Stronger Security for Threshold Blind Signatures

Anja Lehmann¹, Phillip Nazarian², and Cavit Özbay¹

 ¹ Hasso Plattner Institute, University of Potsdam, Germany {anja.lehmann,cavit.oezbay}@hpi.de
 ² University of California Irvine, USA pnazaria@uci.edu

Abstract. Blind signatures allow a user to obtain a signature from an issuer in a privacy-preserving way: the issuer neither learns the signed message, nor can link the signature to its issuance. The threshold version of blind signatures further splits the secret key among n issuers, and requires the user to obtain at least $t \leq n$ of signature shares in order to derive the final signature. Security should then hold as long as at most t-1 issuers are corrupt. Security for blind signatures is expressed through the notion of *one-more unforgeability* and demands that an adversary must not be able to produce more signatures than what is considered trivial after its interactions with the honest issuer(s). While one-more unforgeability is well understood for the single-issuer setting, the situation is much less clear in the threshold case: due to the blind issuance, counting which interactions can yield a trivial signature is a challenging task. Existing works bypass that challenge by using simplified models that do not fully capture the expectations of the threshold setting. In this work, we study the security of threshold blind signatures, and propose a framework of one-more unforgeability notions where the adversary can corrupt c < t issuers. Our model is generic enough to capture both interactive and non-interactive protocols, and it provides a set of natural properties with increasingly stronger guarantees, giving the issuers gradually more control over how their shares can be combined. As a point of comparison, we reconsider the existing threshold blind signature models and show that their security guarantees are weaker and less clearly comprehensible than they seem. We then re-assess the security of existing threshold blind signature schemes - BLS-based and Snowblind in our framework, and show how to lift them to provide stronger security.

1 Introduction

Blind signatures enable a user to obtain a signature from an issuing authority, without the issuer learning the signed message or even being able to link the produced signature to its issuance. Initially, Chaum proposed blind signatures for building e-cash [Cha83], and they have since become a key component of various privacy-preserving protocols such as anonymous credentials [FHS22], anonymous attestation [BFG⁺13], and anonymous tokens [Han23]. They also enjoy increased real-world attention with recent standardization efforts [DJW23, CDVW24, AHWY24] and implementations by Apple [iCl21] and Cloudflare [clo]. Because blind signatures require highly available online access to the issuer's secret signing key, the compromise of that key is a significant risk. Threshold cryptography is a natural approach to remedy the risk of key exposure. The threshold version of blind signatures distributes the secret signing key among n issuers, and allows a user to obtain a blind signature after interacting with at least t of them. The distribution of secret keys ensures security even if up to t-1 parties have been compromised.

Application-specific threshold variants of some privacy-preserving protocols have already been considered such as e-cash [RP23] or anonymous credentials [SABBD18, DKL⁺23] with threshold issuance. However, there are surprisingly few protocols for threshold blind signatures as a standalone primitive, e.g., the non-interactive BLS-based scheme [VZK03] (which essentially combines Boldyreva's threshold and blind variants of BLS signatures [Bol03] into a single protocol) and the recent interactive Snowblind scheme from Crites et al. [CKM⁺23]. Their security has been shown in security models that extend the *one-more unforgeability* notion of blind signatures to the threshold setting.

(Threshold) One-More Unforgeability. One-more unforgeability adjusts the classic unforgeability idea to the fact that signatures are now created blindly. As issuers in a blind signing protocol are not aware of the message they sign, asking for a forgery on a fresh message is meaningless. Instead, security is defined as the impossibility that an adversary produces more signatures than what it can trivially derive through its interactions with the issuers.

In the single-issuer setting, counting the number of trivial signatures is easy: the count increases with every completed interaction the user has with the honest issuer. In the threshold setting, capturing what is trivial becomes much more challenging, since we do not know which issuer has signed which message. The existing security models by Vo et al. [VZK03], Kuchta and Manulis [KM15], and Crites et al. [CKM⁺23] work around this challenge by (implicitly or explicitly) assuming that t-1 issuers are corrupt. In that restricted case, the triviality condition is no more complex than the single-issuer setting: every blinded signature share the user receives should increment the count of allowed signatures by 1. It is tempting to assume that these models are sufficient because maximal corruptions offer the "best case scenario" for an attacker. In fact, the implications of these models in the general case of c < t corruptions is not so clear.

1.1 Our Contributions

In this work we study the security of threshold blind signatures, and introduce a framework of stronger and more natural security notions for one-more unforgeability in that setting. We then re-assess the security of the BLS and Snowblind protocols in our framework and propose new protocol variants that provide stronger security. An overview of all constructions studied in our work is given in Table 1. In more detail, our work comprises of the following results. OMUF Security Framework. To be useful for a broad class of threshold blind signature instantiations, we propose a generic syntax that can capture both interactive and non-interactive signing protocols (Section 3). Non-interactive means that only a single round between the user and each issuer is needed, as e.g., in BLS. Threshold blind signatures are considered secure when they are blind and unforgeable, and we give definitions for both. The core of our model is the framework of four variants of the threshold one-more unforegability (OMUF) definition with increasing security guarantees. They mainly differ in how we count the number of trivial signatures that the adversary is considered to have learned through its interaction with n - c honest issuers.

- OMUF-0 is the simplest but also weakest possible definition. It increases the count for every signature share provided by an honest issuer.
- OMUF-1 requires at least t-c honest shares to increase the count. The challenge thereby is how to compute this count – remember that we are in the *blind* signature setting, i.e., we do not know which issuer has signed which message. We apply a counting mechanism introduced for threshold OPRFs [JKKX17], to determine the optimal combination of the blindly obtained shares, which then yields the maximal number of signatures the adversary could possibly derive. We also give an efficient algorithm for this count function, which had an exponential running time in its existing specification.
- OMUF-2 additionally requires some form of agreement among the issuers, and we only increase the count when t - c issuers produced shares for the same session ssid. We use the postfix-style session identifier approach recently proposed by Barbosa at al. [BGHJ24], to not impose any pre-agreed session identifiers. In concrete protocols, this ssid typically will be (part of) the issuer's view, e.g., the blinded message received from the user. OMUF-2 then ensures that a signature can only be derived when the quorum of issuers has signed the *same* blinded input.
- OMUF-3 builds upon OMUF-2 and requires that all honest issuers have also agreed to the same set of issuers they intended to co-sign ssid. This is particularly desirable for interactive protocols, where t issuers run a multi-round protocol in the belief of communicating with a particular set of co-signers.

We prove that these four notions form a clear hierarchy of strictly increasing security (Section 4). While we believe that OMUF-1-3 are the desirable properties for threshold blind signatures, stronger security is not necessarily better for every setting. The "right" security notion depends on the concrete application: while OMUF-2 and OMUF-3 are strictly stronger than OMUF-1 by enforcing some form of agreement among all issuers, this agreement also takes away the flexibility on the user side to freely combine signature shares from various interactions.

Inadequacy of Existing Models. To capture existing results in our hierarchy, we also restate the two existing models of threshold one-more unforgeability in our syntax (Section 3). The model of Vo et al. [VZK03] and Kuchta and Manulis [KM15] explicitly assumes that the adversary always corrupts t - 1

issuers; we denote this model OMUF-FC (standing for "full corruption"). The security model introduced along with the Snowblind construction [CKM⁺23] is restated as OMUF-SB. That model allows for any number c < t of corruptions, but its security guarantee seems unduly weak in the case of c < t-1 corruptions. In particular, OMUF-SB increases the count of trivial signatures after only a single share is provided by an honest issuer.

We carefully study the relations between our four notions and these existing two (Section 4). By proving that the seemingly intuitive OMUF-FC and the clearly weak OMUF-0 imply each other, we definitively establish that OMUF-FC is an insufficient notion. For OMUF-SB, the situation is more complex. We show that it is in a mysterious position between OMUF-0 and OMUF-3, yet incomparable to OMUF-1 and OMUF-2. Ultimately, we do not believe that OMUF-SB offers a clearly comprehensible security statement, and we therefore favor our hierarchy's definitions over it.

BLS-based Schemes. We then re-assess the security of the simple BLS-based signature scheme (Section 5), which was shown to be OMUF-FC secure [VZK03]. We restate this construction as tBlindBLS-1, and show that it also satisfies OMUF-1 under the threshold version of the BOMDH assumption. tBlindBLS-1 cannot achieve stronger security though, as the user can freely combine signature shares from various "sessions" and there is no binding to a particular signing set. We therefore propose two extensions tBlindBLS-2 and tBlindBLS-3 that provably satisfy OMUF-2 and OMUF-3, respectively. In both protocols, issuers blind their shares with a carefully crafted value. When combining such shares, the blinding values only cancel out when the user has obtained t contributions for the same session, or even for the same session and from a consistent set of issuers. Both constructions rely on the BOMDH assumption, which is weaker and more natural than the assumption needed by tBlindBLS-1. tBlindBLS-3 uses a secure PRF and needs to establish pairwise keys among all issuers.

Pairing-free Schemes: SB and SB+. While BLS-based schemes are highly attractive due to their simple and non-interactive nature, their reliance on pairingfriendly curves can be a burden for practical deployment. We therefore also study and improve the pairing-free threshold blind signature scheme Snowblind (SB) recently introduced by Crites et al. [CKM⁺23] (Section 6). Their interactive protocol supports concurrent signing sessions, and was proven to be OMUF-SB secure. OMUF-SB directly only implies OMUF-0 security, which is too weak. This raises the question of whether SB can satisfy a stronger notion in our hierarchy.

Given that the Snowblind scheme is an interactive protocol that a dedicated set of t issuers runs for each signing request, the intuitive security goal would be OMUF-3 – reflecting this signature creation with a well-defined set of co-signers in the unforgeability guarantee. We first show that this is not the case, i.e. the original SB scheme does not satisfy OMUF-3 security, and then propose a simple variant SB+ that lifts the Snowblind protocol to achieve this strongest security in our hierarchy. Our construction essentially uses the same blinding technique

as tBlindBLS-3, and is given in a semi-generic manner showing how OMUF-SB can be boosted to OMUF-3.

	tBlindBLS-1	tBlindBLS-2	tBlindBLS-3	SB	SB+
OMUF-0	✓ [VZK03]	1	1	✓ [CKM ⁺ 23]	1
OMUF-1	1	1	1	?	1
OMUF-2	×	1	1	?	1
OMUF-3	×	X	1	×	1

Table 1. Overview of security levels of existing and newly proposed constructions.

1.2 Related Work

Our model and constructions are building upon related concepts for threshold but non-blind signatures, as well as recent works on threshold OPRFs and multisignatures. We summarize these works here and give a more detailed comparison in the respective parts of our work.

Stronger Security for Threshold Signatures. Interestingly, a similar discrepancy between the formally modelled vs. the desirable security was already present in threshold, but non-blind, signatures. This was was recently discovered by Bellare et al. [BTZ22, BCK⁺22] who show that the traditional security model is not satisfactory, as it does not capture the expected higher security properties based on the secret's distribution. They propose several stronger unforgeability notions for the threshold setting, with increasingly stricter definitions of what counts as a *trivial* signature in the unforgeability game. Directly applying this framework of stronger definitions to threshold *and blind* signatures is not straightforward, due to the significantly different security notions in both settings. Furthermore, their framework only considers non-interactive threshold signatures whereas our framework does not have such a restriction. We point out similarities and differences when presenting our definitions and relations in Sections 3 and 4.

Cremers et al. [CPZ24] very recently extended the threshold signature security hierarchy of [BTZ22, BCK⁺22] to include consideration of adaptive corruptions. To date there is no threshold blind signature scheme that has been proven secure under adaptive corruptions. Therefore, our work focuses on the static corruption model.

Blind Multi-Signatures. Karantaidou et al. $[KRB^+24]$ recently proposed blind multi-signatures that allow issuers to generate their individual key, aggregate n of them them into group public keys in an ad-hoc manner, and blindly create signatures that verify under these aggregated keys. They also provide two constructions of blind multi-signatures: one based on BLS, and one build upon Snowblind. Their work also introduces the notion of one-more unforgeability for blind multi-signatures, which assumes that there is only a single honest issuer. In Appendix E we study our definitions for the "edge" case of t = n, and show that OMUF-0, -1 and OMUF-SB become equivalent in this setting. Thus, considering multi-signatures as *n-out-of-n* threshold signatures, we can conjecture that their unforgeability notion provides a comparable level of security to these models. Our BLS and Snowblind based constructions aim at the stronger security of OMUF-2 and OMUF-3. These notions prevent signature share aggregation across sessions and signing sets, which contradicts the flexible nature typically desired from multi-signatures with ad-hoc generated aggregate keys. For multi-signatures where issuers should be given more control over the sessions and co-signers they interact with, constructions with ad-hoc group keys on levels similar to OMUF-2 and 3 are an interesting open problem.

Threshold OPRFs. Oblivious pseudorandom functions (OPRFs) allow a user on input x and server holding a key k to engage in a blind evaluation protocol, that returns y = PRF(k, x) to the user. While similar in spirit, OPRFs and blind signatures demand significantly different security properties that are not directly related [CHL22]. Further, security of OPRFs is typically defined through ideal functionalities, which abstract the complexity of counting through bookkeeping within the functionality. Nevertheless, despite the differences in security models, OPRF and blind signatures are often similar in their constructions and our work takes inspiration from a number of threshold and distributed OPRF works: The blinding technique to lift BLS to OMUF-2 security has been used by Gu et al. [GJK⁺25] for stronger security of threshold OPRFs. The idea for the pairwise blinding in tBlindBLS-3 and SB+, where the blinding values only cancel out when shares from the desired set of issuers are combined, was previously used to obtain adaptive and proactive security for distributed OPRFs [CLN15, BFH⁺20]. Finally, the T-(B)OMDH assumptions used for the analysis of tBlindBLS-1 dates back to the first threshold OPRF by Jarecki et al. [JKKX17].

