Boosting the Security of Blind Signature Schemes

Jonathan Katz, Julian Loss, and Michael Rosenberg*

Dept. of Computer Science, University of Maryland
{jkatz2,lossjulian}@gmail.com, micro@cs.umd.edu

Abstract. Existing blind signature schemes that are secure for polynomially many concurrent executions of the signing protocol are either inefficient or rely on non-standard assumptions (even in the random-oracle model). We show the first efficient blind signature schemes achieving this level of security based on the RSA, factoring, or discrete logarithm assumptions (in the random-oracle model). Our core technique involves an extension and generalization of a transform due to Pointcheval (Eurocrypt '98) that allows us to convert certain blind signature schemes that are secure for (concurrently) issuing logarithmically many signatures into ones secure for (concurrently) issuing polynomially many signatures.

1 Introduction

A *blind signature scheme* [6] consists of an interactive protocol executed between a signer $\mathcal{S}$ (holding a secret key sk) and a user $\mathcal{U}$ (holding a message m and the signer's public key pk), by which $\mathcal{U}$ obtains a signature σ on m. *Blindness* ensures that $\mathcal{S}$ learns nothing about m, and in fact is even unable to link (m, σ) to the execution of the protocol in which σ was generated. *One-more unforgeability* means that if $\mathcal{U}$ executes the signing protocol ℓ times, it should be unable to generate valid signatures on more than ℓ messages. Even in the random-oracle model, known blind signature schemes that support polynomially many signatures are either inefficient [18, 7, 16, 12, 11], rely on non-standard assumptions or the algebraic group model [3, 5, 21, 9, 8, 13, 19], or are secure only for sequential executions of the signing protocol [18, 2, 19]. Known efficient schemes that rely on standard assumptions such as RSA, factoring, the hardness of computing discrete logarithms, or the hardness of SIS [23–25, 1, 10, 15] are concurrently secure but only support logarithmically many signatures; moreover, for many schemes this limitation is known to be inherent as

* Work supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship.

there is an efficient attack [26, 27, 4] when running super-logarithmically many concurrent executions.

In an effort to obtain an efficient blind signature scheme secure for issuing polynomially many signatures, Pointcheval [22] showed a transform for "boosting" the security of the Okamoto-Schnorr blind signature scheme [20, 25]. Specifically, under the assumption that the Okamoto-Schnorr blind signature scheme is secure for logarithmically many sequential executions of the signing protocol (which itself can be shown to hold in the random-oracle model, based on the hardness of computing discrete logarithms), the transformed scheme is secure for polynomially many sequential executions of the signing protocol.[1] The resulting scheme, however, has a significant drawback: it requires the signer to *refuse to issue any further signatures* if a user is ever caught cheating. Thus, while the scheme could be used in a setting where the signer interacts with a *single* user repeatedly (and thus the signer would be justified in refusing to interact with that user once that user is caught cheating), the scheme is not appropriate for standard applications of blind signatures where the signer interacts with *many* users, some of whom may collude. Indeed, in the latter setting a single malicious user could easily carry out a devastating denial-of-service attack by interacting with the signer once and cheating, thus preventing the signer from issuing any further signatures. Note further that an abort by the user during an execution of the signing protocol is considered cheating, so even transient network failures during an honest execution of the protocol may lead to the same result.

1.1 Our Contributions

Inspired by Pointcheval's transform, we show a new transform for boosting the security of certain blind signature schemes. Our transform has the following advantages compared to Pointcheval's result:

1. As with Pointcheval's transform, our transform boosts security in the following sense: if the original scheme BS is secure for *logarithmically* many executions of the signing protocol, then the transformed scheme CCBS is secure for *polynomially* many executions. Importantly, however, in our case the transformed scheme CCBS does *not* require the signer to stop issuing signatures if cheating is detected.
2. Moreover, if BS is secure for (logarithmically many) *concurrent* executions of the signing protocol, then CCBS is secure for (polynomially

[1] We have identified a bug in Pointcheval's result, in that the transformed scheme does not satisfy blindness. This is easy to fix, though.

many) concurrent executions as well. This is in contrast to Pointcheval's transform, which is not secure for concurrent executions of the signing protocol even if the original scheme is concurrently secure.
3. Our transform can be applied to any blind signature scheme BS constructed in a certain way from a *linear function family* [14], in contrast to Pointcheval's transform that is specific to the Okamoto-Schnorr scheme.[2] In particular, our transform can be applied to the Fiat-Shamir, Okamoto-Guillou-Quisquater, and Okamoto-Schnorr blind signature schemes, all of which can be proven secure (for logarithmically many executions) under standard assumptions in the random-oracle model. Our transform can also be applied to the Schnorr blind signature scheme, which was recently proven secure (for logarithmically many executions) in the algebraic group model [10].
4. As in Pointcheval's transform, the size of signatures in the transformed scheme CCBS is (almost) the same as in the underlying scheme BS.

Overall, then, our work gives the first efficient blind signature schemes that are secure for polynomially many concurrent executions of the signing protocol based on standard assumptions (in the random-oracle model).

1.2 Overview

In this section we give a high-level overview of our transform and its proof of security. Our treatment is deliberately informal, and we refer the reader to Section 3 for details of our scheme. Throughout this section we let BS be a blind signature scheme that is secure for logarithmically many executions of the signing protocol, and for which our transform applies. This in particular means that the signing protocol BS has a three-round structure in which the signer sends the first message. We denote the messages sent in each round of the protocol as R, c, and s, respectively.

Pointcheval's transform. We begin by recalling Pointcheval's transform and its proof of security. (Pointcheval's transform was only defined for the Okamoto-Schnorr scheme, but our work shows that it can be applied to a larger class of schemes.) The basic idea of Pointcheval's transform is to use 1-out-of-2 cut-and-choose to catch (in a limited sense) cheating behavior of a user $\mathcal{U}$. In more detail, the transformed scheme works roughly as follows for a user who wants to obtain a signature on a message m:

[2] Pointcheval states that his transform can be adapted to apply to the Okamoto-Guillou-Quisquater scheme, but does not give details (or a proof).

1. $\mathcal{U}$ runs two parallel executions of BS (we refer to each as a *session*) where the messages to be signed are $\mu_1 = \mathsf{H}'(m, \varphi_1)$ and $\mu_2 = \mathsf{H}'(m, \varphi_2)$, respectively. (Here, H' is a random oracle and φ_1, φ_2 are random strings.) The transformed protocol begins by having $\mathcal{U}$ send a commitment com_1 to μ_1 and its randomness for the first session, and a commitment com_2 to μ_2 and its randomness for the second session. These commitments also rely on a random oracle, which enables extraction in the proof of one-more unforgeability (see below).
2. $\mathcal{S}$ runs two executions of BS to obtain initial messages R_1, R_2, which it sends to $\mathcal{U}$.
3. $\mathcal{U}$ responds with c_1, c_2, which are the second messages of its two executions of BS.
4. $\mathcal{S}$ then chooses a uniform $I \in \{1, 2\}$ and challenges $\mathcal{U}$ to open commitment com_{3-I} and thus demonstrate that it behaved honestly in the corresponding session. If the commitment is opened correctly, then $\mathcal{S}$ sends the final message s_I for the unopened session and $\mathcal{U}$ uses BS to compute a signature on μ_I (which is defined to be a signature on m in the transformed scheme). If $\mathcal{U}$ is caught cheating, then $\mathcal{S}$ aborts and refuses to issue any more signatures (see further discussion below).

It is not difficult to show that the transformed scheme is blind if BS is blind, and so the main challenge is to prove one-more unforgeability of the transformed scheme for polynomially many executions. This is shown by reduction to the one-more unforgeability of BS for logarithmically many executions. The idea of the reduction is as follows. Each time the adversarial user $\mathcal{U}^*$ sends its commitments in the first round of the transformed protocol, the reduction uses the random-oracle queries of $\mathcal{U}^*$ to try to extract the randomness of $\mathcal{U}^*$ for both sessions. If this cannot be done for either session, then $\mathcal{U}^*$ will not succeed in the cut-and-choose (except with negligible probability) and so simulation is easy. If extraction can be done for both sessions, then the reduction will be able to simulate an execution of BS on its own, regardless of the value of I. The remaining case is when the reduction is able to extract the randomness for only one of the two sessions. (In that case, we say $\mathcal{U}^*$ *attempts to cheat*.) $\mathcal{U}^*$ can then succeed in the cut-and-choose with probability $1/2$ (in which case we say $\mathcal{U}^*$ *successfully cheats*), but the reduction will be unable to simulate BS for the unopened session in that case (since it was unable to extract the randomness for that session). Instead, in this case the reduction will interact with the real signer in the underlying scheme BS, forwarding messages in the obvious way between the signer and $\mathcal{U}^*$. (One must show that a forgery by $\mathcal{U}^*$ in the transformed scheme implies that the reduction

can compute a forgery in BS with high probability, but this is irrelevant for the discussion that follows.)

To complete the proof, we must argue that with overwhelming probability the reduction interacts with the signer for the underlying scheme BS only logarithmically many times. Although the formal analysis is quite involved, intuitively this holds because each time $\mathcal{U}^*$ attempts to cheat it is caught with probability 1/2. Thus, the probability that $\mathcal{U}^*$ successfully cheats t times (and thus causes the reduction to interact with the signer t times) is 2^{-t}, and hence t is super-logarithmic in the security parameter with only negligible probability. This highlights why it is essential that the signer must refuse to run any more executions of the signing protocol once it detects cheating: if it did not, then $\mathcal{U}^*$ could attempt to cheat in polynomially many executions and successfully cheat (in expectation) in half of those. Since each instance of successful cheating requires the reduction to interact with the signer in BS, this would mean that the transform would then at best be able to double the number of executions of the signing protocol that can be supported.

Our transform. We follow a template similar to Pointcheval's transform, but since we wish to support an unbounded (polynomial) number of executions of the signing protocol we need to modify things to bound the number of times an adversarial user $\mathcal{U}^*$ can successfully cheat in the cut-and-choose. Our key insight is that we can do this by using 1-out-of-N cut-and-choose, *where N increases with the number of executions*.[3] That is, we consider following Pointcheval's general approach, but in the $(N-1)$st execution of the transformed protocol we instead use *1-out-of-N cut-and-choose* on the underlying scheme BS. The probability that $\mathcal{U}^*$ can successfully cheat in the $(N-1)$st execution is now $1/N$, so even if $\mathcal{U}^*$ attempts to cheat in every one of its p executions of the transformed protocol, the expected number of times it can successfully cheat is

$$\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{p+1} < \ln(p+1).$$

An appropriate concentration bound implies that for any polynomial p (which bounds the number of executions $\mathcal{U}^*$ runs) the probability that $\mathcal{U}^*$ successfully cheats super-logarithmically many times is negligible.

