Snowblind: A Threshold Blind Signature in Pairing-Free Groups

Elizabeth Crites[1](✉), Chelsea Komlo[2,3], Mary Maller[4,5], Stefano Tessaro[6], and Chenzhi Zhu[6]

[1] University of Edinburgh, Edinburgh, UK
ecrites@ed.ac.uk
[2] University of Waterloo, Waterloo, Canada
ckomlo@uwaterloo.ca
[3] Zcash Foundation, McLean, USA
[4] Ethereum Foundation, Zug, Switzerland
mary.maller@ethereum.org
[5] PQShield, Oxford, UK
[6] Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, USA
{tessaro,zhucz20}@cs.washington.edu

Abstract. Both threshold and blind signatures have, individually, received a considerable amount of attention. However little is known about their combination, i.e., a threshold signature which is also blind, in that no coalition of signers learns anything about the message being signed or the signature being produced. Several applications of blind signatures (e.g., anonymous tokens) would benefit from distributed signing as a means to increase trust in the service and hence reduce the risks of key compromise. This paper builds the first blind threshold signatures in pairing-free groups. Our main contribution is a construction that transforms an underlying blind non-threshold signature scheme with a suitable structure into a threshold scheme, preserving its blindness. The resulting signing protocol proceeds in three rounds, and produces signatures consisting of one group element and two scalars. The underlying non-threshold blind signature schemes are of independent interest, and improve upon the current state of the art (Tessaro and Zhu, EUROCRYPT '22) with shorter signatures (three elements, instead of four) and simpler proofs of security. All of our schemes are proved secure in the Random Oracle and Algebraic Group Models, assuming the hardness of the discrete logarithm problem.

1 Introduction

Blind signatures [11] allow a *user* to interact with a *signer* to obtain a valid signature on a chosen message. The signer learns nothing about the message being signed, and cannot link any signature back to the interaction that produced it. Blind signatures are a key ingredient in e-cash systems [11,12], and play a major role in a number of recent applications and products in industry, such as privacy-preserving ad-click measurement [29], Apple's iCloud Private Relay [21],

© International Association for Cryptologic Research 2023
H. Handschuh and A. Lysyanskaya (Eds.): CRYPTO 2023, LNCS 14081, pp. 710–742, 2023.
https://doi.org/10.1007/978-3-031-38557-5_23

Google One's VPN Service [44], and various forms of anonymous tokens [20,42]. Variants of RSA blind signatures [26] are also covered by an RFC draft [13].

The main aim of this paper is to mitigate the risk of signer's compromise in blind signatures by following the popular approach of distributing the signer's operation across a number of issuers, each holding a share of the secret key, as in threshold signatures [14,15]. This raises the natural question of how easy it is to implement threshold *blind* signatures, a blind analogue of the classical notion of threshold signatures, which has received significantly less attention. Crucially, unlike standard threshold signatures, the signers learn nothing about the message being signed. Moreover, resulting signatures need to remain unlinkable.

It is possible to combine ideas from [9] to obtain a threshold-blind version of BLS [10], as done explicitly in [43]. BLS signing is non-interactive and therefore, by default, concurrently secure. Other works also give pairing-based schemes [24]. Here, in contrast, we focus on designs based on standard, *pairing-free*, elliptic curves. These are appealing, as highly-verified standard cryptographic libraries (such as NSS and BoringSSL) do not provide support for pairing-friendly curves. RSA signatures [34] are pairing free and non-interactive; however, signature sizes are much larger than those defined over elliptic curves. For example, RSA signatures are 6 times larger than Schnorr signatures at the same security level. Our schemes add only one field element to Schnorr signatures, making them an attractive alternative. Designing pairing-free schemes comes with a number of technical challenges to achieve concurrent security and prevent so-called ROS attacks [8], which have affected both threshold and blind signatures alike.

Our contributions. We develop Snowblind, a construction of threshold blind signatures which compiles a suitable underlying (non-threshold) blind signature scheme into a (blind) threshold signing protocol. The resulting signing protocol proceeds in three rounds between the coordinator and the servers. Our instantiations of Snowblind produce signatures that consist of three group elements, and the underlying signature scheme is marginally more complex than standard Schnorr signatures. The unforgeability of these instantiations is proved in the Algebraic Group Model (AGM) [16], assuming the hardness of the discrete logarithm problem. We also assume random oracles.

These schemes satisfy a strong notion of (statistical) blindness that holds even if *all* servers collude. We also present formal security definitions, generalizing the notions of one-more unforgeability and blindness to the threshold setting.

An important remark here is that while the AGM is undoubtedly undesirable, it has been necessary in all recent constructions of pairing-free blind signatures based on the hardness of DL-related problems [17,22,41]. Avoiding its use in the concurrent setting is a well-known and very challenging theoretical question.

Improving blind signatures. Snowblind relies on new, non-threshold, three-move blind signature schemes of independent interest. The technical challenges to build such schemes are captured already by a non-blind *interactive* signing protocol for Schnorr signatures. Here, the signer initially sends $A \leftarrow g^a$ to the user, where $a \leftarrow_{\$} \mathbb{Z}_p$. Subsequently, the user responds with a challenge $c \leftarrow \mathsf{H}_{\mathsf{sig}}(m, A)$, and the signer sends $z = a + c \cdot \mathsf{sk}$, where $\mathsf{sk} \in \mathbb{Z}_p$ is the secret key. In particular, $\mathsf{pk} = g^{\mathsf{sk}}$ is the public key, and (A, z) is a valid Schnorr signature for m.

one-more unforgeability不可再造性：对手无法根据n次互动生成n+1个签名

Benhamouda et al. [8] show that this protocol is completely insecure against a malicious user that interacts *concurrently* with the signer: After obtaining $\ell \geq \log p$ initial values $A_1, \ldots, A_\ell$, one can efficiently compute suitable challenges $c_1, \ldots, c_\ell$ such that their responses yield $\ell + 1$ valid signatures, hence violating *one-more unforgeability*. The attacker achieves this by solving the related ROS problem, for which [8] gives a polynomial-time algorithm.

Tessaro and Zhu [41] recently proposed an approach to mitigate the above attack by having the signer initially send a *pair*

$$A = g^a \ , \quad B \leftarrow g^b h^y \ ,$$

where $a, b, y \leftarrow\!\!\$\ \mathbb{Z}_p$. Then, upon receiving the challenge $c \leftarrow \mathsf{H}_{\mathsf{sig}}(m, A, B)$, the signer responds with

$$z \leftarrow a + c \cdot y \cdot \mathsf{sk} \ , \quad y \ , \quad b \ .$$

The final signature is (A, z, b, y).

This protocol can easily be made blind. In [41], the user masks the values (A, B, c, b, y, z) using randomness r_1, r_2, α, β as follows:

$$\begin{aligned} \bar{A} &= g^{r_1} A^{\alpha/\beta} \ , & \bar{B} &= g^{r_2} B^{\alpha} \ , & \bar{b} &= r_2 + \alpha b \ , \\ \bar{c} &= c/\beta \ , & \bar{y} &= \alpha y \ , & \bar{z} &= r_1 + (\alpha/\beta) z \ . \end{aligned}$$

The final blinded signature is $(\bar{A}, \bar{z}, \bar{b}, \bar{y})$, which is perfectly blinded by the randomness r_1, r_2, α, β.

The crucial point here is that B is a *perfectly hiding* Pedersen commitment to y, and therefore y can be thought as randomly sampled *after* the challenge c is returned to the signer. The format of the final response z, thanks to this "fresh-looking" random y, compromises the linear structure of interactive Schnorr signing which enables ROS attacks. In [41], this scheme is proved one-more unforgeable in the AGM+ROM assuming the hardness of the discrete logarithm problem, along with the hardness of a variant of the ROS problem, called WFROS. However, in contrast to ROS, the WFROS problem is shown *unconditionally* to be exponentially hard.

In this paper, we improve upon [41] along two orthogonal axes:

– We show that the above signing protocol can produce signatures for a different base scheme which consists of *three* elements (one group element, and two scalars), instead of four. Note that [41] also proposes a variant of their scheme with shorter signatures, relying however on the stronger generic group model [27,38].
– We also propose an alternative approach to incorporating the value y in the signing process, where the signer final response uses $z = a + (c + y^k) \cdot \mathsf{sk}$, where $k \geq 2$ such that the map $y \mapsto y^k$ is a permutation in $\mathbb{Z}_p$. (This happens exactly when $\mathsf{gcd}(p-1, k) = 1$.) An important feature of this approach is that it also offers a significantly simpler proof than that of [41], which in particular merely relies on the hardness of the ROS problem for dimension *one*, which is known to be exponentially hard.

Table 1. Pairing-free blind signatures with concurrent security. All schemes are proved OMUF secure in the AGM+ROM, under the given assumption(s). $\mathbb{G}$ denotes a group element, $\mathbb{Z}_p$ denotes a scalar. κ indicates a κ-bit string, where κ is the security parameter. The ROS assumption is subject to a polynomial-time attack for more than $\log p$ concurrent sessions. The mROS assumption is subject to (lightly) sub-exponential attacks [17]. All schemes, except Abe's, have perfect blindness.

	PK Size	Sig size	Communication	Assumption
Blind Schnorr [17]	$1\ \mathbb{G}$	$1\ \mathbb{G} + 1\ \mathbb{Z}_p$	$1\ \mathbb{G} + 2\ \mathbb{Z}_p$	OMDL+ROS
Clause Blind Schnorr [17]	$1\ \mathbb{G}$	$1\ \mathbb{G} + 1\ \mathbb{Z}_p$	$2\ \mathbb{G} + 4\ \mathbb{Z}_p$	OMDL+mROS
Abe [1,22]	$3\ \mathbb{G}$	$2\ \mathbb{G} + 6\ \mathbb{Z}_p$	$3\ \mathbb{G} + 6\ \mathbb{Z}_p + \kappa$	DL
Tessaro-Zhu [41]	$1\ \mathbb{G}$	$1\ \mathbb{G} + 3\ \mathbb{Z}_p$	$2\ \mathbb{G} + 4\ \mathbb{Z}_p$	DL
This work	$1\ \mathbb{G}$	$1\ \mathbb{G} + 2\ \mathbb{Z}_p$	$2\ \mathbb{G} + 4\ \mathbb{Z}_p$	DL

The resulting schemes are the state-of-the-art with respect to schemes with security based solely on discrete-log related assumptions in pairing-free groups. We discuss related work more in depth in Sect. 1.1 below, and give an efficiency comparison in Table 1.

A THRESHOLD VERSION. Our main technical contribution is a threshold signing protocol for the above blind signature schemes. We assume that there are multiple issuers who each possess secret shares and a single user. Our threshold protocol requires three rounds of interaction. The signing is asynchronous and all interactions are initiated by the user. In particular issuers do not speak directly to each other. If the user goes offline then no signature is produced but there are no other negative consequences. In particular we require that signatures are unforgeable unless the user has queried at least one honest party in the third and final round of interaction.

The Snowblind signature scheme is relatively simple. The final signature is identical to the base signature (1 group element and 2 field elements). The basic idea (which will require some adjustments) is rather simple. First a *threshold* of issuers sends a pair

$$A_i = g^{a_i}\ ,\ \ B_i \leftarrow g^{b_i} h^{y_i}\ ,$$

where $a_i, b_i, y_i \leftarrow\!\!\$\ \mathbb{Z}_p$. Then these first-round messages are aggregated by the user into the product $A = \prod_i A_i$ and $B = \prod_i B_i$. Then, upon receiving the challenge $c \leftarrow \mathsf{H}_{\mathsf{sig}}(m, A, B)$, the issuers would like to directly send

$$z_i \leftarrow a_i + f(c, y) \cdot \mathsf{sk}_i\ ,\ \ y_i\ ,\ \ b_i\ ,$$

where $f(c, y) = c \cdot y$ or $f(c, y) = c + y^k$, depending on which base scheme we pick. However, they do not yet know $b = \sum_i b_i$ or $y = \sum_i y_i$. Thus instead they first reveal all b_i, y_i to the user, who sends b, y back. In the final third round the issuers return the z_i's. This protocol can also be easily made blind by masking the values (A, B, c, z, y, b) in the same way as the base blind scheme.

A few more (minor) adjustments need to be made for the scheme to be proved secure. A first one is concerned with the Pedersen commitments not being online extractable – we will resolve this by including an additional extractable commitment cm_i to y_i, along with B_i. The second is that we will need the involved issuers to agree on the set of involved issuers, their commitments cm_i, and the challenge c, before their reveal their own z_i. This will require using an additional (non-threshold, non-blind) signature scheme.

Proving security of Snowblind. Our key technical challenge is now in proving the one-more unforgeability (OMUF) of the above scheme. In particular, the base blind signature schemes discussed above do not have simple security reductions, and we were reluctant to add additional complexity to these arguments. A better approach is to attempt to reduce the OMUF of the threshold blind signature to the OMUF of the blind signature. Unfortunately this modular approach does not quite work. In particular, the reduction has to query its final-round OMUF oracle in the second round of signing in order to simulate responses. Thus when an adversary responds with $\ell + 1$ signatures having made fewer than ℓ queries (over unique sessions) to the final round, the reduction could have made $\ell + 1$ queries to its final-round oracle and thus would not output a valid forgery. Preventing the adversary from forging signatures when it only queries the preliminary rounds (i.e. the rounds before the final round) is important in our asynchronous and concurrent model, where we can make no termination guarantees.

Instead, we consider a less round-efficient base blind signature scheme that mimics the structure of our threshold scheme. In this alternative scheme, rather than sending (z, b, y) in the second round, the issuer sends (b, y) but withholds z for now. Then in a third round it reveals z. We prove the OMUF of this scheme in the algebraic group model under the discrete logarithm assumption. Security of the base two-round scheme is implied by the security of this three-round scheme because the user sends no additional information between the second and third rounds. More importantly, we can prove the security of our threshold scheme based on the security of this three-round scheme. In particular the reduction only queries its final-round OMUF oracle when the adversary queries its OMUF oracle in the final round on at least one honest party.

1.1 Related Work

We give a brief overview of the most relevant related works in greater detail.

Blind signatures in pairing-free groups. There are very efficient blind signature schemes based on pairings (starting from the work of [9], which in turn is based on [10]) and RSA [6,11] which fall outside the scope of this paper.

The space of blind signatures in pairing-free groups is more complex, especially when focusing on schemes that achieve OMUF concurrent security in the context of one-more unforgeability. As explained above, at first glance, Schnorr signatures [35] appear simple to translate into a blind setting. However, a recent algorithm [8] for solving the ROS problem [36] results in a complete break of security for a sufficient number of concurrent sessions (at least $\log p$, where p is the group order), whereas one can expect only sub-exponential security for a smaller number of concurrent sessions. (This has been proved in the AGM [17], where the security of blind Schnorr signatures is reduced to the hardness of ROS, which is sub-exponential for the case necessary to support fewer than $\log p$ sessions.)

Blind Schnorr signatures are also proved [22] to be sequentially OMUF secure in the AGM, although sequential security is too weak to support most applications of blind signatures without introducing significant performance bottlenecks. The situation is similar for a larger class of signatures based on identification schemes, which includes in particular Okamoto-Schnorr blind signatures [28]. For these, however, OMUF security for a bounded number of concurrent sessions (fewer than $\log p$) can be proved without the AGM, although with very poor concrete guarantees, via a complex rewinding argument [19]. Their sequential security follows instead from a simpler use of the Forking Lemma [32]. We note that the AGM is necessary for Schnorr signatures, as opposed to Okamoto-Schnorr, due to the lower bound of Baldimtsi and Lysyanskaya [3].

Table 1 discusses the more limited set of works achieving *concurrent* OMUF security in the pairing-free setting. All of these works rely on security proofs in the AGM+ROM. The first concurrently OMUF secure scheme is due to Abe [1]– its original proof (which did not rely on the AGM) was found to be incorrect, and a proof in the AGM+ROM was only recently given in [22]. This scheme is rather inefficient, and only achieves computational blindness (under the Decisional Diffie-Hellman assumption).[1] Fuchsbauer, Plouviez, and Seurin [17] introduced a new signing protocol for plain Schnorr signatures, called "Clause Blind Schnorr," where the output of the signing protocol is a signature that is valid under plain Schnorr verification. However, their security proof relies on the hardness of a variant of ROS (called mROS) for which sub-exponential attacks exist – instantiating their scheme on a 256-bit curve would only achieve (roughly) 80 bits of security. Finally, Tessaro and Zhu [41] recently proposed the only scheme which achieves concurrent security and perfect blindness, while producing signatures smaller than those of Abe's scheme. They do so by relying on a variant of the ROS problem, called WFROS, for which they prove an unconditional lower bound.

Kastner et al. recently corrected the OMUF security reduction [22] for the Abe-Okamoto partially blind signature scheme [2]. However their techniques do not extend to the concurrent setting.

Concurrently to this work, Fuchsbauer and Wolf [18] present a blind signature scheme that outputs a signature which can be verified with the single-party Schnorr verification algorithm. However, their constructions require zero-knowledge proofs that the challenge – that is itself the output from a hash function – is derived correctly, which requires significant performance overhead and

[1] One motivation for statistical and/or perfect blindness is the looming threat of quantum attacks, which would affect the blindness of current schemes more than they would affect one-more unforgeability, for which the use of quantum-safe assumptions, while important, still remains less critical.

additional complexity. Also concurrently to this work, Barretto and Zanon [4] present blind signatures that are concurrently secure in the random oracle model; however, they rely on a non-black-box adversary. They do not consider the threshold setting.

Threshold signatures. Most relevant to us, there has been significant work on obtaining efficient threshold signature schemes for Schnorr signatures. For example, FROST [5,23] presents a two-round threshold signature scheme that is concurrently secure. Other concurrently-secure Schnorr threshold signatures exist that trade off efficiency for robustness [40] or a direct reduction to standard assumptions in the random oracle model [25]. However, a naive approach to blinding these schemes can open the door to ROS attacks [8].