2 Preliminaries

Here we define building blocks and assumptions we use throughout the paper. This section also contains the definition of the **count** function that will be crucial for our OMUF-1 security notion and a novel way to efficiently compute it.

Building Blocks. We use prime-order group generator $(\mathbb{G}, g, q) \leftarrow \mathsf{GGen}(\lambda)$, and pairing group generator $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, q) \leftarrow \mathsf{BGGen}(\lambda)$ in our concrete constructions. We also use Shamir's secret sharing as a generic building block, abstracted as $\mathsf{Share}(x, n, t) \to (x_1, \ldots, x_n)$ that outputs t-out-of-n shares of input x. The detailed descriptions of all these algorithms, and the definition of pseudorandom functions are presented in Appendix A for completeness.

Assumptions. Our work uses various Diffie-Hellman type assumptions. The classic DDH assumption is deferred to Appendix A.

The One-More Diffie-Hellman (OMDH) assumption in group \mathbb{G} states that, given $g^p \in \mathbb{G}$ for random p, it is difficult to raise Q+1 given (random) \mathbb{G} elements to p without making more than Q calls to an exponentiation oracle for p. The *Bilinear* OMDH assumption (BOMDH) (previously used in [KMMM23, CLN15]) extends that game to the pairing group setting ($\mathbb{G}_1, \mathbb{G}_2$) by providing both $g^p \in \mathbb{G}_1$ and $\hat{g}^p \in \mathbb{G}_2$ as inputs. We present the BOMDH Assumption in Appendix A.

In the *Threshold* OMDH (T-OMDH) assumption, the secret exponent p is distributed via Shamir secret sharing with threshold t into n shares assigned to n separate exponentiation oracles. When [JKKX17] introduced this assumption, they defined a function count_t that computes the expected number Q of group element exponentiations that should be possible given (q_1, \ldots, q_n) calls to the oracles numbered $(1, \ldots, n)$. The T-OMDH assumption states it is difficult to raise Q + 1 group elements to p. The *Bilinear* T-OMDH assumption (previously used in [KMMM23]) extends T-OMDH to the setting of a bilinear pairing setup in the same manner that BOMDH extends OMDH.

Definition 1 (Threshold Bilinear One-More Diffie-Hellman). Let $\mathcal{BG} := (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, q) \leftarrow \mathsf{BGGen}(\lambda)$. In the (t, n, N)-Threshold Bilinear One-More Diffie-Hellman (T-BOMDH) game where $t \leq n$, \mathcal{A} is given $(\mathcal{BG}, g^{p(0)}, \hat{g}^{p(0)}, \hat{g}_1, \ldots, \hat{g}_N)$ for $(\hat{g}_1, \ldots, \hat{g}_N) \leftarrow \mathbb{G}_2^N$ and access to oracle $\mathsf{TBOMDH}_p(\cdot, \cdot) : [n] \times \mathbb{G}_2 \to \mathbb{G}_2$ for a random (t-1)-degree polynomial $p(\cdot)$ over \mathbb{Z}_q , where $\mathsf{TBOMDH}_p(i, x) \to x^{p(i)}$. \mathcal{A} wins if it outputs Q + 1 pairs $(j, \hat{g}_j^{p(0)})$ for distinct $j \in [N]$, where $Q = \mathsf{count}_t(q_1, \ldots, q_n)$ and q_i is the number of queries made to $\mathsf{TBOMDH}_p(i, \cdot)$. The T-BOMDH assumption holds if, for all efficient \mathcal{A} , the probability that \mathcal{A} wins the T-BOMDH game is negligible.

Clearly, the definition of $count_t$ is central to the T-(B)OMDH assumption. This function captures the expectation that a set of t queries to t different oracles is required for each group element the adversary successfully exponentiates to p. However, there is no trivial way to group oracle queries into such sets. The count function therefore maximizes the number of possible exponentiations by assuming the optimal grouping.

For example, let's consider $\operatorname{count}_3(2, 2, 2, 2, 1)$ which denotes the status after a game where the adversary has access to n = 5 oracles initialized with threshold t = 3. The input (2, 2, 2, 2, 1) indicates the number of queries that have been made to each oracle. Then, $\operatorname{count}_3(2, 2, 2, 2, 1) = 3$ because the optimal grouping of queries is (1, 1, 1, 0, 0), (1, 0, 0, 1, 1), and (0, 1, 1, 1, 0). The same total number of queries distributed less evenly between the oracles yields a lesser value, e.g., $\operatorname{count}_3(4, 2, 1, 1, 1) = 2$.

We now define function $\operatorname{count}_t : \mathbb{N}^n \to \mathbb{N}$ formally. Let $W(\boldsymbol{v})$ be the Hamming weight of vector \boldsymbol{v} . Define $\mathcal{V}_{n,t}$ to be the set of *n*-bit binary vectors with Hamming weight t, i.e. $\mathcal{V}_{n,t} := \{\boldsymbol{v} \in \{0,1\}^n : W(\boldsymbol{v}) = t\}$. Further, let $\boldsymbol{q} = (q_1, \ldots, q_n) \in \mathbb{N}^n$. Previous works [JKKX17, AMMM18, KMMM23] have defined $\operatorname{count}_t(\boldsymbol{q})$ as the maximum integer c for which there exist $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c \in \mathcal{V}_{n,t}$ (not necessarily distinct) such that $\boldsymbol{v}_1 + \cdots + \boldsymbol{v}_c \leq \boldsymbol{q}$. We provide an equivalent

recursive definition, using notation $\mathcal{V}_{n,t}(\mathbf{q}) := \{ \mathbf{v} \in \mathcal{V}_{n,t} : \mathbf{v} \leq \mathbf{q} \}$:

$$\mathsf{count}_{t}(\boldsymbol{q}) = \begin{cases} 0 & \mathcal{V}_{n,t}(\boldsymbol{q}) = \emptyset \\ 1 + \max_{\boldsymbol{v} \in \mathcal{V}_{n,t}(\boldsymbol{q})} \mathsf{count}_{t}(\boldsymbol{q} - \boldsymbol{v}) & \mathcal{V}_{n,t}(\boldsymbol{q}) \neq \emptyset \end{cases}$$
(1)

Computing this function naively requires recursively searching through an exponential number of possible vectors $v \in \mathcal{V}_{n,t}(q)$. Prior works have not offered an efficient way of computing count_t, without which the T-(B)OMDH assumption would not be falsifiable. Therefore we provide an efficient alternative algorithm: repeatedly "use up" vector v with 1s in the positions of the t greatest elements of q. Appendix B contains a proof of this algorithm's correctness.

3 Threshold Blind Signatures

In this section, we present our definitional framework for threshold blind signatures. The main contribution of this section is our hierarchy of definitions for the one-more unforgeability (OMUF) property of such signatures.

3.1 Definition of Threshold Blind Signatures

We start by defining the syntax for threshold blind signatures in a way that is generic enough to capture both interactive and non-interactive signing protocols. Non-interactive refers to round-optimal protocols, where no communication beyond the minimally required interaction, i.e., the user sending the blinded message to the issuers, is needed. Our detailed syntax for threshold blind signatures is presented in Definition 2, and consists of algorithms for the public parameter generation (Pg), the key generation of all n signers (Kg), a signing protocol between a user and t of the signers, as well as a verification algorithm Vf.

Setup, KGen & Verification. The Pg algorithm outputs the public parameters pp that can be potentially used as an implicit input to all other algorithms. The key generation algorithm Kg, on input number of issuers n and the threshold t, outputs the public key pk and the secret key shares $\{sk_i\}_{i\in n}$. Kg also outputs some auxiliary information aux, which could e.g., consist of some additional public values used by the signing protocol to validate individual signing contribution. Although pk and aux are both public values, they are represented separately as aux is only needed during signing and not during signature verification. Crites et al. [CKM⁺23] made a similar distinction and we keep this in our model too. We note that [CKM⁺23] defined an additional output to Kg, the issuer public key shares, that we omit as it was not explicitly used in any security definition. The verification algorithm Vf works just like the regular signature verification that checks the validity of a signature σ on a message msq for a public key pk.

Signing Protocol. Our signing protocol is defined as a potentially interactive protocol where a user interacts with a set of t issuers S. The protocol is initiated by the user via the algorithm USign_0 , which takes the public key pk, auxiliary information aux, message msg, and a set of t issuers S as input. The algorithm outputs local user state ust_0 and a protocol message (typically a blinded form of msg) ups_0 . The first round of an issuer i is triggered through ISign_1 on input secret key share sk_i , auxiliary information aux, some additional context data ctx, and the user's input ups_0 . The context ctx becomes useful in our stronger security notions, and allows the issuers to control that only signature shares generated for the same ctx can be combined into the final signature. The issuer i's output is its local state $ist_{1,i}$ and protocol message $ips_{1,i}$.

For interactive protocols, the user and signers may engage in further rounds $USign_j$ and $ISign_j$, getting local state and protocol message from round j - 1 as input, and outputting new tuples (ust_j, ups_j) and $(ist_{j,i}, ips_{j,i})$ respectively.

We make the final round r explicit, with USign_r on the user side outputting a signature σ or \bot . On the issuer side, ISign_r outputs a set of signers \mathcal{S}' and a subsession identifier ssid. Looking ahead to Section 3.2, our different notions of one-more unforgeability enforce different properties conditioned on these two values. In all cases, though, we require protocols to output sensible \mathcal{S}' values. In particular, $\mathcal{S}' \subseteq [n]$, $|\mathcal{S}'| = t$, and $i \in \mathcal{S}'$ where i is the issuer who outputs \mathcal{S}' .

Definition 2 (Threshold Blind Signature). A threshold blind signature scheme TB is a tuple of algorithms such that:

- $\mathsf{Pg}(1^{\lambda}) \to pp$: Outputs public parameters pp for security parameter 1^{λ} , which we assume to be an implicit input to all other algorithms.
- $\mathsf{Kg}(n,t) \to (pk, \{sk_1, \ldots, sk_n\}, aux)$: Probabilistic algorithm that outputs the public key pk, secret key shares sk_i , and auxiliary information aux.
- **Signing Protocol:** It is an (possibly) interactive algorithm between the user and a set of issuers $S \subseteq [n]$, $t \leq |S| \leq n$, to sign a message *msg*. The interaction between the user and issuers are denoted as follows.

$$\begin{split} &(ust_0, ups_0) \leftarrow \mathsf{USign}_0(pk, aux, msg, \mathcal{S}) \\ &(ist_{1,i}, ips_{1,i}) \leftarrow \mathsf{ISign}_1(i, sk_i, aux, \mathsf{ctx}, ups_0) \\ &(ust_1, ups_1) \leftarrow \mathsf{USign}_1(ust_0, \{ips_{1,i}\}_{i\in\mathcal{S}}) \\ &(ist_{j,i}, ips_{j,i}) \leftarrow \mathsf{ISign}_j(i, ist_{j-1,i}, ups_{j-1}) \\ &(ust_j, ups_j) \leftarrow \mathsf{USign}_j(ust_{j-1}, \{ips_{j,i}\}_{i\in\mathcal{S}}) \\ &(\mathsf{ssid}, \mathcal{S}', ips_{r,i}) \leftarrow \mathsf{ISign}_r(i, ist_{r-1,i}, ups_{r-1}) \\ &\sigma/\bot \leftarrow \mathsf{USign}_r(ust_{r-1}, \{ips_{r,i}\}_{i\in\mathcal{S}}) \end{split}$$

 $Vf(pk, \sigma, msg) \rightarrow b \in \{true, false\}$: Verifies if σ is valid on msg for pk.

In non-interactive protocols, i.e., when r = 1, the issuer's algorithms $|\text{Sign}_1|$ and $|\text{Sign}_r|$ collapse, in the sense that $(\text{ssid}, \mathcal{S}', ips_{r,i}) \leftarrow |\text{Sign}_{r=1}(i, sk_i, aux, \text{ctx}, ups_0)$ combines the input behaviour from the first, with the output behaviour of the last round.

Comparison to $[CKM^+23]$. Our notation for the signing protocol is similar to the definition of Crites et al. $[CKM^+23]$. Our version, however, is initiated by the user while theirs is initiated by the issuers. In the context of threshold blind signatures, we believe that having the user as the first party to send a message is more natural as the issuers do not know the message to be signed and maybe even the co-signers in the session. Furthermore, any protocol that has the first interaction from the issuers to the user can be represented in our notation by setting the first user message to \perp . Further, although $[CKM^+23]$'s definition starts with a move from the issuers to the users, their construction, Snowblind, actually has the user as the first message sender.

Correctness & Blindness. Threshold blind signature schemes must satisfy correctness, blindness – and of course one-more unforgeability, which we define in several variants afterwards. We present the formal definitions of the correctness and blindness properties in Appendix C.

Our blindness definition mainly follows Crites et al. [CKM⁺23] and is given in Appendix C, Figure 7. To recap, like their definition, our blindness notion allows the adversary to control all issuers and also to trigger honest users to perform signing protocols with different issuer sets. It demands that an adversary, when given two signatures a user has obtained through the threshold blind signing protocol, cannot link a signature to its issuance session. We note that this blindness definition enforce a synchronized behavior where the user always outputs its partial signatures for msg_b and msg_{1-b} together, not allowing the adversary to choose its partial signatures adaptively for different messages. This limitation is not exploited in the proofs of our constructions and we keep the existing definition of [KRB⁺24] for multi-signatures. Finally, we underline that our definition does not make any assumption or limitation regarding ctx, and thus even using a different context in each signing session must not interfere with the blindness guarantees of such signatures.

