Rai-Choo! Evolving Blind Signatures to the Next Level

Lucjan Hanzlik [1] Julian Loss [1] Benedikt Wagner* [1,2]

February 24, 2023

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
{hanzlik,loss,benedikt.wagner}@cispa.de
[2] Saarland University, Saarbrücken, Germany

Abstract

Blind signatures are a fundamental tool for privacy-preserving applications. Known constructions of concurrently secure blind signature schemes either are prohibitively inefficient or rely on non-standard assumptions, even in the random oracle model. A recent line of work (ASIACRYPT '21, CRYPTO '22) initiated the study of concretely efficient schemes based on well-understood assumptions in the random oracle model. However, these schemes still have several major drawbacks: 1) The signer is required to keep state; 2) The computation grows linearly with the number of signing interactions, making the schemes impractical; 3) The schemes require at least five moves of interaction.

In this paper, we introduce a blind signature scheme that eliminates *all* of the above drawbacks at the same time. Namely, we show a round-optimal, concretely efficient, concurrently secure, and stateless blind signature scheme in which communication and computation are independent of the number of signing interactions. Our construction also naturally generalizes to the partially blind signature setting.

Our scheme is based on the CDH assumption in the asymmetric pairing setting and can be instantiated using a standard BLS curve. We obtain signature and communication sizes of 9 KB and 36 KB, respectively. To further improve the efficiency of our scheme, we show how to obtain a scheme with better amortized communication efficiency. Our approach *batches* the issuing of signatures for multiple messages.

Keywords: Blind Signatures, Standard Assumptions, Random Oracle Model, Cut-and-Choose, Computation Complexity, Round Complexity.

1 Introduction

Blind signatures, introduced by David Chaum in 1982 [Cha82] are an interactive type of signature scheme with special privacy features. Informally, in a blind signature scheme, a Signer, holding a secret key sk, and a User, holding a corresponding public key pk and a message m, engage in a two-party protocol. At the end of the interaction, the user obtains a signature on m that can be verified using pk. *Blindness* ensures that the Signer learns no information about m. On the other hand, *unforgeabillity* guarantees that the User cannot obtain valid signatures without interacting with the Signer. These properties make blind signatures a useful building block for privacy-sensitive applications, e.g. e-cash [Cha82, OO92], anonymous credentials [CG08, CL01], e-voting [GPZZ19], and blockchain-based systems [HBG16].

Unfortunately, even in the random oracle model, existing constructions of blind signatures either rely on non-standard assumptions [Bol03, BNPS03, FHS15], or have parameters (e.g. communication and signature sizes) that grow linearly in the number of concurrent signing interactions [PS00, HKL19, KLR21]. Very recently, Chairattana-Apirom et al. [CAHL+22] gave the first blind signature schemes from standard assumptions in the random oracle model that are simultaneously size and communication efficient. Even

*Main author

so, their schemes cannot be considered practical. For one, they require many rounds of interaction, which may be problematic if network conditions are poor. Second, they still require computation that grows linearly in the number of signatures that the server has already issued. This can become a heavy burden as the number of signatures grows large, say 2^{30}. In this work, we propose a novel construction of a blind signature scheme that overcomes *all* of these limitations. Concretely, our scheme has the following properties:

- Our scheme can be instantiated from the (co)-CDH assumption in type-3 pairings in the random oracle model (to get a proof from plain CDH, we can easily use type-2 or type-1 pairings).
- It has compact signatures and communication complexity.
- Signing and verifying are computationally efficient and require only a few hundred hash and group operations per signature; we provide a prototypical implementation to demonstrate practicality.
- Our scheme is round-optimal, i.e., it requires only a single message from both the signer and the user.

1.1 Background and Limitations of Existing Constructions

A long line of work [PS00, AO00, Abe01, HKL19, HKLN20] has explored constructions of blind signatures from witness indistinguishable linear identification schemes such as the Okamoto-Schnorr and Okamoto-Guillou-Quisquater schemes [Oka93]. The resulting blind signature schemes are secure under well-understood assumptions, such as RSA, factoring, or discrete logarithm. On the downside, some these schemes admit an efficient attack [Sch01, Wag02, BLL+21] if the number of (concurrent) singing interactions ever exceeds a polylogarithmic bound.

Inspired by an early work by Pointcheval [Poi98], Katz, Loss, and Rosenberg [KLR21] recently introduced a boosting transform that turns linear blind signature schemes into fully secure ones (i.e., admitting a polynomial number of concurrent signing interactions). Applying their transform, one obtains schemes that rely on well-studied assumptions and have short signatures. Unfortunately, the resulting communication and computational complexity renders them impractical. This is because in the Nth interaction between Signer and User, the communication and computation depend *linearly on* N. To ameliorate some of these drawbacks, Chairattana-Apirom et al. [CAHL+22] introduced a more compact version of Katz et al.'s generic transform in which the communication only depends *logarithmically on* N. Their work also presents two more optimized blind signature schemes which do not follow from their transform generically. We focus here on their BLS-based [BLS01, Bol03] construction (called 'PI-Cut-Choo') which can be instantiated from CDH.

We briefly highlight the remaining drawbacks of PI-Cut-Choo. The idea of the boosting transform fundamentally relies on a 1-of-N cut-and-choose where N, the number of signing interactions, grows over time. This requires to execute N copies of the base scheme and has the following implications:

- The Signer is stateful, as it has to keep track of the current value of N.
- The computation grows linearly in N for both the Signer and the User. To issue $N \approx 2^{30}$ signatures, this would require prohibitive computational effort (roughly $\sum_{i=1}^{2^{30}} i \approx 2^{59}$ operations).
- Issuing a signature requires five moves of interaction between Signer and User which is a far cry from the theoretical one-round limit achieved by some schemes [Bol03].

Thus, even though PI-Cut-Choo significantly improves over prior schemes, it can still not be considered useful for practical deployment.

1.2 Our Contribution

In this work, we eliminate *all* of the aforementioned drawbacks.

Our Scheme. We construct a practical blind signature scheme using a new variant of the cut-and-choose technique, that is polynomially secure and does not require the signer to keep a state. This eliminates the dependency on a counter N as in [KLR21, CAHL+22] entirely, thereby also significantly reducing the computational complexity, see Table 1. Additionally, in contrast to schemes in [KLR21, CAHL+22], our scheme is round-optimal. Our scheme is statistically blind against malicious signers. We show one-more unforgeability based on the (co)-CDH assumption in asymmetric pairing groups. One-more unforgeability holds for any (a priori unbounded) polynomial number of signing interactions. We obtain several parameter settings for our scheme. This leads to a trade-off between signature and communication size, see Table 2. For example, we can instantiate parameters to obtain 9 KB signature size and 36 KB communication complexity. To demonstrate that our scheme is computationally efficient, we implemented a prototype over the BLS12–381 curve. Our experiments show that signing takes less than 0.2 seconds, see Table 2.

	Boosting [KLR21]	Compact Boosting [CAHL+22]		Our Work
Moves	7	7	5	2
Communication	$\Theta(\lambda N)$	$\Theta(\lambda \log N)$	$\Theta(\lambda \log N + \lambda^2)$	$\Theta(\lambda^2)$
Computation	$\Theta(N)$	$\Theta(N \log N)$	$\Theta(\lambda N)$	$\Theta(\lambda)$

Table 1: Comparison of number of moves, communication and computation for the line of work [KLR21, CAHL+22] and our work in the Nth signing interaction. The security parameter is denoted by λ. Communication is given in bits, and computation is given by treating pairings, group and field operations, and hash evaluations as one unit.

			Communication with batch size L				Running Time	
	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$	$L = 1$	$L = 4$	$L = 16$	$L = 256$	Sign	Verify
(I)	0.14	13.98	33.20	16.98	12.92	11.65	163	54
(II)	0.14	9.41	36.21	20.11	16.08	14.82	169	36
(III)	0.14	5.71	72.79	43.97	36.77	34.52	333	22

Table 2: Efficiency of different parameter settings of our scheme. Sizes and times are given in kilobytes and milliseconds, respectively. Communication is amortized per message. Details can be found in Section 5.

Partial Blindness and Batching. We show that our scheme naturally generalizes to the setting of partially blind signatures. Additionally, we show how we can *batch* multiple signing interactions to improve communication complexity (see also Table 2), and provide the first formal model and analysis for that. Batching has been used in many other contexts as well, e.g. in oblivious transfer [IKNP03, BBDP22]. We believe that batching blind signatures has a lot of natural use-cases. As an example, consider an e-cash scenario. Here, parties withdraw coins from a bank by getting blind signature for a random message. Later, the coin can be deposited by presenting the message-signature pair. Blindness ensures that the process of withdrawal is not linkable to the process of depositing. This approach is also used to do enhance the anonymity in electronic payment systems [HBG16]. We remark that it is crucial that all issued coins are of equal amount to guarantee a large anonymity set. Therefore, any user that wants to retrieve more than one coin has to interact with the bank multiple times to get multiple coins (i.e. signatures). Using batch blind signatures, these interactions can all happen in parallel, leading to improved communication and computational efficiency, as well as reduced overhead to initiate interactions.
Remark on Assumptions. In our construction, we use the asymmetric type-3 pairing setting, as standard in practical pairing-based schemes. This also means that we need to use the standard variant of CDH in this setting, sometimes called co-CDH [CHKM09]. We emphasize that this variant is even needed to prove unforgeability of standard BLS signatures in the asymmetric type-3 setting [BLS01]. On the other hand, it is straight-forward to instantiate our scheme in the symmetric pairing setting, or the asymmetric type-2 pairing setting based on plain CDH. We refer to Section 5 for more details.

1.3 Technical Overview

We give an intuitive overview of our techniques. For full formal details, we refer to the main body.

Boosting and PI-Cut-Choo. We start this overview by recalling the boosting transform [KLR21] and its parallel instance variant [CAHL$^+$22]. Let BS be a blind signature scheme which is secure against an adversary that queries the signer for a small number of signatures (we will give a suitable definition of "small" below). The boosting transform results in a new scheme which is secure for any number of signing interactions between signer and adversary. In the Nth signing interaction, the User and the Signer behave as follows.

1. The user commits to its message m using randomness $\varphi_j,\ j \in [N]$, thereby obtaining N commitments μ_j. It also samples random coins $\rho_j,\ j \in [N]$ for the user algorithm of BS. Then, it commits to each pair (μ_j, ρ_j) using a random oracle, and sends the resulting commitments com_j to the Signer.

2. Signer and User run the underlying scheme BS N times in parallel. We refer to these N parallel runs as *sessions*. More precisely, the Signer uses its secret key sk, and the User uses the public key pk, μ_i as the message, and ρ_j as the random coins in the jth session, for $j \in [N]$.

3. Before the final messages $s_j,\ j \in [N]$ are sent from the Signer to the User, the Signer selects a random session $J \in [N]$. The user now has to open all the commitments com_j for $j \in [N] \setminus \{J\}$ by sending (μ_j, ρ_j). The Signer can now verify that the User behaved honestly for all but the Jth session. In case the User behaved dishonestly in one session, the Signer aborts.

4. The Signer completes the Jth session by sending the final message s_J. Finally, the User derives a signature σ_J from that session as in BS, and outputs $\sigma = (\sigma_J, \varphi_J)$ as its final signature.

Katz, Loss, and Rosenberg [KLR21] show that the above scheme is secure for polynomially many signing interactions, given that the underlying scheme BS is secure for logarithmically many signing interactions. In more detail, they provide a reduction that simulates a signer oracle for the new scheme, given a logarithmic number of queries to the signer oracle for BS. Their reduction distinguishes the following cases for the Nth signing interaction.

1. If the adversary (i.e. the User) is dishonest in at least two sessions, then the adversary is caught. Hence, no response has to be provided and no secret key is needed.

2. If the adversary is honest in all sessions, the reduction can extract all (μ_j, ρ_j) by inspecting random oracle queries. Using a special property of the underlying scheme BS, this allows the reduction to simulate the response, e.g. by programming the random oracle.

3. If the adversary is dishonest in exactly one session j^*, then either $J \neq j^*$ and the reduction works as in the previous case, or $J = j^*$, and the reduction has to use the signer oracle of BS to provide the response s_J. In this case, we say that there is a *successful cheat*.

It is clear that the probability of a successful cheat is at most $1/N$ in the Nth signing interaction. Therefore, the expected number of successful cheats over q signing interactions is at most $\sum_{N=2}^{q+1} 1/N \leq O(\log q)$. Using an appropriate concentration bound, it therefore can be argued that the underlying signer oracle for BS is called logarithmically many times.

Unfortunately, the above transform yields impractical parameter sizes for the resulting signature scheme, which results from a relatively loose reduction to BS. To overcome these issues, recent work introduced a parallel instance version of the boosting transform (hereafter PI-Cut-Choo) [CAHL$^+$22]. The primary goal of this version is to work for key-only secure schemes BS, i.e. such that the reduction can simulate signing queries in the transformed scheme entirely without accessing the signing oracle of BS. First, N is scaled by some constant, such that the expected number of successful cheats is less than 1[1]. Thus, in expectation, the reduction does not need access to a signer oracle for BS. To ensure that this is true with overwhelming probability, the entire boosting transform is repeated with $K = \Theta(\lambda)$ *instances* in parallel. These instances use independent public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ and independent random coins[2]. This implies that with overwhelming probability, there will be an instance $i^* \in [K]$, such that there is no successful cheat in instance i^* over the entire runtime of the reduction. The reduction can now guess

[1] This assumes an upper known bound on the number of signing interactions, which is a minor limitation. Alternatively, one could instead increase N as N^2 to achieve an expected constant number of successful cheats.

[2] In PI-Cut-Choo, this parallel repetition comes almost for free due to a lot of optimizations that we do not cover in this overview.

i^* and embed the target public key of BS in pk_{i^*}. If the guess was correct, the reduction to key-only security of BS goes through.

The above discussion highlights the importance of growing the parameter N as a function of the number of signing interactions over time. In summary, it allows to bound the expected number of successful cheats, which is the central idea of prior work [KLR21, CAHL+22]. Thus, keeping N fixed presents several technical challenges that we discuss in the next paragraph.

Strawman One: Fixed Cut-and-Choose. We are now ready to describe our central ideas to avoid a growing cut-and-choose parameter N. As explained above, the key idea of PI-Cut-Choo is to ensure that for one of the parallel instances i^*, the adversary *never cheats* in any of its interactions with the signer. This argument fails if we set N to be constant, e.g. $N = 2$. However, by keeping the number of parallel instances K the same, we can still argue that with overwhelming probability *in each signing interaction*, there is a non-cheating instance i^*. We highlight the reversed role of quantifiers: The non-cheating instance i^* could now be different for every signing interaction. Unfortunately, the reduction approach presented in PI-Cut-Choo only allows to embed the target public key of the underlying scheme BS in a *fixed key* among the keys $\mathsf{pk}_1, \dots, \mathsf{pk}_K$ corresponding to the K parallel instances. Once this key is fixed, the reduction fails if ever there is a successful cheat with respect to this instance.

Strawman Two: Dynamic Key Structure (Naively). The above discussion shows that we have to support a *dynamic embedding* of the target public key into one of the keys $\mathsf{pk}_1, \dots, \mathsf{pk}_K$. The first (naive) idea would be to use a fresh set of public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_K$ and secret keys $\mathsf{sk}_1, \dots, \mathsf{sk}_K$ in each interaction. Observe that in PI-Cut-Choo, the base scheme BS is a two-move scheme, in which the first message c (challenge) sent from user to signer does not depend on the public key. Thus, our reduction for the resulting scheme can identify the non-cheating instance i^* *after* seeing the commitments $\mathsf{com}_{i,j}$ and the challenges $c_{i,j}$. Using this observation, we could let the Signer send the (fresh) public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_K$ that will be used in the current signing interaction after receiving commitments and challenges. This way, the reduction knows in which key pk_{i^*} to embed the target public key in each signing interaction. To do so, the reduction first identifies the non-cheating instance i^*, and then samples $(\mathsf{pk}_i, \mathsf{sk}_i)$ for $i \neq i^*$ honestly, while setting pk_{i^*} to (a rerandomization of) the target public key. Finally, the reduction can use sk_i to simulate all instances except i^*, while using random oracle programming in instance i^*.