Threshold blind signatures have been considered in prior literature, notably in a setting that requires pairings [24]. Our schemes, however, are pairing-free.

Threshold credential issuance. Coconut [39] is a Threshold Issuance Anonymous Credential (TIAC) system that enables a set of certification authorities to jointly and blindly issue credentials. While the construction is practical, it was presented without a formal security analysis. Rial and Piotrowska [33] proved a modified scheme secure in the UC setting. Both schemes are built upon a threshold variant of Pointcheval-Sanders (PS) signatures [31], which rely on pairings.

2 Preliminaries

Notation. Let $\kappa \in \mathbb{N}$ denote the security parameter and 1^κ its unary representation. A function $\nu : \mathbb{N} \rightarrow \mathbb{R}$ is called *negligible* if for all $c \in \mathbb{R}, c > 0$, there exists $k_0 \in \mathbb{N}$ such that $|\nu(k)| < \frac{1}{k^c}$ for all $k \in \mathbb{N}, k \geq k_0$. For a non-empty set S, let $x \leftarrow\!\!\$\ S$ denote sampling an element of S uniformly at random and assigning it to x. We use $[n]$ to represent the set $\{1, \dots, n\}$ and represent vectors as $\vec{a} = (a_1, \dots, a_n)$. We denote $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$.

Let PPT denote probabilistic polynomial time. Algorithms are randomized unless explicitly noted otherwise. Let $y \leftarrow A(x;\omega)$ denote running algorithm A on input x and randomness ω and assigning its output to y. Let $y \leftarrow\!\!\$\ A(x)$ denote $y \leftarrow A(x;\omega)$ for a uniformly random ω.

Code-based games are used in security definitions [7]. A game $\mathsf{Game}^{\mathrm{sec}}_{\mathcal{A}}(\kappa)$, played with respect to a security notion sec and adversary $\mathcal{A}$, has a MAIN procedure whose output is the output of the game.

Group generators and discrete logs. Throughout this paper, a *group (parameter) generator* GrGen is a polynomial-time algorithm that takes as input a security parameter 1^κ and outputs a group description $\mathcal{G} = (\mathbb{G}, p, g)$ consisting of a group $\mathbb{G}$ of order p, where p is a κ-bit prime, and a generator g of $\mathbb{G}$.

Definition 1 (Discrete Logarithm Assumption (DL)). *The discrete logarithm assumption holds with respect to* GrGen *if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{A},\mathsf{GrGen}}(\kappa) = \Pr[\mathcal{G} \leftarrow\!\!\$\ \mathsf{GrGen}(1^\kappa);\ x \leftarrow\!\!\$\ \mathbb{Z}_p;\ x \leftarrow\!\!\$\ \mathcal{A}(\mathcal{G}, g^x)]$ *is negligible.*

The Algebraic Group Model. We will make use of the algebraic group model (AGM) [16] throughout this paper, which in particular only proves security for *algebraic* adversaries. Somewhat informally, we say that an adversary is *algebraic* if for every group element $Z \in \mathbb{G} = \langle g \rangle$ that it outputs, it is required to output a representation $\vec{a} = (a_0, a_1, a_2, \dots)$ such that $Z = g^{a_0} \prod Y_i^{a_i}$, where $Y_1, Y_2, \dots \in \mathbb{G}$ are group elements that the adversary has seen thus far.

Polynomial Interpolation. Let $\mathbb{F}$ be a field of size at least t, and let $\mathcal{S} \subseteq \mathbb{F}$ be such that $|\mathcal{S}| = t$. Then, any set of at least t evaluations $(i, P(i))_{i \in \mathcal{S}}$ for a polynomial $P(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_{t-1} z^{t-1}$ of degree $t-1$ over $\mathbb{F}$ can be interpolated to evaluate the polynomial on any other point $z_0 \in \mathbb{F}$ as $P(z_0) = \sum_{i \in \mathcal{S}} P(k) \cdot L_i(z_0)$, where $L_i(z)$ is the Lagrange coefficient of form

$$L_i(z) = \prod_{j \in \mathcal{S}; j \neq i} \frac{z-j}{i-j} \,. \tag{1}$$

(Here, the set $\mathcal{S}$ is usually implicit in L_i.)

Shamir Secret Sharing. We will employe Shamir's secret sharing scheme [37] in our threshold blind signature construction.

- $\mathsf{Share}(x, n, t) \to \{(1, x_1), \dots, (n, x_n)\}$: Define a polynomial $P(z) = x + a_1 z + a_2 z^2 + \dots + a_{t-1} z^{t-1}$ by sampling $t-1$ random coefficients $a_1, \dots, a_{t-1} \leftarrow\$ \mathbb{Z}_p$. Output the set of participant shares $\{(i, x_i)\}_{i \in [n]}$, where each $x_i, i \in [n]$, is the evaluation of $P(i)$: $x_i \leftarrow x + \sum_{j \in [t-1]} a_j i^j$.
- $\mathsf{Recover}(t, \{(i, x_i)\}_{i \in \mathcal{S}}) \to x$: The recover algorithm takes as input t shares and returns the original secret. Recover x as follows: $x \leftarrow \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} x_i$, where the Lagrange coefficient for the set $\mathcal{S}$ is defined by $\lambda_i^{\mathcal{S}} = L_i(0) = \prod_{j \in \mathcal{S}, j \neq i} \frac{j}{j-i}$.

3 Definitions

3.1 Blind Signatures

A **blind signature scheme** BS allows a user to interact with an issuer to obtain a valid signature over a message m unknown to the issuer. Importantly, when later presented with this signature, the issuer cannot link it to any particular signing execution. BS is parameterized by the number of rounds r required to perform the signing protocol. The public parameters par are generated by a trusted party and given as input to all other algorithms. A public/private key pair is generated by running $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{BS.KeyGen}()$. To collectively produce a signature, the issuer and user engage in an interactive signing protocol as shown in Eq. 2, wherein the issuer takes as input the secret key sk, but not the message, and the user takes as input the public key pk and the message m. At the end of the signing protocol, the user outputs the blind signature σ, that is valid if $\mathsf{BS.Verify}(\mathsf{pk}, \sigma, m) = 1$.

Definition 2. *A* **blind signature scheme** BS *parameterized by the number of signing rounds r is a tuple of polynomial-time algorithms* $\mathsf{BS} = (\mathsf{BS.Setup}, \mathsf{BS.KeyGen}, \{\mathsf{BS.ISign}_j\}_{j=1}^r, \{\mathsf{BS.USign}_j\}_{j=1}^r, \mathsf{BS.Verify})$*, as follows.*

$\mathsf{BS.Setup}(1^\kappa) \rightarrow \mathsf{par}$*: Accepts as input a security parameter κ and outputs public parameters* par*, which are then implicitly provided as input to all other algorithms.*

$\mathsf{BS.KeyGen}() \rightarrow (\mathsf{pk}, \mathsf{sk})$*: A probabilistic algorithm that generates and outputs a keypair, where* pk *is the public key and* sk *is the secret key.*

The interaction between the user and the issuer to sign a message $m \in \{0,1\}^*$ with respect to pk *is defined by the following experiment:*

$$\begin{aligned}
&(\mathsf{st}^I, \mathsf{pm}_1^I) \leftarrow \mathsf{BS.ISign}_1(\mathsf{sk})\ ,\ (\mathsf{st}^U, \mathsf{pm}_1^U) \leftarrow \mathsf{BS.USign}_1(\mathsf{pk}, m, \mathsf{pm}_1^I)\\
&(\mathsf{st}^I, \mathsf{pm}_j^I) \leftarrow \mathsf{BS.ISign}_j(\mathsf{st}^I, \mathsf{pm}_{j-1}^U)\ ,\ (\mathsf{st}^U, \mathsf{pm}_j^U) \leftarrow \mathsf{BS.USign}_j(\mathsf{st}^U, \mathsf{pm}_{j-1}^I)\\
&\mathsf{pm}_r^I \leftarrow \mathsf{BS.ISign}_r(\mathsf{st}^I, \mathsf{pm}_{r-1}^U)\ ,\ \bot/\sigma \leftarrow \mathsf{BS.USign}_r(\mathsf{st}^U, \mathsf{pm}_r^I)
\end{aligned} \tag{2}$$

In the above experiment, st^I *is the internal state of the issuer,* st^U *is the internal state of the user.* pm^I *is a protocol message sent by the issuer, and* pm^U *is a protocol message sent by the user.*

$\mathsf{BS.Verify}(\mathsf{pk}, \sigma, m) \rightarrow \{0,1\}$*: A deterministic algorithm that outputs a bit indicating if the signature is valid with respect to the message and public key.*

A blind signature scheme is *correct* if for every $m \in \{0,1\}^*$ and for $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{BS.KeyGen}()$, the experiment in (2) returns σ such that $\mathsf{BS.Verify}(\mathsf{pk}, \sigma, m) = 1$. For security, a blind signature scheme must be *one-more unforgeable* and *blind*, which we describe next.

One-More Unforgeability. The standard notion of security for non-blind signature schemes, EUF-CMA security, cannot be applied to the blind setting, as the reduction cannot detect if the signature output by the adversary is a forgery or a valid signature. Instead, one must employ the notion of *one-more unforgeability.* Intuitively, one-more unforgeability requires an adversary that is allowed to query the signing oracle ℓ times to produce $\ell + 1$ valid signatures, guaranteeing that at least one is forged. We consider the setting where ℓ is unbounded and determined dynamically, as opposed to requiring a fixed ℓ.

We show the one-more unforgeability experiment $\mathsf{Game}_{\mathcal{A},\mathsf{BS}}^{\mathrm{omuf}}(\kappa)$ in the full version.

Definition 3 (One More Unforgeability). *Let the advantage of an adversary $\mathcal{A}$ against the one-more unforgeability game* $\mathsf{Game}_{\mathcal{A},\mathsf{BS}}^{\mathrm{omuf}}(\kappa)$ *be as follows:*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{BS}}^{\mathrm{omuf}}(\kappa) = \Pr[\mathsf{Game}_{\mathcal{A},\mathsf{BS}}^{\mathrm{omuf}}(\kappa) = 1]$$

A blind signature scheme BS *is* one-more unforgeable *if for all PPT adversaries $\mathcal{A}$, there exists a negligible function ν such that* $\mathsf{Adv}_{\mathcal{A},\mathsf{BS}}^{\mathrm{omuf}}(\kappa) < \nu(k)$.

Blindness. We employ a similar notion of blindness as in prior literature [17, 41], and rely on a right-or-left indistinguishability-based definition. Intuitively, a signature scheme achieves blindness if the adversary has negligible chance of distinguishing two signatures with respect to two messages of its choosing. Our schemes satisfy the stronger notion of *perfect blindness*, where the adversary's advantage is zero.

We show the blindness experiment $\mathsf{Game}^{\text{blind}}_{\mathcal{A},\mathsf{BS}}(\kappa)$ in the full version.

Definition 4 (Perfect Blindness). *Let the advantage of an adversary $\mathcal{A}$ against the blindness game $\mathsf{Game}^{\text{blind}}_{\mathcal{A},\mathsf{BS}}(\kappa)$ be as follows:*

$$\mathsf{Adv}^{\text{blind}}_{\mathcal{A},\mathsf{BS}}(\kappa) = |\Pr[\mathsf{Game}^{\text{blind}}_{\mathcal{A},\mathsf{BS}}(\kappa) = 1] - 1/2|$$

A blind signature scheme BS *satisfies* perfect blindness *if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\text{blind}}_{\mathcal{A},\mathsf{BS}}(\kappa) = 0$.*

3.2 Threshold Blind Signatures

A threshold blind signature is an interactive signing protocol between a single user and multiple issuers, each with a share of the secret signing key. Similar to blind signatures, a threshold blind signature scheme should satisfy *correctness*, *blindness*, and *one-more unforgeability*. We present formal security definitions, generalizing the notions of one-more unforgeability and blindness from the single-party setting to the threshold setting. Game-based notions of security for threshold blind signatures have been given in prior literature [24], however, our notions explicitly model important details such as concurrency and session management.

A threshold blind signature scheme TB is similarly parameterized by the number of signing rounds r. The public parameters par are generated by a trusted party and given as input to all other algorithms. Key generation is described in a centralized manner with respect to the number of issuers n and threshold t, where public/private key pairs are generated for all n issuers, as well as the joint public key representing all n issuers. To collectively produce a signature, a quorum $\mathcal{S}$ of issuers interact with the user in an interactive signing protocol as defined in Eq. 3, wherein each issuer takes as input its secret key $\mathsf{sk_i}$, but not the message, and the user takes as input the public key pk, the message m, and the signing set $\mathcal{S}$. For a valid signature to be issued, it must be the case that $t \leq \mathcal{S} \leq n$. At the end of the signing protocol, the user outputs the threshold blind signature σ and each issuer learns the set $\mathcal{S}$ of issuers that are involved in the signing protocol. The signature σ on m is valid if $\mathsf{TB.Verify}(\mathsf{pk}, m, \sigma) = 1$.

Definition 5. *A* **threshold blind signature scheme** TB *parameterized by the number of signing rounds r is a tuple of polynomial-time algorithms* $\mathsf{TB} = (\mathsf{TB.Setup}, \mathsf{TB.KeyGen}, \{\mathsf{TB.ISign}_j\}_{j=1}^r, \{\mathsf{TB.USign}_j\}_{j=1}^r, \mathsf{TB.Verify})$*, as follows.*

$\mathsf{TB.Setup}(1^\kappa) \rightarrow \mathsf{par}$*: Accepts as input a security parameter κ and outputs public parameters* par*, which are then implicitly provided as input to all other algorithms.*

$\mathsf{TB.KeyGen}(n,t) \rightarrow (\mathsf{pk}, \{\mathsf{pk}_1,\ldots,\mathsf{pk}_n\}, \{\mathsf{sk}_1,\ldots,\mathsf{sk}_n\}, \mathsf{aux})$*: A probabilistic algorithm that accepts as input the number of signers* n *and the threshold* t*. Outputs the public key* pk *representing the set of all signers, the set* $\{\mathsf{pk}_1,\ldots,\mathsf{pk}_n\}$ *of public keys representing each issuer, the set* $\{\mathsf{sk}_1,\ldots,\mathsf{sk}_n\}$ *of secret keys for each issuer, and additional auxiliary information* aux.

The interaction between the user and a set of issuers $\mathcal{S}, t \le |\mathcal{S}| \le n$*, to sign a message* $m \in \{0,1\}^*$ *with respect to* pk *is defined by the following experiment:*

$$\begin{aligned}
&(\mathsf{st}_{\mathsf{i}}^I, \mathsf{pm}_{1,\mathsf{i}}^I) \leftarrow \mathsf{TB.ISign}_1(\mathsf{i}, \mathsf{sk}_{\mathsf{i}}, \mathsf{aux}) \ , \ (\mathsf{st}^U, \mathsf{pm}_1^U) \leftarrow \mathsf{TB.USign}_1(\mathsf{pk}, \mathsf{aux}, m, \mathcal{S}, \{\mathsf{pm}_{1,\mathsf{i}}^I\}_{\mathsf{i}\in\mathcal{S}})\\
&(\mathsf{st}_{\mathsf{i}}^I, \mathsf{pm}_{j,\mathsf{i}}^I) \leftarrow \mathsf{TB.ISign}_j(\mathsf{i}, \mathsf{st}_{\mathsf{i}}^I, \mathsf{pm}_{j-1}^U) \ , \ (\mathsf{st}^U, \mathsf{pm}_j^U) \leftarrow \mathsf{TB.USign}_j(\mathsf{st}^U, \{\mathsf{pm}_{j-1,\mathsf{i}}^I\}_{\mathsf{i}\in\mathcal{S}})\\
&(\mathsf{pm}_{r,\mathsf{i}}^I, \mathcal{S}) \leftarrow \mathsf{TB.ISign}_r(\mathsf{i}, \mathsf{st}_{\mathsf{i}}^I, \mathsf{pm}_{r-1}^U) \ , \ \perp/\sigma \leftarrow \mathsf{TB.USign}_r(\mathsf{st}^U, \{\mathsf{pm}_{r,\mathsf{i}}^I\}_{\mathsf{i}\in\mathcal{S}})
\end{aligned} \tag{3}$$

Note that in the last round, $\mathsf{TB.ISign}_r$ *also outputs* $\mathcal{S}$ *indicating that issuer* i *learns the set of issuers involved in the signing protocol.*

$\mathsf{TB.Verify}(\mathsf{pk}, \sigma, m) \rightarrow \{0,1\}$*: A deterministic algorithm that outputs a bit indicating if the signature is valid with respect to the message and public key.*

A threshold blind signature scheme is *correct* if for all allowable $1 \le t \le n$, for all $t \le \mathcal{S} \le n$, all messages $m \in \{0,1\}^*$, and for $(\mathsf{pk}, \{\mathsf{pk}_1,\ldots,\mathsf{pk}_n\}, \{\mathsf{sk}_1,\ldots,\mathsf{sk}_n\}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{TB.KeyGen}(n,t)$, the experiment in (3) returns σ such that $\mathsf{TB.Verify}(\mathsf{pk}, \sigma, m) = 1$.

Distributed key generation. Our definition considers a centralized key generation algorithm $\mathsf{TB.KeyGen}$ to generate the public key pk and set of shares $\{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]}$. However, our scheme and proofs can be adapted to use a fully decentralized distributed key generation protocol, such as the Pedersen DKG [30].