3.2 One-More Unforgeability

This section presents six different notions of one-more unforgeability. We introduce four new notions (OMUF-0 to OMUF-3) that form a clear hierarchy, and that we consider the most useful measure of a threshold blind signature scheme's security. In all these notions, the adversary can corrupt up to c < t issuers and freely interact with the remaining honest ones. The adversary wins, if it manages to produce more valid signatures than what is trivially achievable based on the adversary's knowledge. Our notions differ in what is considered trivial. Roughly, the number of trivial signatures is increased based on the following events:

OMUF-0: 1 honest issuer outputs a partial signature OMUF-1: t - c issuers output a partial signature OMUF-2: t - c issuers output a partial signature for the same ssid OMUF-3: t - c issuers output a partial signature for the same ssid and S' We also present two notions (OMUF-FC and OMUF-SB) that originate from prior works, and we include them to analyze existing notions and schemes in our hierarchy too. The six security notions differ from each other mainly in their winning conditions (and the bookkeeping necessary to define their winning conditions). The basic game structure of one-more unforgeability is the same in all six cases. Thus, we first explain this general game structure.

Security Game. The one-more unforgeability security game OMUF-x is defined in Figure 1 for $x \in \{0, 1, 2, 3, FC, SB\}$. This game follows the general behavior of traditional one-more unforgeability. The experiment starts with generating the public parameters and initializing the bookkeeping structure for the games. EC_0 , El_2 , El_3 , ES_3 , and ES_{SB} are variables used to define the winning conditions of specific properties; we defer their explanations to the individual descriptions of each property. The experiment also initializes the sets S_1, \ldots, S_r which are used to keep track of in-progress signing sessions. The adversary must tag each signing session with a unique session identifier *sid*, and S_j is the set of (i, sid)pairs such that issuer *i* has executed session *sid* through (at least) round *j*.

Our game models static corruption and trusted key generation following [CKM⁺23]. The adversary chooses the threshold parameters and the corrupted parties (n, t, C). If these values are not well-formed, the threshold t is not satisfiable (n < t), or the corrupted set leads to trivial insecurity $(|C| \ge t)$, the game directly returns false. Otherwise, the adversary gains access to signing oracles. The signing oracle $\mathcal{O}^{\mathsf{lSign}}$ simply mimics the behavior of honest issuers. Under all six security definitions, only execution of the signing oracle for last signing round j = r causes an update to the adversary's winning condition. That is, we consider a partial signature complete when an issuer has engaged in the final round of the protocol. After its interaction with the signing oracles, the adversary ultimately outputs ℓ message/signature pairs.

The winning condition checks that two requirements are met. First, the number of signatures, ℓ , must exceed the number that is allowed by the adversary's actions in the OMUF-x game. This number is determined by the function allow_x in Figure 1 which takes the game state as input and returns an integer that is the maximum number of signatures the adversary can output without violating the OMUF-x property. This allow_x function is where our security models differ, and we explain how we count the number of admissible signatures in the description of each game. We drop the input of the allow_x function for simplicity and just note that it has access to the game state. The second requirement of the winning condition is simply that all ℓ message/signature pairs are valid and distinct.

Definition 3 (TB-OMUF-x). A threshold blind signature scheme TB is x-onemore unforgeable for $x \in \{0, 1, 2, 3, \text{FC}, \text{SB}\}$ if, for all PPT adversaries \mathcal{A} , and $n, t \in \mathbb{N}^+$ such that $t \leq n$, $\Pr\left[\mathsf{Exp}_{\mathsf{TB},\mathcal{A},n,t}^{\mathsf{OMUF-x}}(\lambda) = \mathsf{true}\right]$ is negligible for the experiment from Figure 1.

Next we describe how each OMUF-x game defines its winning condition and explain the corresponding intuitive security guarantees it gives.

 $\mathsf{Exp}_{\mathsf{TB},\mathcal{A},n,t}^{\mathsf{OMUF-x}}(\lambda)$ $pp \leftarrow \mathsf{Pg}(1^{\lambda}); S_1, \ldots, S_r, := \emptyset; \mathsf{ES}_3, \mathsf{ES}_{\mathsf{SB}} := \emptyset$ $\forall i, \mathsf{ssid}, \mathcal{S}_3 : \mathsf{EC}_0[i] := 0, \mathsf{El}_2[\mathsf{ssid}], \mathsf{El}_3[\mathsf{ssid}, \mathcal{S}_3] := \emptyset; \ (\mathcal{C}) \leftarrow \mathcal{A}(pp, n, t)$ if $|\mathcal{C}| \ge t$: return false if $|\mathcal{C}| \ne t - 1$: return false $(pk, \{sk_i\}_{i \in [n]}, aux) \leftarrow \mathsf{Kg}(n, t)$ $(\ell, \{(msg_k^*, \sigma_k^*)\}_{k \in [\ell]}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{lSign}}}(pk, \{sk_i\}_{i \in \mathcal{C}}, aux)$ $\mathbf{return} \left(\ell > \mathsf{allow}_{\mathsf{x}} \right) \land \forall k \in [\ell] : \left(\begin{array}{c} \mathsf{Vf}(pk, \sigma_k^*, msg_k^*) \\ \land \forall j \in [\ell] \setminus \{k\} : (msg_k^*, \sigma_k^*) \neq (msg_j^*, \sigma_j^*) \end{array} \right)$ |ES_{SB}| $|\mathsf{ES}_3|$ allow_{SB}: allow₃: $\mathcal{O}^{\mathsf{lSign}}(j, sid, i, \mathsf{ctx}_i^{sid}, ups_{j-1}^{sid}) \ \# \text{ issuance in round } j \in \{1, \dots, r\} \text{ of issuer } i \in [n]$ **return** \perp if $(i, sid) \in S_i \ //$ ensure this round has not been queried return \perp if $j > 1: (i, sid) \notin S_1, \dots S_{j-1}$ // ensure previous rounds have been queried $S_i := S_i \cup \{(i, sid)\}$ if j = 1 : $(ist_{1,i}^{sid}, ips_{1,i}^{sid}) \leftarrow \mathsf{ISign}_1(i, sk_i, aux, \mathsf{ctx}_i^{sid}, ups_0^{sid})$ $(ist_{j,i}^{sid}, ips_{j,i}^{sid}) \leftarrow \mathsf{ISign}_i(i, ist_{j-1,i}^{sid}, ups_{j-1}^{sid})$ else : $\mathbf{if}~j=r~:~/\!/~\textit{final round updates the state used to determine $\mathbf{allow_x}$}$ Parse $ist_{r,i}^{sid} := (ssid_i^{sid}, \mathcal{S}'_i^{sid})$ $\mathsf{EC}_0[i] := \mathsf{EC}_0[i] + 1$ // bookkeeping for OMUF-0, 1, FC $\mathsf{El}_2[\mathsf{ssid}_i^{sid}] := \mathsf{El}_2[\mathsf{ssid}_i^{sid}] \cup \{i\}$ // bookkeeping for OMUF-2 $\mathcal{S}_3 := \mathcal{S}'^{sid}_i \setminus \mathcal{C}$ // bookkeeping for OMUF-3 $\mathsf{El}_{3}[\mathsf{ssid}_{i}^{sid}, \mathcal{S}_{3}] := \mathsf{El}_{3}[\mathsf{ssid}_{i}^{sid}, \mathcal{S}_{3}] \cup \{i\}$ $\mathbf{if} \ \mathcal{S}_3 \subseteq \mathsf{El}_3[\mathsf{ssid}_i^{sid}, \mathcal{S}_3] \ : \ \mathsf{ES}_3 := \mathsf{ES}_3 \cup \{\mathsf{ssid}_i^{sid}\}$ $\mathsf{ES}_{\mathsf{SB}} := \mathsf{ES}_{\mathsf{SB}} \cup \{(sid, \mathcal{S}'_i^{sid})\}$ // bookkeeping for OMUF-SB return $(ips_{r,i}^{sid}, ssid_i^{sid}, \mathcal{S}'_i^{sid})$ $/\!\!/$ response for round j = rreturn $ips_{i,i}^{sid}$ $/\!\!/$ response for rounds j < r

Fig. 1. One-More Unforgeability Definition OMUF-x for $x \in \{0, 1, 2, 3, FC, SB\}$. The shadowed text is only for x = FC.

OMUF-FC. With OMUF-FC (standing for "full corruption"), we express the first security definition that was considered for threshold blind signatures in [VZK03]. It was later used again in [KM15]. This notion differs from our other five in that it imposes a restriction on the adversary's behavior: the adversary must corrupt exactly t-1 issuers. Under this condition, it is straightforward to define allow_{FC}. Clearly, the adversary should earn one signature for every signing oracle interaction. For all $i \in [n]$, the security game maintains $\mathsf{EC}_0[i]$ as a counter of the signing queries answered by issuer *i*. allow_{FC} is simply the sum of all $\mathsf{EC}_0[i]$.

OMUF-0. Though easy to understand, OMUF-FC does, at first glance, not provide any intuition about security in the case that less than t - 1 issuers are corrupted. We therefore generalize OMUF-FC to OMUF-0, the weakest of the four notions in our hierarchy. OMUF-0 eliminates the restriction that $|\mathcal{C}| = t - 1$ but is otherwise identical to OMUF-FC, i.e. every single signing query adds one allowed signature to allow₀. When there are few (or no) corruptions, this security definition is obviously far weaker than the level of security one would expect from a threshold blind signature scheme. Seemingly, it allows the adversary to flaunt the scheme's threshold entirely!

In Section 4 we prove the surprising result that OMUF-FC and OMUF-0 are equivalent (Lemma 4). For the purpose of our hierarchy, we prefer OMUF-0 over OMUF-FC for two reasons. First, OMUF-0 imposes no restriction on the adversary's behavior, so it matches the game structure used by all our other definitions. Second, OMUF-0 better exposes the concerning weakness of this security notion. No matter how high the threshold t is set, OMUF-0 allows outputting a signature after interacting with just a single issuer in the game.

Related Concepts. Bellare et al. [BTZ22] define the unforgeability notion TS-UF-0 for *non-blind* threshold signatures which requires as the non-triviality condition that no signature share was requested on a message regardless of what the threshold is. They also observe that many existing threshold signature schemes relied on this relatively weak notion.

OMUF-1. Clearly, the security guarantees of OMUF-0 are not satisfactory, so we strengthen it into OMUF-1, which formalizes a much more natural security expectation. In particular, OMUF-1 allows one signature per each set of t interactions with distinct issuers. If the adversary has corrupted some issuers, then less honest issuer interactions (specifically, $t' := t - |\mathcal{C}|$) are needed for each signature.

We believe that OMUF-1 captures the most obvious and intuitive understanding of threshold blind signature security. This notion has not, however, appeared before in this form. The challenge of this model is how to count the number of admissible signatures allow_1 . Due to the blindness and threshold setting, we are not aware which issuers have signed the same blinded input. Thus, we need to find a way to extrapolate the upper bound of signatures that can be derived using the issued partial signatures. This is done via the function **count**, which we defined in Section 2 as part of the T-BOMDH assumption. We make the top-up threshold $t' := t - |\mathcal{C}|$ explicit and use $\mathsf{allow}_1 := \mathsf{count}_{t'}(\mathsf{EC}_0[1], \ldots, \mathsf{EC}_0[n])$ as the number of allowed signatures. Intuitively, this use of count is very similar to its use in T-BOMDH: count assumes the optimal grouping of partial signatures into sets of size t' in order to establish the greatest number of signatures that should be possible after the adversary's interactions with issuers in the OMUF-1 game.

Related Concepts. For non-blind threshold signatures, our OMUF-1 notion is similar to TS-UF-1 by Bellare et al. [BTZ22] which defines a signature to be trivial when a partial signature was generated by at least t - c honest issuers. As their framework is in the non-blind setting, the definition does not require any complex counting arguments.

Similar security properties have been described for other threshold and blind primitives in the Universal Composability (UC) framework, e.g. the threshold oblivious PRF functionality of [JKKX17] and the threshold anonymous credentials functionalities of [RP22] and [DKL+23]. As these notions are in the UC framework, they did not require any explicit counting of the number of trivial evaluations. The threshold OPRF security definition of [AMMM18] has utilized the function **count** in the game-based setting. That definition, however, describes a game where the adversary must evaluate the PRF on an input that is unknown to it; our notion of one-more unforgeability does not compare to that setting.

OMUF-2. Though OMUF-1 ensures a reasonable signature count, it does not provide for any coordination or agreement mechanism between issuers. We therefore continue our hierarchy with OMUF-2, which requires that the issuers' final signing round outputs a postfix-style subsession identifier ssid. This is already part of our syntax as defined in Def. 2. Our OMUF-2 game then uses this ssid to model some form of consensus among the issuers. In a concrete protocol, ssid can, e.g., be the view the individual issuer's had, and/or some pre-agreed context information ctx. In fact, this approach is inspired by the recent Bare PAKE model of Barbosa et al. [BGHJ24], in which parties output unique and consistent ssids instead of taking them as input (which requires an agreement pre-phase).