We can use random-self reducibility of the underlying signature scheme to ensure blindness of this construction. Namely, the User re-randomizes the keys and signatures it receives from the user. (Otherwise, it would be trivial to link signatures to signing interactions). The final signature then contains the rerandomized set of keys and signatures. Fortunately, the BLS scheme [Bol03], which serves as the basis of PI-Cut-Choo, has such a property.

However, the above scheme is insecure. Since a fresh set of keys $\mathsf{pk}_1, \dots, \mathsf{pk}_K$ is used in every interaction, there is nothing tying signatures to the Signer's actual public and secret key. In particular, there is no way from preventing the adversary from (trivially) creating a forgery containing a set of keys of its own choice. In the security proof, the reduction can not extract a forgery for BS with respect to the target public key in this scenario.

Our Solution: PI-Cut-Choo evolves to Rai-Choo. To overcome the remaining issues of the above strawman approach, we fix one public key pk and one secret key sk for our scheme. Instead of using independent public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_K$ for each interaction, we instead use a sharing

$$(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_K, \mathsf{sk}_K) \text{ such that } \sum_i \mathsf{sk}_i = \mathsf{sk} \text{ and } \prod_i \mathsf{pk}_i = \mathsf{pk}.$$

By setting pk to be the target public key of the underlying scheme BS and carefully working out the details, our reduction is now able to extract a forgery as required. It remains to sketch why the simulation of the signing oracle is still possible with this new structure of the $\mathsf{pk}_1, \dots, \mathsf{pk}_K$. Note that the reduction can define the $\mathsf{pk}_1, \dots, \mathsf{pk}_K$ in a way that allows it to know all but one sk_i. Concretely, after identifying the non-cheating instance i^* in an interaction with the adversary, the reduction first samples $(\mathsf{pk}_i, \mathsf{sk}_i)$ for all $i \in [K] \setminus \{i^*\}$, and then sets $\mathsf{pk}_{i^*} := \mathsf{pk} \cdot \prod_{i \neq i^*} \mathsf{pk}_i^{-1}$. This is identically distributed to the real sharing.

In summary, we have successfully transformed a key-only secure scheme BS into a fully secure one, while using a constant cut-and-choose parameter N. We can further optimize the scheme using many minor tricks, some of them similar to [CAHL+22]. In the process we also manage to reduce the number of moves to two, which is optimal. This is because in our new scheme, we can make the cut-and-choose step completely non-interactive using a random oracle, and the signer does not need to send N anymore, as it is fixed.

1.4 More on Related Work

We discuss related work in more detail.

Impossibility. There are several impossibility results about the construction of blind signatures in the standard model [FS10, Pas11, BL13]. Fischlin and Schröder showed that statistically blind three-move schemes can not be constructed from non-interactive assumptions under certain conditions [FS10]. Pass showed that unique round-optimal blind signatures can not be based on a class of interactive assumptions [Pas11]. Baldimtsi and Lysyanskaya showed that schemes with a unique secret key and a specific structure can not be proven secure, even under interactive assumptions [BL13].

Sequential vs. Concurrent Security. In terms of unforgeability, one distinguishes concurrent and sequential security. For sequential security, the adversary has to finish one interaction with the signer before initiating the following interaction. In contrast, concurrent security allows the adversary to interact with the signer in an arbitrarily interleaved way. In practice, restricting communication with the signer to sequential access opens a door for denial of service attacks. Therefore, concurrent security is the widely accepted notion.

Generic Constructions. One can build blind signatures generically from standard signatures and secure two-party computation (2PC), as shown by Juels, Luby and Ostrovsky [JLO97]. Unfortunately, this construction only achieves sequential security. Contrary to that, Fischlin [Fis06] gave a (round-optimal) generic construction that is secure even in the universal composability framework [Can00]. However, it turns out that instantiating these generic constructions efficiently is highly non-trivial. For example, instantiating Fischlin's construction requires to prove statements in zero-knowledge about a combination of commitment and signature scheme. If we instantiate the signature scheme efficiently in the random oracle model, we end up treating the random oracle as a circuit. This leads to unclear implications in terms of security. Additionally, schemes based on Fischlin's construction inherently require strong decisional assumptions due to the use of zero-knowledge proofs and encryption. The recent work by Katsumata and del Pino [dK22] makes significant progress in this direction. By carefully choosing building blocks and slightly tweaking the construction, they give an instantiation of Fischlin's paradigm in the lattice setting. However, the communication complexity of their protocol is still far from being practical.

Efficiency from Strong Assumptions. In addition to the generic constructions mentioned above, there are direct constructions of blind signatures. While some constructions make use of complexity leveraging [GRS+11, GG14], others are proven secure under non-standard q-type or interactive assumptions [Oka06, GRS+11, FHS15, Gha17]. Notably, there are efficient and round-optimal schemes based on the full-domain-hash paradigm [Bol03, BNPS03, AKSY21]. For example, Boldyreva [Bol03] introduces a blinded version of the BLS signature scheme [BLS01]. To prove security, one relies on the non-standard one-more variant of the underlying assumption (e.g. one-more CDH for BLS).

Idealized Models. In addition to the works in the standard and random oracle model mentioned before, there are also constructions [FPS20, KLX22, TZ22] that are proven secure in more idealized models, such as the algebraic or generic group model [FKL18, Sho97]. While it leads to efficient schemes, we want to avoid using such a model, as it is non-standard.

2 Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$, and assume that all algorithms get 1^λ implicitly as input. Let S be a finite set and $\mathcal{D}$ be a distribution. We write $x \leftarrow_{\$} S$ to indicate that x is sampled uniformly at random from S. We write $x \leftarrow \mathcal{D}$ if x is sampled according to $\mathcal{D}$. Let $\mathcal{A}$ be a (probabilistic) algorithm. We write $y \leftarrow \mathcal{A}(x)$, if y is output from $\mathcal{A}$ on input x with uniformly sampled random coins. To make these random coins ρ explicit, we write $y = \mathcal{A}(x;\rho)$ The notation $y \in \mathcal{A}(x)$ means that y is a possible output of $\mathcal{A}(x)$. As always, an algorithm is said to be PPT if its running time $\mathbf{T}(\mathcal{A})$ is bounded by a polynomial in its input size. A function $f : \mathbb{N} \to \mathbb{R}_+$ is negligible in its input λ, if $f \in \lambda^{-\omega(1)}$. Let $\mathbf{G}$ be a security game. We write $\mathbf{G} \Rightarrow b$ to indicate that $\mathbf{G}$ outputs b. The first K natural numbers are denoted by $[K] := \{1, \ldots, K\}$. Next, we define the cryptographic primitive of interest and the computational assumption that we use.

Definition 1 (Blind Signature Scheme)**.** A blind signature scheme is a quadruple of PPT algorithms $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ takes as input the security parameter 1^λ and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$. We assume that the public key pk defines a message space $\mathcal{M} = \mathcal{M}_{\mathsf{pk}}$ implicitly.
- S and U are interactive algorithms, where S takes as input a secret key sk and U takes as input a key pk and a message $\mathsf{m} \in \mathcal{M}$. After the execution, U returns a signature σ and we write $(\perp, \sigma) \leftarrow \langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, \mathsf{m})\rangle$.
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$ is deterministic and takes as input public key pk, message $\mathsf{m} \in \mathcal{M}$, and a signature σ, and returns $b \in \{0, 1\}$.

We require that BS is complete in the following sense. For all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$ and all $\mathsf{m} \in \mathcal{M}_{\mathsf{pk}}$ it holds that

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) = 1 \mid (\perp, \sigma) \leftarrow \langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, \mathsf{m})\rangle\right] = 1.$$

Definition 2 (One-More Unforgeability)**.** Let $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ be a blind signature scheme and $\ell \colon \mathbb{N} \to \mathbb{N}$. For an algorithm $\mathcal{A}$, we consider the following game $\ell\textbf{-OMUF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda)$:

1. Sample keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Let O be an interactive oracle simulating $\mathsf{S}(\mathsf{sk})$. Run
$$((\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}),$$
where $\mathcal{A}$ can query O in an arbitrarily interleaved way and complete at most $\ell = \ell(\lambda)$ of the interactions with O.
3. Output 1 if and only if all $\mathsf{m}_i, i \in [k]$ are distinct, $\mathcal{A}$ completed at most $k - 1$ interactions with O and for each $i \in [k]$ it holds that $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}_i, \sigma_i) = 1$.

We say that BS is ℓ-one-more unforgeable (ℓ-OMUF), if for every PPT algorithm $\mathcal{A}$ the following advantage is negligible:

$$\mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}}(\lambda) := \Pr\left[\ell\textbf{-OMUF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda) \Rightarrow 1\right].$$

We say that BS is one-more unforgeable (OMUF), if it is ℓ-OMUF for all polynomial ℓ.

Definition 3 (Blindness)**.** Consider a blind signature scheme $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$. For an algorithm $\mathcal{A}$ and bit $b \in \{0, 1\}$, consider the following game $\textbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}(\lambda)$:

1. Run $(\mathsf{pk}, \mathsf{m}_0, \mathsf{m}_1, St) \leftarrow \mathcal{A}(1^\lambda)$.
2. Let O_0 be an interactive oracle simulating $\mathsf{U}(\mathsf{pk}, \mathsf{m}_b)$ and O_1 be an interactive oracle simulating $\mathsf{U}(\mathsf{pk}, \mathsf{m}_{1-b})$. Run $\mathcal{A}$ on input St with arbitrary interleaved one-time access to each of these oracles, i.e. $St' \leftarrow \mathcal{A}^{\mathrm{O}_0, \mathrm{O}_1}(St)$.
3. Let σ_b, σ_{1-b} be the local outputs of $\mathrm{O}_0, \mathrm{O}_1$, respectively. If $\sigma_0 = \perp$ or $\sigma_1 = \perp$, then run $b' \leftarrow \mathcal{A}(St', \perp, \perp)$. Else, obtain a bit b' from $\mathcal{A}$ on input σ_0, σ_1, i.e. run $b' \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$.
4. Output b'.

We say that BS satisfies malicious signer blindness, if for every PPT algorithm $\mathcal{A}$ the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) := \left|\Pr\left[\textbf{BLIND}^{\mathcal{A}}_{0,\mathsf{BS}}(\lambda) \Rightarrow 1\right] - \Pr\left[\textbf{BLIND}^{\mathcal{A}}_{1,\mathsf{BS}}(\lambda) \Rightarrow 1\right]\right|.$$

We make use of the natural variant of the CDH assumption in the asymmetric pairing setting [CHKM09].

Definition 4 (CDH Assumption)**.** Let $\mathsf{PGGen}(1^\lambda)$ be a bilinear group generation algorithm that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a non-degenerate[3] pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$. We say that the CDH assumption holds relative to PGGen, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathsf{PGGen}}(\lambda) := \Pr\left[z = xy \;\middle|\; \begin{array}{l} (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda), \\ x, y \leftarrow_\$ \mathbb{Z}_p,\ X_1 := g_1^x,\ X_2 := g_2^x,\ Y := g_1^y \\ g_1^z \leftarrow \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e, X_1, Y, X_2) \end{array} \right]$$

3 Our Blind Signature Scheme

In this section, we present our blind signature scheme.

3.1 Construction

Let $\mathsf{PGGen}(1^\lambda)$ be a bilinear group generation algorithm that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$. We assume that these system parameters are known to all algorithms. Note that their correctness can be verified efficiently. Our scheme $\mathsf{BS}_R = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ is parameterized by integers $K = K(\lambda), N(\lambda) \in \mathbb{N}$. These do not depend on the number of previous interactions. We only need that N^{-K} is negligible in λ. Our scheme does not require the signer to hold a state. The scheme makes use of random oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$.

Key Rerandomization. Our scheme makes use of an algorithm ReRa, that takes as input tuples $(\mathsf{pk}_i, h_i)_{i \in [K]}$ and an element $\bar{\sigma} \in \mathbb{G}_1$, where $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}) \in \mathbb{G}_1 \times \mathbb{G}_2$, and $h_i \in \mathbb{G}_1$ for all $i \in [K]$. The algorithm is as follows:

1. Choose $r_1, \ldots, r_{K-1} \leftarrow_\$ \mathbb{Z}_p$ and set $r_K := -\sum_{i=1}^{K-1} r_i$.
2. For all $i \in [K]$, set $\mathsf{pk}'_i := \left(\mathsf{pk}'_{i,1}, \mathsf{pk}'_{i,2}\right) := \left(\mathsf{pk}_{i,1} \cdot g_1^{r_i}, \mathsf{pk}_{i,2} \cdot g_2^{r_i}\right)$.
3. Set $\bar{\sigma}' := \bar{\sigma} \cdot \prod_{i=1}^K h_i^{r_i}$ and return $((\mathsf{pk}'_i)_{i \in [K]}, \bar{\sigma}')$.

It is easy to see that $\prod_{i \in K} \mathsf{pk}_{i,j} = \prod_{i \in K} \mathsf{pk}'_{i,j}$ for both $j \in \{1, 2\}$. Further, if we assume that the inputs satisfy $e(\bar{\sigma}, g_2) = \prod_{i=1}^K e\left(h_i, \mathsf{pk}_{i,2}\right)$ and $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$, then the outputs satisfy $e(\bar{\sigma}', g) = \prod_{i=1}^K e\left(h_i, \mathsf{pk}'_{i,2}\right)$ and $e\left(\mathsf{pk}'_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}'_{i,2}\right)$ for all $i \in [K]$. Additionally, the output does not reveal anything about the input, except what is already revealed by these properties. We will make this more formal in Lemma 1 when we analyze the blindness property of our scheme.

Key Generation. To generate keys algorithm $\mathsf{Gen}(1^\lambda)$ does the following:

1. Sample $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_p$, set $\mathsf{pk}_1 := g_1^{\mathsf{sk}}$ and $\mathsf{pk}_2 := g_2^{\mathsf{sk}}$.
2. Return public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$ and secret key sk.

Signature Issuing. The algorithms S, U and their interaction are formally given in Figures 1 and 2.