We remark that although the complexity of our signing protocol grows, the other parameters of the scheme—namely, the size of the keys, the size of the signatures, and the cost of verification—are fixed and essentially the

[3] In fact, it suffices to have N depend linearly on *the number of times cheating is detected* (cf. Section 3.3), but we ignore this optimization in our informal overview.

same as in the original scheme BS. The round complexity of our signing protocol is also constant.

Comparison with generic constructions. The complexity of the signing protocol in our scheme is (roughly) N times the complexity of the signing protocol in BS, where N is linear in the number of executions. (But see footnote 3.) One might therefore wonder whether our scheme is better than a generic construction of a blind signature scheme where signing involves running a secure two-party computation (2PC) protocol for computing any (standard) signature [18]. (The details are more complex, but are unimportant for the purposes of this discussion.) This generic approach, however, has several drawbacks compared to our scheme.

1. Such a generic construction would not be concurrently secure without additional complexity and/or without assuming some form of trusted setup or non-standard hardness assumptions.
2. The efficiency of such a generic construction (even restricting attention to sequential security) is unclear, but we conservatively estimate (based on work of Jayaraman et al. [17]) that secure 2PC of an Okamoto-Schnorr signature at the 96-bit security level would have communication complexity at least $10^9\times$ that of our protocol when $N = 2$. Thus, our signing protocol would have better communication complexity for $N < 10^9$. The comparison would be even more favorable for our scheme at higher security levels.
3. Efficient and provably secure signature schemes rely on the random-oracle model, but secure computation of a signature would require a circuit for the hash function instantiating the random oracle. Security of the resulting protocol in this case is unclear.

Notwithstanding the above, we note that the generic approach does have two advantages compared to our scheme: First, the signer is *stateless*, whereas in our scheme the signer is required to maintain (a small amount of) state. Second, the signatures produced in the generic scheme are identical to signatures in the underlying scheme, whereas in our scheme (as in Pointcheval's) signatures include an additional random value.

Another generic construction of blind signatures is given by Fischlin [7]. Roughly, in his scheme the signer signs a commitment to m; the signature computed by the user consists of a non-interactive zero-knowledge proof of knowledge (NIZKPoK) of a signed commitment on m. Fischlin's scheme is concurrently secure. Nevertheless, the signatures produced by his scheme are much larger than standard signatures (even if SNARKs are used for the NIZKPoK); also, as with the generic construction discussed above,

the concrete efficiency of this approach—especially if one wants to rely on standard assumptions—is unclear.

2 Preliminaries

We give definitions for blind signature schemes and linear function families, and recall a generic construction of blind signature schemes secure for logarithmically many signatures from the latter.

Notation. We denote the security parameter by κ. We write $a \leftarrow S$ to denote that a is drawn uniformly from set S. For a randomized algorithm $\mathcal{A}$ we write $y \leftarrow \mathcal{A}(x)$ to denote that $\mathcal{A}$ returns y when run on input x. For a positive integer N we let $[N] = \{1, \ldots, N\}$.

2.1 Blind Signatures

We define the syntax of a blind signature scheme, followed by definitions of *blindness* and *one-more unforgeability*.

Definition 1 (Blind signature scheme). *A* blind signature scheme *is a tuple of algorithms* $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ *such that:*

- *The* key-generation algorithm Gen *takes as input the security parameter* 1^κ *and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$ *as well as initial state* $\mathsf{st}_{\mathcal{S}}$.
- *The* signer algorithm $\mathcal{S}$ *is an interactive algorithm that takes as input a secret key* sk *and can (atomically) read/write a global variable* $\mathsf{st}_{\mathcal{S}}$ *during its execution. At the end of its execution, it outputs either* $\perp$ *(indicating an abort) or 1 (indicating a successful execution). When* $\mathcal{S}$ *outputs 1 at the end of an execution we call that execution* complete.
- *The* user algorithm $\mathcal{U}$ *is an interactive algorithm that takes as input a public key* pk *and a message* m. *At the end of its execution, it either outputs* $\perp$ *(indicating an abort) or a signature* σ.
- *The* verification algorithm Vrfy *takes as input a public key* pk, *a message* m, *and a signature* σ, *and outputs a bit* b *indicating "accept"* $(b = 1)$ *or "reject"* $(b = 0)$.

We require perfect correctness: for all $(\mathsf{pk}, \mathsf{sk})$ *output by* Gen *and all messages* m, *if* $\mathcal{S}(\mathsf{sk})$ *and* $\mathcal{U}(\mathsf{pk}, m)$ *execute the protocol honestly then* $\mathcal{S}$ *outputs 1 and the signature* σ *output by* $\mathcal{U}$ *satisfies* $\mathsf{Vrfy}_{\mathsf{pk}}(m, \sigma) = 1$.

The above definition allows the signer to be *stateful*, and this will be the case for our construction. For simplicity, however, we leave the state implicit in our definitions.

Definition 2 (Blindness). *For blind signature scheme* $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ *and an adversary* $\mathcal{A}$*, consider the following experiment:*

1. $\mathcal{A}(1^\kappa)$ *outputs a public key* pk *and a pair of messages* m_0, m_1*. A uniform bit* $b \leftarrow \{0,1\}$ *is also chosen.*
2. *Run* $\mathcal{A}^{\mathcal{U}(\mathsf{pk},m_b),\mathcal{U}(\mathsf{pk},m_{1-b})}(1^\kappa)$*, where* $\mathcal{A}$ *may run one execution with each of its oracles, but may arbitrarily interleave its oracle calls.*
3. *When both executions are completed, let* σ_b, σ_{1-b} *be the (local) outputs of the respective oracles. If* $\sigma_0 = \perp$ *or* $\sigma_1 = \perp$*, then* $\mathcal{A}$ *is given* $\perp$*; otherwise,* $\mathcal{A}$ *is given* σ_0, σ_1*. Finally,* $\mathcal{A}$ *outputs* b'*.*
4. $\mathcal{A}$ succeeds *iff* $b' = b$*.*

The advantage *of* $\mathcal{A}$ *is the probability that it succeeds in the above experiment minus* $1/2$*. We say* BS *satisfies* blindness *if for all probabilistic polynomial-time* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *is negligible.*

The above definition allows the malicious signer to use a maliciously generated public key pk. A weaker definition that is often considered in the literature assumes pk is generated honestly using the key-generation algorithm of BS. We refer to the corresponding notion of security as *blindness for honestly generated keys.*

Definition 3 (One-more unforgeability). *Let* $\ell : \mathbb{N} \to \mathbb{N}$*. For blind signature scheme* $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ *and adversary* $\mathcal{A}$*, consider the following experiment:*

1. *Generate keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$*.*
2. *Run* $\mathcal{A}^{\mathcal{S}(\mathsf{sk})}(\mathsf{pk})$*, where* $\mathcal{A}$ *may initiate an arbitrary number of executions with its oracle (arbitrarily interleaving its oracle calls), so long as* $\mathcal{S}$ *completes at most* $\ell = \ell(\kappa)$ *of those executions.*
3. $\mathcal{A}$ *outputs* $\ell + 1$ *message-signature pairs* $(m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1})$*.*
4. $\mathcal{A}$ succeeds *if all* $\{m_i\}$ *are distinct and* $\mathsf{Vrfy}_{\mathsf{pk}}(m_i, \sigma_i) = 1$ *for all* i*.*

BS *satisfies* ℓ-one-more unforgeability *if for all probabilistic polynomial-time* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *succeeds is negligible.* BS *satisfies* one-more unforgeability *if it is* ℓ*-one-more unforgeable for all polynomial* ℓ*.*

The above definition allows concurrent executions of the signing protocol. A weaker definition considers only *sequential* executions. (Formally, this would mean that if $\mathcal{A}$ initiates a new session with its oracle $\mathcal{S}(\mathsf{sk})$, then the oracle terminates the currently active session.) We refer to the corresponding notion of security as *sequential* (ℓ*-)one-more unforgeability.*

2.2 Linear Function Families

A *linear function family* [14] is a tuple of probabilistic polynomial-time algorithms $\mathsf{LF} = (\mathsf{PGen}, \mathsf{F}, \Psi)$. The *parameter-generation algorithm* PGen takes as input the security parameter 1^κ and returns parameters par that, in particular, define abelian groups $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{R}$ (written additively), with $|\mathcal{S}|, |\mathcal{R}| \geq 2^\kappa$. (These correspond to a set of "scalars," a "domain," and a "range," respectively). We require the existence of a "scalar multiplication" map $\cdot : \mathcal{S} \times \mathcal{D} \to \mathcal{D}$ such that for all $s \in \mathcal{S}$ and $x, x' \in \mathcal{D}$ we have $s \cdot (x + x') = s \cdot x + r \cdot x'$ and $0 \cdot x = s \cdot 0 = 0$. (We stress that it is *not* necessarily the case that $(s + s') \cdot x = s \cdot x + s' \cdot x$; see further below.) We also require a map $\cdot : \mathcal{S} \times \mathcal{R} \to \mathcal{R}$ with analogous properties. Finally, it should be possible to efficiently sample uniform elements from $\mathcal{S}$ and $\mathcal{D}$.

For concreteness, the reader may want to keep in mind the linear function family where $\mathcal{S} = \mathcal{D} = \mathbb{Z}_q$ and $\mathcal{R}$ is a cyclic group $\mathbb{G}$ of prime order q (written multiplicatively). (Looking ahead to the next section, this is the linear function family that underlies the Schnorr blind signature scheme.) We have scalar multiplication maps $s \cdot x = s \cdot x \pmod q$ for $s, x \in \mathbb{Z}_q$ and $s \cdot g = g^s$ for $g \in \mathbb{G}$. We give other examples of linear function families in Appendix A.

