Goyal, R., Kalai, Y., Vaikuntanathan, V.: Rate-1 non-interactive arguments for batch-NP and applications. In: 63rd FOCS, pp. 1057–1068. IEEE Computer Society Press (2022)
10. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-63688-7_18

11. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 3–32. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-26954-8_1
12. Garg, S., Srinivasan, A.: A simple construction of iO for turing machines. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 425–454. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-030-03810-6_16
13. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) ITCS 2015, pp. 163–172. ACM (2015)
14. Hulett, J., Jawale, R., Khurana, D., Srinivasan, A.: SNARGs for P from subexponential DDH and QR. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 520–549. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-07085-3_18
15. Kalai, Y., Lombardi, A., Vaikuntanathan, V., Wichs, D.: Boosting batch arguments and ram delegation. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, pp. 1545–1552. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3564246.3585200
16. Kalai, Y.T., Vaikuntanathan, V., Zhang, R.Y.: Somewhere statistical soundness, post-quantum security, and SNARGs. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, vol. 13042, pp. 330–368. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-90459-3_12
17. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, pp. 419–428. ACM Press (2015)
18. Okamoto, T., Pietrzak, K., Waters, B., Wichs, D.: New realizations of somewhere statistically binding hashing and positional accumulators. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 121–145. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_6
19. Paneth, O., Pass, R.: Incrementally verifiable computation via rate-1 batch arguments. In: 63rd FOCS, pp. 1045–1056. IEEE Computer Society Press (2022)
20. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC, pp. 84–93. ACM Press (2005)
21. Waters, B., Wu, D.J.: Batch arguments for NP and more from standard bilinear group assumptions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 433–463. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-15979-4_15