Verification. The resulting signature $\sigma := ((\mathsf{pk}_i, \varphi_i)_{i=1}^{K-1}), \varphi_K, \bar{\sigma})$ for a message m is verified by algorithm $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma)$ as follows:

1. Write $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ for each $i \in [K-1]$.
2. Compute $\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$ and $\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$.
3. If there is an $i \in [K]$ with $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$, return 0.
4. For each instance $i \in [K]$, compute $\mu_i := \mathsf{H}_\mu(\mathsf{m}, \varphi_i)$.
5. Return 1 if and only if
$$e(\bar{\sigma}, g_2) = \prod_{i=1}^K e\left(\mathsf{H}(\mu_i), \mathsf{pk}_{i,2}\right).$$

[3]Non-degenerate means that $e(g_1, g_2)$ is a generator of the group $\mathbb{G}_T$.

```
$\underline{\mathsf{S}(\mathsf{sk})}$
for $i \in [K-1]$ :
    $\mathsf{sk}_i \leftarrow\!\!\$\ \mathbb{Z}_p$
$\mathsf{sk}_K := \mathsf{sk} - \sum_{i=1}^{K-1} \mathsf{sk}_i$
for $i \in [K]$ :
    $\mathsf{pk}_{i,1} = g_1^{\mathsf{sk}_i}$
    $\mathsf{pk}_{i,2} = g_2^{\mathsf{sk}_i}$
    $\mathsf{pk}_i := (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$

if $\mathsf{Check}(\mathsf{open}) = 0$ :        $\xleftarrow{\mathsf{open}}$
    abort
for $i \in [K]$ : $s_i := c_{i,\mathbf{J}_i}^{\mathsf{sk}_i}$
$\bar{s} := \prod_{i=1}^{K} s_i$        $\xrightarrow{(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}}$
```

```
$\underline{\mathsf{U}(\mathsf{pk}, \mathsf{m})}$
for $(i,j) \in [K] \times [N]$ :
    $\varphi_{i,j} \leftarrow\!\!\$\ \{0,1\}^\lambda$,  $\mu_{i,j} := \mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j})$
    $\gamma_{i,j} \leftarrow\!\!\$\ \{0,1\}^\lambda$,  $\alpha_{i,j} := \mathsf{H}_\alpha(\gamma_{i,j})$
    $\mathsf{r}_{i,j} := (\mu_{i,j}, \gamma_{i,j})$,  $\mathsf{com}_{i,j} := \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,j})$
    $c_{i,j} := \mathsf{H}(\mu_{i,j}) \cdot g_1^{\alpha_{i,j}}$
$\mathsf{com} := (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$
$c := (c_{1,1}, \ldots, c_{K,N})$
$\mathbf{J} := \mathsf{H}_{cc}(\mathsf{com}, c)$
$\mathsf{open} := \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j \neq \mathbf{J}_i}, c_{i,\mathbf{J}_i}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i \in [K]}\right)$

$\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$
$\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$
$\mathsf{pk}_K := (\mathsf{pk}_{K,1}, \mathsf{pk}_{K,2})$
for $i \in [K]$ :
    if $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$ : abort
if $e(\bar{s}, g_2) \neq \prod_{i=1}^{K} e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right)$ : abort
$\bar{\sigma} := \bar{s} \cdot \prod_{i=1}^{K} \mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}$
$((\mathsf{pk}'_i)_i, \bar{\sigma}') \leftarrow \mathsf{ReRa}((\mathsf{pk}_i, \mathsf{H}(\mu_{i,\mathbf{J}_i}))_i, \bar{\sigma})$
return $\sigma := ((\mathsf{pk}'_i, \varphi_{i,\mathbf{J}_i})_{i=1}^{K-1}, \varphi_{K,\mathbf{J}_K}, \bar{\sigma}')$
```

Figure 1: Signature issuing protocol of the blind signature scheme BS_R, where algorithm Check is defined in Figure 2.

```
Alg $\mathsf{Check}\left(\mathsf{open} = \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j \neq \mathbf{J}_i}, c_{i,\mathbf{J}_i}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i \in [K]}\right)\right)$
for $i \in [K]$ :
    for $j \in [N] \setminus \{\mathbf{J}_i\}$ :
        parse $\mathsf{r}_{i,j} = (\mu_{i,j}, \gamma_{i,j}) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$
        $\alpha_{i,j} := \mathsf{H}_\alpha(\gamma_{i,j})$,   $c_{i,j} := \mathsf{H}(\mu_{i,j}) \cdot g_1^{\alpha_{i,j}}$,   $\mathsf{com}_{i,j} := \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,j})$
$\mathsf{com} := (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$,   $c := (c_{1,1}, \ldots, c_{K,N})$
if $\mathbf{J} \neq \mathsf{H}_{cc}(\mathsf{com}, c)$ : return 0
return 1
```

Figure 2: The algorithm Check used in the signature issuing protocol of blind signature scheme BS_R.

3.2 Security Analysis

Completeness of the scheme follows by inspection. We show blindness and one-more unforgeability. Before we give the proof of blindness, we first show a lemma that is needed. Intuitively, it states that algorithm ReRa perfectly rerandomizes the key shares.

Lemma 1. *For any* $\mathsf{pk}_1 \in \mathbb{G}_1$ *and* $\mathsf{pk}_{i,1} \in \mathbb{G}_1, i \in [K]$ *such that* $\prod_{i=1}^{K} \mathsf{pk}_{i,1} = \mathsf{pk}$*, the following distributions* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *are identical:*

$$\mathcal{D}_1 := \left\{ \left(\mathsf{pk}_1, (\mathsf{pk}_{i,1})_{i\in[K]}, (\mathsf{pk}'_{i,1})_{i\in[K]}\right) \;\middle|\; \begin{array}{l} r_1, \dots, r_{K-1} \leftarrow\!\!{\scriptstyle\$}\, \mathbb{Z}_p,\ r_K := -\sum_{i=1}^{K-1} r_i \\ \forall i \in [K] : \mathsf{pk}'_{i,1} := \mathsf{pk}_{i,1} \cdot g_1^{r_i} \end{array} \right\}$$

$$\mathcal{D}_2 := \left\{ \left(\mathsf{pk}_1, (\mathsf{pk}_{i,1})_{i\in[K]}, (\mathsf{pk}'_{i,1})_{i\in[K]}\right) \;\middle|\; \begin{array}{l} \forall i \in [K] : \mathsf{pk}'_{i,1} \leftarrow\!\!{\scriptstyle\$}\, \mathbb{G} \\ \mathsf{pk}'_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}'^{-1}_{i,1} \end{array} \right\}$$

Proof. It is sufficient to look at the distributions in terms of their exponents. To this end, let $\mathsf{pk} = g_1^x$, $\mathsf{pk}_{i,1} = g_1^{x_i}$, and $\mathsf{pk}'_{i,1} = g_1^{x'_i}$ for all $i \in [K]$. Consider the homomorphism $f : \mathbb{Z}_p^K \to \mathbb{Z}_p$ with $f(\mathbf{y}) := (1, \dots, 1) \cdot \mathbf{y}$. Note that in distribution $\mathcal{D}_2$, the vector $\mathbf{x}' = (x'_1, \dots, x'_K)^t$ is distributed uniformly over all vectors in $f^{-1}(x)$. Further, note that in distribution $\mathcal{D}_1$, the vector $\mathbf{x}'$ is distributed as $\mathbf{x} + \mathbf{r}$, where $\mathbf{r}$ is uniform in the kernel of f. This gives us the same distribution for $\mathbf{x}'$. Therefore, the distributions are the same. □

Theorem 1. *Let* $\mathsf{H_r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$ *and* $\mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p$ *be random oracles. Then* BS_R *satisfies malicious signer blindness.*

Concretely, for any algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H_r}}, Q_{\mathsf{H}_\mu}, Q_{\mathsf{H}_\alpha}$ *queries to* $\mathsf{H_r}, \mathsf{H}_\mu, \mathsf{H}_\alpha$ *respectively, we have*

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H_r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}.$$

Proof. We set $\mathsf{BS} := \mathsf{BS}_R$ and let $\mathcal{A}$ be an adversary against the blindness of BS. Our proof is presented as a sequence of games $\mathbf{G}_{i,b}$ for $i \in [8]$ and $b \in \{0,1\}$. We set $\mathbf{G}_{0,b} := \mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}(\lambda)$. Then, our goal is bound the distinguishing advantage

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) = \left|\Pr\left[\mathbf{G}_{0,0} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{0,1} \Rightarrow 1\right]\right|.$$

To do that, we will change our game to end up at a game $\mathbf{G}_{8,b}$ for which we have

$$\Pr\left[\mathbf{G}_{8,0} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{8,1} \Rightarrow 1\right].$$

Game $\mathbf{G}_{0,b}$: Game $\mathbf{G}_{0,b}$ is defined as $\mathbf{G}_{0,b} := \mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}(\lambda)$. We recall this game to fix some notation. First, $\mathcal{A}$ outputs a public key pk and two messages $\mathsf{m}_0, \mathsf{m}_1$. Second, $\mathcal{A}$ is run with access to two interactive oracles O_0 and O_1. These simulate $\mathsf{U}(\mathsf{pk}, \mathsf{m}_b)$ and $\mathsf{U}(\mathsf{pk}, \mathsf{m}_{1-b})$, respectively. To distinguish variables used in the two oracles, we use superscripts L and R. That is, variables with superscript L (resp. R) are part of the interaction between $\mathcal{A}$ and O_0 (resp. O_1). For example, $\mathbf{J}^L := \mathsf{H}_{cc}(\mathsf{com}^L, c^L)$ denotes the cut-and-choose vector that O_0 computes, and open^R denotes the first message that O_1 sends to $\mathcal{A}$. For descriptions with variables without a superscript, the reader should understand them as applying to both oracles.

Game $\mathbf{G}_{1,b}$: This game is as $\mathbf{G}_{0,b}$, but we let the game abort on a certain event. Namely, the game aborts if $\mathcal{A}$ ever makes a query of the form $\mathsf{H}_\mu(\cdot, \varphi_{i,j})$ for some $i \in [K]$ and $j \in [N] \setminus \{\mathbf{J}_i\}$. Note that for these values (i,j), $\mathcal{A}$ obtains no information about $\varphi_{i,j}$ throughout the entire game. Thus, the probability that a query is of this form is at most $1/2^\lambda$. A union bound over all such (i,j), the two oracles, and the random oracle queries leads to

$$\left|\Pr\left[\mathbf{G}_{0,b} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{1,b} \Rightarrow 1\right]\right| \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-1}}.$$

Game $\mathbf{G}_{2,b}$: This game is as $\mathbf{G}_{1,b}$, but with another abort event. Concretely, the game aborts if $\mathcal{A}$ ever makes a query $\mathsf{H_r}(\mathsf{r}_{i,\mathbf{J}_i})$, or a query $\mathsf{H}_\alpha(\gamma_{i,\mathbf{J}_i})$ for some $i \in [K]$. Note that $\mathsf{r}_{i,\mathbf{J}_i}$ has the form

$\mathsf{r}_{i,\mathbf{J}_i} = (\mu_{i,\mathbf{J}_i}, \gamma_{i,\mathbf{J}_i})$, where $\gamma_{i,\mathbf{J}_i}$ is sampled uniformly at random from $\{0,1\}^\lambda$. Further, $\mathcal{A}$ obtains no information about $\gamma_{i,\mathbf{J}_i}$ throughout the entire game. Therefore, taking a union bound over all instances $i \in [K]$, the two user oracles, and the random oracle queries for both random oracles H_r and H_α, we get

$$|\Pr[\mathbf{G}_{1,b} \Rightarrow 1] - \Pr[\mathbf{G}_{2,b} \Rightarrow 1]| \le \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-1}}.$$

Game $\mathbf{G}_{3,b}$: In this game, we change how the final signatures are computed. Specifically, suppose that the user oracle does not abort due to the condition $e(\bar{s}, g_2) \neq \prod_{i=1}^K e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right)$ and does not abort due to condition $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$ for any $i \in [K]$. Then, in previous games, the user oracle first computed $\bar{\sigma}$, and then executed $((\mathsf{pk}'_i)_i, \bar{\sigma}') \leftarrow \mathsf{ReRa}((\mathsf{pk}_i, \mathsf{H}(\mu_{i,\mathbf{J}_i}))_i, \bar{\sigma})$. The value $\bar{\sigma}'$ is part of the final signature. In game $\mathbf{G}_{3,b}$, we instead let the user oracle run a brute-force search to compute the unique $\bar{\sigma}''$ such that $e(\bar{\sigma}'', g_2) = \prod_{i=1}^K e\left(\mathsf{H}(\mu_{i,\mathbf{J}_i}), \mathsf{pk}'_{i,2}\right)$. Then, we include $\bar{\sigma}''$ in the final signature instead of $\bar{\sigma}'$. We claim that this does not change the view of $\mathcal{A}$. To see this, first note that we did not change any verification or abort condition of the user oracles. Therefore, we can first consider the case where one of the user oracles locally outputs $\perp$. In this case, $\mathcal{A}$ gets $\perp, \perp$ as its final input in both $\mathbf{G}_{2,b}$ and $\mathbf{G}_{3,b}$. It remains to analyze the case where both user oracles do not abort. We claim that $\bar{\sigma}'$ and $\bar{\sigma}''$ are the same. To see this, assume $e(\bar{s}, g_2) = \prod_{i=1}^K e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right)$, and multiply both sides by $\prod_{i=1}^K e\left(\mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right)$. We obtain

$$\begin{aligned} e(\bar{s}, g_2) \cdot \prod_{i=1}^K e\left(\mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right) &= \prod_{i=1}^K e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right) \cdot \prod_{i=1}^K e\left(\mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right) \\ \Longrightarrow e\left(\bar{s} \cdot \prod_{i=1}^K \mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right) &= \prod_{i=1}^K e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right) \cdot e\left(g_1^{-\alpha_{i,\mathbf{J}_i}}, \mathsf{pk}_{i,2}\right) \\ \Longrightarrow e(\bar{\sigma}, g_2) &= \prod_{i=1}^K e\left(\mathsf{H}(\mu_{i,\mathbf{J}_i}), \mathsf{pk}_{i,2}\right), \end{aligned}$$

where we used $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$ on the right-hand side. Using the definition of algorithm ReRa, it is easy to see that this implies

$$e(\bar{\sigma}', g_2) = \prod_{i=1}^K e\left(\mathsf{H}(\mu_{i,\mathbf{J}_i}), \mathsf{pk}'_{i,2}\right).$$

By definition, $\bar{\sigma}''$ satisfies the same equation. As their is a unique solution to this equation for given $\mathsf{pk}'_{i,2}$ and $\mu_{i,\mathbf{J}_i}, i \in [K]$, we see that $\bar{\sigma}' = \bar{\sigma}''$. We have

$$\Pr[\mathbf{G}_{2,b} \Rightarrow 1] = \Pr[\mathbf{G}_{3,b} \Rightarrow 1].$$

Game $\mathbf{G}_{4,b}$: We make another change to the computation of the final signatures. Again, suppose that the user oracle does not abort. In this game $\mathbf{G}_{4,b}$, we no longer run algorithm ReRa in this case. Instead, we compute the $\mathsf{pk}'_i = (\mathsf{pk}'_{i,1}, \mathsf{pk}'_{i,2})$ as a fresh sharing via

$$\begin{gathered} \mathsf{sk}'_i \leftarrow\!\!\$\, \mathbb{Z}_p, \quad \mathsf{pk}'_{i,1} := g_1^{\mathsf{sk}_i}, \quad \mathsf{pk}'_{i,2} := g_2^{\mathsf{sk}_i} \text{ for } i \in [K-1], \\ \mathsf{pk}'_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}'^{-1}_{i,1}, \quad \mathsf{pk}'_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}'^{-1}_{i,2}. \end{gathered}$$

Note that the other output $\bar{\sigma}'$ of algorithm ReRa is no longer needed due to the previous change. To analyze this change, we first argue that the $\mathsf{pk}'_{i,2}$ are uniquely determined by the $\mathsf{pk}'_{i,1}$. Namely, if the user oracle does not abort, we know that $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$, and $e(\mathsf{pk}_1, g_2) = e(g_1, \mathsf{pk}_2)$. It is easy to see that property is preserved by algorithm ReRa, i.e. $e\left(\mathsf{pk}'_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}'_{i,2}\right)$ for all $i \in [K]$. One can verify that our new definiton of the $\mathsf{pk}'_{i,1}, \mathsf{pk}'_{i,2}$ also satisfies this. It remains to analyze the distribution of the $\mathsf{pk}'_{i,1}$. By Lemma 1 the distribution of the $\mathsf{pk}'_{i,1}$ stays the same. This implies that

$$\Pr[\mathbf{G}_{3,b} \Rightarrow 1] = \Pr[\mathbf{G}_{4,b} \Rightarrow 1].$$

Game $\mathbf{G}_{5,b}$: In game $\mathbf{G}_{5,b}$, we first sample random vectors $\hat{\mathbf{J}}^L \leftarrow\!\!\$\, [N]^K$ and $\hat{\mathbf{J}}^R \leftarrow\!\!\$\, [N]^K$. Then, we let the game abort, if later we do not have $\hat{\mathbf{J}}^L = \mathbf{J}^L$ and $\hat{\mathbf{J}}^R = \mathbf{J}^R$. As the view of $\mathcal{A}$ is independent of $\hat{\mathbf{J}}^L, \hat{\mathbf{J}}^R$ until a potential abort, we have

$$\Pr[\mathbf{G}_{5,b} \Rightarrow 1] = \frac{1}{N^{2K}} \cdot \Pr[\mathbf{G}_{4,b} \Rightarrow 1].$$