In the OMUF-2 game, $El_2[ssid]$ is the set of issuers that have outputted ssid. The allowed number of signatures allow₂ is the number of ssid values that have been outputted by at least t issuers (the participation of all corrupted issuers is assumed). Our definition does not specify exactly how each protocol should define ssid, but two basic conditions must be met. First, it must be impossible to compute a signature without having t issuers output the same ssid. Second, it must be impossible to compute two different signatures using issuer interactions that produce the same ssid (i.e. ssids must be unique). We believe that reasonable protocols satisfying OMUF-2 will choose ctx and/or some or all of the transcript of user messages ups_1, \ldots, ups_r as their ssid. For instance, in our BLS protocols we use ups_1 , which is the blinded message from the user, as part of ssid. Our OMUF-2 notion then guarantees that the user can only obtain a valid signature on a message after sending the same ups_1 to t issuers. Session Agreement. OMUF-2's ssid agreement guarantee is valuable for two main reasons. First, issuers who keep logs of their ssid outputs can later determine which of their blind signing sessions go together. Tracking of this type could be useful for system administration and abuse prevention, and it does not violate blindness. Issuers still cannot learn which signatures were created by which sessions; they can simply "link" their individual sessions together. Second, if ssid includes ctx, then issuers can use the ctx field to enforce agreement on arbitrary side data. If ctx contains a timestamp or counter, this behavior would enforce a form of freshness and implicit coordination between issuers, without requiring an interactive consensus round. If ctx contains a username, this behavior would enforce that the same user authenticates to all issuers. Once again, blindness is not violated. This side data is not a part of the final signature, yet issuers must agree upon it during the signing process.

Considering the possible applications of threshold blind signatures, e.g., distributed e-cash of anonymous tokens for rate-limiting, such an added control for the issuers will be essential for the desired upper-level functionality.

Related Concepts. Though this security property has not previously been considered for threshold blind signatures, there have been similar notions considered in related works. For threshold (non-blind) signatures, Bellare et al. [BTZ22] proposed an unforgeability property (TS-UF-2) that requires t signers to not just sign the same message, but to record the same "leader request" (which may contain some other data in addition to the message). For threshold OPRFs, Gu et al. [GJK⁺25] proposed a UC functionality that requires t servers to run on the same side input (comparable to our notation's ctx).

OMUF-3. The security guarantee of OMUF-2 enables different paradigms of coordination between issuers, but it does not accommodate issuers that wish to coordinate each other's *identities* during the blind signing process. Even if issuers believed they knew the signing set S and included it in ssid, OMUF-2 would offer no assurance of that signing set's correctness. At the extreme, consider the case that t-1 issuers are corrupted. An honest issuer might wish to sign alongside a set S of other honest issuers, but in reality just that one honest issuer's participation is sufficient for the adversary to complete a signature. To fill this gap, we propose OMUF-3, the strongest one-more unforgeability definition of our hierarchy.

In OMUF-3, we require that each issuer's final signing round outputs a supposed signing set S' in addition to ssid. Again, this is already part of our syntax defined in Def. 2, but wasn't needed so far. It is not meaningful to differentiate between corrupted issuers, so the OMUF-3 game first pares down S' to $S_3 := S' \setminus C$, the honest subset of S'. Intuitively, OMUF-3 enforces that the issuer's interaction only counts toward producing a signature if every other issuer in S_3 also executes and agrees upon S_3 and ssid. El₃[ssid, S_3] is the set of issuers that have outputted ssid and S_3 (or, to be specific, any S' whose honest subset is S_3). Once every issuer in S_3 is in El₃[ssid, S_3], ssid is added to the set ES₃, thereby incrementing allow₃. In order to keep OMUF-3 strictly stronger than

OMUF-2, we maintain the semantic that a single ssid can only correspond to a single signature, i.e. two different signing sets that output the same ssid cannot produce two different signatures.

Signer Set Agreement. OMUF-3 allows for the highest level of coordination between issuers. Not only can they enforce agreement on arbitrary side data as in OMUF-2, they can choose whether or not to participate in signing alongside certain other issuers. For example, suppose it is discovered that a certain issuer is acting suspiciously. The other issuers might refuse to participate in signing sessions that include that suspicious issuer in their signing set. In this way, corruptions (below the threshold t) can be handled without losing security or needing to repeat key generation.

Related Concepts. This notion is similar in spirit to TS-UF-3 and TS-UF-4 in the definitional framework by Bellare et al. [BTZ22]. Therein, the security game also demands that a signature can only be created when all honest issuers in the signing set have contributed their shares. The reason why their framework for non-interactive and non-blind schemes has two notions, is that they differentiate whether some preprocessing information can be maliciously generated or not. This is not relevant in our blind setting, as signers always operate on possibly maliciously generated input from the user.

OMUF-SB. Finally we recap the one-more unforgeability definition proposed by Crites et al. [CKM⁺23] for their Snowblind protocol, which we denote as OMUF-SB. This game maintains $\mathsf{ES}_{\mathsf{SB}}$ to be the set of (sid, \mathcal{S}') pairs such that at least one issuer completed a signing session with identifier *sid* that outputted supposed signing set \mathcal{S}' . Recall that *sid* is purely used for bookkeeping within the security game, it is not an explicit input to any of the protocols. The allowed number of signatures $\mathsf{allow}_{\mathsf{SB}}$ is the cardinality of the set $\mathsf{ES}_{\mathsf{SB}}$.

It is difficult to intuitively express the security property implied by OMUF-SB. On the one hand, it seems strong in that it enforces some semantic on the signing set S'. In our hierarchy, only the strongest notion OMUF-3 involves S'. On the other hand, it seems weak in that it allows for a signature to be computed after interacting with just a *single* honest issuer. In our hierarchy, only the weakest notion OMUF-0 exhibits this behavior. If the adversary uses a different *sid* for every issuer interaction, then every single one will increment allow_{SB}, allowing for one more signature. However, an adversary trying to violate the property would want to keep allow_{SB} low and would therefore want to reuse the same *sid* for as many sessions as possible. Since S' must be of size t, the same (*sid*, S') can only result from at most t individual issuer interactions, which seems to enforce some version of a natural threshold counting property. In the case of one or more corruptions, though, the consequences of this property are not very clear.



Fig. 2. Relations between one-more unforgeability notions. An arrow $A \rightarrow B$ indicates that notion A implies notion B. The diagram is exhaustive, i.e. if there is no path from A to B, then notion A does not imply notion B.

4 Relations Between Unforgeability Definitions

We now prove relations between all introduced OMUF notions. Figure 2 depicts the implication relations between the six one-more unforgeability notions that we consider. In this section we prove that the diagram is correct and exhaustive.

Results in a Nutshell. We first prove that our newly introduced notions OMUF-0 to OMUF-3 form a clear hierarchy with strictly increasing security guarantees. We also show that the existing notion OMUF-FC and OMUF-0 imply each other, demonstrating that the seemingly intuitive notion OMUF-FC is not sufficient in a threshold setting.

For OMUF-SB, the situation is more complex. We prove that OMUF-SB is strictly stronger than OMUF-0, strictly weaker than OMUF-3, and yet incomparable to (i.e. neither weaker nor stronger than) OMUF-1 and OMUF-2. It is left, then, in a mysterious position relative to our hierarchy.

<u>OMUF-0 to OMUF-3 Hierarchy</u>. We start with showing that our four security models form a clear hierarchy. Most relations are pretty straightforward, with the exception of the separation between OMUF-0 and OMUF-1 security.

Lemma 1. OMUF-3 security implies OMUF-2 security. OMUF-2 security implies OMUF-1 security. OMUF-1 security implies OMUF-0 security.

Proof. These most basic implications in our OMUF hierarchy are immediate from the security game definitions (Figure 1). In particular, $\mathsf{allow}_3 \leq \mathsf{allow}_2 \leq \mathsf{allow}_1 \leq \mathsf{allow}_0$. Thus, any winning adversary against a lower-numbered security game is also a winning adversary against a higher-numbered security game.

It remains to show that these implications are strict, i.e. none of the 4 notions in our hierarchy are actually equivalent to one another. Lemma 2 states that OMUF-1 security is strictly stronger than OMUF-0 security. This is an interesting result because Bellare et al. [BTZ22] show that their TS-UF-0 notion implies their TS-UF-1 notion for small n, t parameters. Their argument relies on guessing the subset of honest servers that will sign the eventual forgery target of a TS-UF-1 attacker. In contrast, we show that OMUF-0 does not imply OMUF-1 in any case. On a high level, this difference arises because, in the OMUF games, individual server interactions are not associated with particular messages, and the subset-guessing strategy is therefore inapplicable.

Lemma 2. OMUF-0 security does not imply OMUF-1 security.

Proof (Sketch). Starting from any OMUF-1 secure protocol Π^1 , we construct a scheme Π^0 that is OMUF-0 but not OMUF-1 secure. Π^0 key generation runs Π^1 key generation twice, once with the correct threshold parameters (n,t) and a second time with (n,t+1). Both sets of secret keys are distributed to the issuers, and Π^0 verification accepts a Π^1 signature under either public key.

The Π^0 issuers' signing procedure runs in one of two modes based on a bit provided by the user in ups_0 . Honest users always use mode 0, and in this mode issuers perform standard Π^1 signing under their (n,t) keys. If a dishonest user invokes mode 1, then issuers perform a *double* Π^1 signing under their (n, t+1)keys. In other words, a single Π^0 mode 1 issuer interaction is equivalent to two Π^1 issuer interactions.

Clearly, Π^0 is not OMUF-1 secure. Adversary \mathcal{A} can generate 2 signatures after interacting with t + 1 issuers in mode 1. As long as $t \geq 2$, $\mathsf{allow}_1 = 1$ in this case and \mathcal{A} wins OMUF-1. Π^0 is, however, OMUF-0 secure. \mathcal{A} corrupts at most t - 1 servers. To take advantage of mode 1's double-signing behavior, \mathcal{A} must interact with t + 1 servers, but doing so requires interacting with at least 2 honest servers. \mathcal{A} gets 2 signatures this way, but allow_0 also increments by (at least) 2. A complete proof of Lemma 2 is provided in Appendix D.

The separations between OMUF-1, OMUF-2, and OMUF-3 are more easily apparent. Here we can exploit that their security crucially depends on ssid and S', respectively, that are output by the final signing round. Thus, simply omitting or manipulating such outputs will have no impact for the security notion where this is not needed, but trivially break security of the higher level.

Lemma 3. OMUF-1 security does not imply OMUF-2 security, and OMUF-2 security does not imply OMUF-3 security.

Proof. For showing that OMUF-2 is strictly stronger than OMUF-1, we modify an OMUF-2 secure scheme, such that issuers always output ssid := \perp . This preserves OMUF-1 security, but loses OMUF-2 security. For OMUF-3 vs. OMUF-2 we modify a OMUF-3 secure scheme so that issuers receive the supposed signing set S from the user and simply output it. Clearly, the resulting scheme is still OMUF-2 secure, but it is not OMUF-3 secure.

<u>OMUF-FC</u> in our Hierarchy. Next we prove that OMUF-FC is equivalent to $\overline{\text{OMUF-0}}$, which proves that OMUF-FC is not a desirable notion for signatures with t < n, as it does not take any advantage of the threshold setting.

Lemma 4. OMUF-0 security and OMUF-FC security imply one another.

Proof. It is immediate that OMUF-0 security implies OMUF-FC security. The OMUF-FC game is just a restricted version of the OMUF-0 game, so any winning OMUF-FC adversary is also a winning OMUF-0 adversary.

To show that OMUF-FC security implies OMUF-0 security, suppose there exists a winning OMUF-0 adversary \mathcal{A}_0 who corrupts $t^* \leq t-1$ issues \mathcal{C}_0 . Construct

reduction $\mathcal{R}_{\mathsf{FC}}$ which picks any additional $t - 1 - t^*$ issuers to create corrupt set $\mathcal{C}_{\mathsf{FC}}$ of size t - 1, $\mathcal{C}_0 \subseteq \mathcal{C}_{\mathsf{FC}}$. \mathcal{A}_0 makes allow₀ queries to issuers in $[n] \setminus \mathcal{C}_0$ and then outputs $\ell > \mathsf{allow}_0$ signatures. Upon queries to issuers in $\mathcal{C}_{\mathsf{FC}} \setminus \mathcal{C}_0$, $\mathcal{R}_{\mathsf{FC}}$ responds directly; upon queries to issuers in $[n] \setminus \mathcal{C}_{\mathsf{FC}}$, $\mathcal{R}_{\mathsf{FC}}$ queries the OMUF-FC game (denote the number of such queries as q). Clearly, $q \leq \mathsf{allow}_0$. In the end, $\mathcal{R}_{\mathsf{FC}}$ outputs $\ell > \mathsf{allow}_0 \geq q$ signatures and thereby wins the OMUF-FC game. \Box

<u>OMUF-SB</u> in our Hierarchy. We now analyze the recently proposed OMUF-SB notion [CKM⁺23], and show that it lies somewhere between OMUF-0 and OMUF-3 security in our hierarchy. We start with the positive results, and prove that OMUF-SB \rightarrow OMUF-0 and OMUF-3 \rightarrow OMUF-SB. Then we show that OMUF-SB does not imply OMUF-1, and OMUF-SB is not implied by OMUF-2.

Lemma 5. OMUF-SB security implies OMUF-0 security.

Proof. This implication is immediate from the security game definitions. In particular, $\mathsf{allow}_{\mathsf{SB}} \leq \mathsf{allow}_0$. Therefore, any winning adversary against the OMUF-0 game is also a winning adversary against the OMUF-SB game.

Lemma 6. OMUF-3 security implies OMUF-SB security.

The proof of Lemma 6 uses a simple combinatoric argument to show that $\operatorname{allow}_3 \leq \operatorname{allow}_{SB}$. We defer this proof to Appendix D.