One-More Unforgeability. We show the one-more unforgeability game for a threshold blind signature in Fig. 1. Compared to the single-signer notion of unforgeability, the adversary is allowed to participate in the signing protocol in the role of both user and issuer. The adversary is allowed to choose the parameters n and t, as well as the set of honest signers honest and the set of corrupt signers $\mathsf{corrupt}$, not to exceed $t-1$. The environment then performs key generation in a centralized manner with respect to these parameters. The adversary is given as input the public parameters, the joint public key pk, the set of public key shares for each participant $\{\mathsf{pk}_{\mathsf{i}}\}_{\mathsf{i}\in[n]}$, and the set of secret key shares for the corrupted parties $\{\mathsf{sk}_{\mathsf{i}}\}_{\mathsf{i}\in\mathsf{corrupt}}$. Additionally, the adversary is given an auxiliary string aux. When playing the role of the user, the adversary can query the signing oracle $\mathcal{O}^{\mathsf{ISign}_k}$ for each round k in the protocol, for any issuer $\mathsf{i} \in \mathsf{honest}$, session identifier sid, and protocol message pm^U of its choosing.

The adversary wins the one-more unforgeability game if it outputs a set of $\ell+1$ valid signatures with respect to the joint public key pk, for messages of its choosing, where ℓ denotes the number of signatures that are legitimately obtained by the adversary. For a particular signing set $\mathcal{S}$ and a session identifier

```
MAIN $\mathsf{Game}^{\text{omuf-t}}_{\mathcal{A},\mathsf{TB}}(\kappa)$
$\mathsf{par} \leftarrow \mathsf{TB.Setup}(1^\kappa)$
$\ell \leftarrow 0$   // count total # of signing queries
$S_1, \ldots, S_r \leftarrow \emptyset$   // opened signing sessions
$S_{\mathsf{fin}} \leftarrow \emptyset$   // finished signing sessions
$(n, t, \mathsf{corrupt}, \mathsf{st}^{\mathcal{A}}) \leftarrow\$ \mathcal{A}(\mathsf{par})$
return $\perp$ if $n < t$ or $|\mathsf{corrupt}| \geq t$
$\mathsf{honest} \leftarrow [n] \setminus \mathsf{corrupt}$
$(\mathsf{pk}, \{\mathsf{pk_i}, \mathsf{sk_i}\}_1^n, \mathsf{aux}) \leftarrow\$ \mathsf{TB.KeyGen}(n, t)$
$\theta \leftarrow (\mathsf{pk}, \{\mathsf{pk_i}\}_1^n, \{\mathsf{sk_i}\}_{\mathsf{i} \in \mathsf{corrupt}}, \mathsf{aux})$
$\{(m_k^*, \sigma_k^*)\}_{k \in [\ell+1]} \leftarrow\$ \mathcal{A}^{\mathcal{O}^{\mathsf{ISign}_1, \ldots, \mathsf{ISign}_r}}(\mathsf{st}^{\mathcal{A}}, \theta)$
    // $\mathcal{A}$ must output $\ell + 1$
    // message/signature pairs
for all $k \in [\ell+1], i \in [\ell+1], k \neq i$
    return 0 if $(m_k^*, \sigma_k^*) = (m_i^*, \sigma_i^*)$
    // ensure no duplicates
for all $k \in [\ell+1]$
    return 0 if $\mathsf{TB.Verify}(\mathsf{pk}, m_k^*, \sigma_k^*) \neq 1$
return 1

$\mathcal{O}^{\mathsf{ISign}_1}(\mathsf{i}, \mathsf{sid})$
return $\perp$ if $(\mathsf{i}, \mathsf{sid}) \in S_1$
$S_1 \leftarrow S_1 \cup \{(\mathsf{i}, \mathsf{sid})\}$
$(\mathsf{st}^I_{\mathsf{i,sid}}, \mathsf{pm}^I_{1,\mathsf{i,sid}}) \leftarrow \mathsf{TB.ISign}_1(\mathsf{sk_i}, \mathsf{aux})$
return $\mathsf{pm}^I_{1,\mathsf{i,sid}}$

$\mathcal{O}^{\mathsf{ISign}_j}(\mathsf{i}, \mathsf{sid}, \mathsf{pm}^U_{\mathsf{sid}})$   // $j \in \{2, \ldots, r\}$
return $\perp$ if $(\mathsf{i}, \mathsf{sid}) \notin S_1, \ldots, S_{j-1}$
    // ensure prior rounds have been queried
return $\perp$ if $(\mathsf{i}, \mathsf{sid}) \in S_j$
    // ensure this round has not yet been queried
$S_j \leftarrow S_j \cup \{(\mathsf{i}, \mathsf{sid})\}$
┌ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
    // for signing round $j < r$
  $(\mathsf{st}^I_{\mathsf{i,sid}}, \mathsf{pm}^I_{j,\mathsf{i,sid}}) \leftarrow \mathsf{TB.ISign}_j(\mathsf{st}^I_{\mathsf{i,sid}}, \mathsf{pm}^U_{\mathsf{sid}})$
└ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
┌──────────────────────────────────────────┐
    // for the last signing round $j = r$
  $(\mathsf{pm}^I_{r,\mathsf{i,sid}}, \mathcal{S}) \leftarrow \mathsf{TB.ISign}_r(\mathsf{st}^I_{\mathsf{i,sid}}, \mathsf{pm}^U_{\mathsf{sid}})$
  // in the final round, issuers additionally
  // output the signing set $\mathcal{S}$
  if $(\mathsf{sid}, \mathcal{S}) \notin S_{\mathsf{fin}}$ then
      $S_{\mathsf{fin}} \leftarrow S_{\mathsf{fin}} \cup \{(\mathsf{sid}, \mathcal{S})\}$
      $\ell \leftarrow \ell + 1$
└──────────────────────────────────────────┘
return $\mathsf{pm}^I_{j,\mathsf{i,sid}}$
```

Fig. 1. The one-more unforgeability game for a threshold blind signature scheme. The dashed box appears only for Round $2 \leq j < r$, and the solid box appears only for Round r. The public parameters par are implicitly given as input to all algorithms.

sid, there is at most one signature issued, contingent on the adversary completing the signing session with at least one of the honest issuers.

Definition 6 (One-More Unforgeability). *Let the advantage of an adversary $\mathcal{A}$ against the one-more unforgeability game $\mathsf{Game}^{\text{omuf-t}}_{\mathcal{A},\mathsf{TB}}(\kappa)$, as defined in Fig. 1, be as follows:*

$$\mathsf{Adv}^{\text{omuf-t}}_{\mathcal{A},\mathsf{TB}}(\kappa) = |\Pr[\mathsf{Game}^{\text{omuf-t}}_{\mathcal{A},\mathsf{TB}}(\kappa) = 1]|$$

A threshold blind signature scheme TB *satisfies* one-more unforgeability *if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\text{omuf-t}}_{\mathcal{A},\mathsf{TB}}(\kappa)$ is negligible.*

Blindness. We now extend the definition of blindness to the threshold setting. The key difference in the threshold blindness experiment is that the user interacts with *multiple* issuers, as opposed to a single issuer. Hence, the adversary

```
MAIN $\mathsf{Game}^{\text{blind-t}}_{\mathcal{A},\mathsf{TB}}(\kappa)$
$\mathsf{par} \leftarrow \mathsf{TB.Setup}(1^\kappa)$
$S_1, \ldots, S_r \leftarrow \emptyset$   // opened signing sessions
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}^{\mathsf{USign}_1,\ldots,\mathsf{USign}_r}}(\mathsf{par})$
return 0 if $b' \neq b$
return 1

$\mathcal{O}^{\mathsf{USign}_1}(\mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, \mathsf{aux}_{\mathsf{sid}}, m_{0,\mathsf{sid}}, m_{1,\mathsf{sid}}, \mathcal{S}_{0,\mathsf{sid}}, \mathcal{S}_{1,\mathsf{sid}}, \{\mathsf{pm}^I_{0,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{0,\mathsf{sid}}}, \{\mathsf{pm}^I_{1,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{1,\mathsf{sid}}})$
    // $\mathcal{S}_{i,\mathsf{sid}} \subseteq [n]$ is the set of signers chosen by $\mathcal{A}$ for signing session $i \in \{0,1\}$.
return $\perp$ if $\mathsf{sid} \in S_1$
$S_1 \leftarrow S_1 \cup \{\mathsf{sid}\}$
$(\mathsf{st}^U_{0,\mathsf{sid}}, \mathsf{pm}^U_{0,1,\mathsf{sid}}) \leftarrow \mathsf{TB.USign}^{(1)}_1(\mathsf{pk}_{\mathsf{sid}}, \mathsf{aux}_{\mathsf{sid}}, m_{b,\mathsf{sid}}, \mathcal{S}_{0,\mathsf{sid}}, \{\mathsf{pm}^I_{0,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{0,\mathsf{sid}}})$
$(\mathsf{st}^U_{1,\mathsf{sid}}, \mathsf{pm}^U_{1,1,\mathsf{sid}}) \leftarrow \mathsf{TB.USign}^{(2)}_1(\mathsf{pk}_{\mathsf{sid}}, \mathsf{aux}_{\mathsf{sid}}, m_{1-b,\mathsf{sid}}, \mathcal{S}_{1,\mathsf{sid}}, \{\mathsf{pm}^I_{1,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{1,\mathsf{sid}}})$
return $(\mathsf{pm}^U_{0,1,\mathsf{sid}}, \mathsf{pm}^U_{1,1,\mathsf{sid}})$

$\mathcal{O}^{\mathsf{USign}_j}(\mathsf{sid}, \{\mathsf{pm}^I_{0,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{0,\mathsf{sid}}}, \{\mathsf{pm}^I_{1,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{1,\mathsf{sid}}})$   // $j \in \{2,\ldots,r\}$
return $\perp$ if $\mathsf{sid} \notin S_1, \ldots, S_{j-1}$   // ensure prior rounds have been queried
return $\perp$ if $\mathsf{sid} \in S_j$   // ensure this round has not yet been queried
$S_j \leftarrow S_j \cup \{\mathsf{sid}\}$
$\boxed{\sigma_{b,\mathsf{sid}}}$ $(\mathsf{st}^U_{0,\mathsf{sid}}, \mathsf{pm}^U_{0,j,\mathsf{sid}})$ $\leftarrow \mathsf{TB.USign}^{(1)}_j(\mathsf{st}^U_{0,\mathsf{sid}}, \{\mathsf{pm}^I_{0,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{0,\mathsf{sid}}})$
$\boxed{\sigma_{1-b,\mathsf{sid}}}$ $(\mathsf{st}^U_{1,\mathsf{sid}}, \mathsf{pm}^U_{1,j,\mathsf{sid}})$ $\leftarrow \mathsf{TB.USign}^{(2)}_j(\mathsf{st}^U_{1,\mathsf{sid}}, \{\mathsf{pm}^I_{1,\mathsf{i},\mathsf{sid}}\}_{\mathsf{i}\in\mathcal{S}_{1,\mathsf{sid}}})$
$\boxed{\textbf{return } (\perp,\perp) \textbf{ if } \sigma_{0,\mathsf{sid}} = \perp \textbf{ or } \sigma_{1,\mathsf{sid}} = \perp}$
return $\boxed{(\sigma_{0,\mathsf{sid}}, \sigma_{1,\mathsf{sid}})}$ $(\mathsf{pm}^U_{0,j,\mathsf{sid}}, \mathsf{pm}^U_{1,j,\mathsf{sid}})$
```

Fig. 2. The blindness game for a threshold blind signature scheme. The public parameters par are implicitly given as input to all algorithms. Dashed boxes denote Rounds 2 to $r-1$, and solid boxes denote Round r only.

```
$\mathsf{Setup}(1^\kappa)$
    $(\mathbb{G}, p, g) \leftarrow\$ \mathsf{GrGen}(1^\lambda)$
    $h \leftarrow\$ \mathbb{G}$
    Select $\mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathbb{Z}_p$
    $\mathsf{par} \leftarrow ((\mathbb{G}, p, g, h), \mathsf{H}_{\mathsf{sig}})$
    return $\mathsf{par}$

$\mathsf{KeyGen}()$
    $\mathsf{sk} \leftarrow\$ \mathbb{Z}_p;\ \mathsf{pk} \leftarrow g^{\mathsf{sk}}$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Sign}(\mathsf{sk}, m)$
    $r, y \leftarrow\$ \mathbb{Z}_p^*;\ R \leftarrow g^r h^y$
    $c \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, R)$
    $z \leftarrow r + f(c, y) \cdot \mathsf{sk}$
    $\sigma \leftarrow (R, z, y)$
    return $\sigma$

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
    parse $(R, z, y) \leftarrow \sigma$
    return $0$ if $y = 0$
    $c \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, R)$
    return $0$ if $R \cdot \mathsf{pk}^{f(c,y)} \neq g^z h^y$
    return $1$
```

Fig. 3. Our base (non-blind) signature scheme. We can instantiate it with any bivariate function f such that $f(X, y)$ is invertible for all $y \in \mathbb{Z}_p^*$. The public parameters par are implicitly given as input to all algorithms.

queries the oracle $\mathcal{O}^{\mathsf{USign}_1}$ with a public key and two messages of its choosing. Additionally, the adversary is allowed to choose disjoint signing sets $\mathcal{S}_0, \mathcal{S}_1 \subseteq [n]$ and two sets of protocol messages. Hence, the adversary could corrupt *all* issuers, not just a threshold number of them, and the scheme should still preserve blindness. The blindness game for threshold blind signatures is specified in Fig. 2. Our scheme satisfies the stronger notion of *perfect blindness*, where the adversary's advantage in winning the blindness game is zero.

Definition 7 (Perfect Blindness). *Let the advantage of an adversary $\mathcal{A}$ against the threshold blindness game $\mathsf{Game}^{\text{blind-t}}_{\mathcal{A},\mathsf{TB}}(\kappa)$, as defined in Fig. 2, be as follows:*

$$\mathsf{Adv}^{\text{blind-t}}_{\mathcal{A},\mathsf{TB}}(\kappa) = |\Pr[\mathsf{Game}^{\text{blind-t}}_{\mathcal{A},\mathsf{TB}}(\kappa) = 1] - 1/2|$$

A threshold blind signature scheme TB *satisfies* perfect blindness *if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\text{blind-t}}_{\mathcal{A},\mathsf{TB}}(\kappa) = 0$.*

4 Blind Signature Scheme BS

In this section, we present our construction of a blind signature scheme BS. We begin by constructing a base (non-blind) scheme (Fig. 3), from which our blind signature scheme is derived. The base scheme can be instantiated in two different ways, which are parameterized by a non-linear function $f : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_p$.

Explicitly, we consider two possibilities: $f_1(c, y) = c + y^5$ and $f_2(c, y) = cy$. A signature is then given by $\sigma = (R, z, y)$, where

$$R = g^r h^y \qquad \text{and} \qquad z = r + f(c, y) \cdot \mathsf{sk}$$

for $c = \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, R)$. We prove the EUF-CMA security in the full version. We then blind the base scheme by replacing the Sign algorithm with an interactive protocol between an issuer running ISign and a user running USign, wherein the issuer does not learn the message m. The signature size with either function f is one group element plus two scalars, which is only one more scalar than a Schnorr signature [35].

Our blind signature scheme BS is specified in Fig. 4. Of technical interest is the fact the security reductions for the two parametrization options f_1 and f_2 use very different techniques, and yet achieve identical efficiency and security assumptions.

The issuer in the blind protocol behaves identically for either function f. Initially, the issuer sends some g^a and $g^b h^y$ for random a, b, y. The user returns a blinded challenge c. In the second round, the issuer returns the opening (b, y) in the clear, and the response $z = a + f(c, y) \cdot \mathsf{sk}$. This is described formally in Fig. 4.

The user in the blind protocol behaves slightly differently for f_1 and f_2. In Fig. 4, we show the corresponding algorithms USign_1 and USign_2 for each function f. Let us informally describe the user for f_1. In USign_1, the user computes a nonce and challenge as

$$\bar{R} = g^{r + \alpha^5 a + \alpha^5 \beta \mathsf{sk} + \alpha b} h^{\alpha y} \qquad \text{and} \qquad \bar{c} = \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$$

blinded by the random values r, α, β. Note that, up to this point, the random value y is completely hidden from the user. Thus, a malicious user is unable to include factors of g^{y^5} in the nonce $\bar{R}$, preventing potential ROS attacks. Now, the user returns $c = \bar{c}\alpha^{-5} + \beta$ to the issuer, which is randomized by α and β. Then, in USign_2, upon receiving (z, b, y) the user sets

$$\bar{z} \leftarrow r + \alpha^5 z + \alpha b \qquad \text{and} \qquad \bar{y} = \alpha y$$

where $\bar{y}$ is randomized by α. Now $\bar{z}$ is the unique value that satisfies the verifier's equation given $\bar{R}, \bar{z}$. Thus, the final signature $\sigma = (\bar{R}, \bar{z}, \bar{y})$ reveals no information about the session.