Game $\mathbf{G}_{6,b}$: In game $\mathbf{G}_{6,b}$, we change how the values $\mu_{i,j}$ for $i \in [K]$ and $j \in [N] \setminus \{\hat{\mathbf{J}}_i\}$ are computed. Recall that before, they were computed as $\mu_{i,j} = \mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j})$. In $\mathbf{G}_{6,b}$, we sample $\mu_{i,j} \leftarrow\!\!\$\, \{0,1\}^\lambda$ for $i \in [K]$ and $j \in [N] \setminus \{\hat{\mathbf{J}}_i\}$ instead. We highlight that the game still samples the values $\varphi_{i,j}$ to determine when it has to abort according to $\mathbf{G}_{1,b}$. Due to the changes introduced in $\mathbf{G}_{1,b}$ and $\mathbf{G}_{5,b}$, we can assume that $\hat{\mathbf{J}} = \mathbf{J}$ and $\mathcal{A}$ never queries $\mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j})$, and therefore this change does not influence the view of $\mathcal{A}$. We have

$$\Pr[\mathbf{G}_{5,b} \Rightarrow 1] = \Pr[\mathbf{G}_{6,b} \Rightarrow 1].$$

Game $\mathbf{G}_{7,b}$: In game $\mathbf{G}_{7,b}$, we change how the values $\alpha_{i,\hat{\mathbf{J}}_i}$ and $\mathsf{com}_{i,\hat{\mathbf{J}}_i}$ are computed for all $i \in [K]$. Concretely, in this game, $\alpha_{i,\hat{\mathbf{J}}_i}$ is sampled uniformly at random as $\alpha_{i,\hat{\mathbf{J}}_i} \leftarrow\!\!\$\, \mathbb{Z}_p$. Further, $\mathsf{com}_{i,\hat{\mathbf{J}}_i} \leftarrow\!\!\$\, \{0,1\}^\lambda$ is sampled uniformly at random. Assuming that the game does not abort, we argue that the view of $\mathcal{A}$ does not change. This follows directly from the changes in $\mathbf{G}_{5,b}$ and $\mathbf{G}_{2,b}$. Namely, we can assume that $\hat{\mathbf{J}} = \mathbf{J}$ and that $\mathcal{A}$ never makes a query $\mathsf{H}_\mathsf{r}(\mathsf{r}_{i,\hat{\mathbf{J}}_i})$. We have

$$\Pr[\mathbf{G}_{6,b} \Rightarrow 1] = \Pr[\mathbf{G}_{7,b} \Rightarrow 1].$$

Game $\mathbf{G}_{8,b}$: In game $\mathbf{G}_{8,b}$, we change how the values $c_{i,\hat{\mathbf{J}}_i}$ for $i \in [K]$ are computed. First, recall that in the previous games, these are computed as $c_{i,\hat{\mathbf{J}}_i} = \mathsf{H}(\mu_{i,\hat{\mathbf{J}}_i}) \cdot g_1^{\alpha_{i,\hat{\mathbf{J}}_i}}$. Now, we sample it at random using $c_{i,\hat{\mathbf{J}}_i} \leftarrow\!\!\$\, \mathbb{G}_1$. We argue indistinguishability as follows. Due to the change introduced in $\mathbf{G}_{5,b}$, we can assume that $\hat{\mathbf{J}} = \mathbf{J}$. Then, we know that in this case $\alpha_{i,\hat{\mathbf{J}}_i}$ is only used to define $c_{i,\hat{\mathbf{J}}_i}$ and nowhere else. In particular, it is not used to derive the final signatures from the interaction, due to the change introduced in $\mathbf{G}_{3,b}$, and it is not used to define $\mathsf{com}_{i,\hat{\mathbf{J}}_i}$ due to the change in $\mathbf{G}_{7,b}$. As $\alpha_{i,\hat{\mathbf{J}}_i}$ is sampled uniformly at random due to the change in $\mathbf{G}_{7,b}$, we know that $c_{i,\hat{\mathbf{J}}_i}$ is distributed uniformly at random in $\mathbf{G}_{7,b}$. This shows that

$$\Pr[\mathbf{G}_{7,b} \Rightarrow 1] = \Pr[\mathbf{G}_{8,b} \Rightarrow 1].$$

Finally, it can be observed that the view of $\mathcal{A}$ does not depend on the bit b anymore. This is because the messages $\mathsf{m}_0, \mathsf{m}_1$ are not used in the user oracles. Instead, the user oracles use random $\mu_{i,j}$, independent of the messages, for all opened sessions $j \neq \mathbf{J}_i$, and the final signatures σ_0, σ_1 that $\mathcal{A}$ gets are computed using brute-force independent of the interactions, assuming that both interactions accept. This shows that

$$\Pr[\mathbf{G}_{8,0} \Rightarrow 1] = \Pr[\mathbf{G}_{8,1} \Rightarrow 1].$$

To conclude, we upper bound $\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) = |\Pr[\mathbf{G}_{0,0} \Rightarrow 1] - \Pr[\mathbf{G}_{0,1} \Rightarrow 1]|$ by

$$\begin{aligned}
&|\Pr[\mathbf{G}_{4,0} \Rightarrow 1] - \Pr[\mathbf{G}_{4,1} \Rightarrow 1]| + 2\left(\frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-1}}\right)\\
= \;& N^{2K}|\Pr[\mathbf{G}_{5,0} \Rightarrow 1] - \Pr[\mathbf{G}_{5,1} \Rightarrow 1]| + \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}\\
= \;& N^{2K}|\Pr[\mathbf{G}_{8,0} \Rightarrow 1] - \Pr[\mathbf{G}_{8,1} \Rightarrow 1]| + \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}\\
= \;& \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}.
\end{aligned}$$

□

Theorem 2. *Let $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$, and $\mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} \colon \{0,1\}^* \to \mathbb{G}$ be random oracles. If* CDH *assumption holds relative to* PGGen*, then* BS_R *is one-more unforgeable.*

Concretely, for any polynomial ℓ and any PPT algorithm $\mathcal{A}$ that makes at most $Q_{\mathsf{H}_{cc}}, Q_{\mathsf{H}_{\mathsf{r}}}, Q_{\mathsf{H}_{\mu}}, Q_{\mathsf{H}}$ queries to $\mathsf{H}_{cc}, \mathsf{H}_{\mathsf{r}}, \mathsf{H}_{\mu}, \mathsf{H}$ respectively, there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and

$$\begin{aligned}\mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}_R}(\lambda) \leq \frac{Q^2_{\mathsf{H}_\mu} + Q^2_{\mathsf{H}_\mathsf{r}} + Q_{\mathsf{H}_\mathsf{r}}Q_{\mathsf{H}_{cc}} + Q_{\mathsf{H}}Q_{\mathsf{H}_\mu}}{2^\lambda} + \frac{\ell}{N^K} \\ + 4\ell \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).\end{aligned}$$

Proof. We set $\mathsf{BS} := \mathsf{BS}_R$ and let $\mathcal{A}$ be an adversary against the one-more unforgeability of BS. We show the statement by presenting a sequence of games. Before we go into detail, we explain the overall strategy of the proof. In our final step, we give a reduction that breaks the CDH assumption. This reduction works similar to the reduction for the BLS signature scheme [BLS01]. Namely, it embeds one part of the CDH instance in the public key, and one part in some of the random oracle queries for oracle H. In the first part of our proof, we prepare simulation of the signer oracle without using the secret key. Here, the strategy is to extract the users randomness using the cut-and-choose technique. With overwhelming probability, in a fixed interaction, we can extract the randomness for one of the K instances, say instance i^*. Then, we compute the public key shares pk_i in a way that allows us to know all corresponding secret keys except sk_{i^*}. For instance i^*, we can simulate the signing oracle by programming random oracle H. In the second part of our proof, we prepare the extraction of the CDH solution from the forgery that $\mathcal{A}$ returns. Here, it is essential that the scheme uses random oracle H_μ to compute commitments $\mu_{i,j}$. This allows us to embed the part of the CDH input in H in a consistent way. We will now proceed more formally.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is the real one-more unforgeability game, i.e. $\mathbf{G}_0 := \ell\text{-}\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}}$. Let us recall this game. First, the game samples $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Then, $\mathcal{A}$ is executed on input pk, and gets concurrent access to signer oracle O, simulating $\mathsf{S}(\mathsf{sk})$. Additionally, $\mathcal{A}$ gets access to random oracles $\mathsf{H}, \mathsf{H}_\mu, \mathsf{H}_\mathsf{r}, \mathsf{H}_{cc}$. These are simulated by the game in the standard lazy way. Finally, $\mathcal{A}$ outputs pairs $(\mathsf{m}_1,\sigma_1),\dots,(\mathsf{m}_k,\sigma_k)$. Denote the number of completed interactions (i.e. interactions in which O sent $\bar{s}$ to $\mathcal{A}$) by ℓ. If all m_i are distinct, all σ_i are valid signatures for m_i with respect to pk, and $k > \ell$, the game outputs 1. By definition, we have

$$\mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}}(\lambda) = \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Game $\mathbf{G}_1$: Game $\mathbf{G}_1$ is as $\mathbf{G}_0$, but it aborts if a collision for one of the random oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu$ occurs. More precisely, let $* \in \{\mathsf{r},\mu\}$ and consider a query $\mathsf{H}_*(x)$ for which the hash value is not yet defined. The game samples $\mathsf{H}_*(x)$ as in game $\mathbf{G}_0$. Then, the game aborts if there is another $x' \neq x$ such that $\mathsf{H}_*(x')$ is already defined and $\mathsf{H}_*(x) = \mathsf{H}_*(x')$. As the outputs of H_* are sampled uniformly from $\{0,1\}^\lambda$, we can use a union bound over all pairs of queries and get

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{Q^2_{\mathsf{H}_\mu}}{2^\lambda} + \frac{Q^2_{\mathsf{H}_\mathsf{r}}}{2^\lambda}.$$

Game $\mathbf{G}_2$: Game $\mathbf{G}_2$ is as game $\mathbf{G}_1$, but we introduce a bad event and let the game abort if this bad event occurs. Concretely, consider any fixed query to oracle H_{cc} of the form $\mathsf{H}_{cc}(\mathsf{com},c) = \mathbf{J}$ for $\mathsf{com} = (\mathsf{com}_{1,1},\dots,\mathsf{com}_{K,N})$ and $c = (c_{1,1},\dots,c_{K,N})$. For such queries and all $(i,j) \in [K]\times[N]$, the game now tries to extract values $\bar{\mathsf{r}}_{i,j}$ such that $\mathsf{com}_{i,j} = \mathsf{H}_\mathsf{r}(\bar{\mathsf{r}}_{i,j})$. To do that, it searches through the random oracle queries for random oracle H_r. For those (i,j) for which such a value can not be extracted, we write $\bar{\mathsf{r}}_{i,j} = \bot$. Due to the change introduced in $\mathbf{G}_1$, there can be at most one extracted value for each (i,j). The game now aborts, if in such a query, there is some $(i,j) \in [K]\times[N]$ such that $\bar{\mathsf{r}}_{i,j} = \bot$, but later oracle H_r is queried and returns $\mathsf{com}_{i,j}$. Clearly, for a fixed pair of queries to H_{cc} and H_r, respectively, this bad event can only with probability $1/2^\lambda$. By a union bound we get

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{Q_{\mathsf{H}_\mathsf{r}}Q_{\mathsf{H}_{cc}}}{2^\lambda}.$$

Before we continue, we summarize what we established so far and introduce some terminology. For that, we fix an interaction between $\mathcal{A}$ and the signer oracle O. Consider the first message

$$\mathsf{open} = \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j\neq\mathbf{J}_i}, c_{i,\mathbf{J}_i}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i\in[K]}\right)$$

that is sent by $\mathcal{A}$. Recall that after receiving this message, algorithm Check uses open to compute values $\mathsf{com} = (\mathsf{com}_{1,1}, \dots, \mathsf{com}_{K,N})$ and $c = (c_{1,1}, \dots, c_{K,N})$. Then, it also checks if $\mathbf{J} = \mathsf{H}_{cc}(\mathsf{com}, c)$. Also, consider the values $\bar{\mathsf{r}}_{i,j}$ related to the query $\mathsf{H}_{cc}(\mathsf{com}, c)$, as defined in $\mathbf{G}_2$. Assuming Check outputs 1 (i.e. $\mathbf{J} = \mathsf{H}_{cc}(\mathsf{com}, c)$), we make two observations for any instance $i \in [K]$.

1. If for some $j \in [N]$ we have $\bar{\mathsf{r}}_{i,j} = \perp$, then $j = \mathbf{J}_i$. This is due to the bad event introduced in $\mathbf{G}_2$.
2. If for some $j \in [N]$ we have $\bar{\mathsf{r}}_{i,j} = (\mu, \gamma) \neq \perp$ but $c_{i,j} \neq \mathsf{H}(\mu) \cdot g_1^{\alpha}$ for $\alpha := \mathsf{H}_{\alpha}(\gamma)$, then $\mathbf{J}_i = j$. This is because we ruled out collisions for H_{r} in $\mathbf{G}_1$. Namely, as there are no collisions, we know that $\bar{\mathsf{r}}_{i,j} = \mathsf{r}_{i,j}$ for all $j \neq \mathbf{J}_i$. Therefore, $c_{i,j} = \mathsf{H}(\mu) \cdot g_1^{\alpha}$ by definition of Check.

If one of these two events occur for some i, we say that there is a *successful cheat in instance i*. Note that the game can efficiently check if there is a successful cheat in an instance once it received open. Also note that the values $\bar{\mathsf{r}}_{i,j}$ are fixed in the moment $\mathcal{A}$ queries $\mathsf{H}_{cc}(\mathsf{com}, c)$ for the first time. In particular, they are fixed before $\mathcal{A}$ obtains any information about the uniformly random $\mathbf{J} = \mathsf{H}_{cc}(\mathsf{com}, c)$. Therefore, using the two observations above, the probability of a successful cheat in instance i is at most $1/N$. Further, as the components of $\mathbf{J}$ are sampled independently, the probability that there is a successful cheat in all K instances (in this fixed interaction) is at most $1/N^K$.

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, we introduce another abort. Namely, the game aborts, if in some interaction between $\mathcal{A}$ and the signer oracle O, there is a successful cheat in every instance $i \in [K]$, and that interaction is completed. By the discussion above, we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \frac{\ell}{N^K}.$$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, we change the way the signer oracle computes the shares sk_i. Recall that before, these were computed as

$$\mathsf{sk}_i \leftarrow\!\!\$\, \mathbb{Z}_p \text{ for } i \in [K-1], \quad \mathsf{sk}_K := \mathsf{sk} - \sum_{i=1}^{K-1} \mathsf{sk}_i.$$

Then, the corresponding public key shares were computed as $\mathsf{pk}_i = (g_1^{\mathsf{sk}_i}, g_2^{\mathsf{sk}_i})$ for all $i \in [K]$. In game $\mathbf{G}_4$, the game instead defines the sk_i after it received the first message open from $\mathcal{A}$ in the following way. If Check outputs 0 or there is a successful cheat in every instance, the game behaves as before (i.e. it aborts the interaction, or the entire execution). Otherwise, let $i^* \in [K]$ be the first instance in which there is no successful cheat. Then, the game computes

$$\mathsf{sk}_i \leftarrow\!\!\$\, \mathbb{Z}_p \text{ for } i \in [K] \setminus \{i^*\}, \quad \mathsf{sk}_{i^*} := \mathsf{sk} - \sum_{i \in [K] \setminus \{i^*\}} \mathsf{sk}_i.$$

The game defines pk_i for all $i \in [K]$ as before. It is clear that this change is only conceptual, as a uniformly random additive sharing of sk is computed in both $\mathbf{G}_3$ and $\mathbf{G}_4$. Therefore, we have

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_4 \Rightarrow 1].$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we introduce an abort related to the random oracles H and H_{μ}. Namely, the game aborts if the following occurs. The adversary $\mathcal{A}$ first queries $\mathsf{H}(\mu)$ for some $\mu \in \{0,1\}^*$, and after that a hash value $\mathsf{H}_{\mu}(x)$ is defined for some $x \in \{0,1\}^*$, and we have $\mathsf{H}_{\mu}(x) = \mu$. Clearly, once μ is fixed, the probability that a previously undefined hash value $\mathsf{H}_{\mu}(x)$ is equal to μ is at most $1/2^{\lambda}$. Therefore, we can use a union bound over the random oracle queries and get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{Q_{\mathsf{H}} Q_{\mathsf{H}_{\mu}}}{2^{\lambda}}.$$

Game $\mathbf{G}_6$: In this game, we introduce a purely conceptual change. To do that, we introduce maps $b[\cdot]$ and $\hat{b}[\cdot]$. Then, on a query $\mathsf{H}_{\mu}(\mathsf{m}, \varphi)$ for which the hash value is not yet defined, the game samples bit $\hat{b}[\mathsf{m}] \in \{0,1\}$ from a Bernoulli distribution, such that the probability that $\hat{b}[\mathsf{m}] = 1$ is $1/(\ell+1)$. Additionally, on a query $\mathsf{H}(\mu)$ for which the hash value is not yet defined, the game first searches for a previous query $(\mathsf{m}_{\mu}, \varphi)$ to H_{μ} such that $\mathsf{H}_{\mu}(\mathsf{m}_{\mu}, \varphi) = \mu$. Then, it sets $b[\mu] := \hat{b}[\mathsf{m}_{\mu}]$. If no such query can be found, it sets $b[\mu] := 0$. Note that due to the change in $\mathbf{G}_1$, the game can find at most one such query and m_μ is well defined. The view of $\mathcal{A}$ does not change, and we have

$$\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_7$: In this game, we introduce an initially empty set $\mathcal{L}$ and an abort related to it. In each interaction between $\mathcal{A}$ and the signer oracle O, the game simulates the oracle as in $\mathbf{G}_6$. Additionally, if the game has to provide the final message $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$, then we know that Check output 1 and the game did not abort. Therefore, there is at least one instance $i^* \in [K]$ such that $\mathcal{A}$ did not cheat successfully in instance i^*. Fix the first such instance. This means that the game could extract $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\mu, \gamma)$ before (see the discussion after $\mathbf{G}_2$). In game $\mathbf{G}_7$, the game tries to extract m_μ as defined in $\mathbf{G}_6$ from μ using H_μ, and inserts (μ, m_μ) into set $\mathcal{L}$ if it could extract. Also, the game aborts if $b[\mu] = 1$. Otherwise, it computes and sends $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$ as before. We highlight that the size of $\mathcal{L}$ is at most the number of completed interactions ℓ.