Lemmas 7 and 8 offer negative results that complete the comparison of OMUF-SB to our hierarchy. The key observation behind the proof of Lemma 7 is that OMUF-SB and -2 winning conditions concern on independent outputs of the signing protocol, the signer set S' and the signing postfix ssid, respectively. Thus, they do not necessarily imply each other. Below we manipulate the S' outputs of a OMUF-SB and -2 secure scheme and it loses OMUF-SB property, but it still reserves OMUF-2 property.

Lemma 7. OMUF-2 security does not imply OMUF-SB security.

Proof. Consider any OMUF-2 secure scheme. Modify it so that issuers receive the supposed signing set S from the user and simply output it. Clearly, the resulting scheme is still OMUF-2 secure. It is not, however, OMUF-SB secure. Consider the OMUF-SB game for any parameters (n, t) which satisfies t < n. An adversary A who corrupts t - 1 issuers (i.e. the maximum) can interact with 2 different uncorrupted issuers using the same (sid, S), but signing 2 different messages. allow_{SB} = 1, but A outputs 2 signatures.

While OMUF-SB security implies OMUF-0 security, we show below that we cannot hope for any better from OMUF-SB. In fact, in the proof of Lemma 8, we rely on a generic transformation that starts from a scheme Π^0 that is OMUF-0 but not OMUF-1 secure, and turns it into a scheme that also is OMUF-SB secure. We do this by changing Π^0 to always output distinct signing sets S' for each issuer. OMUF-SB counts the fresh (sid, S') values and this scheme creates a fresh (sid, S') for each output signature share, so $\mathsf{allow}_0 = \mathsf{allow}_{SB}$. Thus, OMUF-0 security of this scheme implies its OMUF-SB security.

Lemma 8. OMUF-SB security does not imply OMUF-1 security.

Proof. Suppose there exists an OMUF-SB secure scheme Π^{SB} (if not, we are done). By Lemma 5, Π^{SB} is OMUF-0 secure. Therefore, by Lemma 2, there exists a scheme Π^0 that is OMUF-0 secure but not OMUF-1 secure.

Construct scheme Π^* . If t = n, then Π^* runs Π^{SB} . If t < n, then Π^* runs Π^0 , but modified so that issuer *i* always outputs supposed signing set $S = \{i, i + 1, \ldots, i + t - 1\}$ (any indexes $j \in S$ such that j > n should be "wrapped around" to j - n).

Clearly, Π^* is not OMUF-1 secure (per the proof of Lemma 2, there is a successful adversary against the OMUF-1 security of Π^0 with parameters t < n). Π^* is, however, OMUF-SB secure. When t = n, this is immediate from the OMUF-SB security of Π^{SB} . When t < n, every issuer interaction is guaranteed to output a unique (sid, S); therefore, $\mathsf{allow}_{SB} = \mathsf{allow}_0$. The OMUF-0 security of Π^0 implies that the scheme is OMUF-SB secure.

Special Cases: t = 1 or t = n. Our OMUF definition requires a threshold blind signature scheme to satisfy the property for all parameters (n, t) such that $1 \le t \le n$. Nonetheless, it is also interesting to consider the relations between the different OMUF definitions in the edge cases where t = 1 or t = n. In Appendix E, we show the following results, which simplify our OMUF relations when the parameters are restricted to $t \in \{1, n\}$:

Lemma 9. When $t \in \{1, n\}$, OMUF-0, OMUF-1, and OMUF-SB are equivalent to each other; also, OMUF-2 and OMUF-3 are equivalent to each other.

5 Threshold Blind BLS Signatures

We now analyze and improve the security of BLS-based threshold blind signatures. Standard BLS signatures rely on a bilinear group $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, q)$ and let an issuer with secret key $k \in \mathbb{Z}_q$ and public key $pk = g^k$ sign a message msg as $\sigma := H(msg)^k$. Verification checks that $e(g, \sigma) = e(pk, H(msg))$. Threshold BLS signatures can be realized through secret-sharing of k, and either multiplicative or exponential blinding of H(msg) yields a blind signature [Bol03].

We start by analyzing the construction tBlindBLS-1 that combines the (exponential) blind and threshold signature techniques in a straightforward manner and was made explicit by Vo et al. [VZK03]. This construction was proven OMUF-FC secure, and we show that it also satisfies OMUF-1 security. tBlindBLS-1 cannot achieve OMUF-2 or OMUF-3 security as the signature shares are neither bound to the signing session nor to the signing set. We therefore propose a new scheme tBlindBLS-2 that uses the blinding factor idea from [GJK+25, DR24] and prove that it is OMUF-2 secure. tBlindBLS-2 does not achieve OMUF-3 security though, as it does not tightly bind the signature shares to a particular issuer set. This can also be seen as an advantage, as the user can freely combine any t shares on the same blinded message a. However, for settings where a clear agreement



Fig. 3. Threshold Blind BLS signature scheme with three variants: tBlindBLS-1 (dashed text), tBlindBLS-2 (boxed text), and tBlindBLS-3 (shadowed text). $\Lambda_i^{\mathcal{S}}$ denotes the Lagrange coefficient for the index *i* over the set \mathcal{S} . Note that the input \mathcal{S} to USign₀ can be set to \perp for tBlindBLS-1 and tBlindBLS-2 as it does not serve any role there.

on the signing set is desired, we propose our third variant tBlindBLS-3, which uses pairwise blinding factors. We prove that it is OMUF-3 secure.

All three constructions are presented in Figure 3. The main difference between the schemes lies in how issuers create partial signatures. We note that tBlindBLS-2 requires the DDH assumption to be hard in the pairing group \mathbb{G}_2 , so this construction requires asymmetric pairings (i.e. type-2 or type-3). The other two constructions can be instantiated with all pairing types.

Correctness and Blindness It is easy to see that all three tBlindBLS variants satisfy threshold blind signature correctness. In fact, all three produce standard BLS signatures. They also all satisfy threshold blind signature blindness: blinded messages $a := H(msg)^r$ are uniformly random in \mathbb{G}_2 unless $H(msg) = 1_{\mathbb{G}_2}$, which only occurs with probability q_H/q in ROM for q_H hash queries. We defer the proof of blindness to Appendix F.

One-More Unforgeability We now prove the OMUF-1, 2, and 3 security of tBlindBLS-1, 2, and 3 from Figure 3, respectively. While we present proof sketches

and explain the main strategies of proofs here, the full proofs are presented in Appendix F.

OMUF-1 Security of tBlindBLS-1. The existing BLS-based threshold blind signature scheme tBlindBLS-1 was previously shown to be OMUF-FC secure under the chosen-target CDH assumption [VZK03]. We show that it also satisfies OMUF-1 under the T-BOMDH assumption, which uses a counting-based winning condition similar to the OMUF-1 game. T-BOMDH and its pairing-free variant were used previously to show the security of (verifiable) threshold OPRFs [KMMM23, JKKX17], and our proof strategy is similar to these works.

Theorem 1 (tBlindBLS-1 OMUF-1). If the (t', n, N)-T-BOMDH assumption holds on \mathbb{G} for all $1 \leq t' \leq t$, then tBlindBLS-1 is OMUF-1 secure, where (n, t)are the initialization parameters, N is the number of $H(\cdot)$ queries, and H is modeled as a random oracle.

Proof (Sketch). In the proof, we simulate the public key pk as the challenge $g^{p(0)}$ and the secret shares of corrupted issuers are sampled as random values. Such a simulation of keys requires making a TBOMDH $(i, a)_p$ query for each signing query $(ups_0 := (a, S))$ and an honest issuer i. We further simulate random oracle queries by using the T-BOMDH challenge values \hat{g}_j and setting $H(msg_j) := \hat{g}_j$. This ensures that for each valid forgery σ_k^* , we have $\sigma_k^* = H(msg_k^*) = \hat{g}_j^{p(0)}$ for some j. Finally, the winning condition of OMUF-1 ensures we have the enough σ_k^* values to win the T-BOMDH game.

<u>OMUF-2</u> Security of tBlindBLS-2. In tBlindBLS-2, the issuer secret keys are extended with secret shares of 0 labeled $\{z_i\}$. The signature share of the issuer *i* is computed as $b_i := (a^{k_i} \cdot \tilde{H}(\mathsf{ctx}, a)^{z_i})$ where $a = H(msg)^r$ is the user's blinded message. Thus, the signing postfix of this scheme is set to ssid := (ctx, a) . Intuitively, the signature shares on the blinded message *a* are bound to each other with additional terms $\tilde{H}(\mathsf{ctx}, a)^{z_i}$, so a user needs *t* shares on the same $\tilde{H}(\mathsf{ctx}, a)$ to get a valid signature. To prove that this scheme achieves OMUF-2 security, we rely on the Decisional Diffie-Hellman (DDH) and Bilinear One-More Diffie-Hellman (BOMDH) assumptions. We note that our proof closely follows the proof of Gu et al. [GJK⁺25], as our tBlindBLS-2 construction is similar to their threshold OPRF construction.

Role of ctx. Including ctx in H(ctx, a) gives the issuers more control over their signing. For instance, by setting ctx to the current date, OMUF-2 security guarantees that the user cannot combine signature shares received on different days. Or even more strictly, ctx could be the verified user name, ensuring that a signature on a blinded message a can only be derived when the user properly authenticated to at least t issuers. If this control is not desired, and the user should be flexible in combining her shares, ctx is simply set to \perp .

Theorem 2 (tBlindBLS-2 OMUF-2). If the DDH on \mathbb{G}_2 and (N)-BOMDH assumptions hold, then tBlindBLS-2 is OMUF-2 secure, where N is the number of $H(\cdot)$ queries, and H and \tilde{H} are modeled as random oracles.

Proof (Sketch). Our proof relies on the BOMDH assumption by setting the group public key $pk := g^p$ and $H(msg_j) := \hat{g}_j$ so that for each forgery we have $\sigma_j = \hat{g}_j^p$ for some $j \in [N]$. As the public key is set to g^p our reduction cannot know the individual k_i values of honest issues *i*. Therefore, we must leverage the fact that the first $t - 1 - |\mathcal{C}|$ signature shares requested for any blinded message a do not reveal any information about a^{k_i} because the factor $H(\mathsf{ctx}, a)^{z_i}$ acts as a pseudorandom value. Our proof follows a hybrid argument that systematically replaces $H(\mathsf{ctx}, a)^{z_i}$ values with random values, using the DDH assumption at each step. After this change, we can simply use random values for the first $t-1-|\mathcal{C}|$ signature shares on any blinded message a. For the $(t-|\mathcal{C}|)$ th query, we finally use the BOMDH_p oracle to learn a^p and then compute the response that will aggregate with the previously chosen random signature shares into the correct final signature. Our reduction makes a BOMDH_n query per each ssid := a only when there are at least $t - |\mathcal{C}|$ signing queries that output ssid, so the number of BOMDH_p queries is equal to allow_2 ; our reduction wins BOMDH if \mathcal{A} wins the OMUF-2 game.

<u>OMUF-3</u> Security of tBlindBLS-3. Compared to tBlindBLS-2, tBlindBLS-3 relies on stricter binding factors, which make use of pairwise key material held by all issuers. These pairwise binding factors are computed via a pseudorandom function F, that is invoked on all pairwise keys of the involved issuers in the desired signer set S. These values only cancel out when the user receives t signature shares from exactly the intended issuers. The same blinding technique was used by [CLN15, BFH⁺20] to obtain adaptive security of distributed OPRFs.

Theorem 3 (tBlindBLS-3 **OMUF**). If F is a pseudorandom function and the (N)-BOMDH assumption holds on \mathbb{G} , then tBlindBLS-3 on \mathbb{G} is OMUF-3 secure, where N is the number of $H(\cdot)$ queries, and H is modeled as a random oracle.

Proof (Sketch). Our proof strategy is similar to the proof of OMUF-2 security of tBlindBLS-2. As in the proof of Theorem 2, our proof aims to build an (N)-BOMDH adversary, so it sets the group public key $pk := g^p$ and $H(msg_j) := \hat{g}_j$. To simulate the signature shares of honest parties, we proceed as follows: For any pair of honest issuers $i, j \in [n] \setminus C$, their pairwise key $w_{i,j}$ is unknown to the adversary. Thus, we can replace $F_{w_{i,j}}$ with a truly random function $f_{i,j}(\cdot)$ by relying on the pseudorandomness of F. This means that for each ups = (a, S), signature shares of honest issuers except the last honest issuer in S can be replaced by a random value. The signature share of the last honest issuer in the set is computed by first receiving $a^p = \text{BOMDH}_p(a)$ from the BOMDH challenger and multiplying a^p with the inverse of the previously output honest issuer signature shares to ensure the indistinguishable simulation. It is straightforward to observe that we make a BOMDH_p query only when ES₃ gets updated and the number of BOMDH_p queries is equal to allow₃. Thus, if \mathcal{A} wins the OMUF-3 game, then we break the BOMDH assumption. □



Fig. 4. Threshold blind signature scheme of $[CKM^+23]$. Λ_i^S denotes the Lagrange coefficients. DS := (Pg, Kg, Sign, Vf) is an unforgeable signature scheme and GGen is a secure prime-order group generator.

6 Pairing-Free Threshold Blind Signatures: SB and SB+

Crites et al. [CKM⁺23] recently introduced the pairing-free threshold blind signature scheme Snowblind (SB) and proved it secure in the model we restated as unforgeability notion OMUF-SB. Per our analysis in Section 4, OMUF-SB only immediately implies OMUF-0 security, so the position of Snowblind in relation to our security hierarchy is left in question. The construction SB and security model OMUF-SB clearly evince a desire for issuers to coordinate each other's identities, so OMUF-3 is the natural security goal to consider.