If instead f_2 is used, in USign_1 the user computes a nonce and challenge as

$$\bar{R} = g^{r + \alpha\beta^{-1} a + \alpha b} h^{\alpha y} \qquad \text{and} \qquad \bar{c} = \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$$

blinded by the random values r, α, β. The user returns $c = \beta\bar{c}$ to the issuer, which is randomized by β. Then, in USign_2, upon receiving (z, b, y) the user sets

$$\bar{z} \leftarrow r + \alpha\beta^{-1} z + \alpha b \qquad \text{and} \qquad \bar{y} = \alpha y$$

where $\bar{y}$ is randomized by α. Now $\bar{z}$ is the unique value that satisfies the verifier's equation given $\bar{R}, \bar{z}$. Thus, the final signature $(\bar{R}, \bar{z}, \bar{y})$ reveals no information

```
$\mathsf{BS.Setup}(1^\kappa)$
    $(\mathbb{G}, p, g) \leftarrow\$ \mathsf{GrGen}(1^\kappa)$; $h \leftarrow\$ \mathbb{G}$
    Select $\mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathbb{Z}_p^*$
    $\mathsf{par} \leftarrow ((\mathbb{G}, p, g, h), \mathsf{H}_{\mathsf{sig}}$
    return $\mathsf{par}$

$\mathsf{BS.KeyGen}()$
    $\mathsf{sk} \leftarrow\$ \mathbb{Z}_p$; $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
    return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{BS.Verify}(\mathsf{pk}, m, \sigma)$
    parse $(\bar{R}, \bar{z}, \bar{y}) \leftarrow \sigma$
    $\bar{c} \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$
    if $\bar{y} = 0$ or $\bar{R} \cdot \mathsf{pk}^{f(\bar{c}, \bar{y})} \neq g^{\bar{z}} h^{\bar{y}}$
        return 0
    return 1

$\mathsf{BS.ISign}(\mathsf{sk})$                                  $\mathsf{BS.USign}(\mathsf{pk}, m)$
    $a, b, \leftarrow\$ \mathbb{Z}_p$; $y \leftarrow\$ \mathbb{Z}_p^*$
    $A \leftarrow g^a$; $B \leftarrow g^b h^y$      $\xrightarrow[A,B]{}$      $(\mathsf{st}^U, c) \leftarrow\$ \mathsf{USign}_1(\mathsf{pk}, m, A, B)$
    $z \leftarrow a + f(c, y) \cdot \mathsf{sk}$    $\xleftarrow[c]{}$
                                                   $\xrightarrow[z,b,y]{}$    return $\mathsf{USign}_2(\mathsf{st}^U, z, b, y)$
```

```
$f_1(c, y)$
    return $c + y^5$

$\mathsf{BS}[f_1].\mathsf{USign}_1(\mathsf{pk}, m, A, B)$
    $\alpha \leftarrow\$ \mathbb{Z}_p^*$; $r, \beta \leftarrow\$ \mathbb{Z}_p$
    $\bar{R} \leftarrow g^r A^{\alpha^5} \mathsf{pk}^{\alpha^5 \beta} B^\alpha$
    $\bar{c} \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$
    $c \leftarrow \bar{c}\alpha^{-5} + \beta$
    $\mathsf{st}^U \leftarrow (\bar{R}, r, \alpha, \beta)$
    return $(\mathsf{st}^U, c)$

$\mathsf{BS}[f_1].\mathsf{USign}_2(\mathsf{st}^U, z, b, y)$
    return $\perp$ if $B \neq g^b h^y$
    return $\perp$ if $g^z \neq A\mathsf{pk}^{c+y^5}$
    $\mathsf{st}^U \leftarrow (\bar{R}, r, \alpha, \beta)$
    $\bar{z} \leftarrow r + \alpha^5 z + \alpha b$; $\bar{y} \leftarrow \alpha y$
    $\sigma \leftarrow (\bar{R}, \bar{z}, \bar{y})$
    return $\perp$ if $\mathsf{BS.Verify}(\mathsf{pk}, m, \sigma) = 0$
    return $\sigma$
```

```
$f_2(c, y)$
    return $c \cdot y$

$\mathsf{BS}[f_2].\mathsf{USign}_1(\mathsf{pk}, m, A, B)$
    $\alpha, \beta \leftarrow\$ \mathbb{Z}_p^*$; $r \leftarrow\$ \mathbb{Z}_p$
    $\bar{R} \leftarrow g^r A^{\alpha\beta^{-1}} B^\alpha$
    $\bar{c} \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$
    $c \leftarrow \beta\bar{c}$
    $\mathsf{st}^U \leftarrow (\bar{R}, r, \alpha, \beta)$
    return $(\mathsf{st}^U, c)$

$\mathsf{BS}[f_2].\mathsf{USign}_2(\mathsf{st}^U, z, b, y)$
    return $\perp$ if $B \neq g^b h^y$
    return $\perp$ if $g^z \neq A\mathsf{pk}^{c \cdot y}$
    $\mathsf{st}^U \leftarrow (\bar{R}, r, \alpha, \beta)$
    $\bar{z} \leftarrow r + \alpha\beta^{-1} z + \alpha b$; $\bar{y} \leftarrow \alpha y$
    $\sigma \leftarrow (\bar{R}, \bar{z}, \bar{y})$
    return $\perp$ if $\mathsf{BS}_2.\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 0$
    return $\sigma$
```

Fig. 4. Top: The two-round blind signature scheme $\mathsf{BS}[\mathsf{GrGen}, f]$. Bottom: Two ways of instantiating f and the corresponding $\mathsf{USign}_1, \mathsf{USign}_2$. The power 5 may be replaced with any power $q < \mathsf{poly}(\kappa)$ for which $\mathsf{gcd}(q, p-1) = 1$.

$$\begin{array}{lcl}
\underline{\mathsf{BSr3.ISign}(\mathsf{sk})} & & \underline{\mathsf{BSr3.USign}(\mathsf{pk}, m)} \\
a, b, y \leftarrow\!\!\$\; \mathbb{Z}_p & & \\
A \leftarrow g^a;\; B \leftarrow g^b h^y & \xrightarrow[A,B]{\hspace{4em}} & \text{pick } s \in \mathbb{Z}_p \text{ arbitrarily} \\
 & \xleftarrow[c,s]{\hspace{4em}} & (\mathsf{st}^U, c) \leftarrow\!\!\$\; \mathsf{BS}[f].\mathsf{USign}_1(\mathsf{pk}, m, A, h^s B) \\
 & \xrightarrow[b,y]{\hspace{4em}} & \\
 & \xleftarrow[\perp]{\hspace{4em}} & \\
z \leftarrow a + f(c, y+s) \cdot \mathsf{sk} & \xrightarrow[z]{\hspace{4em}} & \textbf{return } \mathsf{BS}[f].\mathsf{USign}_2(\mathsf{st}^U, z, b, y+s)
\end{array}$$

Fig. 5. The signing protocol of blind signature scheme $\mathsf{BSr3}[f]$, which is a three-round version of $\mathsf{BS}[f]$. Two instantiations of f, $\mathsf{BS}[f].\mathsf{USign}_1$ and $\mathsf{BS}[f].\mathsf{USign}_2$ are shown in Fig. 4.

about the session. When we instantiate our blind signature with f_2, the scheme draws many parallels with [41]; however, the resulting signature is more efficient.

We prove that BS is perfectly blind and one-more unforgeable under the discrete logarithm assumption in the AGM and the ROM. When $f(c, y) = c+y^5$, correct simulation computes 5^{th} roots, so these roots must exist and be unique. We chose the power 5 in our construction for ease of exposition and because they often exist in practice; however, our proofs hold for any prime p and power $q < \mathsf{poly}(\kappa)$ for which unique roots exist (i.e., when $\mathsf{gcd}(q, p-1) = 1$).

Single use and secure state keeping. Our schemes require choosing values uniformly at random, and require that these values be used *strictly* once; otherwise, all security is lost. Like many multi-round protocols, this assumption requires secure state keeping. Our definitions model this state keeping via session identifiers, and implementations of our schemes will similarly need to ensure secure state keeping, to prevent nonce misuse.

4.1 One-More Unforgeability

To demonstrate the one-more unforgeability of BS, we introduce a three-round variant $\mathsf{BSr3}$ (Fig. 5). $\mathsf{BS.Setup}$, $\mathsf{BS.KeyGen}$ and $\mathsf{BS.Verify}$ are identical in both schemes, but the signing protocols differ in two ways. First, the user can additionally pick a scalar s that varies y generated by the signer. Second, z is sent to the user in an additional round. We show that the unforgeability of $\mathsf{BSr3}$ implies the unforgeability of BS in Lemma 1. Indeed, the signing oracles in the one-more unforgeability game of BS can be simulated by the signing oracles in the one-more unforgeability game of $\mathsf{BSr3}$. We introduce $\mathsf{BSr3}$ as an intermediate step towards proving security for our threshold blind signature scheme $\mathsf{Snowblind}$, presented in the next section. The relationships between our blind and threshold blind constructions and the assumptions on which they rely are outlined in Fig. 6.

$$\begin{tikzcd}
\mathrm{DL} \arrow[r, "\text{Thm. 1,2 (AGM+ROM)}"] & \mathsf{BSr3} \arrow[r, "\text{Lem. 1}"] \arrow[dr, bend right, "\text{Thm. 5 (ROM)}"] & \mathsf{BS} \\
 & & \mathsf{Snowblind}
\end{tikzcd}$$

Fig. 6. Underlying assumptions for our blind and threshold blind signature constructions. DL denotes the Discrete Logarithm problem, AGM denotes the Algebraic Group Model, and ROM denotes the Random Oracle Model.

Lemma 1. *Let* GrGen *be a group generator. For any* $f \in \{f_1, f_2\}$*, and nay adversary* $\mathcal{A}$ *for the game* $\mathsf{Game}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BS}[\mathsf{GrGen},f]}$ *there exists an adversary* $\mathcal{B}$ *for the game* $\mathsf{Game}^{\mathrm{omuf}}_{\mathcal{B},\mathsf{BSr3}[\mathsf{GrGen},f]}$ *making the same number of oracle queries as* $\mathcal{A}$ *running in a similar running time as* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BS}[\mathsf{GrGen},f]}(\kappa) = \mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{B},\mathsf{BSr3}[\mathsf{GrGen},f]}(\kappa)$$

Proof. Let $\mathcal{A}$ be an adversary against the one-more unforgeability of the two-round protocol in Fig. 4. We construct an adversary $\mathcal{B}$ against the one-more unforgeability of the three-round protocol in Fig. 5 as follows.

$\mathcal{B}$ has access to its own signing oracles, denoted by $\hat{\mathcal{O}}^{\mathsf{ISign}_1,\mathsf{ISign}_2,\mathsf{ISign}_3}$. Upon receiving a challenge public key pk, $\mathcal{B}$ runs $\mathcal{A}^{\mathcal{O}^{\mathsf{ISign}_1,\mathsf{ISign}_2}}(\mathsf{pk})$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{ISign}_1}$, $\mathcal{B}$ queries $\hat{\mathcal{O}}^{\mathsf{ISign}_1}$ and receives (A, B), which it returns to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{Sign}_2}$ on c, $\mathcal{B}$ queries its $\hat{\mathcal{O}}^{\mathsf{ISign}_2}$ oracle on $(c, 0)$ to get (b, y). Next, $\mathcal{B}$ queries $\hat{\mathcal{O}}^{\mathsf{Sign}_3}$ and receives z. $\mathcal{B}$ returns (b, y, z) to $\mathcal{A}$. When $\mathcal{A}$ returns a forgery $\{(m_k, \sigma_k)\}_{k=1}^{\ell+1}$, $\mathcal{B}$ outputs the same forgery.

If $\mathcal{A}$ succeeds, then both $\mathcal{A}$ and $\mathcal{B}$ have made fewer than ℓ final-round queries, and $\mathcal{B}$'s forgery also verifies. Thus, $\mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BS}[\mathsf{GrGen},f]}(\kappa) = \mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{B},\mathsf{BSr3}[\mathsf{GrGen},f]}(\kappa)$. □

We prove that $\mathsf{BSr3}$ is one-more unforgeable under the discrete logarithm assumption in the algebraic group model (AGM) and random oracle model (ROM) for both instantiations of f and provide proof outlines here. The game models the hash function $\mathsf{H}_{\mathsf{sig}}$ as a random oracle, which on each different input outputs an uniformly random value over $\mathbb{Z}_p^*$, and to which the adversary is given oracle access. The full proofs can be found in the full version.

Theorem 1. *Let* GrGen *be a group generator. For any algebraic adversary* $\mathcal{A}$ *for the game* $\mathsf{Game}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BSr3}[\mathsf{GrGen},f_1]}$ *making at most* q_S *signing queries and* q_H *queries to the random oracle, there exists adversaries* $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$ *for the discrete logarithm problem running in a similar running time as* $\mathcal{A}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BSr3}[\mathsf{GrGen},f_1]}(\kappa) \leq & q_S \mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{B}_0,\mathsf{GrGen}}(\kappa) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{B}_1,\mathsf{GrGen}}(\kappa) \\ & + \mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{B}_2,\mathsf{GrGen}}(\kappa) + \frac{q_S + 2 + 4q_H + q_S q_H^2}{p}\end{aligned}$$

where p *denotes the group size.*

Proof Outline. Suppose the session id is from 1 to q_S. For session id $i \in [q_S]$, denote the output of $\mathcal{O}^{\mathsf{ISign}_1}$ as (A_i, B_i), the input of $\mathcal{O}^{\mathsf{ISign}_2}$ as (c_i, s_i), the output of $\mathcal{O}^{\mathsf{ISign}_2}$ as (b_i, y_i), and the output of $\mathcal{O}^{\mathsf{ISign}_3}$ as z_i. An adversary $\mathcal{A}$ wins the one-more unforgeability game if it returns distinct forged signatures $\{(m_k^*, \sigma_k^* = (\bar{R}_k^*, \bar{z}_k^*, \bar{y}_k^*)\}_{k \in [\ell+1]}$ satisfying the verification equation:

$$\bar{R}_k^* = g^{\bar{z}_k^*} h^{\bar{y}_k^*} \mathsf{pk}^{-(\bar{c}_k^* + (\bar{y}_k^*)^5)}$$

where $\bar{c}_k^* = \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m_k^*, \bar{R}_k^*)$. An algebraic adversary must also output a representation of each $\bar{R}_k^*$:

$$\bar{R}_k^* = g^{\varsigma_k^*} h^{\eta_k^*} \mathsf{pk}^{\chi_k^*} \prod_{i \in [q_S]} A_i^{\rho_{k,i}^*} B_i^{\tau_{k,i}^*} \ .$$

To prove the theorem, we define a series of games, beginning with the one-more unforgeability game $\mathsf{Game}_{\mathcal{A},\mathsf{BSr3}}^{\mathrm{omuf}}$ and concluding with a game Game_2 in which $\mathcal{A}$ wins if the following conditions hold: (1) $\rho_{k,i}^* = 0$ for each (A_i, B_i) queried in the first round with no corresponding (i, c_i) query in the second round, and (2) $\bar{y}_k^*$ satisfies the following equation:

$$\bar{y}_k^* = \eta_k^* + \sum_{i \in [q_S]} y_i \tau_{k,i}^* \ .$$

Intuitively, these conditions say that each $\bar{R}_k^*$ must have a representation over completed signing sessions only and that $\bar{R}_k^*$ must commit to $\bar{y}_k^*$. We show that if $\mathcal{A}$ wins Game_2, its responses must satisfy a polynomial expression in the secret key sk, and a reduction $\mathcal{B}_2$ can use this expression to compute sk with overwhelming probability.

We first jump to a hybrid game Game_1 in which condition (1) holds but not (2). To do this we design a reduction $\mathcal{B}_0$ that randomly selects a signing query $i' \in [q_S]$ for which the condition fails, and can compute the discrete logarithm of $A_{i'}$ with overwhelming probability. To jump from Game_1 to Game_2, we show that if $\mathcal{A}$ wins Game_1, its responses must satisfy a polynomial expression in the discrete logarithm ω of h, which a reduction $\mathcal{B}_1$ can use to compute ω with overwhelming probability.

The most technical aspect of our proof is constructing the reduction $\mathcal{B}_2$. Indeed, we demonstrate that whenever $\mathcal{A}$ wins Game_2, then either $\mathcal{B}_2$ returns the secret key as

$$\mathsf{sk} = \frac{-\varsigma_k^* - \sum_{i \in S_3} (z_i \rho_{k,i}^* + b_i \tau_{k,i}^*) + \bar{z}_k^*}{\chi_k^* + \bar{c}_k^* + (\eta_k^* + \sum_{i \in [q_S]} y_i \tau_{k,i}^*)^5 - \sum_{i \in S_3} (c_i + (y_i + s_i)^5) \rho_{k,i}^*} \ ,$$

where S_3 is defined in the game denoting the set of completed signing sessions, or the denominator is zero:

$$\chi_k^* + \bar{c}_k^* + (\eta_k^* + \sum_{i \in [q_S]} y_i \tau_{k,i}^*)^5 - \sum_{i \in S_3} (c_i + (y_i + s_i)^5) \rho_{k,i}^* = 0 \ . \tag{4}$$

We then proceed with a timing argument that shows Eq. (4) holds with probability less than the probability that the *one-dimensional* ROS problem has a solution. The one-dimensional ROS problem is proven to be statistically hard in [17, Lemma 2]. □

Theorem 2. *Let* GrGen *be a group generator. For any algebraic adversary* $\mathcal{A}$ *for the game* $\mathsf{Game}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BSr3}[\mathsf{GrGen},f_2]}$ *making at most* q_S *signing queries and* q_H *queries to the random oracle, there exists adversaries* $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$ *for the discrete logarithm problem running in a similar running time as* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{A},\mathsf{BSr3}[\mathsf{GrGen},f_2]}(\kappa) \leq q_S \cdot \mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{B}_0,\mathsf{GrGen}}(\kappa) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{B}_1,\mathsf{GrGen}}(\kappa) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{B}_2,\mathsf{GrGen}}(\kappa) + \frac{(q_S+1)(q_H+q_S+1)^2}{p-1}$$

where p *denotes the group size.*

Proof Outline. We follow the same technique as f_1 to construct $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$ and switch to Game_2 in which: (1) $\rho^*_{k,i} = 0$ for each (A_i, B_i) queried in the first round with no corresponding query for i in the third round, and (2) $\bar{y}^*_k$ satisfies $\bar{y}^*_k = \eta^*_k + \sum_{i\in[q_S]} y_i \tau^*_{k,i}$. In Game_2 we can similarly show that either the reduction $\mathcal{B}_2$ returns the secret key sk, or

$$\chi^*_k + \bar{c}^*_k(\eta^*_k + \sum_{i\in[q_S]} y_i \tau^*_{k,i}) - \sum_{i\in S_3} c_i(y_i + s_i)\rho^*_{k,i} = 0\ . \tag{5}$$

Our proof that Eq. (5) holds with negligible probability differs substantially from our proof for f_1. Here, we reduce it to a modified version of the WFROS problem [41] which has been shown to be information-theoretically hard. The reduction and the hardness proof of the modified WFROS problem follow from similar ideas from [41]. More details are given in the full version. □

Remark 1. The main difference between the modified WFROS problem and the original WFROS game is that the adversary is allowed to send an additional offset s_i in each query to one of the oracles, which leads to an additional loss factor of q_S in the advantage bound. However, this is because we are proving the OMUF of the more complex three-round scheme $\mathsf{BSr3}$ which is stronger than the OMUF of BS. In fact, we can remove the q_S factor and get the same bound as [41] for the OMUF advantage of $\mathsf{BS}[\mathsf{GrGen}, f_2]$.