Next, consider the final output $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$ of $\mathcal{A}$, write $\sigma_r = ((\mathsf{pk}_{r,i}, \varphi_{r,i})_{i=1}^{K-1}), \varphi_{r,K}, \bar{\sigma}_r)$, and set $\mu_{r,i} := \mathsf{H}_\mu(\mathsf{m}_r, \varphi_{r,i})$ for all $r \in [k], i \in [K]$. If $\mathcal{A}$ is successful, we know that $k > \ell$. Therefore, by the pigeonhole principle, there is at least one $(\tilde{r}, \tilde{i}) \in [k] \times [K]$ such that $(\mu_{\tilde{r},\tilde{i}}, \mathsf{m}_{\tilde{r}}) \notin \mathcal{L}$. Game $\mathbf{G}_7$ finds the first such $\mu_{\tilde{r},\tilde{i}}$, sets $\mu^* := \mu_{\tilde{r},\tilde{i}}$ and aborts if $b[\mu^*] = 0$. Note that we can assume that $b[\mu^*]$ is defined, as verification of $\mathcal{A}$'s output involves computing $\mathsf{H}(\mu^*)$. For the sake of analysis, $\mathbf{G}_7$ also appends further entries of the form (μ, m_μ) to $\mathcal{L}$ such that $|\mathcal{L}| = \ell$ and all entries in $\mathcal{L} \cup \{\mu^*\}$ have distinct components m_μ. It queries $\mathsf{H}(\mu)$ for all $(\mu, \mathsf{m}_\mu) \in \mathcal{L}$. Then, it aborts if for some $(\mu, \mathsf{m}_\mu) \in \mathcal{L}$ it holds that $b[\mu] = 1$.

To analyze the change we introduced, note that $\mathbf{G}_6$ and $\mathbf{G}_7$ only differ if $b[\mu^*] = 0$ or $b[\mu] = 1$ for some $(\mu, \mathsf{m}_\mu) \in \mathcal{L}$. This is because if the game could not extract m_μ in some interaction, then due to the changes in $\mathbf{G}_5$ and $\mathbf{G}_6$, we know that $b[\mu] = 0$. The view of $\mathcal{A}$ is independent of these bits until a potential abort occurs. This implies that

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1] \cdot \Pr[b[\mu^*] = 1 \wedge \forall (\mu, \mathsf{m}_\mu) \in \mathcal{L} : b[\mu] = 0].$$

By definition of the bits $b[\cdot]$, and the change in $\mathbf{G}_5$, we can rewrite the latter term in the product as

$$\begin{aligned}\Pr\left[\hat{b}[\mathsf{m}_{\tilde{r}}] = 1 \wedge \forall (\mu, \mathsf{m}_\mu) \in \mathcal{L} : \hat{b}[\mathsf{m}_\mu] = 0\right] &= \frac{1}{\ell+1}\left(1 - \frac{1}{\ell+1}\right)^{\ell} \\ &= \frac{1}{\ell}\left(1 - \frac{1}{\ell+1}\right)^{\ell+1} \geq \frac{1}{4\ell},\end{aligned}$$

where we used the fact $(1 - 1/x)^x \geq 1/4$ for all $x \geq 2$, and that all bits $\hat{b}[\cdot]$ are independent. Thus, we have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] \geq \frac{1}{4\ell} \cdot \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_8$: In this game, we change how random oracle H is simulated. Namely, in the beginning of the game, the game samples $Y \leftarrow\!\!\$\, \mathbb{G}_1$ and initiates a map $t[\cdot]$. Then, on a query $\mathsf{H}(\mu)$ for which the hash value is not yet defined, the game first determines bit $b[\mu]$ as before. Then, it samples $t[\mu] \leftarrow\!\!\$\, \mathbb{Z}_p$ and sets $\mathsf{H}(\mu) := Y^{b[\mu]} \cdot g_1^{t[\mu]}$. Clearly, all hash values are still uniformly random and independent. Therefore, we have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_8 \Rightarrow 1].$$

Game $\mathbf{G}_9$: In this game, we change how the signing oracle computes public keys $(\mathsf{pk}_i)_i$ and the values $s_i, i \in [K]$ used to compute the final message $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$. Consider an interaction between $\mathcal{A}$ and the signer oracle and recall the definition of the instance i^* as in game $\mathbf{G}_4$. This is the first instance for which there is no successful cheat in this interaction, i.e. $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\mu, \gamma) \neq \bot$ could be extracted and $c_{i^*,\mathbf{J}_{i^*}} = \mathsf{H}(\mu) \cdot g_1^\alpha$ for $\alpha := \mathsf{H}_\alpha(\gamma)$. In $\mathbf{G}_9$, the public keys $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ are computed via

$$\mathsf{pk}_{i,1} = g_1^{\mathsf{sk}_i} \text{ for } i \in [K] \setminus \{i^*\}, \quad \mathsf{pk}_{i^*,1} := \mathsf{pk}_1 \cdot \prod_{i \in [K] \setminus \{i^*\}} \mathsf{pk}_{i,1}^{-1},$$

$$\mathsf{pk}_{i,2} = g_2^{\mathsf{sk}_i} \text{ for } i \in [K] \setminus \{i^*\}, \quad \mathsf{pk}_{i^*,2} := \mathsf{pk}_2 \cdot \prod_{i \in [K] \setminus \{i^*\}} \mathsf{pk}_{i,2}^{-1}.$$

Further, due to the aborts introduced in previous games, we know that the game only has to send $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$ if i^* is defined and $b[\mu] = 0$, where μ is as above. In this case, game $\mathbf{G}_8$ would compute

$$s_{i^*} = c_{i^*,\mathbf{J}_{i^*}}^{\mathsf{sk}_{i^*}} = \mathsf{H}(\mu)^{\mathsf{sk}_{i^*}} \cdot g_1^{\alpha \cdot \mathsf{sk}_{i^*}} = \left(Y^{b[\mu]} \cdot g_1^{t[\mu]}\right)^{\mathsf{sk}_{i^*}} \cdot \mathsf{pk}_{i^*,1}^{\alpha} = \mathsf{pk}_{i^*,1}^{\alpha + t[\mu]}.$$

Game $\mathbf{G}_9$ computes s_{i^*} directly as $\mathsf{pk}_{i^*,1}^{\alpha+t[\mu]}$, and all other $s_i,\ i \neq i^*$ as before using sk_i. Both changes are only conceptual and allow the game to provide the signer oracle without using the secret key sk at all. We have

$$\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1].$$

Finally, we give a reduction $\mathcal{B}$ against the CDH assumption that is successful if $\mathbf{G}_9$ outputs 1. We argue that

$$\Pr[\mathbf{G}_9 \Rightarrow 1] \leq \mathsf{Adv}_{\mathcal{B},\mathsf{PGGen}}^{\mathsf{CDH}}(\lambda).$$

The reduction $\mathcal{B}$ is as follows.

- Reduction $\mathcal{B}$ gets as input $g_1, g_2, e, p,\ X_1, Y \in \mathbb{G}_1$, and $X_2 \in \mathbb{G}_2$. It sets $\mathsf{pk}_1 := X_1, \mathsf{pk}_2 := X_2$ and uses Y as explained in $\mathbf{G}_8$.
- Reduction $\mathcal{B}$ simulates $\mathbf{G}_9$ for $\mathcal{A}$. Note that it can do that efficiently, as sk is not needed.
- When $\mathcal{A}$ terminates with its final output $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$, the reduction $\mathcal{B}$ writes $\sigma_r = ((\mathsf{pk}_{r,i}, \varphi_{r,i})_{i=1}^{K-1}), \varphi_{r,K}, \bar{\sigma}_r)$, $\mathsf{pk}_{r,i} = (\mathsf{pk}_{r,i,1}, \mathsf{pk}_{r,i,2})$, sets $\mu_{r,i} := \mathsf{H}_\mu(\mathsf{m}_r, \varphi_{r,i})$ for all $r \in [k], i \in [K]$ and $\mathsf{pk}_{r,K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{r,i,1}^{-1}$ and $\mathsf{pk}_{r,K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{r,i,2}^{-1}$ for all $r \in [k]$. It performs all checks as in $\mathbf{G}_9$. If $\mathbf{G}_9$ outputs 1, we know that $\mathcal{B}$ defined $\mu^* := \mu_{\tilde{r},\tilde{i}}$ as $\mathbf{G}_9$ does. Then, $\mathcal{B}$ outputs

$$Z := \bar{\sigma}_{\tilde{r}} \cdot \prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{-t[\mu_{\tilde{r},i}]}.$$

It is clear that $\mathcal{B}$ perfectly simulates $\mathbf{G}_9$ and the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. Thus, it remains to argue that if $\mathbf{G}_9$ outputs 1, the Z is a valid CDH solution. To this end, assume that $\mathbf{G}_9$ outputs 1. It is sufficient to show that $e(Y, X_2) = e(Z, g_2)$.

First, note that due to the abort that we introduced in $\mathbf{G}_5$, we know that for all $i \in [K]$, the query $\mathsf{H}_\mu(\mathsf{m}_{\tilde{r}}, \varphi_{\tilde{r},i})$ was made before bit $b[\mu_{\tilde{r},i}]$ was defined. Therefore, due to the change in $\mathbf{G}_6$, we obtain for all $i \in [K]$

$$b[\mu_{\tilde{r},i}] = \hat{b}[\mathsf{m}_{\tilde{r}}] = b[\mu_{\tilde{r},\tilde{r}}] = b[\mu^*] = 1.$$

Second, we know that we have $\prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,2} = X_2$, and by definition of the verification algorithm we have

$$\begin{aligned} e(\bar{\sigma}_{\tilde{r}}, g_2) &= \prod_{i=1}^{K} e\left(\mathsf{H}(\mu_{r,i}), \mathsf{pk}_{\tilde{r},i,2}\right) = \prod_{i=1}^{K} e\left(Y \cdot g^{t[\mu_{\tilde{r},i}]}, \mathsf{pk}_{\tilde{r},i,2}\right) \\ &= \prod_{i=1}^{K} e\left(Y, \mathsf{pk}_{\tilde{r},i,2}\right) \cdot e\left(\mathsf{pk}_{\tilde{r},i,1}^{t[\mu_{\tilde{r},i}]}, g_2\right) = e(Y, X_2) \cdot e\left(\prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{t[\mu_{\tilde{r},i}]}, g_2\right). \end{aligned}$$

In the third equation we used $e\left(\mathsf{pk}_{\tilde{r},i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{\tilde{r},i,2}\right)$ for all $i \in [K]$. This implies that

$$e(Z, g_2) = e\left(\bar{\sigma}_{\tilde{r}} \cdot \prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{-t[\mu_{\tilde{r},i}]}, g_2\right) = e(Y, X_2).$$

□

4 Extension: Partial Blindness and Batching

In this section, we present a batching technique for our blind signature scheme, which leads to a significant efficiency improvement in terms of communication. At the same time, we show how to make our scheme partially blind. We first give an informal overview. In the second part of the section, we present the formal model for batching (partially) blind signatures. Then, we present our scheme and its analysis.

4.1 Overview

We give an overview of the extensions we present in this section. These cover partial blindness, and batching to further improve the communication complexity.

Partially Blind Signatures. Recall that a partially blind signature scheme allows to sign messages with respect to some public information string info, that the signer knows. This string acts as a form of domain separator. Namely, one-more unforgeability now guarantees that the user can output at most ℓ valid message signature pairs with respect to any public information string info, for which it interacted at most ℓ times with the signer oracle. It turns out that we can extend our blind signature scheme into a partially blind signature scheme, by changing the definition of the values $c_{i,j}$ from $c_{i,j} = \mathsf{H}(\mu_{i,j}) \cdot g_1^{\alpha_{i,j}}$ to $c_{i,j} = \mathsf{H}(\mathsf{info}, \mu_{i,j}) \cdot g_1^{\alpha_{i,j}}$. Intuitively, the cut-and-choose technique now ensures that the user uses the correct info to compute the $c_{i,j}$'s.

Batching. We show how we can batch multiple signing interactions. Namely, we observe that if we sign multiple messages in one interaction, the (amortized) communication complexity decreases. Batching has been subject of study for other primitives, e.g. in oblivious transfer [IKNP03, BBDP22]. Let us briefly sketch how we can apply batching to our blind signature scheme. For that, consider one signing interaction in which a batch $\mathsf{m}_1, \ldots, \mathsf{m}_L$ of L messages should be signed. Recall that in our scheme, cut-and-choose ensured that there is an instance $i^* \in [K]$, such that the user does not cheat successfully in instance i^*. Then, the purpose of sending a fresh public key sharing $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ was to dynamically embed the unknown share of the secret key in instance i^*. For this strategy, it is not relevant that we cover one message per instance. Therefore we can use the same public key sharing $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$, and the same cut-and-choose index for every instance, leading to our batched scheme.

4.2 Model for Batched (Partially) Blind Signatures

In this section, we sketch the definition of batched (partially) blind signatures and their security. For formal definitions, we refer to Supplementary Material Section B. The reader should observe that batched partially blind signatures imply partially blind signatures by fixing the batch size $L = 1$. Further, the partial blindness can be lifted to standard blindness by fixing a default public information string. We start with the syntax of batched partially blind signatures. Recall that in partially blind signatures, the signer gets the public information string info, while the user gets info and the message m. Here, we generalize the syntax of partially blind signatures to the setting, where both user and signer get the batch size L as input, and multiple pairs $(\mathsf{info}_l, \mathsf{m}_l)$ are signed. This models that the batch size is not fixed, but instead it can be chosen dynamically. More precisely, while the syntax of key generation and verification is as for partially blind signatures, an interaction between S and U can now be described as

$$(\bot, (\sigma_1, \ldots, \sigma_L)) \leftarrow \langle \mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]}), \mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l\in[L]} \rangle.$$

Completeness requires that for all $l \in [L]$, it holds that $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}_l, \mathsf{m}_l, \sigma_l) = 1$.

In terms of security, we require the same security guarantees, as if we just run a normal (partially) blind signature scheme L times in parallel. We let the adversary determine the batch size in each interaction separately. This leads to a natural definition of batch one-more unforgeability. As for unforgeability, blindness should give the same guarantees as if we just run a normal (partially) blind signature scheme L times in parallel. Especially, it should not be possible to tell if two signatures result from the same interaction or not. In our security game, we let the malicious signer choose two batches of (potentially different) sizes L_0 and L_1. The signer also points to one element for each batch. Then, the game either swaps these two elements, or not, and the signer has to distinguish these two cases. Via a hybrid argument, this implies that the signer does not know which message is signed in which interaction.