We first show that SB does not satisfy OMUF-3 security, but we then show how to lift it into a stronger version that does. We do this in a partly generic manner, by proposing a construction that builds on any OMUF-SB secure scheme and applies a similar pairwise binding technique as our tBlindBLS-3 construction. To prove OMUF-3 security, we require the underlying scheme to satisfy an additional property, *issuer set consistency*, which we define and show to be met by [CKM⁺23]'s SB. Snowblind SB Scheme [CKM⁺23]. The threshold blind signature scheme of [CKM⁺23] is presented in Figure 4. The non-threshold version of their blind signature follows the design idea of Tessaro-Zhu blind signatures [TZ22], but [CKM⁺23] provides shorter signatures. The scheme can be seen as an extension to Schnorr blind signatures and verifies as $\overline{R} \cdot pk^{\overline{c}\cdot\overline{y}} \stackrel{?}{=} g^{\overline{z}} \cdot h^{\overline{y}}$ for a non-zero \overline{y} . It follows a novel approach by having the issuer contribute to sampling \overline{y} uniformly with a uniformly random Υ , and answer Fiat-Shamir response z over $c \cdot \Upsilon$ instead of a maliciously chosen challenge c. This allows them to avoid relying on the broken ROS assumption [BLL⁺21]. Crites et al. [CKM⁺23] thresholdize it by collaboratively sampling Υ with t issuers. For simplicity, we state the protocol in the original form, i.e., without fully mapping to our syntax, but Appendix G contains the scheme as written in our syntax for completeness.

Crites et al. [CKM⁺23] actually propose two variants of their signature scheme. In a nutshell, while the first variant checks if $\overline{R} \cdot pk^{\overline{c}\cdot\overline{y}} = g^{\overline{z}}h^{\overline{y}}$ during the signature verification, the second variant checks if $\overline{R} \cdot pk^{\overline{c}+\overline{y}^5} = g^{\overline{z}}h^{\overline{y}}$. The two variants also differ in how the blinding on the user side, and the partial signature in the last signing round $|\text{Sign}_3|$ are computed. We only study the first variant, but note that our attack against OMUF-3 security can be easily applied to the second variant as well. Further, as our boost is built generically from any OMUF-SB secure scheme, it can be applied on both variants. Finally, note that we omit the public key shares $pk'_j := g^{sk'_j}$ that were computed in the original Kg algorithm as they were not used in the protocol.

6.1 SB Is Not OMUF-3 Secure

We show that the SB construction in Figure 4 is not OMUF-3 secure. This might be surprising, as SB seems to perform some agreement on the set of co-issuers S in the second issuer signing round. However, the final signature shares are not bound to the co-issuer set. Due to this behavior, SB cannot guarantee that a signature is created by the exact co-issuers that was intended by an individual issuer – which violates our OMUF-3 notion.

Lemma 10. The SB construction [CKM⁺23] (Figure 4) is not OMUF-3 secure.

We provide a sketch of the attack here and refer to Appendix H for the detailed description. We build a simple adversary against the OMUF-3 security of SB that uses $(n := 3, t := 2, C := \{3\})$. \mathcal{A} pretends to start a signing session between issuers 1 and 2, but gets a signature share only from issuer 2, and creates a valid signature using it. As there will only be a single final round signing query, allow₃ = 0, no matter what ssid value we use for SB.

This is done as follows. \mathcal{A} makes first round queries via $\mathcal{O}^{\mathsf{lSign}}$ to issuers 1 and 2 for $ups_1 := (\mathcal{S} := \{1, 2\})$, and gets $ips_{1,1} := (A_1, B_1, \mathsf{cm}_1)$ and $ips_{1,2} := (A_2, B_2, \mathsf{cm}_2)$, respectively. Then \mathcal{A} sets $\bar{R} := A_2^{A_2^{\mathcal{S}'}/A_2^{\mathcal{S}}} \cdot B_1 \cdot B_2$ where $\mathcal{S}' := \{2, 3\}$ and sets $c := \mathsf{H}_{sig}(pk, \bar{R}, msg^*)$ for an arbitrary message msg^* . Here A_2 is omitted while computing \bar{R} as \mathcal{A} cannot compute a valid Fiat-Shamir response \bar{z} for such a \bar{R} without making a final round signing query to issuer 1. \mathcal{A} then makes a second round signing query to both issuers with $ups_1 := (c, \{\mathsf{cm}_1, \mathsf{cm}_2\})$. As a result, both issuers respond with valid $ips_{2,i} := (b_i, y_i, ds_i)$ values, and \mathcal{A} makes a third round signing query via to the issuer 2 to get z_2 such that $A_2 \cdot pk_j^{\Lambda_2^S \cdot c \cdot \Upsilon} = g^{z_2}$ for $\Upsilon := y_1 + y_2$. Finally, \mathcal{A} sets $\sigma^* := (\bar{R}, \bar{y}, \bar{z})$ for $\bar{y} := \Upsilon$, $z_3 := c \cdot \Lambda_3^{S'} \cdot sk_3 \cdot \Upsilon$, and $\bar{z} := z_2 \cdot \Lambda_2^{S'} / \Lambda_2^S + z_3 + b_1 + b_2$ which is a valid signature on message msg^* . As there is no third round query for issuer 1, allow_{SB} = 0 and the adversary wins by outputting msg^*, σ^* .

6.2 SB+: OMUF-3 Variant of Snowblind

We now show how the original Snowblind construction can be lifted to obtain OMUF-3 security. Instead of directly extending the concrete SB construction, we do this in a somewhat generic way be describing how OMUF-SB security can be boosted to OMUF-3 security. This has the advantage that we do not have to repeat the security proof of the core SB construction and that our boost directly applies to both protocol variants proposed by Crites et al. [CKM⁺23].

Our boost Π^3 is presented in Figure 5 and closely follows the idea behind the OMUF-3 secure tBlindBLS-3 scheme from Section 5. It is constructed from a PRF F and a OMUF-SB secure threshold blind signature scheme Π^{SB} . We note that Π^3 assumes that the underlying Π^{SB} has $r \ge 2$ rounds. This is satisfied by SB and makes the transform simpler. If this condition is not met for a future construction that is shown to be OMUF-SB secure, it can be "padded" to r = 2by adding a dummy first round in which the user and issuers simply exchange acknowledgments.

The challenge in upgrading OMUF-SB security to OMUF-3 security lies in how each notion counts the required number of forgeries. In OMUF-SB, a signature query with a fresh (sid, S) immediately increments allow_{SB}, whereas in OMUF-3, additional conditions must be met before allow₃ is incremented. Our construction employs PRF-based maskings as in tBlindBLS-3 to align the counting mechanisms of allow_{SB} and allow₃.

Our construction first sets pairwise keys $w_{i,j}$'s for all issuers to be used as PRF keys. In the signing protocol, the postfix ssid³ for a signing session is created with an additional round. This value is simply set by collecting fresh nonces from each issuer. Then the remaining rounds mainly follow the underlying OMUF-SB secure scheme Π^{SB} until the final round. In the final round, similar to tBlindBLS, issuers set a pairwise blinding value for the signature shares using PRF masks with $F_{w_{i,j}}$ evaluations. Thus, in order to get the signature shares of the underlying scheme Π^{SB} , the user has to contact each issuer from the issuer set that was initially intended for the signing session. The freshness of the signing postfix ssid³ ensures that the blinding values from F evaluations of different sessions cannot cancel out, and thus the attack described above no longer applies.

One More Ingredient: Issuer Set Consistency. While the blueprint above may seem to be enough to show the OMUF-3 security of Π^3 by relying on a OMUF-SB secure scheme and a secure PRF, this is not the case. In OMUF-SB, the count of

 $\Pi^3 . \mathsf{Pg}(1^{\lambda})$ Set PRF $F: \mathcal{K} \times \{0,1\}^* \to \{0,1\}^l$, where l is the bitlength of the ips_{π} output by Π^{SB} . ISign_... Run $pp^{\mathsf{SB}} \leftarrow \Pi^{\mathsf{SB}}.\mathsf{Pg}(1^{\lambda}).$ Return $(pp^{\mathsf{SB}}, F).$ $\frac{\boldsymbol{\varPi}^{3}.\mathsf{Kg}(n,t)}{(pk,\{sk_{i}^{\mathsf{SB}}\}_{i\in[n]},aux)\leftarrow\boldsymbol{\varPi}^{\mathsf{SB}}.\mathsf{Kg}(n,t)}$ $\frac{\varPi^3.\mathsf{USign}_j(ust_{j-1}^3,\{ips_{j,i}^3\}_{i\in\mathcal{S}}) \quad \# j \in \{2,\ldots,r\}}{\mathbf{if} \ j=1}$ $\mathbf{return} \ (ust_1^3 := ust_0^3, ups_1^3 := \{ips_{1,j}^3\}_{j \in \mathcal{S}})$ $\{w_{i,j}\}_{i,j\in[n],i< j} \leftarrow_{\$} \mathcal{K}^{n(n-1)/2}$ $\forall_{i,j\in[n],i>j} \ w_{i,j} := w_{j,i} \forall_{i\in[n]}$ if $j \in \{2, ..., r\}$: Parse $ips_{j,i}^3 = ips_{j-1,i}^{SB}$ and $ust_{j-1}^3 := ust_{j-2}^{SB}$ $sk_i^3 := (sk_i^{\mathsf{SB}}, \{w_{i,j}\}_{j \in [n] \setminus \{i\}})$ return Π^{SB} .USign_{*j*-1}($ust_{j-2}^{SB}, \{ips_{j-1,i}^{SB}\}_{i \in S}$) return $(pk, \{sk_i^{\mathsf{SB}}\}_{i \in [n]}, aux)$ **if** j = r + 1 : $\Pi^{3}.\mathsf{ISign}_{1}(i, sk_{i}^{3} = (sk_{i}^{\mathsf{SB}}, \{w_{i,j}\}_{j \in [n] \setminus \{i\}}),$ Parse $ips_{r+1,i}^3 := \{\overline{ips}_{r,j,i}^{\mathsf{SB}}\}_{j \in \mathcal{S}}$ $\frac{aux,\mathsf{ctx},ups_0=(ups_0',\mathcal{S}))}{\mathsf{nonce}_i \leftarrow_{\$} \{0,1\}^{\lambda}}$ $\forall_{i \in \mathcal{S}} \ ips_{r,i}^{\mathsf{SB}} := \bigoplus_{i \in \mathcal{S}} \overline{ips}_{r,i,j}^{\mathsf{SB}}$ $ist_{1,i}^3 := (ups'_0, \mathcal{S}, sk_i^{\mathsf{SB}}, \{w_{i,j}\}_{j \in [n] \setminus \{i\}},$ return Π^1 . USign_r($ust_{r-1}, \{ips_{r,i}^{SB}\}_{i \in S}$) $aux, ctx, nonce_i)$ **return** $(ist_{1,i}^3, ips_{1,i}^3 := nonce_i)$ $\Pi^{3}.\mathsf{ISign}_{i}(i, ist_{i-1,i}^{3} = (ist_{i-2,i}^{\mathsf{SB}}, \mathsf{ssid}^{3}, \{w_{i,j}\}_{j \in [n] \setminus \{i\}}),$ $ups_{j-1}^3 = ups_{j-2}^{SB}) // j \in \{3, ..., r+1\}$ $\Pi^3.\mathsf{ISign}_2(i, ist^3_{1,i} = (ups'_0, \mathcal{S}, sk^{\mathsf{SB}}_i,$ $\overline{\mathbf{if} \ i \leq r}$: $\{w_{i,j}\}_{j\in[n]\setminus\{i\}}, aux, \mathsf{ctx},$ $(ist_{j-1,i}^{\mathsf{SB}}, ips_{j-1,i}^{\mathsf{SB}}) \leftarrow \Pi^{\mathsf{SB}}\mathsf{ISign}_{j-1}(i, ist_{j-2,i}^{\mathsf{SB}}, ups_{j-2}^{\mathsf{SB}})$ $\operatorname{nonce}_{i}^{\prime}$, $ups_{1} = \{\operatorname{nonce}_{i}\}_{i \in S}$) $ist_{j,i}^3 := (ist_{j-1,i}^{\mathsf{SB}}, \mathsf{ssid}^3, \{w_{i,j}\}_{j \in [n] \setminus \{i\}})$ $ssid^3 := \{nonce_i\}_{i \in S}$ return $(ist_{i,i}^3, ips_{i,i}^3 := ips_{i-1,i}^{SB})$ if $nonce'_i \neq nonce_i : abort$ $(ist_{1,i}^{SB}, ips_{1,i}^{SB}) \leftarrow \Pi^{SB}.ISign_1(i, sk_i^{SB})$ else : $(ips_{r,i}^{SB}, ssid^{SB}, S^{SB}) \leftarrow \Pi^{SB}.ISign_r(i, ist_{r-1,i}^{SB}, ups_{r-1}^{SB})$ $aux, \mathsf{ctx}, (ups'_0, \mathcal{S}))$ $\forall_{j\in\mathcal{S}} \ \overline{ips}^{\mathrm{SB}}_{r,j,i} := \bigoplus_{k\in\mathcal{S}\backslash\{i\}} F_{w_{i,k}}(\mathrm{ssid}^3, j, ups^{\mathrm{SB}}_{r-1}, \mathcal{S})$ $ist_{2,i}^{3} := (ist_{1,i}^{SB}, ssid^{3}, \{w_{i,j}\}_{j \in [n] \setminus \{i\}})$ **return** $(ist_{2i}^3, ips_{2i}^3) := ips_{1i}^{SB}$ $\overline{ips}_{r,i,i}^{\mathsf{SB}} := \overline{ips}_{r,i,i}^{\mathsf{SB}} \oplus ips_{r,i}^{\mathsf{SB}}, \quad ips_{r+1,i}^3 := \{\overline{ips}_{r,j,i}^{\mathsf{SB}}\}_{j \in \mathcal{S}}$ $\mathbf{return} \ (ips^3_{r+1,i},\mathsf{ssid}^3,\mathcal{S})$

Fig. 5. A generic construction of an OMUF-3 secure threshold blind signature scheme Π^3 from any OMUF-SB secure scheme Π^{SB} and PRF *F*. $\Pi^3.USign_0$ and $\Pi^3.Vf$ algorithms are identical to Π^{SB} .

trivial forgeries $\mathsf{allow}_{\mathsf{SB}}$ counts the fresh $(sid^{\mathsf{SB}}, \mathcal{S})$ values, so we must ensure that the number of fresh $(sid^{\mathsf{SB}}, \mathcal{S})$ values cannot exceed allow_3 . The intuitive way to ensure this is making OMUF-SB signing queries that will have the identical $(sid^{\mathsf{SB}}, \mathcal{S})$ for a signing session of Π^3 . We can ensure such queries to have identical session identifiers by setting $sid^{\mathsf{SB}} := \mathsf{ssid}^3$. However, we cannot control which issuer set \mathcal{S} will be output by Π^{SB} . As OMUF-SB security does not guarantee any particular property for this, the issuers can always output distinct issuer sets – even in the same signing session. This behavior would cause $\mathsf{allow}_{\mathsf{SB}}$ to increase much faster than allow_3 , and we do not have any way to argue that the \mathcal{S} outputs of a generic OMUF-SB secure scheme. Thus, we define an additional property: *issuer set consistency*. Issuer set consistency requires that the first user message in a signing session contains the intended issuer set S and the signing algorithm outputs this value as it is in the last signing round. We formally define this property below.