4.2 Blindness

The following two theorems establish the perfect blindness of $\mathsf{BS}[\mathsf{GrGen}, f_1]$ and $\mathsf{BS}[\mathsf{GrGen}, f_2]$. (The proofs are very similar.)

Theorem 3. *Let* GrGen *be a group generator. Then, the blind signature scheme* $\mathsf{BS}[\mathsf{GrGen}, f_1]$ *is perfectly blind.*

Proof. Let $\mathcal{A}$ be an adversary playing $\mathsf{Game}^{\text{blind}}_{\mathcal{A},\mathsf{BS}[f_1]}(\kappa)$ against the blind signature scheme as described in Fig. 4. Without loss of generality, we assume the randomness of $\mathcal{A}$ is fixed. As we prove perfect blindness, we can focus on adversaries that only run one signing session, i.e., they use a single sid, as security for a more general adversary follows by a standard hybrid argument. Further, we also assume that $\mathcal{A}$ always finishes both signing sessions and receives valid signatures (σ_0, σ_1) from $\mathcal{O}^{\mathsf{USign}_2}$. (Otherwise, the output of $\mathcal{O}^{\mathsf{USign}_1}$ are two blinded challenges which are both uniformly random over $\mathbb{Z}_p$, and blindness trivially holds.)

Let V_A denote the set of all possible views of $\mathcal{A}$ that can occur after one single interaction with $\mathcal{O}^{\mathsf{USign}_1}, \mathcal{O}^{\mathsf{USign}_2}$. In particular, any such view $\Delta \in V_A$ takes form $\Delta = (\mathsf{pk}, m_0, m_1, T_0, T_1, \sigma_0, \sigma_1)$. Here, $\sigma_i = (\bar{R}_i, \bar{c}_i, \bar{z}_i, \bar{y}_i)$, where $\bar{c}_i = \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m_i, \bar{R}_i)$. (Note that $\bar{c}_i$ is redundant here, and does not need to be included, but it will make the argument easier.) Moreover, T_0 and T_1 are the signing protocol transcripts for the left and right interactions, respectively, and take form $T_i = (A_i, B_i, c_i, z_i, b_i, y_i)$. We need to show that the distribution of the actual adversarial view, which we denote as v_A, is the same when $b = 0$ and $b = 1$. Because we assume the randomness of $\mathcal{A}$ is fixed, the distribution of v_A only depends on the randomness $\eta = (r_0, \alpha_0, \beta_0, r_1, \alpha_1, \beta_1)$ required to respond to $\mathcal{O}^{\mathsf{USign}_1}$ and $\mathcal{O}^{\mathsf{USign}_2}$ queries, and we write $v_A(\eta)$ to make this fact explicit.

Concretely, fix some $\Delta \in V_A$. We now show that there exists a unique η that makes it occur, i.e., $v_A(\eta) = \Delta$, regardless of whether we are in the $b = 0$ or in the $b = 1$ case. In particular, we claim that, in both cases $b = 0, b = 1$, $v_A(\eta) = \Delta$ if and only if for $i \in \{0, 1\}$, η satisfies

$$\begin{aligned} r_i &= \bar{z}_{\omega_i} - z_i \alpha_i^5 - \alpha_i b_i \\ \alpha_i &= \bar{y}_{\omega_i} / y_i \\ \beta_i &= c_i - \bar{c}_{\omega_i} \alpha_i^{-5} , \end{aligned} \tag{6}$$

where $\omega_0 = b$ and $\omega_1 = 1 - b$.

In the "only if" direction, from Fig. 4, it is clear that when $v_A(\eta) = \Delta$, then η satisfies all constraints in Eq. 6.

To prove the "if" direction, assume that η satisfies all constraints in Eq. 6. We need to show that $v_A(\eta) = \Delta$. This means in particular verifying that the challenges output by $\mathcal{O}^{\mathsf{USign}_1}$ are indeed (c_0, c_1) and the signatures output by $\mathcal{O}^{\mathsf{USign}_2}$ are indeed (σ_0, σ_1).

Note that because we only consider Δ's that result in $\mathcal{O}^{\mathsf{USign}_2}$ not producing output $(\bot, \bot)$, for $i \in \{0, 1\}$, we have $\bar{y}_i \neq 0$, as well as,

$$g^{z_i} = A_i \mathsf{pk}^{c_i + y_i^5} \ , \ \ B_i = g^{b_i} h^{y_i} \ , \ \bar{R}_{\omega_i} = g^{\bar{z}_{\omega_i}} h^{\bar{y}_{\omega_i}} \mathsf{pk}^{-\bar{c}_{\omega_i} - \bar{y}^5_{\omega_i}} \ .$$

Therefore, using Eq. 6,

$$\begin{aligned} \bar{R}_{\omega_i} &= g^{r_i + z_i \alpha_i^5 + \alpha_i b_i} h^{\alpha_i y_i} \mathsf{pk}^{\alpha_i^5 (\beta_i - c_i - y_i^5)} \\ &= B_i^{\alpha_i} g^{r_i + z_i \alpha_i^5} \mathsf{pk}^{\alpha_i^5 (\beta_i - c_i - y_i^5)} \\ &= g^{r_i} A_i^{\alpha_i^5} B_i^{\alpha_i} \mathsf{pk}^{\alpha_i^5 \beta_i} \ . \end{aligned}$$

Consequently, for $i \in \{0,1\}$, $\mathcal{O}^{\mathsf{USign}_1}$ outputs the challenge

$$\alpha_i^{-5}\mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m_{\omega_i}, g^{r_i}A^{\alpha_i^5}\mathsf{pk}^{\alpha^5\beta_i}B^{\alpha_i}) + \beta_i = \alpha_i^{-5}\mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m_{\omega_i}, \bar{R}_{\omega_i}) + \beta_i = c_i\ ,$$

i.e., the two challenges are consistent with the view $\varDelta$. Furthermore, for $i \in \{0,1\}$, the signatures $(\tilde{\sigma}_1, \tilde{\sigma}_2)$ output by $\mathcal{O}^{\mathsf{USign}_2}$ are such that

$$\tilde{\sigma}_{\omega_i} = (g^{r_i}A_i^{\alpha_i^5}B_i^{\alpha_i}\mathsf{pk}^{\alpha_i^5\beta_i}, r_i + \alpha_i^5 z_i + \alpha_i b_i, \alpha_i y_i) = (\bar{R}_{\omega_i}, \bar{z}_{\omega_i}, \bar{y}_{\omega_i}) = \sigma_{\omega_i}\ ,$$

i,.e., these are exactly the signatures from $\varDelta$. □

Theorem 4. *Let* GrGen *be a group generator. Then, the blind signature scheme* BS[GrGen, f_2] *is perfectly blind.*

The proof is very similar to that of Theorem 3, and we defer it to the full version.

5 Threshold Blind Signature Scheme Snowblind

Here we present Snowblind, an efficient threshold blind signature scheme (Fig. 7). Snowblind extends single-party blind signing to the multi-issuer setting. In this setting, the user determines the signing set $\mathcal{S}$, such that $t \leq |\mathcal{S}| \leq n$. The user plays the role of the coordinator of the protocol; each issuer interacts directly with the user, and the user relays protocol messages between issuers for each round, for a total of three signing rounds. At the end of the protocol, the user aggregates the signature shares received from each issuer and publishes the resulting signature.

We provide a modular approach to proving the one-more unforgeability (OMUF) of Snowblind (Fig. 1). Indeed, we are able to reduce the OMUF of Snowblind to the OMUF of our three-round blind signature scheme BSr3 in Sect. 4, which more closely resembles the structure of Snowblind. We cannot directly reduce to the OMUF of our more efficient two-round scheme BS because in the simulation, the BS adversary might make more queries to its final-round signing oracle than the Snowblind adversary does, resulting in an invalid forgery. Preventing the adversary from forging signatures when it only queries the preliminary rounds is important in our asynchronous and concurrent model, where we can make no termination guarantees.

Concretely, we employ a centralized key generation mechanism, but, alternatively, a distributed key generation protocol (DKG) could be used. The public parameters par generated during setup are provided as input to all other algorithms and protocols. We assume some external mechanism to choose the set of signers $\mathcal{S} \subseteq \{1, \ldots, n\}$, where $t \leq |\mathcal{S}| \leq n$ and $\mathcal{S}$ is ordered to ensure consistency. Snowblind additionally makes use of a standard EUF-CMA-secure single-party signature scheme DS, used to authenticate messages sent in the signing rounds.

Parameter Generation. On input the security parameter 1^κ, the setup algorithm runs $(\mathbb{G}, p, g) \leftarrow_\$ \mathsf{GrGen}(1^\kappa)$ and selects a random group element $h \leftarrow_\$ \mathbb{G}$

```
$\mathsf{Snowblind.Setup}(1^\kappa)$
$(\mathbb{G}, p, g) \leftarrow\!\!\$\ \mathsf{GrGen}(1^\kappa);\ h \leftarrow\!\!\$\ \mathbb{G}$
**select two hash functions** $\mathsf{H}_{\mathsf{cm}}, \mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathbb{Z}_p^*$
$\mathsf{par}_{\mathsf{sig}} \leftarrow \mathsf{DS.Setup}(1^\kappa)$
**return** $\mathsf{par} \leftarrow ((\mathbb{G}, p, g, h), \mathsf{H}_{\mathsf{cm}}, \mathsf{H}_{\mathsf{sig}}, \mathsf{par}_{\mathsf{sig}})$

$\mathsf{Snowblind.KeyGen}(n, t)$
$\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p; \mathsf{pk} \leftarrow g^{\mathsf{sk}}$
$\{(\mathsf{i}, \mathsf{sk_i})\}_{\mathsf{i} \in [n]} \leftarrow\!\!\$\ \mathsf{Share}(\mathsf{sk}, n, t)$
  // Shamir secret sharing of sk
**for** $\mathsf{i} \in [n]$ **do**
  $\mathsf{pk_i} \leftarrow g^{\mathsf{sk_i}};\ (\hat{\mathsf{pk}}_\mathsf{i}, \hat{\mathsf{sk}}_\mathsf{i}) \leftarrow \mathsf{DS.KeyGen}()$
$\mathsf{aux} \leftarrow \{\hat{\mathsf{pk}}_\mathsf{i}\}_{\mathsf{i} \in [n]}$
**return** $(\mathsf{pk}, \{\mathsf{pk_i}\}_{\mathsf{i} \in [n]} \{\mathsf{sk_i}, \hat{\mathsf{sk}}_\mathsf{i}\}_{\mathsf{i} \in [n]}, \mathsf{aux})$

$\mathsf{Snowblind.Verify}(\mathsf{pk}, m, \sigma)$
**parse** $(\bar{R}, \bar{z}, \bar{y}) \leftarrow \sigma$
$\bar{c} \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$
**if** $\bar{y} = 0$ **or** $\bar{R} \cdot \mathsf{pk}^{f(\bar{c}, \bar{y})} \neq g^{\bar{z}} h^{\bar{y}}$
    **return** 0
**return** 1

$\mathsf{Snowblind.ISign}(\mathsf{i}, \mathsf{sk_i}, \mathsf{aux})$                          $\mathsf{Snowblind.USign}(\mathsf{pk}, \mathsf{aux}, m, \mathcal{S})$
parse $\{\hat{\mathsf{pk}}_\mathsf{i}\}_{\mathsf{i} \in [n]} \leftarrow \mathsf{aux}$                         parse $\{\hat{\mathsf{pk}}_\mathsf{i}\}_{\mathsf{i} \in [n]} \leftarrow \mathsf{aux}$
**abort if** $\mathsf{i} \notin \mathcal{S}$                  $\xleftarrow{\mathcal{S}}$
$a_\mathsf{i}, b_\mathsf{i} \leftarrow\!\!\$\ \mathbb{Z}_p;\ y_\mathsf{i} \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$\mathsf{cm_i} \leftarrow \mathsf{H}_{\mathsf{cm}}(\mathsf{sid}, \mathsf{i}, y_\mathsf{i})$
$A_\mathsf{i} \leftarrow g^{a_\mathsf{i}};\ B_\mathsf{i} \leftarrow g^{b_\mathsf{i}} h^{y_\mathsf{i}}$      $\xrightarrow{A_\mathsf{i}, B_\mathsf{i}, \mathsf{cm_i}}$   $A \leftarrow \prod_{j \in \mathcal{S}} A_j,\ B \leftarrow \prod_{j \in \mathcal{S}} B_j$
$\mathsf{msg} \leftarrow (\mathsf{sid}, \mathcal{S}, c, \{\mathsf{cm}_j\}_{j \in \mathcal{S}})$   $\xleftarrow{c, \{\mathsf{cm}_j\}_{j \in \mathcal{S}}}$   $(\mathsf{st}^U, c) \leftarrow\!\!\$\ \mathsf{BS}[f].\mathsf{USign}_1(\mathsf{pk}, m, A, B)$
$\sigma_i \leftarrow \mathsf{DS.Sign}(\hat{\mathsf{sk}}_\mathsf{i}, \mathsf{msg})$          $\xrightarrow{b_\mathsf{i}, y_\mathsf{i}, \sigma_\mathsf{i}}$
**abort if** $\exists\, j$ such that              $\xleftarrow{\{\sigma_j, y_j\}_{j \in \mathcal{S}}}$
  $\mathsf{cm}_j \neq \mathsf{H}_{\mathsf{cm}}(\mathsf{sid}, j, y_j)$ **or**
  $\mathsf{DS.Verify}(\hat{\mathsf{pk}}_j, \mathsf{msg}, \sigma_j) \neq 1$
$y \leftarrow \sum_{j \in \mathcal{S}} y_j$                                 $y \leftarrow \sum_{j \in \mathcal{S}} y_j$
$z_i \leftarrow a_i + f(c, y) \cdot (\lambda_\mathsf{i}^{\mathcal{S}} \mathsf{sk_i})$       $\xrightarrow{z_i}$   $z = \sum_{j \in \mathcal{S}} z_j;\ \ b \leftarrow \sum_{j \in \mathcal{S}} b_j$
                                                   $(\bar{R}, \bar{z}, \bar{y}) \leftarrow \mathsf{BS}[f].\mathsf{USign}_2(\mathsf{st}^U, z, b, y)$
                                                   **return** $\sigma \leftarrow (\bar{R}, \bar{z}, \bar{y})$
```

Fig. 7. The signing protocol of the threshold blind signature scheme $\mathsf{Snowblind}[\mathsf{GrGen}, \mathsf{DS}, f]$ derived from our blind signature scheme $\mathsf{BSr3}[f]$ (Fig. 5), where DS is an arbitrary EUF-CMA-secure digital signature scheme. sid denotes the id of the signing session. Snowblind assumes an external mechanism to choose the set $\mathcal{S} \subseteq \{1, \ldots, n\}$ of signers, where $t \leq |\mathcal{S}| \leq n$. $\mathcal{S}$ is required to be ordered to ensure consistency. Each issuer must respond to each round in a session no more than once or else all security is lost. Implementations of our scheme should ensure secure state keeping as described in our definitions.

as well as two hash functions $\mathsf{H}_{\mathsf{cm}}, \mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \rightarrow \mathbb{Z}_p$. It also runs the setup algorithm for a signature scheme $\mathsf{par}_{\mathsf{sig}} \leftarrow \mathsf{DS.Setup}(1^\kappa)$ used for authentication in Signing Rounds 1 and 2. It outputs public parameters $\mathsf{par} \leftarrow ((\mathbb{G}, p, g, h), \mathsf{H}_{\mathsf{cm}}, \mathsf{H}_{\mathsf{sig}}, \mathsf{par}_{\mathsf{sig}})$.

Key Generation. On input the number of signers n and the threshold t, this algorithm first generates the secret key $\mathsf{sk} \leftarrow\$ \mathbb{Z}_p$ and joint public key $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$. It then performs Shamir secret sharing of sk: $\{(\mathsf{i}, \mathsf{sk}_\mathsf{i})\}_{\mathsf{i}\in[n]} \leftarrow\$ \mathsf{Share}(\mathsf{sk}, n, t)$. It computes the corresponding public key for each participant as $\mathsf{pk}_\mathsf{i} \leftarrow g^{\mathsf{sk}_\mathsf{i}}$. It then runs the key generation algorithm $(\hat{\mathsf{pk}}_\mathsf{i}, \hat{\mathsf{sk}}_\mathsf{i}) \leftarrow \mathsf{DS.KeyGen}()$. It sets $\mathsf{aux} \leftarrow \{\hat{\mathsf{pk}}_\mathsf{i}\}_{\mathsf{i}\in[n]}$. To guarantee identification of misbehaving issuers by verifying each issuer's signature share (i.e., identifiable abort), aux may additionally include the set of public key shares $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$. Finally, it outputs $(\mathsf{pk}, \{\mathsf{pk}_\mathsf{i}\}_{\mathsf{i}\in[n]}\{\mathsf{sk}_\mathsf{i}, \hat{\mathsf{sk}}_\mathsf{i}\}_{\mathsf{i}\in[n]}, \mathsf{aux})$.