4.3 Construction

As for BS_R, we let $\mathsf{PGGen}(1^\lambda)$ be a bilinear group generation algorithm that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$. Again, we assume that these system parameters are known to all algorithms and note that their correctness can be verified efficiently. Our scheme $\mathsf{BPBS}_R = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ is parameterized by integers $K = K(\lambda), N(\lambda) \in \mathbb{N}$, where we need that N^{-K} is negligible in λ. We assume that the space $\mathcal{I}$ contains bitstrings of bounded length [4]. The scheme makes use of random oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$.

We verbally describe the signature issuing protocol (S, U) and verification of scheme BPBS_R. Key generation (algorithm Gen) is exactly as in BS_R.

Signature Issuing. The interactive signature issuing protocol between algorithms $\mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]})$ and $\mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l\in[L]})$ is given as follows.

1. User U does the following.

   (a) *Preparation.* First, for each instance $i \in [K]$ and session $j \in [N]$, U commits to all L messages via
   $$\varphi_{i,j,l} \leftarrow_\$ \{0,1\}^\lambda, \quad \mu_{i,j,l} := \mathsf{H}_\mu(\mathsf{m}_l, \varphi_{i,j,l}) \text{ for all } (i,j,l) \in [K] \times [N] \times [L].$$

   (b) *Commitments.* Next, for each instance $i \in [K]$ and session $j \in [N]$, U samples a seed $\gamma_{i,j} \leftarrow_\$ \{0,1\}^\lambda$. It then defines
   $$\mathsf{r}_{i,j} := (\gamma_{i,j}, \mu_{i,j,1}, \ldots, \mu_{i,j,L}), \quad \mathsf{com}_{i,j} := \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,j}) \text{ for all } (i,j) \in [K] \times [N].$$
   Then, U sets $\mathsf{com} := (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$.

   (c) *Challenges.* Now, U derives randomness $\alpha_{i,j,l}$ and computes challenges $c_{i,j,l}$ via $\alpha_{i,j,l} := \mathsf{H}_\alpha(\gamma_{i,j}, l)$ and
   $$c_{i,j,l} := \mathsf{H}(\mathsf{info}_l, \mu_{i,j,l}) \cdot g_1^{\alpha_{i,j,l}} \text{ for all } (i,j,l) \in [K] \times [N] \times [L].$$
   Then, U sets $c := (c_{1,1,1}, \ldots, c_{K,N,L})$.

   (d) *Cut-and-Choose.* Next, U derives a cut-and-choose vector $\mathbf{J} \in [N]^K$ as $\mathbf{J} := \mathsf{H}_{cc}(\mathsf{com}, c)$. It then defines an opening
   $$\mathsf{open} := \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j \neq \mathbf{J}_i}, (c_{i,\mathbf{J}_i,l})_{l\in[L]}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i\in[K]}\right).$$
   Finally, U sends open to S.

2. Signer S does the following.

   (a) *Key Sharing.* First, S samples $\mathsf{sk}_i \leftarrow_\$ \mathbb{Z}_p$ for $i \in [K-1]$. It computes $\mathsf{sk}_K := \mathsf{sk} - \sum_{i=1}^{K-1} \mathsf{sk}_i$ and $\mathsf{pk}_i := (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}) := (g_1^{\mathsf{sk}_i}, g_2^{\mathsf{sk}_i})$ for all $i \in [K]$.

   (b) *Cut-and-Choose Verification.* To verify the opening, S runs algorithm $\mathsf{Check}(L, (\mathsf{info}_l)_{l\in[L]}, \mathsf{open})$ (see Figure 3). If this algorithm returns 0, S aborts the interaction.

   (c) *Responses.* For each instance $i \in [K]$ and each $l \in [L]$, S computes responses $s_{i,l} := c_{i,\mathbf{J}_i,l}^{\mathsf{sk}_i}$. Then, it aggregates them for each $l \in [L]$ by computing $\bar{s}_l := \prod_{i=1}^K s_{i,l}$. Finally, S sends $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}_1, \ldots, \bar{s}_L$ to U.

3. User U does the following.

   (a) *Key Sharing Verification.* First, U recomputes key pk_K as $\mathsf{pk}_K := (\mathsf{pk}_{K,1}, \mathsf{pk}_{K,2})$ for $\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$ and $\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$. Next, U checks validity of the pk_i by checking if
   $$e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right) \text{ for all } i \in [K].$$
   If any of these equations does not hold, U aborts the interaction.

   (b) *Response Verification.* Then, U verifies the responses $\bar{s}_l$ by checking
   $$e\left(\bar{s}_l, g_2\right) = \prod_{i=1}^K e\left(c_{i,\mathbf{J}_i,l}, \mathsf{pk}_{i,2}\right) \text{ for all } l \in [L].$$
   If any of these equations does not hold, U aborts the interaction. Otherwise, it computes
   $$\bar{\sigma}_l := \bar{s}_l \cdot \prod_{i=1}^K \mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i,l}} \text{ for all } l \in [L].$$

[4]This is without loss of generality, using a collision-resistant hash function.

(c) *Key Rerandomization.* Next, U computes rerandomized key sharings via

$$((\mathsf{pk}'_{i,l})_i, \bar{\sigma}'_l) \leftarrow \mathsf{ReRa}((\mathsf{pk}_i, \mathsf{H}(\mathsf{info}_l, \mu_{i,\mathbf{J}_i,l}))_i, \bar{\sigma}_l) \text{ for all } l \in [L].$$

It then defines signatures

$$\sigma_l := ((\mathsf{pk}'_{i,l}, \varphi_{i,\mathbf{J}_i,l})_{i=1}^{K-1}, \varphi_{K,\mathbf{J}_K,l}, \bar{\sigma}'_l) \text{ for all } l \in [L].$$

(d) Finally, U outputs the signatures $\sigma_1, \ldots, \sigma_L$.

Verification. The resulting signature $\sigma := ((\mathsf{pk}_i, \varphi_i)_{i=1}^{K-1}), \varphi_K, \bar{\sigma})$ for a message m and string info is verified by algorithm $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}, \mathsf{m}, \sigma)$ as follows:

1. Write $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ for each $i \in [K-1]$.
2. Compute $\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$ and $\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$.
3. If there is an $i \in [K]$ with $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$, return 0.
4. For each instance $i \in [K]$, compute $\mu_i := \mathsf{H}_\mu(\mathsf{m}, \varphi_i)$.
5. Return 1 if and only if

$$e\left(\bar{\sigma}, g_2\right) = \prod_{i=1}^{K} e\left(\mathsf{H}(\mathsf{info}, \mu_i), \mathsf{pk}_{i,2}\right).$$

Alg $\mathsf{Check}\left(L, (\mathsf{info}_l)_{l\in[L]}, \mathsf{open} = \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j\neq\mathbf{J}_i}, (c_{i,\mathbf{J}_i,l})_{l\in[L]}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i\in[K]}\right)\right)$

```
01 for $i \in [K]$ :
02   for $j \in [N] \setminus \{\mathbf{J}_i\}$ :
03     $\mathsf{com}_{i,j} := \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,j})$
04     parse $\mathsf{r}_{i,j} = (\gamma_{i,j}, \mu_{i,j,1}, \ldots, \mu_{i,j,L}) \in (\{0,1\}^\lambda)^{L+1}$
05     for $l \in [L]$ : $\alpha_{i,j,l} := \mathsf{H}_\alpha(\gamma_{i,j}, l)$, $c_{i,j,l} := \mathsf{H}(\mathsf{info}_l, \mu_{i,j,l}) \cdot g_1^{\alpha_{i,j,l}}$
06 $\mathsf{com} := (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$, $c := (c_{1,1,1}, \ldots, c_{K,N,L})$
07 if $\mathbf{J} \neq \mathsf{H}_{cc}(\mathsf{com}, c)$ : return 0
08 return 1
```

Figure 3: The algorithm Check used in the signature issuing protocol of batched blind signature scheme BPBS_R.

4.4 Security Analysis

Completeness of the scheme follows by inspection. The proofs and concrete security bounds for blindness and one-more unforgeability are almost identical to the proofs of the corresponding theorems in Section 3. We postpone the formal analysis to Supplementary Material Section C.

5 Concrete Parameters and Efficiency

In this section, we discuss concrete parameters and efficiency of our scheme.

Instantiating Parameters. We instantiate our scheme over the BLS12-381 curve, using SHA-256 as a hash function. It remains to determine appropriate choices for parameters K and N. To do that, we first fix some choice of N and a bit security level $\kappa = 128$. Then, we assume a maximum number of $\ell = 2^{30}$ signing interactions with the same key. Following the security bound, we can now set $K := \lceil (\kappa + \log \ell)/\log N \rceil + 1$. This approach leads to the instantiations

$$\text{(I)}\ K = 80,\ N = 4, \quad \text{(II)}\ K = 54,\ N = 8, \quad \text{(III)}\ K = 33,\ N = 32.$$

For these, we compute the sizes of signatures and communication in a Python script (see Supplementary Material Section D). Our results are presented in Table 2. Let us briefly discuss the results. Especially, we want to compare the communication complexity of our scheme to the communication complexity of PI-Cut-Choo [CAHL+22]. For that, we use instantiation (I) of our scheme, which has roughly 33 KB of communication. For PI-Cut-Choo, we base our calculation on the numbers in Table 1 in the eprint version of PI-Cut-Choo, which says that the communication is roughly $26 + 3 \cdot \log(N)$ KB. For $N = 6$, the communication size of PI-Cut-Choo already exceeds the size of Rai-Choo. Thus, even assuming that N is exactly the number of interactions, PI-Cut-Choo is only better in the first 5 interactions. This assumption does not even hold, as N is not identical to the number of interactions, see function f in Theorem 4.2 of PI-Cut-Choo. In addition, we remark that when setting parameters, one has to be very conservative about the number of signing interactions. Especially, for PI-Cut-Choo, the signer's counter will increase quickly, because for each (even non-malicious) timeout, the counter has to be increased. Such timeouts are even more likely when the scheme requires multiple signing rounds, as PI-Cut-Choo does. An adversary can amplify this, leading to a kind of DoS attack. Thus, Rai-Choo outperforms PI-Cut-Choo immediately in terms of communication.

Implementation. To demonstrate computational practicality, we prototypically implemented our scheme in C++ using above parameter settings. Our implementation uses the MCL library[5] and can be found at

`https://github.com/b-wagn/Raichoo`

Although our scheme is highly parallelizable, we did not implement any parallelization. To evaluate the efficiency of our implementation, we determined the average running time over 100 runs of the signing interaction (i.e. running U_1, then S, then U_2), and the verification algorithm. For our tests, we used a Intel Core i5-7200U processor @2,5 GHz with 4 cores and 8 GB of RAM, running Ubuntu 20.04.4 LTS 64-bit. Our results are presented in Table 2. In general, the table shows a tradeoff between signature size, communication complexity, and computational efficiency.

Other Pairing Settings. We could also instantiate our scheme in the type-2 or type-1 pairing setting. To recall, in the type-2 setting, there is an efficient isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$. In the type-1 setting, we have $\mathbb{G}_1 = \mathbb{G}_2$. In both cases, the public keys pk_i in our scheme only have to be sent in $\mathbb{G}_2$. Type-1 pairing-friendly curves are usually constructed from supersingular curves over a field with small characteristics (2 or 3). They were shown to be insecure [BGJT14]. The alternative are supersingular curves over a larger field, but we only know how to construct them with an embedding degree 3. Assuming the same target group size as for BLS12-381 (4572 bits), we get impractical group element sizes of 1525 bits. For type-2 pairings, we can start with the BLS12-381 parameters, and replace the curve $\mathbb{G}_2$ with a curve $\mathbb{G}_2'$ defined over a larger extension field[CM09]. The most efficient way to represent elements in $\mathbb{G}_2'$ is to represent them as an element of $\mathbb{G}_1 \times \mathbb{G}_2$. For more details, see [CM09]. Therefore, by sending all keys in $\mathbb{G}_2'$, we obtain the same communication and signature sizes as in the type-3 setting. The security of such a variant of our scheme will solely rely on the computational Diffie-Hellman assumption in the group $\mathbb{G}_2'$.

Concrete Bit Security. In contrast to [CAHL+22], we compute our parameters using standardized curves and hash functions instead of estimating parameters based on the security loss. The reason for this is twofold. First, we want our numbers be consistent with our implementation and therefore have to rely on standardized components. Second, the estimations in [CAHL+22] assume a generic mapping from the bit security of CDH to the size of an appropriate group. This is not always given. To discuss the effect of the security loss, we now assume all components are roughly 128 bit secure. Then, the guaranteed security for our scheme is roughly $128 - \log \ell = 98$ bit. This is the same for the PI-Cut-Choo scheme [CAHL+22], and the standard BLS signature scheme [BLS01].

References

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001. (Cited on page 2.)

[5] See `https://github.com/herumi/mcl`

[AKSY21] Shweta Agrawal, Elena Kirshanova, Damien Stehle, and Anshu Yadav. Can round-optimal lattice-based blind signatures be practical? Cryptology ePrint Archive, Report 2021/1565, 2021. `https://eprint.iacr.org/2021/1565`. (Cited on page 6.)

[AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, August 2000. (Cited on page 2.)

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 157–186. Springer, Heidelberg, May / June 2022. (Cited on page 3, 17.)

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 1–16. Springer, Heidelberg, May 2014. (Cited on page 20.)

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 82–99. Springer, Heidelberg, December 2013. (Cited on page 6.)

[BLL$^+$21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021. (Cited on page 2.)

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001. (Cited on page 2, 3, 6, 13, 20.)

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. (Cited on page 1, 6.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003. (Cited on page 1, 2, 5, 6.)

[CAHL$^+$22] Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 3–31. Springer, Heidelberg, August 2022. (Cited on page 1, 2, 3, 4, 5, 20.)

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. (Cited on page 6.)

[CG08] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 345–356. ACM Press, October 2008. (Cited on page 1.)

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982. (Cited on page 1.)

[CHKM09] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two pairing-based aggregate signature schemes. Cryptology ePrint Archive, Report 2009/060, 2009. `https://eprint.iacr.org/2009/060`. (Cited on page 3, 7.)

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001. (Cited on page 1.)

[CM09] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings – the role of Ψ revisited. Cryptology ePrint Archive, Report 2009/480, 2009. `https://eprint.iacr.org/2009/480`. (Cited on page 20.)

[dK22] Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336. Springer, Heidelberg, August 2022. (Cited on page 6.)

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015. (Cited on page 1, 6.)

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006. (Cited on page 6.)

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. (Cited on page 6.)

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020. (Cited on page 6.)

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, Heidelberg, May / June 2010. (Cited on page 6.)

[GG14] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, Heidelberg, May 2014. (Cited on page 6.)

[Gha17] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Heidelberg, April 2017. (Cited on page 6.)

[GPZZ19] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 210–231. Springer, Heidelberg, March 2019. (Cited on page 1.)

[GRS$^+$11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Heidelberg, August 2011. (Cited on page 6.)

[HBG16] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 43–60. Springer, Heidelberg, February 2016. (Cited on page 1, 3.)

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019. (Cited on page 1, 2.)

[HKLN20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, August 2020. (Cited on page 2.)

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003. (Cited on page 3, 17.)

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164. Springer, Heidelberg, August 1997. (Cited on page 6.)

[KLR21] Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 468–492. Springer, Heidelberg, December 2021. (Cited on page 1, 2, 3, 4, 5.)

[KLX22] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2022. (Cited on page 6.)

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993. (Cited on page 2.)

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer, Heidelberg, March 2006. (Cited on page 6.)

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Heidelberg, August 1992. (Cited on page 1.)

[Pas11] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 109–118. ACM Press, June 2011. (Cited on page 6.)

[Poi98] David Pointcheval. Strengthened security for blind signatures. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 391–405. Springer, Heidelberg, May / June 1998. (Cited on page 2.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. (Cited on page 1, 2.)

[Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, November 2001. (Cited on page 2.)

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. (Cited on page 6.)

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811. Springer, Heidelberg, May / June 2022. (Cited on page 6.)

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002. (Cited on page 2.)