The linear *evaluation function* $\mathsf{F} = \mathsf{F}_{\mathsf{par}}$ takes as input a point $x \in \mathcal{D}$ and returns an element $y \in \mathcal{R}$. We require that for all $s \in \mathcal{S}$ and $x, y \in \mathcal{D}$, it holds that $\mathsf{F}(s \cdot x + y) = s \cdot \mathsf{F}(x) + \mathsf{F}(y)$. We also assume that F has has min-entropy at least κ, i.e., that the min-entropy of $\mathsf{F}(x)$ is at least κ when x is uniform in $\mathcal{D}$. We say LF *has a pseudo torsion-free element in the kernel* if there exists $z^* \in \mathcal{D}$ such that (1) $\mathsf{F}(z^*) = 0$, and (2) for all distinct $s, s' \in \mathcal{S}$, we have $s \cdot z^* \neq s' \cdot z^*$. (Note this implies $z^* \neq 0$.)

Returning to our running example: if par includes a uniformly selected generator $g \in \mathbb{G}$ we can define $\mathsf{F}(x) = g^x$, which is clearly linear. In this example, however, the linear function family does not have a pseudo torsion-free element in the kernel.

The *distributor function* $\Psi = \Psi_{\mathsf{par}}$ takes as input an element $y \in \mathcal{R}$ and points $s, s' \in \mathcal{S}$, and outputs a point in $\mathcal{D}$. For all y in the range of F and $s, s' \in \mathcal{S}$, we require

$$(s + s') \cdot y = s \cdot y + s' \cdot y + \mathsf{F}(\Psi(y, s, s')).$$

Intuitively, the distributor function Ψ outputs a *correction term* that corrects for the fact that the group operation in $\mathcal{S}$ may not distribute over $\mathcal{R}$. (Thus, the distributor function is the zero function whenever the scalar multiplication map does distribute, as in our running example).

We define two security properties for linear function families.

Definition 4 (Preimage resistance). *For a linear function family* LF *and an adversary* $\mathcal{A}$ *consider the following experiment:*

1. *Generate parameters* $\mathsf{par} \leftarrow \mathsf{PGen}(1^\kappa)$ *and choose* $x \leftarrow \mathcal{D}$.
2. *Run* $\mathcal{A}(\mathsf{par}, \mathsf{F}(x))$ *to obtain* $x' \in \mathcal{D}$.
3. $\mathcal{A}$ succeeds *if* $\mathsf{F}(x') = \mathsf{F}(x)$.

LF *is* preimage resistant *if for all probabilistic polynomial-time* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *succeeds is negligible.* LF *is* $(t, \epsilon_{\mathsf{PRE}})$-preimage resistant *if every* $\mathcal{A}$ *running in time at most* t *succeeds with probability at most* ϵ_{PRE} *in the above experiment.*

Definition 5 (Collision resistance). *For a linear function family* LF *and an adversary* $\mathcal{A}$ *consider the following experiment:*

1. *Generate parameters* $\mathsf{par} \leftarrow \mathsf{PGen}(1^\kappa)$.
2. *Run* $\mathcal{A}(\mathsf{par})$ *to obtain* $x_1, x_2 \in \mathcal{D}$.
3. $\mathcal{A}$ succeeds *if* $\mathsf{F}(x_1) = \mathsf{F}(x_2)$ *and* $x_1 \neq x_2$.

LF *is* collision resistant *if for all probabilistic polynomial-time* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *succeeds is negligible.* LF *is* $(t, \epsilon_{\mathsf{PRE}})$-collision resistant *if every* $\mathcal{A}$ *running in time at most* t *succeeds with probability at most* ϵ_{CR} *in the above experiment.*

The linear function family in our running example is preimage resistant if the discrete-logarithm problem is hard in $\mathbb{G}$, and trivially collision resistant (since F is a bijection).

2.3 Blind Signatures from Linear Function Families

Hauck et al. [14] showed that several blind signature schemes from the literature, including the Schnorr, Okamoto-Schnorr, Fiat-Shamir, and Okamoto-Guillou-Quisquater schemes, can be viewed as being derived from linear function families. We recall their generic construction of a blind signature scheme $\mathsf{BS}[\mathsf{LF}]$ from a linear function family LF. The secret key is a uniform element $\mathsf{sk} \leftarrow \mathcal{D}$ and the corresponding public key is $\mathsf{pk} := \mathsf{F}(\mathsf{sk})$. The signing protocol, where $\mathcal{U}$ holds a message m, proceeds as follows. (See Figure 1.) In the first step, $\mathcal{S}$ samples $r \leftarrow \mathcal{D}$ and sends $R := \mathsf{F}(r)$ to $\mathcal{U}$. Then $\mathcal{U}$ samples blinding parameters $\alpha \leftarrow \mathcal{D}$ and $\beta \leftarrow \mathcal{S}$ that it uses to compute a "blinded commitment" $R' := R + \mathsf{F}(\alpha) + \beta \cdot \mathsf{pk}$, computes $c' := \mathsf{H}(m, R')$, and sends the blinded challenge $c := c' + \beta$ to $\mathcal{S}$. In the last round of the protocol, $\mathcal{S}$ replies with $s := r + c \cdot \mathsf{sk}$, and $\mathcal{U}$ checks that $\mathsf{F}(s) = R + c \cdot \mathsf{pk}$. (If not, $\mathcal{U}$ aborts). Finally, $\mathcal{U}$

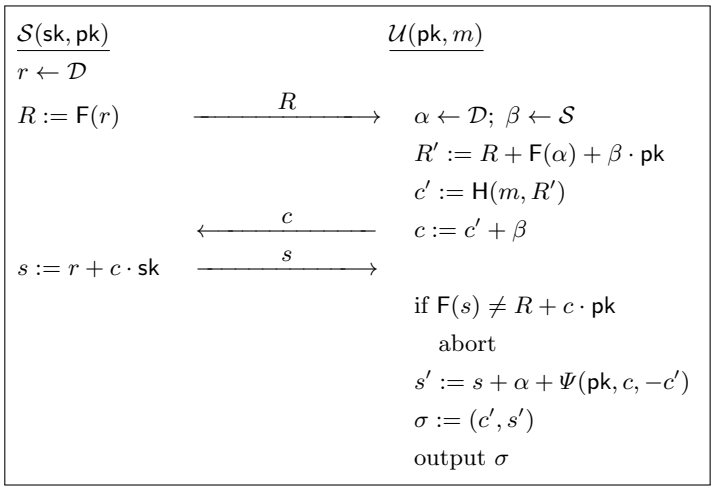

Fig. 1. The signing protocol for blind signature scheme $\mathsf{BS}[\mathsf{LF}]$, where LF is a linear function family and $\mathsf{H} : \{0,1\}^* \to \mathcal{S}$ is modeled as a random oracle.

computes $s' := s + \alpha + \Psi(\mathsf{pk}, c, -c')$ and outputs the signature $\sigma := (c', s')$. Verification is done by checking whether $c' = \mathsf{H}(m, \mathsf{F}(s') - c' \cdot \mathsf{pk})$.

If both parties follow the protocol honestly, then

$$s' = s + \alpha + \Psi(\mathsf{pk}, c, -c') = c \cdot \mathsf{sk} + r + \alpha + \Psi(\mathsf{pk}, c, -c').$$

Thus,

$$\begin{aligned}
\mathsf{F}(s') - c' \cdot \mathsf{pk} &= \mathsf{F}(c \cdot \mathsf{sk} + r + \alpha + \Psi(\mathsf{pk}, c, -c')) - c' \cdot \mathsf{pk} \\
&= c \cdot \mathsf{pk} - c' \cdot \mathsf{pk} + \mathsf{F}(\Psi(\mathsf{pk}, c, -c')) + \mathsf{F}(r) + \mathsf{F}(\alpha) \\
&= (c - c') \cdot \mathsf{pk} + \mathsf{F}(r) + \mathsf{F}(\alpha) \\
&= \beta \cdot \mathsf{pk} + R + \mathsf{F}(\alpha) \ = \ R',
\end{aligned}$$

and so verification succeeds. This demonstrates correctness of the scheme.

Hauck et al. [14] show that $\mathsf{BS}[\mathsf{LF}]$ is statistically blind for honestly generated keys. Their proof extends to full blindness (i.e., even for maliciously generated keys) when $\mathsf{BS}[\mathsf{LF}]$ corresponds to the Schnorr or Okamoto-Schnorr blind signature scheme. More interestingly, $\mathsf{BS}[\mathsf{LF}]$ is ℓ-one-more unforgeable for any $\ell = \mathcal{O}(\log \kappa)$:

Theorem 1 ([14]). *Let* $\mathsf{LF} = (\mathsf{PGen}, \mathsf{F}, \Psi)$ *be a collision-resistant linear function family with a torsion-free element in the kernel, and let* H *be*

modeled as a random oracle. Then $\mathsf{BS}[\mathsf{LF}]$ *is* ℓ*-one-more unforgeable for any* $\ell = \mathcal{O}(\log \kappa)$*.*

Concretely, if there is an adversary against ℓ*-one-more unforgeability of* $\mathsf{BS}[\mathsf{LF}]$ *that runs in time* t*, initiates at most* p *executions, makes at most* q_{H} *queries to* H*, and has success probability* ϵ*, then there is an adversary against collision resistance of* LF *running in time* $t' = 2t$ *and having success probability at least*

$$\epsilon' = \Omega\left(\left(\frac{\epsilon}{2} - \frac{(q_{\mathsf{H}} \cdot (p-\ell))^{\ell+1}}{2^{\kappa}}\right)^3 \cdot \frac{1}{q_{\mathsf{H}}^2 \cdot \ell^3}\right).$$

Theorem 1 requires LF to have a pseudo torsion-free element in the kernel, and thus applies to the Okamoto-Schnorr, Okamoto-Guillou-Quisquater, and Fiat-Shamir blind signature schemes. (See Appendix A.) However, there are examples of other schemes matching the template of Figure 1 that can be proven secure without relying on Theorem 1. In particular, recent work [10] has shown that the Schnorr blind signature scheme is ℓ-one-more unforgeable for any $\ell = \mathcal{O}(\log \kappa)$ in the algebraic group model under the one-more discrete logarithm assumption.