Signing Round 1. In the first round, the issuers compute a shared nonce $A = g^a$ and $B = g^b h^y$. On input a signing set $\mathcal{S}$ determined by the user, each issuer $\mathsf{i} \in \mathcal{S}$, chooses random values $a_\mathsf{i}, b_\mathsf{i} \leftarrow\$ \mathbb{Z}_p, y_\mathsf{i} \leftarrow\$ \mathbb{Z}_p^*$, computes a commitment $\mathsf{cm}_\mathsf{i} \leftarrow \mathsf{H}_{\mathsf{cm}}(\mathsf{sid}, \mathsf{i}, y_\mathsf{i})$ and two nonces $A_\mathsf{i} \leftarrow g^{a_\mathsf{i}}, B_\mathsf{i} \leftarrow g^{b_\mathsf{i}} h^{y_\mathsf{i}}$, and outputs $(A_\mathsf{i}, B_\mathsf{i}, \mathsf{cm}_\mathsf{i})$.

The user receives the set of all $\{(A_j, B_j, \mathsf{cm}_j)\}_{j\in\mathcal{S}}$, from which it computes the aggregate nonces $A \leftarrow \prod_{j\in\mathcal{S}} A_j, B \leftarrow \prod_{j\in\mathcal{S}} B_j$. The user then computes the blinded challenge $c \leftarrow\$ \mathsf{BS}[f].\mathsf{USign}_1(\mathsf{pk}, m, A, B)$ on the message m and outputs it together with the set of commitments $\{\mathsf{cm}_j\}_{j\in\mathcal{S}}$. Here $\mathsf{BS}[f].\mathsf{USign}_1$ is the same algorithm as described for the non-threshold blind signature in Fig. 4.

Signing Round 2. In the second round, the issuers jointly reveal b and y such that $B = g^b h^y$. On input the set of commitments $\{\mathsf{cm}_j\}_{j\in\mathcal{S}}$ and challenge c, each issuer $\mathsf{i} \in \mathcal{S}$ forms the message $\mathsf{msg} = (\mathsf{sid}, \mathcal{S}, c, \{\mathsf{cm}_j\}_{j\in\mathcal{S}})$ and runs the signing algorithm $\sigma_i \leftarrow \mathsf{DS.Sign}(\hat{\mathsf{sk}}_\mathsf{i}, \mathsf{msg})$, used to authenticate the messages sent in Signing Rounds 1 and 2. Each issuer then outputs their committed values $b_\mathsf{i}, y_\mathsf{i}$ and signature σ_i.

The user receives the set of all $\{b_j, y_j, \sigma_j\}_{j\in\mathcal{S}}$ and echoes $\{y_j, \sigma_j\}_{j\in\mathcal{S}}$ back to all parties in the signing set.

Signing Round 3. In the third and final round, the issuers jointly compute $z \leftarrow a + f(c, y) \cdot \mathsf{sk}$. On input $\{(\sigma_j, y_j)\}_{j\in\mathcal{S}}$, each issuer i first checks that the commitments received in the first round are valid, i.e., $\mathsf{cm}_j = \mathsf{H}_{\mathsf{cm}}(\mathsf{sid}, j, y_j)$ for all $j \in \mathcal{S}$ and aborts if for some j, $\mathsf{cm}_j \neq \mathsf{H}_{\mathsf{cm}}(\mathsf{sid}, j, y_j)$.

This ensures that no malicious issuer can cancel out the honest contributions to y.

It then checks that all signatures σ_j verify: $\mathsf{DS.Verify}(\hat{\mathsf{pk}}_j, \mathsf{msg}, \sigma_j) = 1$ and aborts if not. The signature ensures that the honest signing parties all agree on the shared y.

Otherwise, issuer i computes the aggregate y-value $y \leftarrow \sum_{j\in\mathcal{S}} y_j$. It then computes the value $f(c, y)$ according to the chosen base blind signature scheme (Fig. 4) and $z_\mathsf{i} \leftarrow a_\mathsf{i} + f(c, y) \cdot (\lambda_\mathsf{i}^{\mathcal{S}} \mathsf{sk}_\mathsf{i})$, where $\lambda_\mathsf{i}^{\mathcal{S}}$ is the i^{th} Lagrange coefficient corresponding to $\mathcal{S}$. The Lagrange coefficients are computed as

shown in Eq. (1). Finally, the issuer outputs z_{i}.
The user receives the set of all $\{z_j\}_{j\in\mathcal{S}}$, from which it computes the aggregate z-value $z \leftarrow \sum_{j\in\mathcal{S}} z_j$ and $y \leftarrow \sum_{j\in\mathcal{S}} y_j$. The user then computes and outputs the blind signature $\sigma \leftarrow \mathsf{BS}[f].\mathsf{USign}_2(z, b, y)$.

Verification. On input the joint public key pk, a message m, and a signature $\sigma = (\bar{R}, \bar{z}, \bar{y})$, the verifier computes $c \leftarrow \mathsf{H}_{\mathsf{sig}}(\mathsf{pk}, m, \bar{R})$ and accepts if $\bar{R} \cdot \mathsf{pk}^{f(\bar{c},\bar{y})} = g^{\bar{z}}h^{\bar{y}}$ and $\bar{y} \neq 0$.

Note that verification of the threshold signature σ is identical to verification of the single-party signatures with respect to the aggregate nonce $\bar{R}$ and joint public key pk.

Complexity Analysis. For a signing session between the user and a set of issuers $\mathcal{S}$, each issuer sends two group elements and one scalar in the first round, two scalars and one signature in the second round, and one scalar in the third round. Therefore, the total communication complexity of each issuer is $2\mathbb{G}+4\mathbb{Z}_p+\mathsf{sig}$, where $\mathbb{G}$ denotes a group element, $\mathbb{Z}_p$ denotes a scalar, and sig denotes a signature of DS. The user sends the set $\mathcal{S}$ in the first round, $(1+|\mathcal{S}|)$ scalars in the second round, and $|\mathcal{S}|$ signatures and $|\mathcal{S}|$ scalars in the third round. Therefore, the total communication complexity of the user is $(2|\mathcal{S}| + 1)\mathbb{Z}_p + |\mathcal{S}|\mathsf{sig}$ plus $|\mathcal{S}| \log(n)$ bits.

For computation, the total computation complexity of each issuer is $3\mathsf{GExp}+\mathsf{GMul}+3\mathsf{SMul}$ plus one signing operation and $|\mathcal{S}|$ verifications of DS, where GExp denotes one group exponentiation, GMul denotes one group multiplication, and SMul denotes one scalar multiplication. The total computation complexity of the user is $6\mathsf{GExp} + (2n + 4)\mathsf{GMul} + 6\mathsf{SMul}$.

One-More Unforgeability. We reduce the OMUF of $\mathsf{Snowblind}$ to the OMUF of our three-round blind signature defined in Fig. 5 and EUF-CMA security of the underlying signature scheme DS, which is formally stated in the following theorem. The OMUF game models the hash functions H_{cm} and $\mathsf{H}_{\mathsf{sig}}$ as random oracles, to which the adversary is given oracle access.

Theorem 5. *Let* GrGen *be a group generator and* DS *be a digital signature scheme. For any adversary* $\mathcal{A}$ *for the game* $\mathsf{Game}^{\text{omuf-t}}_{\mathsf{Snowblind}[\mathsf{GrGen},f,\mathsf{DS}]}$ *making at most* q_S *queries to* $\mathcal{O}^{\mathsf{ISign}_1}$ *and* q_H *queries to the random oracles, there exists an adversary* $\mathcal{B}$ *for the game* $\mathsf{Game}^{\text{omuf}}_{\mathsf{BSr3}[\mathsf{GrGen},f]}$ *making at most* q_S *queries to* $\mathcal{O}^{\mathsf{ISign}_1}$ *and* q_H *queries to the random oracle running in a similar running time as* $\mathcal{A}$ *and an adversary* $\mathcal{C}$ *for the game* $\mathsf{Game}^{\text{euf-cma}}_{\mathsf{DS}}$ *making at most* q_S *queries to* $\mathcal{O}^{\mathsf{Sign}}$ *running in a similar running time as* $\mathcal{A}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\text{omuf-t}}_{\mathcal{A},\mathsf{Snowblind}[\mathsf{GrGen},f,\mathsf{DS}]}(\kappa) \leq &\mathsf{Adv}^{\text{omuf}}_{\mathcal{B},\mathsf{BSr3}[\mathsf{GrGen},f]}(\kappa)\\ &+ \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{C},\mathsf{DS}}(\kappa) + \frac{(2q_S + q_H + 1)(nq_S + q_H)}{p-1}\end{aligned}$$

where n *denotes the number of signers and* p *denotes the group size.*

Let us give some intuition behind the security reduction of Theorem 5. We design a reduction $\mathcal{B}$ that takes as input a public key pk for the 3-round blind signature. $\mathcal{B}$ simulates the key generation process such that the threshold public key is equal to pk. The secret shares of the corrupt parties are chosen by $\mathcal{B}$. The secret keys of the honest parties are unknown by $\mathcal{B}$, but $\mathcal{B}$ internally computes γ_k and δ_k so that $\mathsf{pk}_\mathsf{k} = \mathsf{pk}^{\gamma_\mathsf{k}} g^{\delta_\mathsf{k}}$ for all $\mathsf{k} \in \mathsf{honest}$, where honest denotes the set of honest signers.

We then specify how $\mathcal{B}$ simulates the signature oracles in a manner which is statistically indistinguishable from the real oracles. For simplicity of the explanation here, let us consider a signing session for $\mathcal{S}$ that consists of only honest signers. In the first signing round, $\mathcal{B}$ embeds exactly one $\hat{\mathcal{O}}^{\mathsf{ISign}_1}$ response $(\hat{A}, \hat{B})$ into the messages from honest signers. For each $\mathsf{i} \in \mathcal{S}$, $\mathcal{B}$ sets $A_\mathsf{i} = \hat{A}^{\gamma_\mathsf{i} \lambda_\mathsf{i}^{\mathcal{S}}} g^{\tilde{a}_\mathsf{i}}$ and $B_\mathsf{i} = \hat{B}^{\frac{1}{|\mathcal{S}|}} g^{\tilde{b}_\mathsf{i}} h^{\tilde{y}_\mathsf{i}}$, where $\tilde{a}_\mathsf{i}, \tilde{b}_\mathsf{i}, \tilde{y}_\mathsf{i}$ are sampled randomly.

In the second signing round, $\mathcal{B}$ sets $\hat{s} = \sum_{\mathsf{i} \in \mathcal{S}} \tilde{y}_\mathsf{i}$, queries $\hat{\mathcal{O}}^{\mathsf{ISign}_2}$ on $(c, \hat{s})$, and receives $\hat{b}, \hat{y}$. Then $\mathcal{B}$ sets $b_\mathsf{i} = \frac{\hat{b}}{|\mathcal{S}|} + \tilde{b}_\mathsf{i}$, $y_\mathsf{i} = \frac{\hat{y}}{|\mathcal{S}|} + \tilde{y}_\mathsf{i}$. It is easy to see that $B_\mathsf{i} = g^{b_\mathsf{i}} h^{y_\mathsf{i}}$.

In the third signing round $\mathcal{B}$ gets $\hat{z}$ by querying $\hat{\mathcal{O}}^{\mathsf{ISign}_3}$. Then $\mathcal{B}$ sets $z_\mathsf{i} = \lambda_\mathsf{i}^{\mathcal{S}} \gamma_\mathsf{i} \hat{z} + \tilde{a}_\mathsf{i} + f(c, y) \lambda_\mathsf{i}^{\mathcal{S}} \delta_\mathsf{i}$, where $y = \sum_{\mathsf{i} \in \mathcal{S}} y_\mathsf{i} = \hat{y} + \hat{s}$. It is not hard to see that z_i is correct since $\hat{z} = \hat{a} + f(c, \hat{y} + \hat{s})\mathsf{sk}$ and thus

$$g^{z_\mathsf{i}} = g^{\left[\lambda_\mathsf{i}^{\mathcal{S}} \gamma_\mathsf{i} \hat{a} + \tilde{a}_\mathsf{i}\right] + \left[\lambda_\mathsf{i}^{\mathcal{S}} f(c,y)(\gamma_\mathsf{i} \mathsf{sk} + \delta_\mathsf{i})\right]} = A_\mathsf{i} \cdot (\mathsf{pk}_\mathsf{i})^{\lambda_\mathsf{i}^{\mathcal{S}} f(c,y)} .$$

Proof (of Theorem 5). For $\mathcal{A}$ described in the theorem, we construct an adversary $\mathcal{B}$ for $\mathsf{Game}^{\mathrm{omuf}}_{\mathsf{BSr3}}$ as follows. $\mathcal{B}$ is responsible for simulating oracle responses for the three rounds of signing, and queries to H_cm and H_sig. $\mathcal{B}$ may program H_cm and H_sig. $\mathcal{B}$ has access to its own random oracle, denoted by $\hat{\mathsf{H}}_\mathsf{sig}$, and signing oracles, denoted by $\hat{\mathcal{O}}^{\mathsf{ISign}_1}$, $\hat{\mathcal{O}}^{\mathsf{ISign}_2}$, and $\hat{\mathcal{O}}^{\mathsf{ISign}_3}$, from $\mathsf{Game}^{\mathrm{omuf}}_{\mathsf{BSr3}}$. $\mathcal{B}$ cannot program $\hat{\mathsf{H}}_\mathsf{sig}$ because it is part of $\mathcal{B}$'s challenge. Let Q_cm be the set of H_cm queries and their responses.

To start with, $\mathcal{B}$ receives as input group parameters $\mathcal{G} = (\mathbb{G}, p, g)$ and a challenge public key $\overline{\mathsf{pk}}$ issued by the BSr3 OMUF game. $\mathcal{B}$ randomly samples $h \leftarrow\!\!\$\ \mathbb{G}$. Then, $\mathcal{B}$ initializes Q_cm and $\mathsf{SignQuery}$ to empty sets and also initializes S_1, S_2, S_3 and ℓ as in $\mathsf{Game}^{\mathrm{omuf\text{-}t}}_{\mathsf{Snowblind}}$.

Key Generation. After receiving $(n, t, \mathsf{corrupt}, \mathsf{st}^{\mathcal{A}})$ from $\mathcal{A}$, assuming without loss of generality that $|\mathsf{corrupt}| = t-1$, $\mathcal{B}$ simulates the key generation algorithm as follows. First, $\mathcal{B}$ sets the joint public key $\mathsf{pk} \leftarrow \overline{\mathsf{pk}}$. $\mathcal{B}$ then simulates a Shamir secret sharing of the discrete logarithm of pk by performing the following steps.

1. For all $\mathsf{j} \in \mathsf{corrupt}$, $\mathcal{B}$ samples a random value $x_\mathsf{j} \leftarrow\!\!\$\ \mathbb{Z}_p$ and defines the secret key as $\mathsf{sk}_\mathsf{j} \leftarrow x_\mathsf{j}$ and corresponding public key as $\mathsf{pk}_\mathsf{j} \leftarrow g^{x_\mathsf{j}}$.
2. To generate the public keys of the honest parties $\mathsf{k} \in \mathsf{honest} = [n] \setminus \mathsf{corrupt}$, $\mathcal{B}$ proceeds as follows:
   (a) For all $i \in \tilde{\mathcal{S}} := \mathsf{corrupt} \cup \{0\}$, it computes the Lagrange polynomials evaluated at point k: $\tilde{\lambda}_{\mathsf{k}i} = L_i^{\tilde{\mathcal{S}}}(\mathsf{k}) = \prod_{j \in \tilde{\mathcal{S}}, j \neq i} \frac{(j-\mathsf{k})}{(j-i)}$.

(b) It takes the public keys of the corrupted parties $\{\mathsf{pk_j}\}_{\mathsf{j}\in\mathsf{corrupt}}$ and the joint public key pk and computes: $\mathsf{pk_k} = \mathsf{pk}^{\tilde{\lambda}_{\mathsf{k}0}} \prod_{\mathsf{j}\in\mathsf{corrupt}} \mathsf{pk_j}^{\tilde{\lambda}_{\mathsf{kj}}}$. $\mathcal{B}$ internally sets $\gamma_{\mathsf{k}} \leftarrow \tilde{\lambda}_{\mathsf{k}0}$ and $\delta_{\mathsf{k}} \leftarrow \sum_{\mathsf{j}\in\mathsf{corrupt}} x_{\mathsf{j}}\tilde{\lambda}_{\mathsf{kj}}$ so that $\mathsf{pk_k} = \mathsf{pk}^{\gamma_{\mathsf{k}}} g^{\delta_{\mathsf{k}}}$ for all $\mathsf{k} \in \mathsf{honest}$.

3. For all $\mathsf{i} \in [n]$, $\mathcal{B}$ runs $(\hat{\mathsf{pk}}_{\mathsf{i}}, \hat{\mathsf{sk}}_{\mathsf{i}}) \leftarrow\$ \mathsf{DS.KeyGen}()$.

$\mathcal{B}$ runs $\mathcal{A}^{\mathcal{O}^{\mathsf{ISign}_1,\mathsf{ISign}_2,\mathsf{ISign}_3},\mathsf{H_{cm}},\mathsf{H_{sig}}}(\mathsf{st}^{\mathcal{A}}, \mathsf{pk}, \{\mathsf{pk_i}\}_{i\in[n]}, \{\mathsf{sk_j}, \hat{\mathsf{sk}}_{\mathsf{j}}\}_{\mathsf{j}\in\mathsf{corrupt}}, \mathsf{aux})$ where $\mathsf{aux} \leftarrow \{\hat{\mathsf{pk}}_{\mathsf{i}}\}_{\mathsf{i}\in[n]}$. $\mathcal{B}$ simulates the oracles as follows. In the following, we use the same notations as the $\mathsf{Snowblind}$ protocol to denote variables from the game $\mathsf{Game}^{\mathsf{omuf\text{-}t}}_{\mathsf{Snowblind}}$. We use $\hat{A}, \hat{B}, \hat{b}, \hat{y}, \hat{z}, \hat{s}$ to denote variables from the game $\mathsf{Game}^{\mathsf{omuf}}_{\mathsf{BSr3}}$ and $\tilde{(\cdot)}$ to denote variables generated by $\mathcal{B}$ itself during the simulation.