Supplementary Material

A Postponed Proofs from Section 3

Proof of Lemma 1. It is sufficient to look at the distributions in terms of their exponents. To this end, let $\mathsf{pk} = g_1^x$, $\mathsf{pk}_{i,1} = g_1^{x_i}$, and $\mathsf{pk}'_{i,1} = g_1^{x'_i}$ for all $i \in [K]$. Consider the homomorphism $f : \mathbb{Z}_p^K \to \mathbb{Z}_p$ with $f(\mathbf{y}) := (1, \ldots, 1) \cdot \mathbf{y}$. Note that in distribution $\mathcal{D}_2$, the vector $\mathbf{x}' = (x'_1, \ldots, x'_K)^t$ is distributed uniformly over all vectors in $f^{-1}(x)$. Further, note that in distribution $\mathcal{D}_1$, the vector $\mathbf{x}'$ is distributed as $\mathbf{x} + \mathbf{r}$, where $\mathbf{r}$ is uniform in the kernel of f. This gives us the same distribution for $\mathbf{x}'$. Therefore, the distributions are the same. $\square$

B Formal Definitions for Batch Partially Blind Signatures

Definition 5 (Batched Partially Blind Signature Scheme)**.** A batched partially blind signature scheme is a quadruple of PPT algorithms $\mathsf{BPBS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ takes as input the security parameter 1^λ and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$. We assume that the public key pk defines a message space $\mathcal{M} = \mathcal{M}_{\mathsf{pk}}$, and a public information space $\mathcal{I} = \mathcal{I}_{\mathsf{pk}}$ implicitly.
- S and U are interactive algorithms, where S takes as input a secret key sk, a batch size $L \in \mathbb{N}$, and L strings $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$, and U takes as input a key pk, a batch size $L \in \mathbb{N}$, and L pairs of messages $\mathsf{m}_1, \ldots, \mathsf{m}_L \in \mathcal{M}$ and strings $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$. After the execution, U returns L signatures $\sigma_1, \ldots, \sigma_L$ and we write
$$(\bot, (\sigma_1, \ldots, \sigma_L)) \leftarrow \langle \mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l \in [L]}), \mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l \in [L]} \rangle.$$
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}, \mathsf{m}, \sigma) \to b$ is deterministic and takes as input public key pk, a string $\mathsf{info} \in \mathcal{I}$, message $\mathsf{m} \in \mathcal{M}$, and a signature σ, and returns $b \in \{0, 1\}$.

We require that BPBS is complete in the following sense. For all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$, all $L = \mathsf{poly}(\lambda)$, all $\mathsf{m}_1, \ldots, \mathsf{m}_L \in \mathcal{M}_{\mathsf{pk}}$, and all $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}_{\mathsf{pk}}$ it holds that
$$\Pr\left[\forall l \in [L] : b_l = 1 \;\middle|\; \begin{array}{l} (\bot, (\sigma_1, \ldots, \sigma_L)) \\ \quad \leftarrow \langle \mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l \in [L]}), \mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l \in [L]} \rangle, \\ \forall l \in [L] : b_l := \mathsf{Ver}(\mathsf{pk}, \mathsf{info}_l, \mathsf{m}_l, \sigma_l) \end{array}\right] = 1.$$

Definition 6 (Batch One-More Unforgeability)**.** Let $\mathsf{BPBS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ be a batched partially blind signature scheme and $\ell \colon \mathbb{N} \to \mathbb{N}$. For an algorithm $\mathcal{A}$, we consider the following game $\ell\textbf{-OMUF}^{\mathcal{A}}_{\mathsf{BPBS}}(\lambda)$:

1. Sample keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Let O be an interactive oracle, taking a batch size $L \in \mathbb{N}$ and L strings $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$ as input, and then simulating $\mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l \in [L]})$. Run
$$((\mathsf{info}_1, \mathsf{m}_1, \sigma_1), \ldots, (\mathsf{info}_k, \mathsf{m}_k, \sigma_k)) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}),$$
where $\mathcal{A}$ can query O in an arbitrarily interleaved way. Let $\mathcal{C}$ denote the list of all tuples $(i, L, (\mathsf{info}_l)_{l \in [L]})$ such that $\mathcal{A}$ submitted batch size L and strings $(\mathsf{info}_l)_{l \in [L]}$ in the ith completed interaction with O. It is required to hold that $\sum_{(i, L, (\mathsf{info}_l)_{l \in [L]}) \in \mathcal{C}} L \leq \ell$.
3. For each $\mathsf{info} \in \mathcal{I}$, define the sets completed interactions and outputs
$$\begin{aligned} \mathsf{Compl}[\mathsf{info}] &:= \{(i, l_0) \mid \exists (i, L, (\mathsf{info}_l)_{l \in [L]}) \in \mathcal{C} : \mathsf{info}_{l_0} = \mathsf{info}\} \\ \mathsf{Out}[\mathsf{info}] &:= \{i \in [k] \mid \mathsf{info}_i = \mathsf{info}\}. \end{aligned}$$
Output 1 if and only if there is some $\mathsf{info}^* \in \mathcal{I}$ such that all $\mathsf{m}_i, i \in \mathsf{Out}[\mathsf{info}^*]$ are distinct, $|\mathsf{Compl}[\mathsf{info}^*]| < |\mathsf{Out}[\mathsf{info}^*]|$, and for each $i \in \mathsf{Out}[\mathsf{info}^*]$ it holds that $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}_i, \mathsf{m}_i, \sigma_i) = 1$.

We say that BPBS is ℓ-batch-one-more unforgeable (ℓ-BOMUF), if for every PPT algorithm $\mathcal{A}$ the following advantage is negligible:
$$\mathsf{Adv}^{\ell\text{-}\mathsf{BOMUF}}_{\mathcal{A}, \mathsf{BPBS}}(\lambda) := \Pr\left[\ell\textbf{-BOMUF}^{\mathcal{A}}_{\mathsf{BPBS}}(\lambda) \Rightarrow 1\right].$$
We say that BPBS is batch one-more unforgeable (BOMUF), if it is ℓ-BOMUF for all polynomial ℓ.

Definition 7 (Batch Partial Blindness)**.** Consider a batch partially blind signature scheme $\mathsf{BPBS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$. For an algorithm $\mathcal{A}$ and bit $b \in \{0, 1\}$, consider the following game $\textbf{BBLIND}^{\mathcal{A}}_{b, \mathsf{BPBS}}(\lambda)$:

1. Run $\big(\mathsf{pk}, (\mathsf{info}_{l,0}, \mathsf{m}_{l,0})_{l \in [L_0]}, (\mathsf{info}_{l,1}, \mathsf{m}_{l,1})_{l \in [L_1]}, l_{*,0}, l_{*,0}, St\big) \leftarrow \mathcal{A}(1^\lambda)$. Then, if $\mathsf{info}_{l_{*,0},0} \neq \mathsf{info}_{l_{*,1},1}$, then return 0.
2. If $b = 1$, then swap $\mathsf{m}_{l_{*,0},0}$ and $\mathsf{m}_{l_{*,0},1}$. That is, set $\mathsf{m}' := \mathsf{m}_{l_{*,0},0}, \mathsf{m}_{l_{*,0},0} := \mathsf{m}_{l_{*,1},1}, \mathsf{m}_{l_{*,1},1} := \mathsf{m}'$.

3. Let O_0 and O_1 be interactive oracles simulating

$$\mathsf{U}(\mathsf{pk}, L_0, (\mathsf{m}_{l,0}, \mathsf{info}_{l,0})_{l \in [L_0]}) \textit{ and } \mathsf{U}(\mathsf{pk}, L_1, (\mathsf{m}_{l,1}, \mathsf{info}_{l,1})_{l \in [L_1]}),$$

respectively. Run $\mathcal{A}$ on input St with arbitrary interleaved one-time access to each of these oracles, i.e. $St' \leftarrow \mathcal{A}^{\mathrm{O}_0, \mathrm{O}_1}(St)$.

4. Let

$$\sigma_{1,0}, \ldots, \sigma_{l_{*,0}-1,0}, \sigma_{l_{*,b},b}, \sigma_{l_{*,0}+1,0}, \ldots, \sigma_{L_0} \textit{ and}$$
$$\sigma_{1,1}, \ldots, \sigma_{l_{*,1}-1,1}, \sigma_{l_{*,1-b},1-b}, \sigma_{l_{*,1}+1,1}, \ldots, \sigma_{L_1}$$

be the local outputs of $\mathrm{O}_0, \mathrm{O}_1$, respectively. If $\sigma_{i,0} = \perp$ or $\sigma_{i',1} = \perp$ for some $i \in [L_0]$ or some $i' \in [L_1]$, then run $b' \leftarrow \mathcal{A}(St', \perp)$. Else, obtain a bit b' from $\mathcal{A}$ on input $(\sigma_{l,0})_{l \in [L_0]}, (\sigma_{l,1})_{l \in [L_1]}$. That is, run $b' \leftarrow \mathcal{A}\left(St', (\sigma_{l,0})_{l \in [L_0]}, (\sigma_{l,1})_{l \in [L_1]}\right)$.

5. Output b'.

We say that BPBS satisfies malicious signer batch partial blindness, if for every PPT algorithm $\mathcal{A}$ the following advantage is negligible:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{bblind}}_{\mathcal{A},\mathsf{BPBS}}(\lambda) := |\Pr &\left[\mathbf{BBLIND}^{\mathcal{A}}_{0,\mathsf{BPBS}}(\lambda) \Rightarrow 1\right] \\ -\Pr &\left[\mathbf{BBLIND}^{\mathcal{A}}_{1,\mathsf{BPBS}}(\lambda) \Rightarrow 1\right]|.\end{aligned}$$

C Security Analysis for the Batched Construction

C.1 Batch Partial Blindness

Theorem 3. *Let $\mathsf{H_r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p$ be random oracles. Then BPBS_R satisfies malicious signer batch partial blindness.*

Concretely, for any algorithm $\mathcal{A}$ that makes at most $Q_{\mathsf{H_r}}, Q_{\mathsf{H}_\mu}, Q_{\mathsf{H}_\alpha}$ queries to $\mathsf{H_r}, \mathsf{H}_\mu, \mathsf{H}_\alpha$ respectively, we have

$$\mathsf{Adv}^{\mathsf{bblind}}_{\mathcal{A},\mathsf{BPBS}_R}(\lambda) \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H_r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}.$$

Proof. The proof is almost identical to the proof of Theorem 1, and we encourage the reader to read the proof of Theorem 1 first. Here, we only sketch the differences. Set $\mathsf{BPBS} := \mathsf{BPBS}_R$ and let $\mathcal{A}$ be an adversary against the batch partial blindness of BPBS. As in the proof of Theorem 1, we prove the statement using games $\mathbf{G}_{i,b}$ for $i \in [8]$ and $b \in \{0,1\}$ such that

$$\Pr[\mathbf{G}_{8,0} \Rightarrow 1] = \Pr[\mathbf{G}_{8,1} \Rightarrow 1].$$

Game $\mathbf{G}_{0,b}$: We set $\mathbf{G}_{0,b}$ as $\mathbf{G}_{0,b} := \mathbf{BBLIND}^{\mathcal{A}}_{b,\mathsf{BPBS}}(\lambda)$. Recall that in this game, $\mathcal{A}$ outputs a public key pk, lists $(\mathsf{info}^L_l, \mathsf{m}^L_l)_{l\in[L^L]}$ and $(\mathsf{info}^R_l, \mathsf{m}^R_l)_{l\in[L^R]}$, and indices l^L_* and l^R_*. The game outputs 0 if $\mathsf{info}^L_{l^L_*} \neq \mathsf{info}^R_{l^R_*}$. For the rest of the proof, we can assume that $\mathsf{info}^L_{l^L_*} = \mathsf{info}^R_{l^R_*}$. Then, if $b = 1$, the messages $\mathsf{m}^L_{l^L_*}$ and $\mathsf{m}^R_{l^R_*}$ are swapped. Adversary $\mathcal{A}$ gets access to oracles O_0 and O_1 simulating

$$\mathsf{U}(\mathsf{pk}, L^L, (\mathsf{m}^L_l, \mathsf{info}^L_l)_{l\in[L^L]}) \text{ and } \mathsf{U}(\mathsf{pk}, L^R, (\mathsf{m}^R_l, \mathsf{info}^R_l)_{l\in[L^R]}),$$

respectively. As in the proof of Theorem 1, we use superscripts L and R to distinguish variables used in these oracles. If no superscript is given, the description refers to both oracles.

Game $\mathbf{G}_{1,b}$: Game $\mathbf{G}_{1,b}$ is as $\mathbf{G}_{0,b}$, but we add an abort on a certain event. Namely, the game aborts if $\mathcal{A}$ ever queries $\mathsf{H}_\mu(\cdot, \varphi_{i,j,l_*})$ for some $i \in [K]$ and $j \in [N] \setminus \{\mathbf{J}_i\}$. As $\mathcal{A}$ obtains no information about φ_{i,j,l_*} over the entire game, we have

$$|\Pr[\mathbf{G}_{0,b} \Rightarrow 1] - \Pr[\mathbf{G}_{1,b} \Rightarrow 1]| \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-1}}.$$

Game $\mathbf{G}_{2,b}$: Game $\mathbf{G}_{2,b}$ is as $\mathbf{G}_{1,b}$, but with another abort event. The game aborts, if $\mathcal{A}$ ever makes a query $\mathsf{H_r}(\mathsf{r}_{i,\mathbf{J}_i})$, or a query $\mathsf{H}_\alpha(\gamma_{i,\mathbf{J}_i}, l_*)$ for some $i \in [K]$. As in the proof of Theorem 1, the probability of such an abort is negligible due to the entropy of $\gamma_{i,\mathbf{J}_i}$, and we get

$$|\Pr[\mathbf{G}_{1,b} \Rightarrow 1] - \Pr[\mathbf{G}_{2,b} \Rightarrow 1]| \leq \frac{KQ_{\mathsf{H_r}}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-1}}.$$

Games $\mathbf{G}_{3,b}$-$\mathbf{G}_{4,b}$: The changes we introduce in these games are exactly as in the proof of Theorem 1, but it is sufficient to apply them to the final signatures for messages $\mathsf{m}^L_{l^L_*}$ and $\mathsf{m}^R_{l^R_*}$. As in the proof of Theorem 1, we have

$$\Pr[\mathbf{G}_{2,b} \Rightarrow 1] = \Pr[\mathbf{G}_{4,b} \Rightarrow 1].$$

Game $\mathbf{G}_{5,b}$: This change is exactly as in the proof of Theorem 1, i.e. we let the game sample random vectors $\hat{\mathbf{J}}^L \leftarrow\!\!\$\, [N]^K$ and $\hat{\mathbf{J}}^R \leftarrow\!\!\$\, [N]^K$, and later the game aborts if we do not have $\hat{\mathbf{J}}^L = \mathbf{J}^L$ and $\hat{\mathbf{J}}^R = \mathbf{J}^R$. We have

$$\Pr[\mathbf{G}_{5,b} \Rightarrow 1] = \frac{1}{N^{2K}} \cdot \Pr[\mathbf{G}_{4,b} \Rightarrow 1].$$

Game $\mathbf{G}_{6,b}$: In this game, we change how the values μ_{i,j,l_*} for $i \in [K]$ and $j \in [N] \setminus \{\hat{\mathbf{J}}_i\}$ are computed. Namely, while they were defined as $\mu_{i,j,l} := \mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j,l})$ before, we now sample them at random, i.e. $\mu_{i,j,l} \leftarrow\!\!\$\, \{0,1\}^\lambda$. We can argue using the change we introduced in $\mathbf{G}_{1,b}$, and similar to the proof of Theorem 1, to get

$$\Pr[\mathbf{G}_{5,b} \Rightarrow 1] = \Pr[\mathbf{G}_{6,b} \Rightarrow 1].$$

Game $\mathbf{G}_{7,b}$: We change how the values $\alpha_{i,\hat{\mathbf{J}}_i,l_*}$ and $\mathsf{com}_{i,\hat{\mathbf{J}}_i}$ are computed for all $i \in [K]$. Namely we sample $\alpha_{i,\hat{\mathbf{J}}_i,l_*} \leftarrow\!\!\$\, \mathbb{Z}_p$ and $\mathsf{com}_{i,\hat{\mathbf{J}}_i} \leftarrow\!\!\$\, \{0,1\}^\lambda$. We can argue using the changes in $\mathbf{G}_{5,b}$ and $\mathbf{G}_{2,b}$, and similar to the proof of Theorem 1, to get

$$\Pr[\mathbf{G}_{6,b} \Rightarrow 1] = \Pr[\mathbf{G}_{7,b} \Rightarrow 1].$$

Game $\mathbf{G}_{8,b}$: We change how the values $c_{i,\hat{\mathbf{J}}_i,l_*}$ are computed. Namely, we now sample these at random, i.e. $c_{i,\hat{\mathbf{J}}_i,l_*} \leftarrow\!\!\$\, \mathbb{G}_1$. We can argue as in the proof of Theorem 1, to get

$$\Pr[\mathbf{G}_{7,b} \Rightarrow 1] = \Pr[\mathbf{G}_{8,b} \Rightarrow 1].$$

Finally, it can be observed that the view of $\mathcal{A}$ is independent of the bit b. The statement follows as in the proof of Theorem 1. □

C.2 Batch One-More Unforgeability

Theorem 4. *Let $\mathsf{H_r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$, and $\mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} \colon \{0,1\}^* \to \mathbb{G}$ be random oracles. If CDH assumption holds relative to PGGen, then BPBS_R is batch one-more unforgeable.*

Concretely, for any polynomial ℓ and any PPT algorithm $\mathcal{A}$ that makes at most $Q_{\mathsf{H}_{cc}}, Q_{\mathsf{H_r}}, Q_{\mathsf{H}_\mu}, Q_{\mathsf{H}}$ queries to $\mathsf{H}_{cc}, \mathsf{H_r}, \mathsf{H}_\mu, \mathsf{H}$ respectively, there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and

$$\mathsf{Adv}^{\ell\text{-}\mathsf{BOMUF}}_{\mathcal{A},\mathsf{BPBS}_R}(\lambda) \leq \frac{Q^2_{\mathsf{H}_\mu} + Q^2_{\mathsf{H_r}} + Q_{\mathsf{H_r}} Q_{\mathsf{H}_{cc}} + Q_{\mathsf{H}} Q_{\mathsf{H}_\mu}}{2^\lambda} + \frac{\ell}{N^K} + 4\ell \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

Proof. The proof is a direct generalization of the proof of Theorem 2. The reader should read the proof of Theorem 2 first. We only sketch the differences.