3 Boosting Security of Blind Signatures

We now present our cut-and-choose blind signature scheme $\mathsf{CCBS}[\mathsf{LF}]$. (We assume the reader has read the informal overview in Section 1.2.) As in $\mathsf{BS}[\mathsf{LF}]$, the secret key is a uniform element $\mathsf{sk} \leftarrow \mathcal{D}$ and the corresponding public key is $\mathsf{pk} := \mathsf{F}(\mathsf{sk})$. Now, however, the signer $\mathcal{S}$ additionally maintains a counter N that is initialized to 1. The signing protocol for a message m then proceeds as follows (cf. Figure 2):

1. $\mathcal{S}$ atomically increments its counter (see further discussion below) and sends the updated counter N to the user $\mathcal{U}$.
2. Informally, $\mathcal{U}$ runs N executions of $\mathsf{BS}[\mathsf{LF}]$, using the "message" $\mu_i = \mathsf{H}'(m, \varphi_i)$ in the ith execution. (We refer to each execution of the underlying scheme $\mathsf{BS}[\mathsf{LF}]$ as a *session*.) Here, $\varphi_i \in \{0,1\}^{\kappa}$ is a uniform string and H' is modeled as a random oracle. Thus, in the first step, for $i \in [N]$ the user chooses randomness α_i, β_i for the ith session of $\mathsf{BS}[\mathsf{LF}]$ and sends a commitment $\mathsf{com}_i = \mathsf{H}'(\alpha_i, \beta_i, \mu_i, \gamma_i)$, where $\gamma_i \in \{0,1\}^{\kappa}$ is another uniform string.
3. $\mathcal{S}$ runs N sessions of $\mathsf{BS}[\mathsf{LF}]$ to obtain initial messages $R_1, \ldots, R_N$, which it sends to $\mathcal{U}$. In response, $\mathcal{U}$ computes $c_1, \ldots, c_N$ using $\mathsf{BS}[\mathsf{LF}]$ and the randomness it chose earlier.

4. $\mathcal{S}$ then chooses a uniform index $I \in [N]$ and sends it to $\mathcal{U}$. The user reveals $(\alpha_i, \beta_i, \mu_i, \gamma_i)$ for all $i \neq I$ (thus opening all but its Ith commitment), and $\mathcal{S}$ verifies that $\mathcal{U}$ behaved honestly in all the opened sessions. If cheating is detected, then $\mathcal{S}$ aborts the entire execution.
5. If $\mathcal{U}$ behaved honestly in the opened sessions, $\mathcal{S}$ uses $\mathsf{BS}[\mathsf{LF}]$ to compute a response $s := r_I - c_I \cdot \mathsf{sk}$ for the Ith (unopened) session.
6. $\mathcal{U}$ computes a signature (c'_I, s'_I) on μ_I using $\mathsf{BS}[\mathsf{LF}]$. It then outputs the signature (c'_I, s'_I, φ_I) on m.

A signature $\sigma = (c', s', \varphi)$ on a message m is verified by checking that (c', s') is a valid signature on $\mu = \mathsf{H}'(m, \varphi)$ in the underlying scheme $\mathsf{BS}[\mathsf{LF}]$.

The counter is used to ensure that each execution of the protocol uses a different value for the cut-and-choose parameter N. (In Section 3.3, we show that it is possible to do better.) In the concurrent setting, it is therefore important to ensure that the counter is incremented atomically so that this property holds across all the concurrent executions.

Theorem 2. *Let* LF *be a linear function family that is preimage resistant and let* H, H' *be modeled as random oracles. If* $\mathsf{BS}[\mathsf{LF}]$ *satisfies blindness (for honestly generated keys), then* $\mathsf{CCBS}[\mathsf{LF}]$ *satisfies blindness (for honestly generated keys). If* $\mathsf{BS}[\mathsf{LF}]$ *is (sequentially)* ℓ*-one-more unforgeable for any* $\ell \in \mathcal{O}(\log \kappa)$*, then* $\mathsf{CCBS}[\mathsf{LF}]$ *is (sequentially)* ℓ*-one-more unforgeable for any* $\ell = \mathsf{poly}(\kappa)$*.*

We separately consider blindness and one-more unforgeability in the sections that follow.

3.1 Blindness

This section is dedicated to a proof of the following:

Theorem 3. *Let* H' *be modeled as a random oracle. If* $\mathsf{BS}[\mathsf{LF}]$ *satisfies blindness (resp., blindness for honestly generated keys), then* $\mathsf{CCBS}[\mathsf{LF}]$ *satisfies blindness (resp., blindness for honestly generated keys).*

Concretely, if there is an adversary $\mathcal{A}$ *against blindness of* $\mathsf{CCBS}[\mathsf{LF}]$ *that runs in time* t*, makes at most* $q_{\mathsf{H}'}$ *queries to* H'*, uses counters* $N^{\mathsf{L}}, N^{\mathsf{R}}$ *in its executions with the user, and has advantage* ϵ*, then there is an adversary* $\mathcal{B}$ *against blindness of* $\mathsf{BS}[\mathsf{LF}]$ *that runs in time* $t' \approx t$ *and has advantage at least* $\frac{1}{N^{\mathsf{L}} \cdot N^{\mathsf{R}}} \cdot \left(\epsilon - \frac{2 \cdot (N^{\mathsf{L}} + N^{\mathsf{R}}) \cdot q_{\mathsf{H}'}}{2^{\kappa}}\right)$.

Proof. We consider the case of blindness for maliciously generated keys, but the proof holds also for honestly generated keys. Fix an adversary $\mathcal{A}$

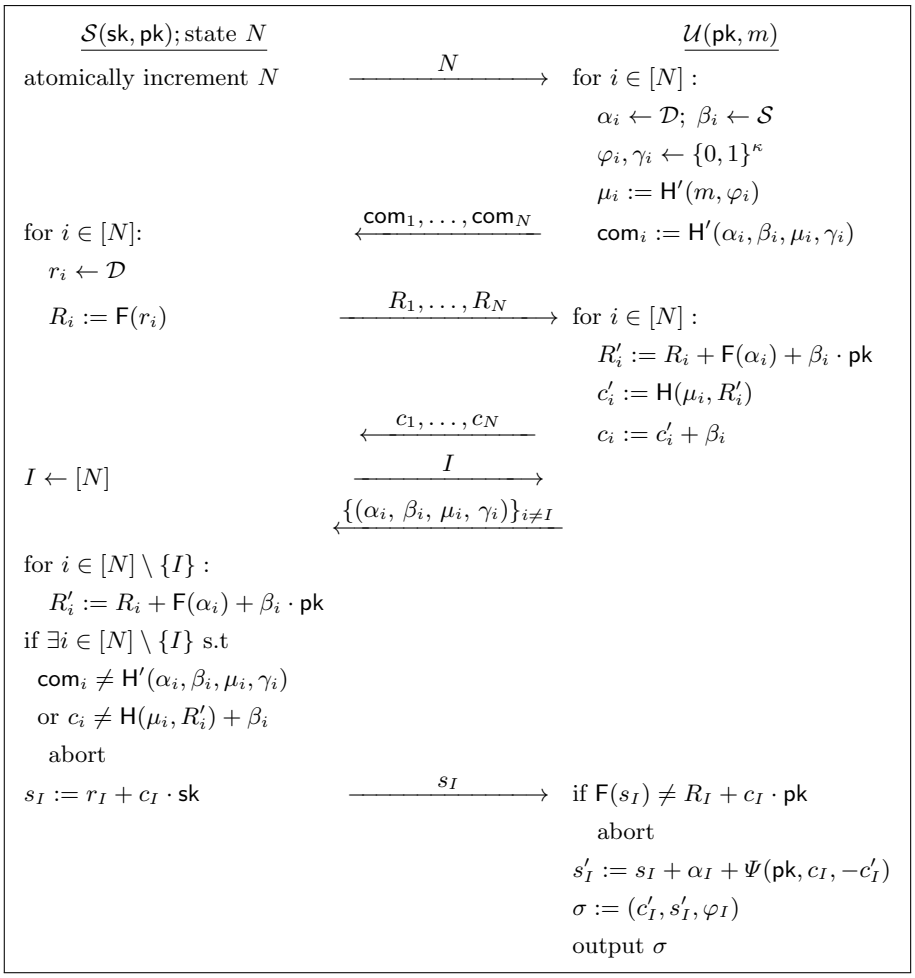

Fig. 2. The signing protocol for blind signature scheme $\mathsf{CCBS}[\mathsf{LF}]$, where LF is a linear function family and $\mathsf{H} : \{0,1\}^* \to \mathcal{S}$, $\mathsf{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ are modeled as random oracles.

attacking blindness of $\mathsf{CCBS}[\mathsf{LF}]$, let $\mathsf{Succ}_{\mathcal{A}}$ be the event that $\mathcal{A}$ succeeds, and let $\epsilon = \epsilon(\kappa)$ be the advantage of $\mathcal{A}$ so that $\Pr[\mathsf{Succ}_{\mathcal{A}}] = \frac{1}{2} + \epsilon$. In an execution of the experiment used to define blindness of $\mathsf{CCBS}[\mathsf{LF}]$, the adversary interacts with two instances of $\mathcal{U}$; we use superscripts L, R to denote variables used in the left and right interactions, respectively. Let $N^{\mathsf{L}}, N^{\mathsf{R}}$ be the values of the counters that $\mathcal{A}$ sends in its two interactions with $\mathcal{U}$, and let Bad be the event that $\mathcal{A}$ makes any H'-queries of the following form:

- $\mathsf{H}'(\star, \varphi_i^{\mathsf{L}})$ for $i = 1, \ldots, N^{\mathsf{L}}$ (resp., $\mathsf{H}'(\star, \varphi_i^{\mathsf{R}})$ for $i = 1, \ldots, N^{\mathsf{R}}$). (In the case of $\varphi_{I^{\mathsf{L}}}^{\mathsf{L}}, \varphi_{I^{\mathsf{R}}}^{\mathsf{R}}$, this must occur *before* those values are revealed to $\mathcal{A}$ as part of the signatures output by $\mathcal{U}$.)
- $\mathsf{H}'(\star, \star, \star, \gamma_i^{\mathsf{L}})$ for $i = 1, \ldots, N^{\mathsf{L}}$ (resp., $\mathsf{H}'(\star, \star, \star, \gamma_i^{\mathsf{R}})$ for $i = 1, \ldots, N^{\mathsf{R}}$) *before* γ_i^{L} (resp., γ_i^{R}) is sent by $\mathcal{U}$ to $\mathcal{A}$ in round 6.