Hash Queries. When $\mathcal{A}$ queries $\mathsf{H_{cm}}$ on $(\mathsf{sid}, \mathsf{i}, y)$, $\mathcal{B}$ checks whether $((\mathsf{sid}, \mathsf{i}, y), \mathsf{cm}) \in \mathsf{Q_{cm}}$ for some cm and, if so, returns cm. Else, $\mathcal{B}$ samples $\mathsf{cm} \leftarrow\$ \mathbb{Z}_p$, appends $((\mathsf{sid}, \mathsf{i}, y), \mathsf{cm})$ to $\mathsf{Q_{cm}}$, and returns cm.

When $\mathcal{A}$ queries $\mathsf{H_{sig}}$ on (pk, m, R), $\mathcal{B}$ returns $c \leftarrow \hat{\mathsf{H}}_{\mathsf{sig}}(\mathsf{pk}, m, R)$.

Signing Round 1 ($\mathcal{O}^{\mathsf{ISign}_1}$ Queries). When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{ISign}_1}$ on $(\mathsf{i}, \mathsf{sid}, \mathcal{S})$, $\mathcal{B}$ returns $\bot$ if $(\mathsf{i}, \mathsf{sid}) \in S_1$ or $\mathsf{i} \notin \mathcal{S}$. If $(\mathsf{sid}, \mathcal{S}) \in \mathsf{SignQuery}$, which means $\mathcal{A}$ has made a query $(\mathsf{k}, \mathsf{sid}, \mathcal{S})$ for an honest party $\mathsf{k} \in \mathsf{honest} \cap \mathcal{S}$ for the same sid and $\mathcal{S}$ before, then $\mathcal{B}$ looks up the previously computed values $(\tilde{A}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}, \tilde{B}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}, \tilde{\mathsf{cm}}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})})$.

Otherwise, $\mathcal{B}$ sets $\mathsf{SignQuery} \leftarrow \mathsf{SignQuery} \cup \{(\mathsf{sid}, \mathcal{S})\}$, creates a session id for the game $\mathsf{Game}^{\mathsf{omuf}}_{\mathsf{BSr3}}$ as $\mathsf{sid}^{(\mathsf{sid},\mathcal{S})}_{\mathsf{BSr3}} \leftarrow (\mathsf{i}, \mathsf{sid})$, and queries $\hat{\mathcal{O}}^{\mathsf{ISign}_1}$ on $\mathsf{sid}^{(\mathsf{sid},\mathcal{S})}_{\mathsf{BSr3}}$ and receives a nonce pair $(\hat{A}, \hat{B})$. Then, for each $\mathsf{k} \in \mathsf{hon}_{\mathcal{S}}$, $\mathcal{B}$ samples $\tilde{a}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})}$, $\tilde{b}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})} \leftarrow\$ \mathbb{Z}_p, \tilde{y}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})}, \tilde{\mathsf{cm}}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})} \leftarrow\$ \mathbb{Z}_p^*$ and computes

$$\tilde{A}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})} \leftarrow \hat{A}^{\gamma_{\mathsf{k}}\lambda_{\mathsf{k}}^{\mathcal{S}}} g^{\tilde{a}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})}}, \quad \tilde{B}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})} \leftarrow \hat{B}^{\frac{1}{|\mathsf{hon}_{\mathcal{S}}|}} g^{\tilde{b}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})}} h^{\tilde{y}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})}}, \quad \tilde{s}^{(\mathsf{sid},\mathcal{S})} \leftarrow \sum_{\mathsf{k}\in\mathsf{hon}_{\mathcal{S}}} \tilde{y}_{\mathsf{k}}^{(\mathsf{sid},\mathcal{S})}$$

Finally, $\mathcal{B}$ sets $\mathcal{S}_{\mathsf{i},\mathsf{sid}} \leftarrow \mathcal{S}$, $S_1 \leftarrow S_1 \cup \{(\mathsf{i}, \mathsf{sid})\}$, and returns $(A_{\mathsf{i}} \leftarrow \tilde{A}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}$, $B_{\mathsf{i}} \leftarrow \tilde{B}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}$, $\mathsf{cm_i} \leftarrow \tilde{\mathsf{cm}}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})})$ to $\mathcal{A}$.

Signing Round 2 ($\mathcal{O}^{\mathsf{ISign}_2}$ queries). When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{ISign}_2}$ on $(\mathsf{i}, \mathsf{sid}, c,$ $\{\mathsf{cm}_j\}_{j\in\mathcal{S}})$ where $\mathcal{S} = \mathcal{S}_{\mathsf{i},\mathsf{sid}}$, $\mathcal{B}$ checks if $(\mathsf{i}, \mathsf{sid}) \in S_1$, $(\mathsf{i}, \mathsf{sid}) \notin S_2$, and $\mathsf{cm_i} = \tilde{\mathsf{cm}}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}$ and returns $\bot$ if not. If $\mathcal{B}$ has not queried $\hat{\mathcal{O}}^{\mathsf{ISign}_2}$ for session id $\mathsf{sid}^{(\mathsf{sid},\mathcal{S})}_{\mathsf{BSr3}}$, then, for each $j \in \mathcal{S} \cap \mathsf{corrupt}$, $\mathcal{B}$ finds y_j such that $((\mathsf{sid}, j, y_j), \mathsf{cm}_j) \in \mathsf{Q_{cm}}$ and sets $\hat{s} = \tilde{s}^{(\mathsf{sid},\mathcal{S})} + \sum_{j\in\mathcal{S}\cap\mathsf{corrupt}} y_j$. If there exists more than one such y_j for some j, $\mathcal{B}$ aborts and we denote this abort event as $\mathsf{HashColl}$. If such y_j does not exists for some j, $\mathcal{B}$ sets $\hat{s} = 0$ and we denote this event as $\mathsf{BadCm}^{(\mathsf{sid},\mathcal{S})}$. Then, $\mathcal{B}$ queries $\hat{\mathcal{O}}^{\mathsf{ISign}_2}$ on $(\mathsf{sid}^{(\mathsf{sid},\mathcal{S})}_{\mathsf{BSr3}}, (c, \hat{s}))$ and receives $\hat{b}, \hat{y}$. If $\mathcal{B}$ has queried $\hat{\mathcal{O}}^{\mathsf{ISign}_2}$ for session id $\mathsf{sid}^{(\mathsf{sid},\mathcal{S})}_{\mathsf{BSr3}}$, $\mathcal{B}$ retrieves $\hat{b}, \hat{y}$ from the previous query.

Then, $\mathcal{B}$ sets $b_{\mathsf{i}} = \frac{\hat{b}}{|\mathsf{hon}_{\mathcal{S}}|} + \tilde{b}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}$, $y_{\mathsf{i}} = \frac{\hat{y}}{|\mathsf{hon}_{\mathcal{S}}|} + \tilde{y}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}$ and appends $((\mathsf{sid}, \mathsf{i}, y_{\mathsf{i}}),$ $\tilde{\mathsf{cm}}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})})$ to $\mathsf{Q_{cm}}$. If there exists $((\mathsf{sid}, \mathsf{i}, y_{\mathsf{i}}), \mathsf{cm}) \in \mathsf{Q_{cm}}$ such that $\mathsf{cm} \neq \tilde{\mathsf{cm}}_{\mathsf{i}}^{(\mathsf{sid},\mathcal{S})}$, $\mathcal{B}$ aborts and we denote the abort event as YColl. If $y_{\mathsf{i}} = 0$, $\mathcal{B}$ aborts and we denote the abort event as YZero. Then, $\mathcal{B}$ sets $\mathsf{msg}_{\mathsf{i},\mathsf{sid}} \leftarrow (\mathsf{sid}, \mathcal{S}, c, \{\mathsf{cm}_j\}_{j\in\mathcal{S}})$, computes $\sigma_\mathsf{i} \leftarrow \mathsf{DS.Sign}(\hat{\mathsf{pk}}_\mathsf{i}, \mathsf{msg}_{\mathsf{i},\mathsf{sid}})$, $S_2 \leftarrow S_2 \cup \{(\mathsf{i}, \mathsf{sid})\}$, and returns $(b_\mathsf{i}, y_\mathsf{i}, \sigma_\mathsf{i})$.

Signing Round 3 ($\mathcal{O}^{\mathsf{ISign}_3}$ queries). When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{ISign}_3}$ on $(\mathsf{i}, \mathsf{sid}, \{\sigma_j, y_j\}_{j\in\mathcal{S}})$ where $\mathcal{S} = \mathcal{S}_{\mathsf{i},\mathsf{sid}}$, $\mathcal{B}$ retrieves $(\mathsf{sid}, \mathcal{S}, c, \{\mathsf{cm}_j\}_{j\in\mathcal{S}}) \leftarrow \mathsf{msg}_{\mathsf{i},\mathsf{sid}}$, checks

1. if $(\mathsf{i}, \mathsf{sid}) \in S_2$ and $(\mathsf{i}, \mathsf{sid}) \notin S_3$ # Round 2 has completed but Round 3 has not completed yet.
2. if $\mathsf{cm}_j = \mathsf{H}_{\mathsf{cm}}(\mathsf{sid}, j, y_j)$ for all $j \in \mathcal{S}$ # In Round 2, for all corrupt j the record $((\mathsf{sid}, j, y_j), \mathsf{cm}_j)$ does exists.
3. if $\mathsf{DS.Verify}(\mathsf{pk}_j, \mathsf{msg}_{\mathsf{i},\mathsf{sid}}, \sigma_j)$ for all $j \in \mathcal{S}$ # All honest parties in $\mathcal{S}$ received the same $(\mathcal{S}, c, \{\mathsf{cm}_j\}_{j\in\mathcal{S}})$ for sid.

and returns $\perp$ if not. If all the checks pass but $\mathsf{BadCm}^{(\mathsf{sid},\mathcal{S})}$ occurs, $\mathcal{B}$ aborts and we denote this abort event as $\mathsf{ForgeCm}$. If all the checks pass but there exists $\mathsf{k} \in \mathsf{hon}_\mathcal{S}$ such that $\mathsf{msg}_{\mathsf{k},\mathsf{sid}} = \perp$ or $\mathsf{msg}_{\mathsf{k},\mathsf{sid}} \neq \mathsf{msg}_{\mathsf{i},\mathsf{sid}}$, $\mathcal{B}$ aborts and we denote this abort event as $\mathsf{ForgeSig}$.

Otherwise, $\mathcal{B}$ gets $\hat{z}$ by querying $\hat{\mathcal{O}}^{\mathsf{ISign}_3}$ on $\mathsf{sid}_{\mathsf{BSr3}}^{(\mathsf{sid},\mathcal{S})}$ if it has not done the query before. Else $\mathcal{B}$ recalls $\hat{z}$ from the previous query. Finally $\mathcal{B}$ returns $z_\mathsf{i} \leftarrow \lambda_\mathsf{i}^\mathcal{S}\gamma_\mathsf{i}\hat{z} + \tilde{a}_\mathsf{i}^{(\mathsf{sid},\mathcal{S})} + f(c, y)\lambda_\mathsf{i}^\mathcal{S}\delta_\mathsf{i}$, where $y = \sum_{i\in\mathcal{S}} y_i$.

Output. When $\mathcal{A}$ returns $\{(m_k^*, \sigma_k^*)\}_{k\in[\ell+1]}$, $\mathcal{B}$ then outputs $\{(m_k^*, \sigma_k^*)\}_{k\in[\ell+1]}$.

Analysis of $\mathcal{B}$. To complete the proof, we show that (1) whenever $\mathcal{A}$ wins the game simulated by $\mathcal{B}$, $\mathcal{B}$ also wins; (2) if none of the abort events occurs, $\mathcal{B}$ simulates the game $\mathsf{Game}^{\mathsf{omuf\text{-}t}}_{\mathsf{Snowblind}[\mathsf{GrGen},f,\mathsf{DS}]}$ perfectly; (3) $\mathcal{B}$ only aborts with negligible probability.

(1) From the simulation, $\mathcal{B}$ makes at most one query to $\hat{\mathcal{O}}^{\mathsf{ISign}_I}$ when $\mathcal{A}$ makes one query to $\mathcal{O}^{\mathsf{ISign}_I}$ for $I = 1, 2$. Also, for each $(\mathsf{sid}, \mathcal{S})$, $\mathcal{B}$ makes at most one query to $\hat{\mathcal{O}}^{\mathsf{ISign}_3}$ if $\mathcal{A}$ make queries to $\mathcal{O}^{\mathsf{ISign}_3}$ corresponding to $(\mathsf{sid}, \mathcal{S})$. Therefore, $\mathcal{B}$ at most makes q_S queries to $\hat{\mathcal{O}}^{\mathsf{ISign}_1}$ and at most ℓ queries to $\hat{\mathcal{O}}^{\mathsf{ISign}_3}$. Also, it is clear $\mathcal{B}$ at most makes q_H queries to $\hat{\mathsf{H}}_{\mathsf{sig}}$.

Since $\mathcal{B}$ sets $\mathsf{pk} = \overline{\mathsf{pk}}$ and all random oracle queries to $\mathsf{H}_{\mathsf{sig}}$ are forwarded to $\hat{\mathsf{H}}_{\mathsf{sig}}$, each valid message-signature pair for pk is also valid for $\overline{\mathsf{pk}}$ in the game $\mathsf{Game}^{\mathrm{omuf}}_{\mathsf{BSr3}[\mathsf{GrGen},f]}$. Therefore, if $\mathcal{A}$ wins, $\mathcal{B}$ wins the game $\mathsf{Game}^{\mathrm{omuf}}_{\mathsf{BSr3}[\mathsf{GrGen},f]}$. Denote Win as the event $\mathcal{A}$ wins $\mathsf{Game}^{\mathsf{omuf\text{-}t}}_{\mathcal{A},\mathsf{Snowblind}}(\kappa)$ simulated by $\mathcal{B}$ and $\mathsf{Abort} := \mathsf{YZero} \vee \mathsf{YColl} \vee \mathsf{HashColl} \vee \mathsf{ForgeCm} \vee \mathsf{ForgeSig}$, and we have $\Pr[\mathsf{Win} \wedge (\neg\mathsf{Abort})] \leq \mathsf{Adv}^{\mathrm{omuf}}_{\mathcal{B},\mathsf{BSr3}[\mathsf{GrGen},f]}$.

(2) It is clear that the key generation and signing round 1 are simulated perfectly. For signing round 2, on a query for $(\mathsf{i}, \mathsf{sid}, \mathcal{S})$, if $\mathcal{B}$ does not abort, we know y_i and b_i are computed correctly since $B_\mathsf{i} = \tilde{B}_\mathsf{i}^{(\mathsf{sid},\mathcal{S})} = \hat{B}^{\frac{1}{|\mathsf{hon}_\mathcal{S}|}} g^{\tilde{b}_\mathsf{i}} h^{\tilde{y}_\mathsf{i}^{(\mathsf{sid},\mathcal{S})}} = g^{\frac{\hat{b}}{|\mathsf{hon}_\mathcal{S}|}+\tilde{b}_\mathsf{i}} h^{\frac{\hat{y}}{|\mathsf{hon}_\mathcal{S}|}+\tilde{y}_\mathsf{i}^{(\mathsf{sid},\mathcal{S})}} = g^{b_\mathsf{i}} h^{y_\mathsf{i}}$. Also, since $\tilde{y}_\mathsf{i}$ is randomly sampled from $\mathbb{Z}_p$ and YZero does not occur, we know y_i is uniformly distributed in $\mathbb{Z}_p^*$, which implies the simulation of signing round 2 is perfect. Also, given $\mathsf{HashColl}$ does not occur, the simulation of H_{cm} is perfect.