Definition 4 (Issuer Set Consistency). A threshold blind signature scheme Π is issuer set consistent if

- For all (pk, aux, msg, S) and for $(ups_0, \cdot) \leftarrow \Pi.\mathsf{USign}_0(pk, aux, msg, S),$ ups_0 can be parsed as $ups_0 := (ups'_0, S).$
- ISign_r outputs S as the issuer gets it through ups_0 .

Equipped with this additional property, we are now able to prove the OMUF-3 security of Construction Π^3 from Figure 5. Our proof strategy is similar to the proof of Theorem 3, OMUF-3 security of tBlindBLS-3, and we refer to Appendix I for the full proof.

Theorem 4 (OMUF-SB to OMUF-3 Boost). If F is a pseudorandom function and Π^{SB} is OMUF-SB secure and issuer-set consistent, then Π^3 in Figure 5 is OMUF-3 secure.

Correctness and Blindness. The correctness and blindness of Π^3 follow directly from the corresponding properties of the underlying Π^{SB} scheme. Blindness of Π^3 can be proven through a straightforward reduction to the blindness of Π^{SB} that echoes all the oracle queries in the blindness game until the final round. In the final round, the reduction performs the XOR operations to unwrap the Π^{SB} signature shares and then forwards the resulting signature shares to the Π^{SB} blindness challenger. The correctness of Π^3 can be shown in a similar way.

Boosting Snowblind: SB+. It is easy to see that the Snowblind scheme of Crites et al. [CKM⁺23] (Figure 4) achieves that additional property of issuer set consistency. Note that this behaviour is only needed from *honest* issuers – and trivially achieved by simply channeling the initial user input to the issuer's final output.

Lemma 11. The SB construction [CKM⁺23] (Figure 4) is issuer set consistent.

Thus, based on this observation and using the original SB construction in our Π^3 boost, which we denote as SB+, we can finally conjecture the following:

Corollary 1. If SB is OMUF-SB secure, then SB+ is OMUF-3 secure.

Note that our generic construction Π^3 does not make any assumptions on the structure of the final round signature shares $ips_{r,i}^{\text{SB}}$ and how they are used to build the final signature. Thus, we model our final round signature shares to simply have the size $t \cdot l$ bits and divide this size into l bit slots. Each issuer iperforms the XOR of the corresponding slot to it's own share $ips_{r,i}^{\text{SB}}$. This allows the user to recover $t \Pi^{\text{SB}}$ signature shares, and run Π^{SB} .USign_r in a black-box manner. More efficient, yet non-generic, variants can be built by exploiting the structure of $ips_{r,i}^{\text{SB}}$ and how Π^{SB} .USign_r uses them. For example, SB's $ips_{r,i}^{\text{SB}}$ values are \mathbb{Z}_q elements and SB.USign_r just uses the sum of elements. Thus, by simply applying the same trick as used in tBlindBLS-3, adapted to \mathbb{Z}_q , we get rid of the $t \cdot l$ increase and have the same signature share size as SB. Acknowledgments. This research was partially funded by the HPI Research School on Systems Design. It was also supported by the German Federal Ministry of Education and Research (BMBF) through funding of the ATLAS project under reference number 16KISA037.

References

- AHWY24. Ghous Ali Amjad, Scott Hendrickson, Christopher A. Wood, and Kevin W. L. Yeo, *Partially Blind RSA Signatures*, Internet-Draft draft-irtf-cfrgpartially-blind-rsa-00, Internet Engineering Task Force, September 2024, Work in Progress.
- AMMM18. Shashank Agrawal, Peihan Miao, Payman Mohassel, and Pratyay Mukherjee, Pasta: Password-based threshold authentication, Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, 2018.
- BCK⁺22. Mihir Bellare, Elizabeth Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu, Better than advertised security for non-interactive threshold signatures, Advances in Cryptology – CRYPTO 2022, 2022.
- BFG⁺13. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P Smart, and Bogdan Warinschi, Anonymous attestation with user-controlled linkability, International Journal of Information Security 12 (2013), 219–249.
- BFH⁺20. Carsten Baum, Tore Frederiksen, Julia Hesse, Anja Lehmann, and Avishay Yanai, Pesto: proactively secure distributed single sign-on, or how to trust a hacked server, 2020 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2020.
- BGHJ24. Manuel Barbosa, Kai Gellert, Julia Hesse, and Stanislaw Jarecki, Bare pake: Universally composable key exchange from just passwords, Advances in Cryptology – CRYPTO 2024, 2024.
- BLL⁺21. Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova, On the (in)security of ROS, Advances in Cryptology – EUROCRYPT 2021, 2021.
- Bol03. Alexandra Boldyreva, Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme, Public Key Cryptography — PKC 2003, 2003.
- BTZ22. Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu, Stronger security for non-interactive threshold signatures: BLS and FROST, Cryptology ePrint Archive, Paper 2022/833, 2022.
- CDVW24. Sofia Celi, Alex Davidson, Steven Valdez, and Christopher A. Wood, Privacy Pass Issuance Protocols, RFC 9578, June 2024.
- Cha83. David Chaum, Blind signatures for untraceable payments, Advances in Cryptology, 1983.
- CHL22. Sílvia Casacuberta, Julia Hesse, and Anja Lehmann, SoK: Oblivious Pseudorandom Functions, 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), 2022.
- CKM⁺23. Elizabeth Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu, Snowblind: A threshold blind signature in pairing-free groups, Advances in Cryptology – CRYPTO 2023, 2023.
- CLN15. Jan Camenisch, Anja Lehmann, and Gregory Neven, Optimal Distributed Password Verification, Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, 2015.

- clo. Privacy Pass: https://developers.cloudflare.com/waf/tools/privacy-pass, Date Accessed: 01.10.2024.
- CPZ24. Cas Cremers, Aleksi Peltonen, and Mang Zhao, An extended hierarchy of security notions for threshold signature schemes and automated analysis of protocols that use them, Cryptology ePrint Archive, Paper 2024/1920, 2024.
- DJW23. Frank Denis, Frederic Jacobs, and Christopher A. Wood, *RSA Blind Signatures*, RFC 9474, October 2023.
- DKL⁺23. Jack Doerner, Yashvanth Kondi, Eysa Lee, abhi shelat, and LaKyah Tyner, Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance, 2023, Publication info: Published elsewhere. IEEE S&P 2023.
- DR24. Sourav Das and Ling Ren, Adaptively secure bls threshold signatures from ddh and co-cdh, Advances in Cryptology CRYPTO 2024, 2024.
- FHS22. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig, *Practical* round-optimal blind signatures in the standard model, Advances in Cryptology – CRYPTO 2015, 2022.
- GJK⁺25. Yanqi Gu, Stanislaw Jarecki, Pawel Kedzior, Phillip Nazarian, and Jiayu Xu, Threshold pake with security against compromise of all servers, Advances in Cryptology – ASIACRYPT 2024, 2025.
- Han23. Lucjan Hanzlik, Non-interactive blind signatures for random messages, Advances in Cryptology EUROCRYPT 2023, 2023.
- iCl21. *iCloud Private Relay Overview: https://www.apple.com/icloud/docs/iCloud* _*Private_Relay_Overview_Dec2021.pdf, Date Accessed: 01.10.2024*, Dec 2021.
- JKKX17. Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu, TOPPSS: Cost-minimal password-protected secret sharing based on threshold OPRF, Applied Cryptology and Network Security – ACNS 2017, 2017. KM15. Veronika Kuchta and Mark Manulis, Rerandomizable Threshold Blind Siq-
- natures, Trusted Systems, 2015.
- KMMM23. Aniket Kate, Easwar Vivek Mangipudi, Siva Maradana, and Pratyay Mukherjee, *Flexirand: Output private (distributed) vrfs and application to blockchains*, Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS '23, 2023.
- KRB⁺24. Ioanna Karantaidou, Omar Renawi, Foteini Baldimtsi, Nikolaos Kamarinakis, Jonathan Katz, and Julian Loss, *Blind multisignatures for anonymous tokens with decentralized issuance*, Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS '24, 2024.
- NR04. Moni Naor and Omer Reingold, Number-theoretic constructions of efficient pseudo-random functions, J. ACM 51 (2004), no. 2, 231–262.
- RP22. Alfredo Rial and Ania M. Piotrowska, Security Analysis of Coconut, an Attribute-Based Credential Scheme with Threshold Issuance, 2022, Publication info: Preprint. MINOR revision.
- RP23. _____, Compact and Divisible E-Cash with Threshold Issuance, Proceedings on Privacy Enhancing Technologies **2023** (2023), no. 4, 381–415 (en).
- SABBD18. Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, and George Danezis, Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers, ISOC Network and Distributed System Security Symposium – NDSS, 02 2018.
- Sta96. Markus Stadler, Publicly verifiable secret sharing, Advances in Cryptology — EUROCRYPT '96, 1996.

TZ22. Stefano Tessaro and Chenzhi Zhu, Short pairing-free blind signatures with exponential security, Advances in Cryptology – EUROCRYPT 2022, 2022.
VZK03. Duc Liem Vo, Fangguo Zhang, and Kwangjo Kim, A new threshold blind signature scheme from pairings, SCIS2003, SCIS, 2003, pp. 233–238.

A Further Preliminaries

For completeness, we formally define all needed building blocks and assumptions here, that were omitted for space reasons from Section 2.

Definition 5 (Pseudorandom Function). Consider function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. Denote by $\mathbb{F}[\mathcal{X}, \mathcal{Y}]$ the set of all functions with domain \mathcal{X} and codomain \mathcal{Y} . In the pseudorandomness game, \mathcal{A} is given oracle access to either F or to a truly random function. F is pseudorandom if, for all efficient \mathcal{A} , the advantage $|\Pr_{\mathbf{k} \leftarrow_{\mathfrak{K}} \mathcal{K}}[\mathcal{A}^{F(k,\cdot)}(1^{\lambda})] - \Pr_{\mathbf{f} \leftarrow_{\mathfrak{K}} \mathbb{F}[\mathcal{X}, \mathcal{Y}]}[\mathcal{A}^{f(\cdot)}(1^{\lambda})]|$ with which \mathcal{A} distinguishes F from a random function is negligible.

Secret Sharing. We use Shamir's secret sharing several times during the paper and we present the algorithm Share below as a shortcut notation for it.

Share $(x, n, t) \rightarrow (x_1, ..., x_n)$: For $a_1, ..., a_{t-1} \leftarrow \mathbb{Z}_q$, define the polynomial $p(X) := x + \sum_{j \in [t-1]} a_j \cdot X^j$. Output (p(1), ..., p(n)).

Cyclic and Bilinear Groups. While the Snowblind schemes require only a primeorder group, the BLS schemes need a bilinear pairing as defined below. We first define their generation and then state the ommitted assumptions (DDH and BOMDH) needed in our work.

Definition 6 (Cyclic Group Generator). A group generator $GGen(\lambda)$ on input the security parameter λ outputs (\mathbb{G}, g, q) such that $\langle g \rangle = \mathbb{G}$ is a cyclic group of prime order q and $\lceil \log_2 p \rceil = \lambda$.

Definition 7 (Bilinear Pairing). For $\langle g \rangle = \mathbb{G}_1$, $\langle \hat{g} \rangle = \mathbb{G}_2$ and \mathbb{G}_T which are groups of prime order $q, e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing if it is efficiently computable and bilinear: $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab} = e(g^b, \hat{g}^a) \quad \forall a, b \in \mathbb{Z}_q$, and non-degenerate: $\langle e(g, \hat{g}) \rangle = \mathbb{G}_T$, so $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$. A bilinear group generator BGGen is a p.p.t. algorithm which outputs a bilinear pairing description $\mathcal{BG} =$ $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, q)$ such that $\lceil \log_2 q \rceil = \lambda$ the requirements above hold.