In particular, since $\gamma_{I^{\mathsf{L}}}^{\mathsf{L}}$ (resp., $\gamma_{I^{\mathsf{R}}}^{\mathsf{R}}$) is not sent in round 6, event Bad occurs if $\mathcal{A}$ makes a query of the form $\mathsf{H}'(\star, \star, \star, \gamma_{I^{\mathsf{L}}}^{\mathsf{L}})$ (resp., $\mathsf{H}'(\star, \star, \star, \gamma_{I^{\mathsf{R}}}^{\mathsf{R}})$) at any point during the experiment. If $q_{\mathsf{H}'}$ denotes the number of queries $\mathcal{A}$ makes to H', it is immediate that

$$\Pr[\mathsf{Succ}_{\mathcal{A}} \wedge \overline{\mathsf{Bad}}] \geq \frac{1}{2} + \epsilon - \frac{2 \cdot (N^{\mathsf{L}} + N^{\mathsf{R}}) \cdot q_{\mathsf{H}'}}{2^{\kappa}} .$$

We now construct an adversary $\mathcal{B}$ attacking blindness of $\mathsf{BS}[\mathsf{LF}]$. Intuitively, $\mathcal{B}$ simulates $\mathcal{A}$'s oracle calls by locally running all-but-one of the sessions of $\mathsf{BS}[\mathsf{LF}]$ honestly, and using its own oracles (which correspond to two executions of the user algorithm for $\mathsf{BS}[\mathsf{LF}]$) to simulate the remaining instance. $\mathcal{B}$ works as follows:

1. Throughout, H'-oracle calls made by $\mathcal{A}$ are handled in the natural way.[4] If event Bad occurs, $\mathcal{B}$ aborts and outputs a uniform bit.
2. $\mathcal{B}$ runs $\mathcal{A}$ to obtain pk, m_0, m_1. It then chooses uniform $\mu_0, \mu_1 \in \{0,1\}^{\kappa}$ and outputs $\mathsf{pk}, \mu_0, \mu_1$.
3. $\mathcal{B}$ handles the interaction of $\mathcal{A}$ with its left oracle by playing the role of $\mathcal{U}$ in an execution of $\mathsf{CCBS}[\mathsf{LF}]$, as follows:
   (a) When $\mathcal{A}$ sends N^{L}, choose uniform $i^{\mathsf{L}} \in [N^{\mathsf{L}}]$ and uniform values $\gamma_{i^{\mathsf{L}}}^{\mathsf{L}}, \varphi_{i^{\mathsf{L}}}^{\mathsf{L}}, \mathsf{com}_{i^{\mathsf{L}}}^{\mathsf{L}} \in \{0,1\}^{\kappa}$. For $i \in [N^{\mathsf{L}}] \setminus \{i^{\mathsf{L}}\}$, run $\mathcal{U}$ honestly to obtain $\mathsf{com}_i^{\mathsf{L}}$. Send $\mathsf{com}_1^{\mathsf{L}}, \ldots, \mathsf{com}_{N^{\mathsf{L}}}^{\mathsf{L}}$ to $\mathcal{A}$.
   (b) When $\mathcal{A}$ sends $R_1^{\mathsf{L}}, \ldots, R_{N^{\mathsf{L}}}^{\mathsf{L}}$, then $\mathcal{B}$ forwards $R_{i^{\mathsf{L}}}^{\mathsf{L}}$ to its own left oracle to receive response $c_{i^{\mathsf{L}}}^{\mathsf{L}}$. For $i \in [N^{\mathsf{L}}] \setminus \{i^{\mathsf{L}}\}$, it runs $\mathcal{U}$ honestly to obtain c_i^{L}, and then sends $c_1^{\mathsf{L}}, \ldots, c_{N^{\mathsf{L}}}^{\mathsf{L}}$ to $\mathcal{A}$.

[4] We do not need to model H as a random oracle; our proof holds as long as $\mathsf{BS}[\mathsf{LF}]$ is secure when using H. For this reason we do not mention how calls to H are handled.

(c) When $\mathcal{A}$ sends I^{L}, then $\mathcal{B}$ aborts and outputs a uniform bit if $I^{\mathsf{L}} \neq i^{\mathsf{L}}$. Otherwise, it responds in the natural way.
(d) When $\mathcal{A}$ sends the final message $s^{\mathsf{L}}_{I^{\mathsf{L}}}$, then $\mathcal{B}$ forwards this to its own left oracle.

$\mathcal{B}$ handles the interaction of $\mathcal{A}$ with its right oracle in an exactly analogous manner.

4. When $\mathcal{B}$ is given the output of its own oracles, it does the following. If the output was $\perp$, it gives $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ is given signature (c'_0, s'_0) on μ_0 and signature (c'_1, s'_1) on μ_1; it gives $(c'_0, s'_0, \varphi^{\mathsf{L}}_{i^{\mathsf{L}}})$ and $(c'_1, s'_1, \varphi^{\mathsf{R}}_{i^{\mathsf{R}}})$ to $\mathcal{A}$ and programs $\mathsf{H}'(m_0, \varphi^{\mathsf{L}}_{i^{\mathsf{L}}}) = \mu_0$ and $\mathsf{H}'(m_1, \varphi^{\mathsf{R}}_{i^{\mathsf{R}}}) = \mu_1$. Finally, it outputs whatever bit is output by $\mathcal{A}$.

First observe that the probability of event Bad is unchanged in the above. Let Guess be the event that $I^{\mathsf{L}} = i^{\mathsf{L}}$ and $I^{\mathsf{R}} = i^{\mathsf{R}}$. If Bad does not occur by the time $\mathcal{A}$ sends the latter of I^{L} or I^{R}, then the view of $\mathcal{A}$ at that point is independent of $i^{\mathsf{L}}, i^{\mathsf{R}}$ and so $\Pr[\mathsf{Guess}] = 1/N^L N^{\mathsf{R}}$. Furthermore, if Guess occurs and Bad does not occur then the simulation provided by $\mathcal{B}$ is perfect, and $\mathcal{B}$ succeeds iff $\mathcal{A}$ succeeds. Letting $\mathsf{Succ}_{\mathcal{B}}$ be the event that $\mathcal{B}$ succeeds, we thus have

$$\begin{aligned}\Pr[\mathsf{Succ}_{\mathcal{B}}] &= \frac{1}{2} \cdot \Pr[\overline{\mathsf{Guess}} \vee \mathsf{Bad}] + \Pr[\mathsf{Succ}_{\mathcal{A}} \wedge \mathsf{Guess} \wedge \overline{\mathsf{Bad}}] \\ &\geq \frac{1}{2} + \frac{1}{N^{\mathsf{L}} \cdot N^{\mathsf{R}}} \cdot \left(\epsilon - \frac{2 \cdot (N^{\mathsf{L}} + N^{\mathsf{R}}) \cdot q_{\mathsf{H}'}}{2^{\kappa}} \right).\end{aligned}$$

Since the advantage of $\mathcal{B}$ must be negligible (by blindness of $\mathsf{BS}[\mathsf{LF}]$), and $N^{\mathsf{L}}, N^{\mathsf{R}}, q_{\mathsf{H}'}$ are polynomial,[5] we conclude that ϵ must be negligible.

3.2 One-More Unforgeability

In this section we show:

Theorem 4. *Let* LF *be a linear function family that is preimage resistant and let* H, H' *be modeled as random oracles. If* $\mathsf{BS}[\mathsf{LF}]$ *is (sequentially)* ℓ*-one-more unforgeable for any* $\ell \in \mathcal{O}(\log \kappa)$*, then* $\mathsf{CCBS}[\mathsf{LF}]$ *is (sequentially)* ℓ*-one-more unforgeable for any* $\ell = \mathsf{poly}(\kappa)$*.*

Concretely, assume LF *is* $(t, \epsilon_{\mathsf{PRE}})$*-preimage resistant and there is an adversary against (sequential)* ℓ*-one-more unforgeability of* $\mathsf{CCBS}[\mathsf{LF}]$ *that runs in time* t*, initiates* p *executions, makes at most* q_{H} *queries to* H

[5] Technically, we can enforce that $N^{\mathsf{L}}, N^{\mathsf{R}}$ are polynomial by requiring the counter N sent by $\mathcal{S}$ to be represented in unary (so $N^{\mathsf{L}}, N^{\mathsf{R}}$ are bounded by the running time of $\mathcal{A}$). In practice one might fix a large polynomial bound B and require $N \leq B$.

and $q_{\mathsf{H}'}$ queries to H', and has success probability ϵ. Then there is an adversary against (sequential) λ-one-more unforgeability of $\mathsf{BS}[\mathsf{LF}]$, where $\lambda = 3\ln(p+1) + \ln(2/\epsilon)$, that runs in time $t' \approx t$, initiates p executions, makes at most q_{H} queries to H, and has success probability at least

$$\frac{\epsilon}{2} - \frac{q_{\mathsf{H}'}^2 + p \cdot q_{\mathsf{H}'} + p^2 \cdot (p^2 + q_{\mathsf{H}})}{2^\kappa} - p \cdot \epsilon_{\mathsf{PRE}}.$$

Proof. Let $\mathcal{A}$ be an adversary attacking the one-more unforgeability of $\mathsf{CCBS}[\mathsf{LF}]$ and having success probability ϵ. We let $q_{\mathsf{H}}, q_{\mathsf{H}'}$ denote the number of queries $\mathcal{A}$ makes to H, H', respectively, let ℓ denote the number of complete executions of the signing protocol run by $\mathcal{A}$, and let p denote the total number of executions of the signing protocol by $\mathcal{A}$, including ones that are aborted early by $\mathcal{S}$. (These are all polynomial in the security parameter, but we leave this dependence implicit.) For simplicity, we make some assumptions about the behavior of $\mathcal{A}$ that are without significant loss of generality; specifically, we assume that if $\mathcal{A}$ sends $\alpha_i, \beta_i, \mu_i, \gamma_i$ during an execution of the signing protocol where the corresponding message from the signer was R_i then it had previously queried $\mathsf{H}'(\alpha_i, \beta_i, \mu_i, \gamma_i)$ as well as $\mathsf{H}(\mu_i,\ R_i + \mathsf{F}(\alpha_i) + \beta_i \cdot \mathsf{pk})$, and that if $\mathcal{A}$ outputs a message/signature pair $(m, (c', s', \varphi))$ then it had previously queried $\mathsf{H}'(m, \varphi)$.

We prove the theorem via a sequence of hybrid experiments.

Expt G_0. This is the one-more unforgeability experiment where $\mathcal{A}$ interacts with the transformed scheme $\mathsf{CCBS}[\mathsf{LF}]$.

When $\mathcal{A}$ sends a commitment com as part of the second message of an execution of the signing protocol, we say com is *extractable* if it was previously returned as output from a query of the form $\mathsf{H}'(\alpha, \beta, \mu, \gamma)$.