For signing round 3, on a query for $(\mathsf{i}, \mathsf{sid}, \mathcal{S})$, we only need to show that if none of the abort events occurs, then $g^{z_\mathsf{i}} = A_\mathsf{i}\mathsf{pk}_\mathsf{i}^{f(c,y)\lambda_\mathsf{i}^{\mathcal{S}}}$. Since $\mathsf{ForgeCm}$ does not occur, we know for each $j \in \mathcal{S} \cap \mathsf{corrupt}$, y_j received in round 3 is exactly the same as the one $\mathcal{B}$ finds in round 2. Since $\mathsf{ForgeSig}$ does not occur, we know for each $\mathsf{k} \in \mathsf{hon}_\mathcal{S}$, y_k received in round 3 is exactly the one $\mathcal{B}$ computes in round 2. Therefore, we have

$$y = \sum_{j \in \mathcal{S} \cap \mathsf{corrupt}} y_j + \sum_{\mathsf{k} \in \mathsf{hon}_\mathcal{S}} y_\mathsf{k} = \sum_{j \in \mathcal{S} \cap \mathsf{corrupt}} y_j + \sum_{\mathsf{k} \in \mathsf{hon}_\mathcal{S}} \left(\frac{\hat{y}}{|\mathsf{hon}_\mathcal{S}|} + \tilde{y}_\mathsf{k}^{(\mathsf{sid},\mathcal{S})}\right)$$
$$= \hat{y} + \left(\tilde{s}^{(\mathsf{sid},\mathcal{S})} + \sum_{j \in \mathcal{S} \cap \mathsf{corrupt}} y_j\right) = \hat{y} + \hat{s} \ .$$

Since $g^{\hat{z}} = \hat{A}\mathsf{pk}^{f(c,\hat{y}+\hat{s})}$ and $A_\mathsf{i} = \tilde{A}_\mathsf{i}^{(\mathsf{i},\mathsf{sid})} = \hat{A}^{\lambda_\mathsf{i}^{\mathcal{S}}\gamma_\mathsf{i}} g^{\tilde{a}_\mathsf{i}}$, we have

$$g^{z_\mathsf{i}} = g^{\lambda_\mathsf{i}^{\mathcal{S}}\gamma_i \hat{z} + \tilde{a}_\mathsf{i} + f(c,y)\lambda_\mathsf{i}^{\mathcal{S}}\delta_\mathsf{i}} = \hat{A}^{\lambda_\mathsf{i}^{\mathcal{S}}\gamma_\mathsf{i}} g^{\tilde{a}_\mathsf{i} + f(c,y)\lambda_\mathsf{i}^{\mathcal{S}}\delta_\mathsf{i}} \mathsf{pk}^{f(c,\hat{y}+\hat{s})\lambda_\mathsf{i}^{\mathcal{S}}\gamma_\mathsf{i}}$$
$$= \hat{A}^{\lambda_\mathsf{i}^{\mathcal{S}}\gamma_\mathsf{i}} g^{\tilde{a}_\mathsf{i}} (\mathsf{pk}^{\gamma_\mathsf{i}} g^{\delta_\mathsf{i}})^{f(c,y)\lambda_\mathsf{i}^{\mathcal{S}}} = A_\mathsf{i}\mathsf{pk}_\mathsf{i}^{f(c,y)\lambda_\mathsf{i}^{\mathcal{S}}} \ .$$

Therefore, $\mathcal{B}$ simulates the game $\mathsf{Game}^{\mathrm{omuf}}_{\mathsf{BSr3}[\mathsf{GrGen},f]}$ perfectly if $\mathcal{B}$ does not abort, which implies

$$\mathsf{Adv}^{\mathrm{omuf\text{-}t}}_{\mathcal{A},\mathsf{Snowblind}[\mathsf{GrGen},f,\mathsf{DS}]}(\kappa) \leq \Pr[\mathsf{Win} \ \wedge \ (\neg\mathsf{Abort})] + \Pr[\mathsf{Abort}]$$

(3) For YZero, since $\tilde{y}_\mathsf{i}$ is randomly sampled from $\mathbb{Z}_p$ indepedent of $\hat{y}$ and the number of $\mathcal{O}^{\mathsf{ISign}_2}$ valid queries is bounded by q_S, we have $\Pr[\mathsf{YZero}] \leq \frac{q_S}{p}$. For YColl, since when y_i is computed, given the view of $\mathcal{A}$, $\tilde{y}_\mathsf{i}$ is uniformly distributed over $\mathbb{Z}_p$, which implies y_i is uniformly distributed over $\mathbb{Z}_p$. Since $|\mathsf{Q}_\mathsf{cm}| \leq nq_S + q_H$,[2] the probability that YColl occurs in one query is bounded by $\frac{nq_S+q_H}{p}$. Therefore, we have $\Pr[\mathsf{YColl}] \leq \frac{q_S(nq_S+q_H)}{p}$.

$\mathsf{HashColl}$ corresponds to the event that there exists $\mathsf{sid}, j, y, y', \mathsf{cm}$ such that $j \in \mathsf{corrupt}$, $((\mathsf{sid}, j, y), \mathsf{cm}) \in \mathsf{Q}_\mathsf{cm}$, $((\mathsf{sid}, j, y'), \mathsf{cm}) \in \mathsf{Q}_\mathsf{cm}$, and $y \neq y'$. For each query (sid, j, y) to H_cm where $j \in \mathsf{corrupt}$, since the number of entries Q_cm that corresponds to (sid, j) is bounded by q_H, the probability that $\mathsf{H}_\mathsf{cm}(\mathsf{sid}, j, y)$ collides with an existing entry in Q_cm corresponds to (sid, j) is bounded by $\frac{q_H}{p-1}$. Since the number of such query is bounded by q_H, we have $\Pr[\mathsf{HashColl}] \leq \frac{q_H^2}{p-1}$.

If $\mathsf{ForgeCm}$ occurs, we know $\mathsf{BadCm}^{(\mathsf{sid},\mathcal{S})}$ occurs for some $(\mathsf{sid}, \mathcal{S})$. Given the event $\mathsf{BadCm}^{(\mathsf{sid},\mathcal{S})}$ occurs, $\mathsf{ForgeCm}$ occurs during the $\mathcal{O}^{\mathsf{ISign}_3}$ query corresponding to $(\mathsf{sid}, \mathcal{S})$ only if $\mathcal{A}$ makes a new query (sid, j, y) to H_cm and gets back with cm where j and cm are fixed after $\mathsf{BadCm}^{(\mathsf{sid},\mathcal{S})}$ occurs, the probability of which, thus, is bounded by $\frac{q_H}{p-1}$. Therefore, $\Pr[\mathsf{ForgeCm}] \leq \frac{q_S q_H}{p-1}$.

Finally, the event that $\mathsf{ForgeSig}$ occurs implies $\mathcal{A}$ breaks EUF-CMA security of DS for some public key $\hat{\mathsf{pk}}_\mathsf{k}$. Therefore, we can construct an adversary $\mathcal{C}$ for

[2] For each $\mathcal{O}^{\mathsf{ISign}_2}$ query at most n entries are added, and for each H_cm query at most one entry is added.

the game $\mathsf{Game}_{\mathsf{DS}}^{\text{euf-cma}}$ as follows. To start with, $\mathcal{C}$ receives a public key $\hat{\mathsf{pk}}^*$ from $\mathsf{Game}_{\mathsf{DS}}^{\text{euf-cma}}$ and runs $\mathcal{A}$ by simulating the game $\mathsf{Game}_{\mathsf{DS}}^{\text{euf-cma}}$ faithfully except $\mathcal{C}$ randomly samples $\mathsf{k}^* \leftarrow\!\!\$\ [n] \setminus \mathsf{corrupt}$ and sets $\hat{\mathsf{pk}}_{\mathsf{k}^*} = \hat{\mathsf{pk}}^*$. Whenever $\mathcal{C}$ need to generate a signature for public key $\hat{\mathsf{pk}}_{\mathsf{k}^*}$, $\mathcal{C}$ makes a query to $\mathcal{O}^{\mathsf{Sign}}$. $\mathcal{C}$ also maintains $\mathsf{msg}_{\mathsf{i},\mathsf{sid}}$, which is defined in the construction of $\mathcal{B}$. Then, if $\mathsf{ForgeSig}$ occurs, there exists $\mathsf{k} \in [n] \setminus \mathsf{corrupt}$ and sid such that $\mathcal{A}$ sends a signature for $m_{\mathsf{k},\mathsf{sid}}$ and public key $\hat{\mathsf{pk}}_{\mathsf{k}}$ but never receives a signature for $m_{\mathsf{k},\mathsf{sid}}$ before. Therefore, if $\mathsf{k} = \mathsf{k}^*$, $\mathcal{C}$ can win the game $\mathsf{Game}_{\mathsf{DS}}^{\text{euf-cma}}$. It is easy to see that $\mathcal{C}$ makes at most q_S query to $\mathcal{O}^{\mathsf{Sign}}$. Since the probability that $\mathsf{k} = \mathsf{k}^*$ is at least $1/n$, we have $\Pr[\mathsf{ForgeSig}] \le n\mathsf{Adv}_{\mathcal{C},\mathsf{DS}}^{\text{euf-cma}}$, which concludes the theorem. □

Blindness. The following theorem implies that our threshold scheme satisfies perfect blindness as long as the underlying base scheme is perfectly blind. The proof proceeds by a straighforward simulation argument.

Theorem 6. *For any* GrGen*,* f*, and* DS*, the scheme* $\mathsf{Snowblind}[\mathsf{GrGen}, f, \mathsf{DS}]$ *is perfectly blind if* $\mathsf{BS}[\mathsf{GrGen}, f]$ *is perfectly blind.*

Proof. The main idea is to show that for any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A},\mathsf{Snowblind}[\mathsf{GrGen},f,\mathsf{DS}]}^{\text{blind-t}}(\kappa) = \mathsf{Adv}_{\mathcal{B},\mathsf{BS}[\mathsf{GrGen},f]}^{\text{blind}}(\kappa)$.

It is not hard to see, by a standard hybrid argument, that it suffices to look at adversaries $\mathcal{A}$ that only query a single sid in $\mathsf{Game}_{\mathcal{A},\mathsf{Snowblind}}^{\text{blind-t}}(\kappa)$, i.e., they only attack a single session.

We construct $\mathcal{B}$ which has access to oracles $\hat{\mathcal{O}}^{\mathsf{USign}_1,\mathsf{USign}_2}$ and internally runs $\mathcal{A}$ by simulating the oracles $\mathcal{O}^{\mathsf{USign}_1,\ldots,\mathsf{USign}_4}$ as follows. Suppose $\mathcal{A}$ starts a signing session by querying $\mathcal{O}^{\mathsf{USign}_1}$ on input $(\mathsf{sid}, \mathsf{pk}, \mathsf{aux}, m_0, m_1, \mathcal{S}_0, \mathcal{S}_1)$. Since USign_1 takes no issuer's message as input and returns the signer set $\mathcal{S}$, $\mathcal{B}$ simulates $\mathcal{O}^{\mathsf{USign}_1}$ the same as $\mathsf{Game}_{\mathsf{Snowblind}}^{\text{blind-t}}$. For a query to $\mathcal{O}^{\mathsf{USign}_2}$ on input $(\mathsf{sid}, \{(A_{0,\mathsf{i}}, B_{0,\mathsf{i}}, \mathsf{cm}_{0,\mathsf{i}})\}_{\mathsf{i}\in\mathcal{S}_0}, \{(A_{1,\mathsf{i}}, B_{1,\mathsf{i}}, \mathsf{cm}_{1,\mathsf{i}})\}_{\mathsf{i}\in\mathcal{S}_1})$, $\mathcal{B}$ computes $A_I = \prod_{\mathsf{i}\in\mathcal{S}_I} A_{I,\mathsf{i}}$ and $B_I = \prod_{\mathsf{i}\in\mathcal{S}_I} B_{I,\mathsf{i}}$ for $I \in \{0,1\}$, queries $(c_0, c_1) \leftarrow \hat{\mathcal{O}}^{\mathsf{USign}_1}(\mathsf{sid}, \mathsf{pk}, m_0, m_1(A_0, B_0), (A_1, B_1))$, and finally returns $((c_0, \{\mathsf{cm}_{0,\mathsf{i}}\}_{\mathsf{i}\in\mathcal{S}_0}), (c_1, \{\mathsf{cm}_{1,\mathsf{i}}\}_{\mathsf{i}\in\mathcal{S}_1}))$. For a query to $\mathcal{O}^{\mathsf{USign}_3}$ on input $(\mathsf{sid}, \{(y_{0,\mathsf{i}}, b_{0,\mathsf{i}}, \sigma_{0,\mathsf{i}})\}_{\mathsf{i}\in\mathcal{S}_0}, \{(y_{1,\mathsf{i}}, b_{1,\mathsf{i}}, \sigma_{1,\mathsf{i}})\}_{\mathsf{i}\in\mathcal{S}_1})$, $\mathcal{B}$ returns $(\{(y_{0,\mathsf{i}}, \sigma_{0,\mathsf{i}})\}_{\mathsf{i}\in\mathcal{S}_0}, \{(y_{1,\mathsf{i}}, \sigma_{1,\mathsf{i}})\}_{\mathsf{i}\in\mathcal{S}_1})$. For a query to $\mathcal{O}^{\mathsf{USign}_3}$ on input $(\mathsf{sid}, \{z_{0,\mathsf{i}}\}_{\mathsf{i}\in\mathcal{S}_0}, \{z_{1,\mathsf{i}}\}_{\mathsf{i}\in\mathcal{S}_1})$, $\mathcal{B}$ computes $b_I = \sum_{\mathsf{i}\in\mathcal{S}_I} b_{I,\mathsf{i}}$, $y_I = \sum_{\mathsf{i}\in\mathcal{S}_I} y_{I,\mathsf{i}}$, $z_I = \sum_{\mathsf{i}\in\mathcal{S}_I} z_{I,\mathsf{i}}$ for $I \in \{0,1\}$ and returns $\hat{\mathcal{O}}^{\mathsf{USign}_2}(\mathsf{sid}, (z_0, b_0, y_0), (z_1, b_1, y_1))$.

Finally, after $\mathcal{A}$ returns b', $\mathcal{B}$ returns b'. It is clear that if $b = I$ for $I \in \{0,1\}$ in the game $\mathsf{Game}_{\mathsf{BS}}^{\text{blind}}$, $\mathcal{B}$ simulates the game $\mathsf{Game}_{\mathsf{Snowblind}}^{\text{blind-t}}$ for $b = I$ perfectly. Therefore, $\mathcal{B}$ has the same advantage as $\mathcal{A}$. □

Acknowledgements. Elizabeth Crites is supported by Input Output through their funding of the Blockchain Technology Lab at the University of Edinburgh. Tessaro and Zhu are supported in part by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

References

1. Abe, M.: A secure three-move blind signature scheme for polynomially many signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_9
2. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_17
3. Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 82–99. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_5
4. Barreto, P.L., Zanon, G.H.M.: Blind signatures from zero-knowledge arguments. Cryptology ePrint Archive, Paper 2023/067 (2023). https://eprint.iacr.org/2023/067
5. Bellare, M., Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Better than advertised security for non-interactive threshold signatures. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13510, pp. 517–550. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_18
6. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. J. Cryptol. **16**(3), 185–215 (2003). https://doi.org/10.1007/s00145-002-0120-1
7. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_25
8. Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in)security of ROS. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 33–53. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_2
9. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_3
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_30
11. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, Santa Barbara, California, USA, 23–25 August 1982, pp. 199–203. Plenum Press, New York (1982). https://doi.org/10.1007/978-1-4757-0602-4_18
12. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_25
13. Denis, F., Jacobs, F., Wood, C.A.: RSA Blind Signatures. Internet-Draft draft-IRTF-CFRG-RSA-blind-signatures-02. Work in Progress. Internet Engineering Task Force (2021). https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-rsa-blind-signatures-02
14. Desmedt, Y.: Society and group oriented cryptography: a new concept. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 120–127. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_8

15. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_28
16. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
17. Fuchsbauer, G., Plouviez, A., Seurin, Y.: Blind Schnorr signatures and signed ELGamal encryption in the algebraic group model. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 63–95. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_3
18. Fuchsbauer, G., Wolf, M.: (concurrently secure) blind schnorr from schnorr. Cryptology ePrint Archive, Paper 2022/1676 (2022). https://eprint.iacr.org/2022/1676
19. Hauck, E., Kiltz, E., Loss, J.: A modular treatment of blind signatures from identification schemes. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 345–375. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_12
20. Hendrickson, S., Iyengar, J., Pauly, T., Valdez, S., Wood, C.A.: Private Access Tokens. Internet-Draft draft-private-access-tokens-01. Work in Progress. Internet Engineering Task Force (2021). https://datatracker.ietf.org/doc/html/draft-private-access-tokens-01
21. icloud private relay overview. https://www.apple.com/privacy/docs/iCloud_Private_Relay_Overview_Dec2021.PDF. Accessed 03 Feb 2023
22. Kastner, J., Loss, J., Xu, J.: The Abe-Okamoto partially blind signature scheme revisited. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 279–309. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22972-5_10
23. Komlo, C., Goldberg, I.: FROST: flexible round-optimized schnorr threshold signatures. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 34–65. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_2
24. Kuchta, V., Manulis, M.: Rerandomizable threshold blind signatures. In: Yung, M., Zhu, L., Yang, Y. (eds.) INTRUST 2014. LNCS, vol. 9473, pp. 70–89. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27998-5_5
25. Lindell, Y.: Simple three-round multiparty schnorr signing with full simulatability. IACR Cryptology ePrint Archive, p. 374 (2022). https://eprint.iacr.org/2022/374
26. Lysyanskaya, A.: Security analysis of RSA-BSSA. Cryptology ePrint Archive, Paper 2022/895, PKC 2023 (2022). https://doi.org/10.1007/978-3-031-31368-4_10
27. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
28. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_3
29. PCM: Click fraud prevention and attribution sent to advertiser. https://webkit.org/blog/11940/pcm-click-fraud-prevention-and-attribution-sent-to-advertiser/. Accessed 03 Feb 2023
30. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9

31. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_7
32. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000). https://doi.org/10.1007/s001450010003
33. Rial, A., Piotrowska, A.M.: Security analysis of coconut, an attribute-based credential scheme with threshold issuance. IACR Cryptology ePrint Archive, p. 11 (2022). https://eprint.iacr.org/2022/011
34. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978). https://doi.org/10.1145/359340.359342
35. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22
36. Schnorr, C.P.: Security of blind discrete log signatures against interactive attacks. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 2001. LNCS, vol. 2229, pp. 1–12. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45600-7_1
37. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979). https://doi.org/10.1145/359168.359176
38. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
39. Sonnino, A., Al-Bassam, M., Bano, S., Meiklejohn, S., Danezis, G.: Coconut: threshold issuance selective disclosure credentials with applications to distributed ledgers. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, 24–27 February 2019. The Internet Society (2019). https://www.ndss-symposium.org/ndss-paper/coconut-threshold-issuance-selective-disclosure-credentials-with-applications-to-distributed-ledgers/
40. Stinson, D.R., Strobl, R.: Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 417–434. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47719-5_33
41. Tessaro, S., Zhu, C.: Short pairing-free blind signatures with exponential security. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022. LNCS, vol. 13276, pp. 782–811. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07085-3_27
42. Trust tokens. https://developer.chrome.com/docs/privacy-sandbox/trust-tokens/. Accessed 03 Feb 2023
43. Vo, D.-L., Zhang, F., Kim, K.: A new threshold blind signature scheme from pairings. In: Proceedings of the 2003 Symposium on Cryptography and Information Security (SCIS 2003) (2003)
44. VPN by google one, explained. https://one.google.com/about/vpn/howitworks. Accessed 02 Feb 2023