Set $\mathsf{BPBS} := \mathsf{BPBS}_R$ and let $\mathcal{A}$ be an adversary against the batch one-more unforgeability of BPBS. As in the proof of Theorem 2, we prove the theorem using a sequence of games $\mathbf{G}_i, i \in [9]$.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is defined to be $\mathbf{G}_0 := \ell\text{-}\mathbf{BOMUF}^{\mathcal{A}}_{\mathsf{BPBS}}$. That is, first a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ is sampled. Then, $\mathcal{A}$ can access a signer oracle O. In each interaction this oracle takes as input a batch size $L \in \mathbb{N}$ and L strings $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$, and then simulates $\mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]})$. Further, $\mathcal{C}$ denotes the list of all tuples $(i, L, (\mathsf{info}_l)_{l\in[L]})$ such that $\mathcal{A}$ submitted batch size L and strings $(\mathsf{info}_l)_{l\in[L]}$ in the ith completed interaction with O. We have $\sum_{(i,L,(\mathsf{info}_l)_{l\in[L]})\in\mathcal{C}} L \leq \ell$. In the end, $\mathcal{A}$ outputs tuples $(\mathsf{info}_1, \mathsf{m}_1, \sigma_1), \ldots, (\mathsf{info}_k, \mathsf{m}_k, \sigma_k)$. Adversary $\mathcal{A}$ wins, if there is some info^*, such that, considering only the tuples with first component $\mathsf{info}*$, all messages m_i are distinct, all signatures σ_i are valid, and there are more tuples of this form than completed interactions with this info^*. We have

$$\mathsf{Adv}^{\ell\text{-}\mathsf{BOMUF}}_{\mathcal{A},\mathsf{BPBS}}(\lambda) = \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Games $\mathbf{G}_1$-$\mathbf{G}_5$: These games are exactly as in the proof of Theorem 2. Namely, we rule out collisions for random oracles $\mathsf{H}_\mu, \mathsf{H_r}$, let the game extract values $\bar{\mathsf{r}}_{i,j}$ during queries to H_{cc}, and establish that there is at least one instance i^* per interaction, for which the adversary does not cheat successfully. Recall that this means that the game can extract $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\gamma, \mu_1, \ldots, \mu_L) \neq \bot$ and $c_{i^*,\mathbf{J}_{i^*},l} = \mathsf{H}(\mathsf{info}_l, \mu_l) \cdot g_1^{\alpha_l}$ for $\alpha_l := \mathsf{H}_\alpha(\gamma, l)$ and all $l \in [L]$. Here, L and $(\mathsf{info}_l)_{l\in[L]}$ denote the batch size and the public strings that the adversary submitted to oracle O in this interaction. Also, in this sequence of games we change the way the secret keys sk_i are sampled, and introduce an abort related to the order of queries for H and H_μ All of this is done exactly as in the proof of Theorem 2. We have

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{Q^2_{\mathsf{H}_\mu}}{2^\lambda} + \frac{Q^2_{\mathsf{H_r}}}{2^\lambda} + \frac{Q_{\mathsf{H_r}} Q_{\mathsf{H}_{cc}}}{2^\lambda} + \frac{\ell}{N^K} + \frac{Q_{\mathsf{H}} Q_{\mathsf{H}_\mu}}{2^\lambda}.$$

Game $\mathbf{G}_6$: Game $\mathbf{G}_6$ is as $\mathbf{G}_5$, but we introduce a conceptual change. Namely, the game now holds maps $b[\cdot]$ and $\hat{b}[\cdot]$. On a query of the form $\mathsf{H}(\mathsf{info}, \mu)$, the game first searches for a previous query $(\mathsf{m}_\mu, \varphi)$ to H_μ such that $\mathsf{H}_\mu(\mathsf{m}_\mu, \varphi) = \mu$. If no such query is found, the game sets $b[\mathsf{info}, \mu] := 0$. If such a query $(\mathsf{m}_\mu, \varphi)$ is found, and $\hat{b}[\mathsf{info}, \mathsf{m}_\mu]$ is not yet defined, the game samples $\hat{b}[\mathsf{info}, \mathsf{m}_\mu] \in \{0,1\}$ from a Bernoulli distribution, such that the probability that $\hat{b}[\mathsf{info}, \mathsf{m}_\mu] = 1$ is $1/(\ell+1)$. The game then sets $b[\mathsf{info}, \mu] := \hat{b}[\mathsf{info}, \mathsf{m}_\mu]$. Note that the game can find at most one m_μ for a given μ, due to the change introduced in $\mathbf{G}_1$. Clearly, the view of the adversary $\mathcal{A}$ does not change, and we have

$$\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_7$: As in the proof of Theorem 2, we introduce an initially empty set $\mathcal{L}$ in this game, and add a new abort event. To this end, consider an interaction between the adversary $\mathcal{A}$ and the signer oracle. In this interaction, we know that there is at least one instance i^* for which the adversary does not cheat successfully. That is, the game can extract $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\gamma, \mu_1, \ldots, \mu_L) \neq \bot$ and $c_{i^*,\mathbf{J}_{i^*},l} = \mathsf{H}(\mathsf{info}_l, \mu_l) \cdot g_1^{\alpha_l}$ for $\alpha_l := \mathsf{H}_\alpha(\gamma, l)$ and all $l \in [L]$. Additionally, in $\mathbf{G}_7$, the game now tries to extract queries $(\mathsf{m}_{\mu_l}, \varphi_l)$ for all $l \in [L]$ such that $\mu_l = \mathsf{H}_\mu(\mathsf{m}_{\mu_l}, \varphi_l)$. For those $l \in [L]$ for which it can extract, it inserts $(\mathsf{info}_l, \mu_l, \mathsf{m}_{\mu_l})$ into $\mathcal{L}$. It is clear that the size of $\mathcal{L}$ is at most $\sum_{(i,L,(\mathsf{info}_l)_{l\in[L]})\in\mathcal{C}} L \leq \ell$. Additionally, if there is some $l \in [L]$ such that $b[\mathsf{info}_l, \mu_l] = 1$, the game aborts.

Further, consider the final output $(\mathsf{info}_1, \mathsf{m}_1, \sigma_1), \ldots, (\mathsf{info}_k, \mathsf{m}_k, \sigma_k)$ of $\mathcal{A}$. Write $\sigma_r = ((\mathsf{pk}_{r,i}, \varphi_{r,i})_{i=1}^{K-1}), \varphi_{r,K}, \bar{\sigma}_r)$, and set $\mu_{r,i} := \mathsf{H}_\mu(\mathsf{m}_r, \varphi_{r,i})$ for all $r \in [k], i \in [K]$. If $\mathcal{A}$ is successful, we know that there is some info^* such that $|\mathsf{Compl}[\mathsf{info}^*]| < |\mathsf{Out}[\mathsf{info}^*]|$, where

$$\mathsf{Compl}[\mathsf{info}^*] := \{(i, l_0) \mid \exists (i, L, (\mathsf{info}_l)_{l\in[L]}) \in \mathcal{C} : \mathsf{info}_{l_0} = \mathsf{info}^*\}$$
$$\mathsf{Out}[\mathsf{info}^*] := \{i \in [k] \mid \mathsf{info}_i = \mathsf{info}^*\}.$$

It is easy to see that $|\mathsf{Compl}[\mathsf{info}^*]|$ is an upper bound for the number of tuples $(\mathsf{info}, \mu, \mathsf{m}_\mu)$ in $\mathcal{L}$ with $\mathsf{info} = \mathsf{info}^*$. Therefore, by the pigeonhole principle, we know that there is at least one $(\tilde{r}, \tilde{i}) \in \mathsf{Out}[\mathsf{info}^*] \times [K]$ such that $(\mathsf{info}_{\tilde{r}}, \mu_{\tilde{r},\tilde{i}}, \mathsf{m}_{\tilde{r}}) \notin \mathcal{L}$. Here we have $\mathsf{info}_{\tilde{r}} = \mathsf{info}^*$. Game $\mathbf{G}_7$ finds the first such $(\tilde{r}, \tilde{i})$, sets $\mu^* := \mu_{\tilde{r},\tilde{i}}$ and aborts if $b[\mathsf{info}^*, \mu^*] = 0$. As in the proof of Theorem 2, we can assume that $b[\mathsf{info}^*, \mu^*]$ is defined. Also, as in the proof of Theorem 2, the game adds further entries $(\mathsf{info}, \mu, \mathsf{m}_\mu)$ to $\mathcal{L}$ such that $|\mathcal{L}| = \ell$, and aborts if $b[\mathsf{info}, \mu] = 1$.

The analysis of this change is as in the proof of Theorem 2. Namely, we first see that $\mathbf{G}_7$ outputs 1 if $\mathbf{G}_6$ outputs 1 and

$$b[\mathsf{info}^*, \mu^*] = 1 \wedge \forall (\mathsf{info}, \mu, \mathsf{m}_\mu) \in \mathcal{L} : b[\mathsf{info}, \mu] = 0.$$

Then, we use the same calculation and arguments as in the proof of Theorem 2 to get

$$\Pr[\mathbf{G}_7 \Rightarrow 1] \geq \frac{1}{4\ell} \cdot \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_8$: As in the proof of Theorem 2, we change how random oracle H is simulated. In the beginning of the game, the game samples $Y \leftarrow\!\!\$\ \mathbb{G}_1$ and initiates an empty map $t[\cdot]$. On a query $\mathsf{H}(\mathsf{info}, \mu)$ for which the hash value is not yet defined, the game first determines $b[\mathsf{info}, \mu]$ as explained in $\mathbf{G}_6$. Then, it samples $t[\mathsf{info}, \mu] \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets $\mathsf{H}(\mu) := Y^{b[\mathsf{info},\mu]} \cdot g_1^{t[\mathsf{info},\mu]}$. This does not change the view of the adversary. We have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_8 \Rightarrow 1].$$

Game $\mathbf{G}_9$: This change is as in the proof of Theorem 2. Namely, we change how the public key sharing $(\mathsf{pk}_i)_{i\in[K]}$ and the values $s_{i^*,l}$ for all $l \in [L]$ are computed in an interaction. The sharing $(\mathsf{pk}_i)_{i\in[K]}$ is computed exactly as in the proof of Theorem 2, and the values $s_{i^*,l}$ are now computed as $s_{i^*,l} = \mathsf{pk}_{i^*,1}^{\alpha_l + t[\mathsf{info}_l, \mu_l]}$, where $\alpha_l := \mathsf{H}_\alpha(\gamma, l)$, and $(\gamma, \mu_1, \ldots, \mu_L)$ has been extracted by the game. Due to this change, sk is no longer needed. As in the proof of Theorem 2, we have

$$\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1].$$

Finally, as in the proof of Theorem 2, we can give a reduction $\mathcal{B}$ against the CDH assumption that is successful if $\mathbf{G}_9$ outputs 1. This reduction gets as input $g_1, g_2, e, p,\ X_1, Y \in \mathbb{G}_1$, and $X_2 \in \mathbb{G}_2$. It sets $\mathsf{pk}_1 := X_1, \mathsf{pk}_2 := X_2$ and uses Y as explained in $\mathbf{G}_8$. Then, it simulates $\mathbf{G}_9$ for $\mathcal{A}$. Finally, it outputs

$$Z := \bar{\sigma}_{\tilde{r}} \cdot \prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{-t[\mathsf{info}^*, \mu_{\tilde{r},i}]}.$$

Analysis of $\mathcal{B}$ is as in the proof of Theorem 2, and we conclude with

$$\Pr\left[\mathbf{G}_9 \Rightarrow 1\right] \leq \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

□

D Script for Parameter Computation

Listing 1: Python Script to compute the parameters for our scheme. A discussion is given in Section 5.

```python
#!/usr/bin/env python

import math
from tabulate import tabulate


#Fixed paramters, e.g. group element size
size_group_one_element = 48*8 #use BLS12-381
size_group_two_element = 96*8 #use BLS12-381
secpar = 256 #use SHA-256
level = 128
log_q = 30




############################################
# Functions to determine the bit sizes of  #
# signatures, keys, and communication for  #
# given parameters                         #
############################################

def size_pk(K, N):
        return size_group_one_element+size_group_two_element

def size_sig(K, N):
        size_comm_rand = K*secpar
        size_pks = (K-1)*(size_group_one_element+size_group_two_element)
        size_aggregate_sig = size_group_one_element
        return size_comm_rand+size_pks+size_aggregate_sig

def size_communication_batched(K, N, L):
        size_cc_index = K*math.ceil(math.log(N,2))
        size_opening = K*((N-1)*(secpar+ L*secpar)+L*size_group_one_element+secpar)
        size_pks = (K-1)*(size_group_one_element+size_group_two_element)
        size_response = L*size_group_one_element
        return (size_cc_index + size_opening + size_pks + size_response)/float(L)


##########################################################


# Determine a minimum value K for given security level,
# number of queries and N, such that the term in the omuf
# bound q/N^K becomes small enough
# Setting K smaller would lead to no solution at all
def min_plausible_K(N):
        return math.ceil((level+log_q) / math.log(N,2))+ 1

# Compute a row of the table, i.e. efficiency measures for given
# parameters K, N, and batch sizes
def table_row(K, N, logLs):
        pk = size_pk(K, N)
        sig = size_sig(K, N)

        row = [level,log_q,K,N,pk/8000.0,sig/8000.0]
        for logL in logLs:
                L = 2**logL
                comm = size_communication_batched(K, N, L)
                row.append(comm/8000.0)

        return row



#HERE you can insert the combinations you want to try.
logNs = [2,3,5]
logLs = [0,2,4,8]

#tabulate preparation
data = [["Level", "log_q", "K", "N", "|pk|", "|sigma|"] + ["Comm L = " + str(2**logL) for logL in logLs]]
print("")

for logN in logNs:
        N = 2**logN
        K = min_plausible_K(N)
        row = table_row(K,N,logLs)
        data.append(row)

print(tabulate(data,headers='firstrow',tablefmt='fancy_grid'))
```