Expt G_1. This experiment is identical to G_0 except that it aborts (and $\mathcal{A}$ does not succeed) if (1) at any point in the experiment, there is a collision in H' or (2) in some execution of the signing protocol, some commitment com_i sent by $\mathcal{A}$ is not extractable, but later in the same execution $I \neq i$ and the signer does not abort (so, in particular, $\mathcal{A}$ sends $\alpha_i, \beta_i, \mu_i, \gamma_i$ for which $\mathsf{H}'(\alpha_i, \beta_i, \mu_i, \gamma_i) = \mathsf{com}_i$). The probability of the first event is at most $q_{\mathsf{H}'}^2/2^\kappa$. Focusing on the least $i \neq I$ in each execution of the signing protocol for which com_i is not extractable (if one exists), we see that the probability of the second event is at most $p \cdot q_{\mathsf{H}'}/2^\kappa$. Hence, $\mathcal{A}$'s success probability in G_1 is at least $\epsilon - (q_{\mathsf{H}'}^2 + p \cdot q_{\mathsf{H}'})/2^\kappa$.

Note that in G_1 and all subsequent experiments, as long as the experiment is not aborted, any extractable commitment com was previously returned as output from a *unique* query of the form $\mathsf{H}'(\alpha, \beta, \mu, \gamma)$. We say that α, β, μ are *associated with* com in that case.

In an execution of the signing protocol, we say $\mathcal{A}$ *successfully cheats* if the signer does not abort the execution (nor does the experiment itself abort), yet either (1) some commitment sent by $\mathcal{A}$ in that execution was not extractable or (2) for some i, the commitment com_i sent in that execution was extractable with associated values α_i, β_i, μ_i, but $c_i \neq \mathsf{H}(\mu_i, R_i + \mathsf{F}(\alpha_i) + \beta_i \cdot \mathsf{pk})$ (where R_i is the value sent by the signer in the corresponding session). In G_1, the only way $\mathcal{A}$ can successfully cheat in some execution is if $\mathcal{A}$ sends a *single* non-extractable commitment com_i and/or a single incorrect c_i in that execution, and the challenge I sent by the signer is equal to i. For an integer N, we let cheat_N be the indicator variable that is equal to 1 iff $\mathcal{A}$ successfully cheats in the (unique) execution of the signing protocol that uses cut-and-choose parameter N. Let $\mathsf{cheat}^* = \sum_{N=2}^{p+1} \mathsf{cheat}_N$ be the number of times $\mathcal{A}$ successfully cheats in the entire experiment. By the observation made a moment ago, we have $\mathbb{E}[\mathsf{cheat}_N] \leq 1/N$ for all N, and so

$$\mathbb{E}[\mathsf{cheat}^*] \leq \sum_{N=2}^{p+1} \frac{1}{N} \leq \ln(p+1).$$

Expt G_2. This experiment is identical to G_1 except that it aborts (and $\mathcal{A}$ does not succeed) if $\mathsf{cheat}^* > 3\ln(p+1) + \ln(2/\epsilon)$. As the cheat_N are (dominated by) independent Bernoulli variables, and cheat^* is their sum, we can apply the Chernoff bound to conclude that

$$\Pr[\mathsf{cheat}^* > 3\ln(p+1) + \ln(2/\epsilon)] \leq \epsilon/2.$$

(We defer the full calculation to Appendix B). Hence, $\mathcal{A}$'s success probability in G_2 is at least $\epsilon/2 - (q_{\mathsf{H}'}^2 + p \cdot q_{\mathsf{H}'})/2^\kappa$.

Expt G_3. Here, we change the way each execution of the signing protocol is run. Now, for each execution of the signing protocol—say, using cut-and-choose parameter N—first choose uniform $j \in [N]$. Then:

- For $i \in [N]$, if com_i is not extractable then compute R_i (and s_i, if needed) as before. Set $C_i := \perp$. (The purpose of C_i will be clear later.)
- For $i \in [N] \setminus \{j\}$, if com_i is extractable with associated values α_i, β_i, μ_i, then compute R_i as before and set $R_i' := R_i + \mathsf{F}(\alpha_i) + \beta_i \cdot \mathsf{pk}$. If $\mathsf{H}(\mu_i, R_i')$ is already defined (before R_i is sent to $\mathcal{A}$), the experiment aborts and $\mathcal{A}$ does not succeed. Otherwise, set $\mathsf{H}(\mu_i, R_i')$ to a uniform value and set $C_i := \mathsf{H}(\mu_i, R_i') + \beta_i$. Compute s_i (if needed) as before.

– If com_j is extractable with associated values α_j, β_j, μ_j, we refer to j as a *programmed session*. In this case, choose $r_j \leftarrow \mathcal{D}$ and $C_j \leftarrow \mathcal{S}$, compute $R_j := \mathsf{F}(r_j) + C_j \cdot (-\mathsf{pk})$ and $R'_j := R_j + \mathsf{F}(\alpha_j) + \beta_j \cdot \mathsf{pk}$, and program $\mathsf{H}(\mu_j, R'_j) := C_j - \beta_j$. (This programming is done before R_j is sent to $\mathcal{A}$.) If $\mathsf{H}(\mu_j, R'_j)$ is already defined, the experiment aborts (and $\mathcal{A}$ does not succeed).
  Later in the execution, if $I = j$ and neither the execution nor the experiment is aborted, compute and send $s_j := r_j + C_j \cdot (-\mathsf{sk}) + c_j \cdot \mathsf{sk}$, where c_j is the corresponding value sent by $\mathcal{A}$.

Ignoring for a moment the aborts introduced in this experiment, we claim that the view of the adversary in each execution of the signing protocol is identical to its view in G_2. This is immediate for all but a programmed session. But it can be verified that in a programmed session j, the joint distribution of s_j and R_j is identical to the distribution of those values in G_2. Moreover, C_j is uniform even conditioned on s_j, R_j, and so $\mathsf{H}(\mu_j, R'_j)$ is programmed to be a uniform value. The latter can be seen as follows. As long as c_j has not been sent by $\mathcal{A}$, R_j is uniform, and hence so is C_j. After c_j is sent by $\mathcal{A}$, s_j and c_j together fully determine R_j as $R_j = \mathsf{F}(s_j) - c_j \cdot \mathsf{pk}$. Hence, for all values of c_j, conditioning on s_j, R_j is the same as conditioning on only s_j. Since $s_j = r_j + C_j \cdot (-\mathsf{sk}) + c_j \cdot \mathsf{sk}$ and r_j is a uniform value, C_j is also uniform.

As for the aborts introduced in G_3, note that whenever the experiment checks whether $\mathsf{H}(\mu, R')$ is already defined it is the case that R' has min-entropy at least κ. (This follows because $R = \mathsf{F}(r)$ for uniform r and F has min-entropy at least κ.) Thus, the probability that G_3 aborts where G_2 would not is at most $p^2 \cdot (p^2 + q_{\mathsf{H}})/2^\kappa$. We conclude that $\mathcal{A}$ succeeds in G_3 with probability at least $\epsilon/2 - (q_{\mathsf{H}'}^2 + p \cdot q_{\mathsf{H}'} + p^2 \cdot (p^2 + q_{\mathsf{H}}))/2^\kappa$.

Expt G_4. Here, we again change each execution of the signing protocol. Consider an execution with cut-and-choose parameter N, and let $j, \{C_i\}_{i \in [N]}$ be as in the previous experiment. After $\mathcal{A}$ sends $c_1, \ldots, c_N$, if it holds that $(c_1, \ldots, c_n) = (C_1, \ldots, C_n)$ then set $I := j$; otherwise, set $I := j + 1 \pmod N$. The rest of the execution is as in G_3.

We claim that $\mathcal{A}$'s view in G_4 is identically distributed to its view in G_3, and hence its success probability is unchanged. Indeed, in any particular execution of the protocol, the value of j is independent of both the view of $\mathcal{A}$ before I is sent as well as the $\{C_i\}_{i \in [N]}$. Thus, regardless of whether $(c_1, \ldots, c_n)$ is equal to $(C_1, \ldots, C_n)$ or not, I is uniformly distributed in $[N]$ in experiment G_4 just as in experiment G_3.

In an execution of the signing protocol, we say the programmed session is *completed* if $I = j$ and the signer does not abort during the remainder

of the execution of the signing protocol. Note that when the programmed session is completed, $c_j = C_j$ and hence

$$s_I = s_j = r_j + C_j \cdot (-\mathsf{sk}) + c_j \cdot \mathsf{sk} = r_j = r_I.$$

Thus, the only time sk is needed when executing the signing protocol in G_4 is when $\mathcal{A}$ successfully cheats, in which case the programmed session is not completed.

For a valid message/signature pair $(m, \sigma) = (m, (c', s', \varphi))$ output by $\mathcal{A}$, let $R' = \mathsf{F}(s') - c' \cdot \mathsf{pk}$ and $\mu = \mathsf{H}'(m, \varphi)$; we say this message/signature pair is *fake* if there is a programmed session in which H was programmed at the point (μ, R') and, if so, we associate (m, σ) with the unique such session. (There cannot be more than one programmed session where H' is programmed at the same point, or else the experiment aborts.) A fake message/signature pair can thus be associated with a particular commitment com_j having associated values $\alpha_j, \beta_j, \mu_j = \mu$ (recall that a session is only programmed if the corresponding commitment is extractable), as well as values r_j, R_j, C_j defined by the experiment. Since (c', s', φ) is a valid signature on m, we have $c' = \mathsf{H}(\mu, R')$; we also have $\mathsf{H}(\mu, R') = C_j - \beta_j$ (by definition of how programming is done) and thus $\beta_j = C_j - c'$. Therefore

$$\begin{aligned}
\mathsf{F}(s') &= R' + c' \cdot \mathsf{pk} \\
&= R_j + \mathsf{F}(\alpha_j) + \beta_j \cdot \mathsf{pk} + c' \cdot \mathsf{pk} \\
&= R_j + C_j \cdot (-\mathsf{pk}) + \mathsf{F}(\alpha_j) + (C_j - c') \cdot \mathsf{pk} + c' \cdot \mathsf{pk} \\
&= R_j + C_j \cdot (-\mathsf{pk}) + \mathsf{F}(\alpha_j) \\
&\quad + C_j \cdot \mathsf{pk} - c' \cdot \mathsf{pk} + \mathsf{F}(\Psi(\mathsf{pk}, C_j, -c')) + c' \cdot \mathsf{pk} \\
&= R_j + C_j \cdot (-\mathsf{pk}) + \mathsf{F}(\alpha_j) + C_j \cdot \mathsf{pk} + \mathsf{F}(\Psi(\mathsf{pk}, C_j, -c')) \\
&= R_j + \mathsf{F}(\alpha_j) + \mathsf{F}(\Psi(\mathsf{pk}, C_j, -c')),
\end{aligned}$$

and so

$$\mathsf{F}(s' - \alpha_j - \Psi(\mathsf{pk}, C_j, -c')) = R_j. \tag{1}$$

There is at most one fake message/signature pair associated with any programmed session (since the distinct $\{m_i\}$ in $\mathcal{A}$'s output correspond to distinct $\{\mu_i = \mathsf{H}'(m_i, \varphi_i)\}$ or else the experiment aborts), and so the number of fake pairs is at most the number of programmed sessions.

Expt G_5. Experiment G_5 aborts (and $\mathcal{A}$ does not succeed) if the number F of fake pairs exceeds the number of *completed*, programmed sessions. Before we bound the probability of this event, note that the number of

completed, programmed sessions is at most $\ell - \mathsf{cheat}^*$; therefore, if F is at most the number of completed, programmed sessions, then if $\mathcal{A}$ succeeds the number of valid message/signature pairs that are *not* fake is

$$(\ell + 1) - F \geq (\ell + 1) - (\ell - \mathsf{cheat}^*) = \mathsf{cheat}^* + 1.$$

Claim. The probability (in G_4) that $\mathcal{A}$ succeeds and the number of fake message/signature pairs exceeds the number of completed, programmed sessions is at most $p \cdot \epsilon_{\mathsf{PRE}}$.

Proof. Let E be the event that $\mathcal{A}$ succeeds and the number of fake message/signature pairs exceeds the number of completed, programmed sessions. We construct an adversary $\mathcal{C}$ attacking preimage resistance of LF that succeeds with probability at least $\Pr[E]/p$. The claim follows.

$\mathcal{C}$ is given parameters par and a challenge $R \in \mathcal{R}$. It honestly generates $(\mathsf{pk}, \mathsf{sk})$ and runs experiment G_5 with $\mathcal{A}$ with the following exception:

- For a uniformly chosen execution of the signing protocol (say, the kth execution), $\mathcal{C}$ sets $R_j := R + C_j \cdot (-\mathsf{pk})$ in the programmed session of that execution. If in that execution, $I = j$ and s_j must be sent to the adversary (so the programmed session is to be completed), $\mathcal{C}$ aborts.

Note that when $\mathcal{C}$ does not abort, $\mathcal{C}$ never needs to use a preimage of R. At the end of the experiment, $\mathcal{C}$ aborts if E has not occurred. If E has occurred, $\mathcal{C}$ finds the first fake message/signature pair $(m, (c', s', \varphi))$ associated with a non-completed, programmed session and aborts if that pair is not associated with the programmed session in execution k. If $\mathcal{C}$ has not aborted, $\mathcal{C}$ has values α_j, C_j, used as part of the programmed session in execution k, such that $s' - \alpha - \Psi(\mathsf{pk}, C, -c')$ is a preimage of R (using Equation (1)). The probability that $\mathcal{C}$ does not abort is precisely $\Pr[E]/p$.

Using the above claim, we see that $\mathcal{A}$ succeeds in G_5 with probability at least $\epsilon/2 - (q_{\mathsf{H}'}^2 + p \cdot q_{\mathsf{H}'} + p^2 \cdot (p^2 + q_{\mathsf{H}}))/2^\kappa - p \cdot \epsilon_{\mathsf{PRE}}$.

Bounding $\mathcal{A}$'s success probability in G_5. To conclude the proof, we show that the success probability of $\mathcal{A}$ in G_5 is negligible. We do so by defining an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and attacks the λ-one-more unforgeability of $\mathsf{BS}[\mathsf{LF}]$, where $\lambda = 3\ln(p+1) + \ln(2/\epsilon)$. Adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ is given a public key pk as well as access to a signing oracle for $\mathsf{BS}[\mathsf{LF}]$ and an oracle H. It runs $\mathcal{A}$ on pk, and simulates experiment G_5 for $\mathcal{A}$ as described below. Queries that $\mathcal{A}$ makes to H' are answered by $\mathcal{B}$

with uniform values in the natural way. Queries that $\mathcal{A}$ makes to H are in general answered by simply relaying those queries to $\mathcal{B}$'s oracle H, except that in programmed sessions $\mathcal{B}$ programs H to a different value (as described in G_3).

2. $\mathcal{B}$ simulates an execution of the signing protocol for $\mathcal{A}$ using cut-and-choose parameter N as follows. $\mathcal{B}$ selects a uniform $j \leftarrow [N]$ and initiates an interaction with its signing oracle for $\mathsf{BS}[\mathsf{LF}]$. Let R^* be the value that $\mathcal{B}$ receives from its signing oracle in the first round. When $\mathcal{A}$ sends $\mathsf{com}_1, \ldots, \mathsf{com}_N$, then:
   - $\mathcal{B}$ sets $R_{j+1} := R^*$ and generates the remaining $\{R_i\}_{i \neq j+1}$ as in G_5. It then sends these values to $\mathcal{A}$.
   - $\mathcal{B}$ then continues to run the signing protocol as in G_5. If $I = j + 1$ and $\mathcal{B}$ needs to send s_I (i.e., neither the current execution of the signing protocol nor the experiment itself is aborted) then $\mathcal{B}$ forwards c_I to its signing oracle for $\mathsf{BS}[\mathsf{LF}]$, and returns the response s^* to $\mathcal{A}$.
3. At the end of the experiment, if $\mathcal{A}$ outputs $\ell + 1$ valid message/signature pairs $(m, (c', s', \varphi))$ (where validity is determined relative to $\mathsf{CCBS}[\mathsf{LF}]$ and the oracles H, H' that $\mathcal{B}$ simulated for $\mathcal{A}$), then $\mathcal{B}$ aborts if the number of fake message/signature pairs exceeds the number of completed, programmed sessions. Assuming it has not aborted, $\mathcal{B}$ identifies $\mathsf{cheat}^* + 1$ valid message/signature pairs that are not fake, and for each such pair $(m, (c', s', \varphi))$ outputs $(\mathsf{H}'(m, \varphi), (c', s'))$.

The simulation provided by $\mathcal{B}$ is perfect, and thus the probability that $\mathcal{A}$ succeeds when run by $\mathcal{B}$ is exactly the probability that $\mathcal{A}$ succeeds in G_5. The number of executions of the signing protocol that $\mathcal{B}$ initiates with $\mathsf{BS}[\mathsf{LF}]$ is p, while the number that $\mathcal{B}$ completes is exactly cheat^* and so is at most $3\ln(p+1) + \ln(2/\epsilon)$. Finally, whenever $\mathcal{A}$ succeeds then for any message/signature pair $(m, (c', s', \varphi))$ output by $\mathcal{A}$ that is not fake, the message/signature pair $(\mathsf{H}'(m, \varphi), (c', s'))$ output by $\mathcal{B}$ is a valid message/signature pair relative to $\mathsf{BS}[\mathsf{LF}]$ and the oracle H provided to $\mathcal{B}$; additionally, the messages $\mathsf{H}'(m, \varphi)$ are distinct since no collisions were found in H'. We conclude that the success probability of $\mathcal{B}$ is equal to the success probability of $\mathcal{A}$ in G_5, which is negligible since $\mathsf{BS}[\mathsf{LF}]$ is secure. This completes the proof of the theorem.

3.3 Improving the Complexity of the Signing Protocol

The complexity of the signing protocol is linear in the cut-and-choose parameter N, and it is therefore important to minimize that parameter.

In the scheme analyzed thus far, N is incremented each time the signing protocol is executed. Here, we argue that it suffices to increment the cut-and-choose parameter *only when cheating is detected*. Not only is this strictly better in theory (assuming at least some interactions are with honest users), but we expect that this optimization would have a significant impact on efficiency in practice where (1) the signer would likely know the identity of each user executing the protocol, and could ban any user the first time they are caught cheating, and (2) we expect that a majority of users are honest.

The discussion that follows assumes familiarity with the high-level overview from Section 1.2 and/or the proof of one-more unforgeability from the previous section. We focus our treatment on the sequential setting, and briefly discuss at the end how it can be extended to handle concurrent executions of the protocol.

Recall that in an execution of the signing protocol of our transformed scheme, we say the adversary *successfully cheats* if it cheats in a single session and is not caught by the signer. In a given execution using cut-and-choose parameter N, the adversary successfully cheats with probability at most $1/N$. For the proof of one-more unforgeability, it is crucial that (over the course of the entire experiment) the adversary successfully cheats at most logarithmically many times, except with negligible probability.

Let cheat_N be a random variable denoting the number of times, over the course of the entire one-more unforgeability experiment, the adversary successfully cheats when the cut-and-choose parameter is N. In the scheme analyzed thus far, each value of the cut-and-choose parameter is used only once and so $\mathbb{E}[\mathsf{cheat}_N] \leq 1/N$. Thus, assuming the attacker runs p executions of the signing protocol overall, the expected number of times the attacker successfully cheats is

$$\sum_{N=2}^{p+1} \mathbb{E}[\mathsf{cheat}_N] \leq \sum_{N=2}^{p+1} \frac{1}{N} \leq \ln(p+1).$$

Consider now what happens if we modify our scheme so that the counter is only incremented when cheating is detected. (We also assume for simplicity that the attacker cheats in exactly one session each time it runs the protocol; it is clear that this maximizes the number of times it can successfully cheat.) Then cheat_N is equal to the number of times the attacker successfully cheats (when the cut-and-choose parameter is N) before being caught. This is one less than the number of trials (when the cut-and-choose parameter is N) until the adversary is caught. (Recall that here we are assuming sequential executions of the signing protocol only.)

Since the probability of being caught in each such trial is $(N-1)/N$, we now have

$$\mathbb{E}[\mathsf{cheat}_N] = \frac{N}{N-1} - 1 = \frac{1}{N-1},$$

and so if the attacker runs p executions of the signing protocol overall, the expected number of times the attacker successfully cheats is at most

$$\sum_{N=2}^{p+1} \mathbb{E}[\mathsf{cheat}_N] = \sum_{N=2}^{p+1} \frac{1}{N-1} \leq 1 + \ln p.$$

Proceeding as in[6] the proof of Theorem 4, we can show that the adversary successfully cheats at most logarithmically many times, except with negligible probability.

Handling concurrent executions. The optimization described above does not work when there may be concurrent executions of the signing protocol. (To see what goes wrong, consider the case where the adversary runs p parallel executions, all using cut-and-choose parameter $N = 2$. Then the adversary successfully cheats in roughly half those executions before the signer detects cheating and has any chance to increment the counter.) For the argument outlined above to work, the key property we need to ensure is that the adversary can successfully cheat *at most once* for each value of the cut-and-choose parameter. To enforce this, the signer just needs to make sure that any currently active executions of the signing protocol use distinct values of the cut-and-choose parameter; moreover, once cheating is detected in an execution using cut-and-choose parameter N, no subsequent executions may use cut-and-choose parameter N. So, for example, the signer can store the largest value of the cut-and-choose parameter N^* for which cheating has been detected, and then when initiating an execution of the signing protocol can use as the cut-and-choose parameter the least value $N > N^*$ that is not currently being used by any active execution.

References

1. M. Abe and T. Okamoto. Provably secure partially blind signatures. In M. Bellare, editor, *Crypto 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, Aug. 2000.
2. F. Baldimtsi and A. Lysyanskaya. Anonymous credentials light. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, Nov. 2013.

[6] As cheat_N now may take values larger than 1, we use Hoeffding's inequality instead of a Chernoff bound (which results in a slightly looser reduction).

3. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In P. F. Syverson, editor, *Financial Cryptography 2001*, volume 2339 of *LNCS*, pages 319–338. Springer, Heidelberg, Feb. 2002.
4. F. Benhamouda, T. Lepoint, J. Loss, M. Orrù, and M. Raykova. On the (in)security of ROS. In *Eurocrypt 2021 (to appear)*, 2021.
5. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, Jan. 2003.
6. D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Crypto '82*, pages 199–203. Plenum Press, New York, USA, 1982.
7. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In C. Dwork, editor, *Crypto 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, Aug. 2006.
8. G. Fuchsbauer, C. Hanser, C. Kamath, and D. Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In V. Zikas and R. De Prisco, editors, *SCN 2016*, volume 9841 of *LNCS*, pages 391–408. Springer, Heidelberg, Aug. / Sept. 2016.
9. G. Fuchsbauer, C. Hanser, and D. Slamanig. Practical round-optimal blind signatures in the standard model. In R. Gennaro and M. J. B. Robshaw, editors, *Crypto 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, Aug. 2015.
10. G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In A. Canteaut and Y. Ishai, editors, *Eurocrypt 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.
11. S. Garg and D. Gupta. Efficient round-optimal blind signatures. In P. Q. Nguyen and E. Oswald, editors, *Eurocrypt 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, Heidelberg, May 2014.
12. S. Garg, V. Rao, A. Sahai, D. Schröder, and D. Unruh. Round optimal blind signatures. In P. Rogaway, editor, *Crypto 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Heidelberg, Aug. 2011.
13. E. Ghadafi. Efficient round-optimal blind signatures in the standard model. In A. Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Heidelberg, Apr. 2017.
14. E. Hauck, E. Kiltz, and J. Loss. A modular treatment of blind signatures from identification schemes. In Y. Ishai and V. Rijmen, editors, *Eurocrypt 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.
15. E. Hauck, E. Kiltz, J. Loss, and N. K. Nguyen. Lattice-based blind signatures, revisited. In D. Micciancio and T. Ristenpart, editors, *Crypto 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, Aug. 2020.
16. C. Hazay, J. Katz, C.-Y. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 323–341. Springer, Heidelberg, Feb. 2007.
17. B. Jayaraman, H. Li, and D. Evans. Decentralized certificate authorities. Available at `https://arxiv.org/abs/1706.03370`.
18. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In B. S. Kaliski Jr., editor, *Crypto '97*, volume 1294 of *LNCS*, pages 150–164. Springer, Heidelberg, Aug. 1997.

19. J. Kastner, J. Loss, and J. Xu. On pairing-free blind signature schemes in the algebraic group model, 2020. Available at `https://eprint.iacr.org/2020/1071`.
20. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. F. Brickell, editor, *Crypto '92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, Aug. 1993.
21. T. Okamoto. Efficient blind and partially blind signatures without random oracles. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer, Heidelberg, Mar. 2006.
22. D. Pointcheval. Strengthened security for blind signatures. In K. Nyberg, editor, *Eurocrypt '98*, volume 1403 of *LNCS*, pages 391–405. Springer, Heidelberg, May / June 1998.
23. D. Pointcheval and J. Stern. Provably secure blind signature schemes. In K. Kim and T. Matsumoto, editors, *Asiacrypt '96*, volume 1163 of *LNCS*, pages 252–265. Springer, Heidelberg, Nov. 1996.
24. D. Pointcheval and J. Stern. New blind signatures equivalent to factorization (extended abstract). In R. Graveman, P. A. Janson, C. Neuman, and L. Gong, editors, *ACM CCS '97*, pages 92–99. ACM Press, Apr. 1997.
25. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.
26. C.-P. Schnorr. Security of blind discrete log signatures against interactive attacks. In S. Qing, T. Okamoto, and J. Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, Nov. 2001.
27. D. Wagner. A generalized birthday problem. In M. Yung, editor, *Crypto 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, Aug. 2002.

A Additional Examples of Linear Function Families

In Section 2.2 we defined linear function families, and described the linear function family that underlies the Schnorr blind signature scheme. Here we recall additional examples of linear function families from the work of Hauck et al. [14].

Okamoto-Schnorr. Here, par defines a cyclic group $\mathbb{G}$ of prime order q, and also includes uniformly selected generators $g_1, g_2 \in \mathbb{G}$. We let $\mathcal{S} = \mathbb{Z}_q$, $\mathcal{D} = \mathbb{Z}_q^2$, and $\mathcal{R} = \mathbb{G}$, with the scalar multiplication maps $s \cdot (x, y) = (s \cdot x, s \cdot y)$ (for $s, x, y \in \mathbb{Z}_q$) and $s \cdot g = g^s$ (for $g \in \mathbb{G}$). Defining $\mathsf{F}(x, y) := g_1^x \cdot g_2^y$, a pseudo torsion-free element in the kernel is given by $z^* = (-1, \log_{g_2} g_1)$. Since scalar multiplication between $\mathcal{S}$ and $\mathcal{R}$ is distributive, Ψ is the zero function. Finally, LF is preimage resistant and collision resistant under the discrete logarithm assumption in $\mathbb{G}$.

Okamoto-Guillou-Quisquater. Here, par contains $N = pq$ for distinct primes p, q, along with a uniform value $a \in \mathbb{Z}_N^*$ and a prime λ with $\gcd(\varphi(N), \lambda) = \gcd(N, \lambda) = 1$. We define $\mathcal{S} = \mathbb{Z}_\lambda$ under addition modulo λ; define $\mathcal{R} = \mathbb{Z}_N^*$ under multiplication modulo N; and define $\mathcal{D} = \mathbb{Z}_\lambda \times \mathbb{Z}_N^*$

with group operation given by

$$(x_1, y_1) \circ (x_2, y_2) := \left(x_1 + x_2 \bmod \lambda,\ \ y_1 \cdot y_2 \cdot a^{\lfloor \frac{x_1+x_2}{\lambda} \rfloor} \bmod N \right).$$

(It can be shown [14] that this is indeed a group.) Scalar multiplication maps $s \cdot b$ for $b \in \mathcal{R}$ or $b \in \mathcal{D}$ are defined as s-fold iteration of the corresponding group operation. Moreover, define $\mathsf{F}(x, y) := a^x y^\lambda \bmod N$ and $\Psi(x, s, s') := (0, x^{\lfloor -\frac{s+s'}{\lambda} \rfloor} \bmod N)$. A pseudo torsion-free element in the kernel is given by $z^* = (\lambda - 1,\ a^{\lambda^{-1}-1} \pmod N)$, where λ^{-1} is the inverse of λ modulo $\varphi(N)$. LF is preimage resistant and collision resistant under a suitable version of the RSA assumption.

Fiat-Shamir. Here, par contains $N = pq$ for distinct primes p, q, and we define $\mathcal{S} = \mathbb{Z}_2^\kappa$, $\mathcal{D} = \mathcal{R} = (\mathbb{Z}_N^*)^\kappa$. The scalar multiplication maps are

$$(s_1, \ldots, s_\kappa) \cdot (x_1, \ldots, x_\kappa) = (x_1^{s_1}, \ldots, x_\kappa^{s_\kappa}).$$

Let $\mathsf{F}(x_1, ..., x_k) := (x_1^2 \pmod N, ..., x_k^2 \pmod N)$, and define $\Psi(\vec{x}, \vec{r}, \vec{s})$ component-wise with $\Psi(x_i, r_i, s_i) := x_i^{-(r_i > s_i + r_i \pmod 2)}$ (where $r_i > s_i + r_i \pmod 2$ denotes the predicate that returns 1 iff $r_i = s_i = 1 \pmod 2$). A pseudo torsion-free element in the kernel is $z^* = (-1, ..., -1)$. LF is preimage resistant and collision resistant under the factoring assumption.

B Deferred Calculations

Let X be a sum of independent $\{0, 1\}$-random variables with $\mu = \mathbb{E}[X]$. The multiplicative Chernoff bound states that for all $\delta > 0$

$$\Pr[X \geq (1 + \delta) \cdot \mu] \leq \exp\left(-\frac{\mu \delta^2}{2 + \delta} \right).$$

Let $X = \mathsf{cheat}^* = \sum_{N=2}^{p+2} \mathsf{cheat}_N$. Then for any $s > \ln(p + 1) \geq \mathbb{E}[\mathsf{cheat}_N]$ we have

$$\begin{aligned} \Pr[\mathsf{cheat}^* \geq s] &= \Pr\left[\mathsf{cheat}^* \geq \left(1 + \left(\frac{s}{\mu} - 1 \right) \right) \cdot \mu \right] \\ &\leq \exp\left(-\frac{\mu(s/\mu - 1)^2}{2 + (s/\mu - 1)} \right). \end{aligned}$$

Using the fact that $x^2/(2 + x) > x - 2$ for all $x \geq 0$, the above is at most

$$\exp\left(-\mu \left(\frac{s}{\mu} - 3 \right) \right) = \exp(3\mu - s)$$

If we set $s := 3 \ln(p + 1) + \ln(1/\epsilon)$, the above equals ϵ.