Definition 8 (Decisional Diffie-Hellman). Let \mathbb{G} be a cyclic group of prime order q with generator g. In the Decisional Diffie-Hellman (DDH) game on \mathbb{G} , \mathcal{A} is given either a random tuple or a Diffie-Hellman tuple. The DDH assumption holds on \mathbb{G} if, for all efficient \mathcal{A} , the advantage $|\operatorname{Pr}_{a,b,c\leftarrow_{\mathbb{S}}\mathbb{Z}_q}[\mathcal{A}(g,g^a,g^b,g^c) =$ $1] - \operatorname{Pr}_{a,b\leftarrow_{\mathbb{S}}\mathbb{Z}_q}[\mathcal{A}(g,g^a,g^b,g^{ab}) = 1]|$ with which \mathcal{A} distinguishes the distributions is negligible. $\begin{array}{l} \displaystyle \operatornamewithlimits{count}_t'(\boldsymbol{q}) \\ \hline C := \emptyset \quad /\!\!/ C \text{ is a multiset} \\ \mathbf{while} \ \mathcal{V}_{n,t}(\boldsymbol{q}) \neq \emptyset : \\ & \text{Find} \ S \subseteq [n] \text{ s.t. } |S| = t \text{ and } \forall i \in S, j \in [n] \setminus S : \ \boldsymbol{q}[i] \geq \boldsymbol{q}[j] \wedge (\boldsymbol{q}[i] > \boldsymbol{q}[j] \lor i < j) \\ & \text{Set} \ \boldsymbol{v} \in \{0,1\}^n \text{ s.t. } \boldsymbol{v}[i] = 1 \text{ iff } i \in S \\ & C := C \cup \{\boldsymbol{v}\} \\ & \boldsymbol{q} := \boldsymbol{q} - \boldsymbol{v} \\ & \text{return} \ |C| \end{array}$

Fig. 6. An efficient algorithm count' to compute count.

Definition 9 (Bilinear One-More Diffie-Hellman). Let $\mathcal{BG} := (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, q) \leftarrow \mathsf{BGGen}(\lambda)$. In the (N)-Bilinear One-More Diffie-Hellman (BOMDH) game, \mathcal{A} is given $(\mathcal{BG}, g^p, \hat{g}^p, \hat{g}_1, \ldots, \hat{g}_N)$ for $(\hat{g}_1, \ldots, \hat{g}_N) \leftarrow \mathbb{G}_2^N$ and access to oracle $\mathsf{BOMDH}_p(\cdot) : \mathbb{G}_2 \to \mathbb{G}_2$ for a random $p \leftarrow \mathbb{Z}_q$, where $\mathsf{BOMDH}_p(x) \to x^p$. \mathcal{A} wins if it outputs Q + 1 pairs (j, \hat{g}_j^p) for distinct $j \in [N]$, where Q is the number of queries made to $\mathsf{BOMDH}_p(\cdot)$. The BOMDH assumption holds if, for all efficient \mathcal{A} , the probability that \mathcal{A} wins the BOMDH game is negligible.

B Efficient Computation of T-(B)OMDH's count Function

The definition of the T-(B)OMDH assumption makes use of the function count, which is defined by Equation 1. Computing that function naively is not efficient, so in Figure 6 we provide an efficient algorithm count'. This algorithm works by greedily picking the t greatest elements of the input vector \mathbf{q} (with ties broken in favor of lesser indices). We prove that this algorithm is optimal, i.e. $\operatorname{count}_t(\mathbf{q}) = \operatorname{count}_t'(\mathbf{q})$ for all inputs t, \mathbf{q} .

Theorem 5. Let count_t' be defined as in Figure 6. count_t' is an efficient algorithm to compute count_t .

Proof. For some inputs n, t, and q, suppose there exists an optimal multiset of vectors $C = \{v_1, \ldots, v_c\}$ that doesn't include any vector matching our criterion (if no such C exists, we're done). In particular, the following three conditions state, respectively, that C is valid, is optimal, and does not include the vector chosen in the first iteration of the count' algorithm:

$$\begin{aligned} \boldsymbol{v}_1, \dots, \boldsymbol{v}_c \in \mathcal{V}_{n,t} & \boldsymbol{v}_1 + \dots + \boldsymbol{v}_c \leq \boldsymbol{q} \\ |C| = c = \mathsf{count}_t(\boldsymbol{q}) \\ \forall_{\boldsymbol{v} \in C} \quad \exists_{i,j \in [n]} \quad \boldsymbol{v}[i] = 1 \land \boldsymbol{v}[j] = 0 \land (\boldsymbol{q}[i] < \boldsymbol{q}[j] \lor (\boldsymbol{q}[i] = \boldsymbol{q}[j] \land i > j)) \end{aligned}$$

Pick any $v \in C$. We will repeatedly alter C into a new solution C^* (that is still valid and optimal) until v matches our criterion in C^* , i.e. v is the vector chosen in the first iteration of the count' algorithm.

- Case: $\exists_{\boldsymbol{v}'\in C} \ \boldsymbol{v}'[i] = 0 \land \boldsymbol{v}'[j] = 1$. Construct C^* by switching the choice of i in \boldsymbol{v} with the choice of j in \boldsymbol{v}' . In other words, set $\boldsymbol{v}[i] := 0$, $\boldsymbol{v}[j] := 1$, $\boldsymbol{v}'[i] := 1$, and $\boldsymbol{v}'[j] := 0$. Clearly, C^* is still valid and optimal.
- Case: $\forall_{v'\in C} v'[i] = 1 \lor v'[j] = 0$. Define C(i) to be the number of vectors in Cthat choose i, i.e. $C(i) := |\{v' \in C : v'[i] = 1\}|$. This case's condition implies that $C(j) \leq C(i)$. Since C is valid and our condition for choosing i and jrequires that $q[i] \leq q[j]$, we know overall that $C(j) \leq C(i) \leq q[i] \leq q[j]$. Construct C^* by switching all choices of i with choices of j. In other words, for each $v' \in C$ (including v), set v'[i] := v'[j] and v'[j] := v'[i]. Now, $C^*(i) = C(j)$ and $C^*(j) = C(i)$, so the inequality chain above shows that C^* is still valid. Clearly, it is still optimal.

In either case, we eliminate from \boldsymbol{v} one pair (i, j) such that $\boldsymbol{v}[i] = 1 \land \boldsymbol{v}[j] = 0 \land (\boldsymbol{q}[i] < \boldsymbol{q}[j] \lor (\boldsymbol{q}[i] = \boldsymbol{q}[j] \land i > j))$. Any other violating pairs (i, k) involving i are replaced by at most the same number of new violating pairs (j, k) involving j instead; any other violating pairs (k, j) involving j are replaced by at most the same number of new violating pairs (j, k) involving j instead; any other violating pairs (k, j) involving j are replaced by at most the same number of new violating pairs (k, i) involving i instead. Therefore, the total number of violating pairs is reduced. After repeating this process enough times, there are no more such (i, j) pairs for \boldsymbol{v} . In other words, \boldsymbol{v} is now a vector matching our criterion, and C^* is an optimal solution that includes a vector matching our criterion. Thus, there exists a valid and optimal solution that follows the first iteration of the count' algorithm. By induction over equation 1, the count' algorithm is optimal.

C Definition of Correctness and Blindness

We formally define correctness and blindness properties of threshold blind signatures below.

Definition 10 (TB Correctness). A threshold blind signature scheme TB is correct if, for all $(pk, \{sk_1, \ldots, sk_n\}, aux) \leftarrow \mathsf{TB}.\mathsf{Kg}(n, t)$, all messages msg, and all signer sets $S \subseteq [n]$ such that |S| = t, the above protocol always produces a signature σ such that $\mathsf{TB}.\mathsf{Vf}(pk, \sigma, msg) = \mathsf{true}$.

Definition 11 (TB Blindnesss). A threshold blind signature scheme TB is blind if, for all PPT adversaries \mathcal{A} , $\left|\Pr\left[\mathsf{Exp}_{\mathsf{TB},\mathcal{A}}^{\mathsf{blind}}(\lambda) = \mathsf{true}\right] - 1/2\right|$ is negligible for the experiment from Figure 7.

D Additional Proofs of Relations Between OMUF Notions

In this section we include proofs deferred from Section 4.



Fig. 7. Blindness for TB. Dashed boxes only appear for the signing round j < r and black boxes only appear when j = r.

D.1 Proof of Lemma 2 – OMUF-0 Does Not Imply OMUF-1

In this section we formally prove Lemma 2, which states that OMUF-0 security does not imply OMUF-1 security.

Proof. Suppose there exists an OMUF-1 secure scheme Π^1 . We then construct a scheme Π^0 from Π^1 that is OMUF-0 but not OMUF-1 secure as follows

- $\Pi^0.\mathsf{Pg}(1^{\lambda})$: Run $\Pi^1.\mathsf{Pg}(1^{\lambda})$ and return the result pp.
- Π^{0} .Kg(n, t): Run Π^{1} .Kg(n, t) and Π^{1} .Kg(n, t + 1). Concatenate the keys of these two Π^{1} instances and return the result $((pk, \hat{pk}), \{(sk_{i}, \hat{sk}_{i})\}_{i \in [n]}, (aux, a\hat{u}x))$. (In the special case that t = n, omit the second Π^{1} instance: $\hat{pk}, \hat{sk}_{1}, \ldots, \hat{sk}_{n}, a\hat{u}x := \bot$.)
- $\begin{array}{l} \ \varPi^0.\mathsf{USign}_0((pk,\hat{pk}),(aux,a\hat{u}x),msg,\mathcal{S}): \ \mathrm{Run} \ \mathrm{the} \ \mathrm{first} \ \varPi^1 \ \mathrm{instance} \ \mathrm{as} \ \mathrm{normal}: (ust_0^1,ups_0^1) \leftarrow \varPi^1.\mathsf{USign}_0(pk,aux,msg,\mathcal{S}). \ \mathrm{Return} \ (ust_0,ups_0^0) := (ust_0^1,(ups_0^1,\bot)). \end{array}$
- $\Pi^0.\mathsf{USign}_j(ust_{j-1}, \{ips_{j,i}\}_{i\in\mathcal{S}})$ for $j \in \{2, \ldots, r\}$: Continue to run as normal using the first Π^1 instance. On the final round, the output of USign is structured differently. (ust_j^1, ups_j^1) or $\sigma \leftarrow \Pi^1.\mathsf{USign}_j(i, ust_{j-1}, \{ups_{j,i}\}_{i\in\mathcal{S}})$. Return $(ust_j, ups_j^0) := (ust_j^1, (ups_j^1, \bot))$ or return σ .
- $\Pi^0.\mathsf{ISign}_1(i, (sk_i, \hat{sk}_i), (aux, a\hat{u}x), \mathsf{ctx}, ups_0^0):$
 - Parse ups_0^0 as (ups_0, ups_0') .

- If $ups'_0 = \bot$ (or t = n), then run the first Π^1 instance on ups_0 : $(ist_{1,i}, i)$ $ips_{1,i}$ $\leftarrow \Pi^1$. $\mathsf{ISign}_1(i, sk_i, aux, \mathsf{ctx}, ups_0)$ and $(ist'_{1,i}, ips'_{1,i}) := (\bot, \bot)$.
- If $ups'_0 \neq \bot$ (and t < n), then run a "double execution" of the second Π^1 instance on both ups_0 and ups'_0 : $(ist_{1,i}, ips_{1,i}) \leftarrow \Pi^1.\mathsf{ISign}_1(i, \hat{sk}_i, a\hat{ux}, ips_{1,i})$ ctx, ups_0 and $(ist'_{1,i}, ips'_{1,i}) \leftarrow \Pi^1.\mathsf{lSign}_1(i, \hat{sk}_i, a\hat{u}x, \mathsf{ctx}, ups'_0).$

• Return $(ist_1^0, ips_{1,i}^0) := ((ist_{1,i}, ist'_{1,i}), (ips_{1,i}, ips'_{1,i})).$ - $\Pi^0.\mathsf{ISign}_j(i, ist_{j-1,i}^0, ups_{j-1}^0)$ for $j \in \{2, \ldots, r\}$: Continue to run either a single execution of the first Π^1 instance or a double execution of the second Π^1 instance. On the final round, the output of ISign is structured differently.

- Parse $ist_{j-1,i}^0$ as $(ist_{j-1,i}, ist'_{j-1,i})$ and ups_{j-1}^0 as (ups_{j-1}, ups'_{j-1}) .
- $(ist_{j,i}, ips_{j,i})$ or $(ips_{r,i}, \mathsf{ssid}, \mathcal{S}) \leftarrow \Pi^1.\mathsf{lSign}_j(i, ist_{j-1,i}, ups_{j-1}).$ If $ist'_{j-1,i} = \bot$ (or t = n), then $(ist'_{j,i}, ips'_{j,i}) := (\bot, \bot)$ or $(ips'_{r,i}, \mathsf{ssid}', \mathcal{S}')$ $:= (\breve{\bot}, \breve{\bot}, \breve{\bot}).$
- If $ist'_{j-1,i} \neq \bot$ (and t < n), then $(ist'_{j,i}, ips'_{j,i})$ or $(ips'_{r,i}, ssid', S') \leftarrow$ $\Pi^1.\mathsf{lSign}_j(i, ist'_{j-1,i}, ups'_{j-1}).$
- Return $(ist_{j}^{0}, ips_{j,i}^{0}) := ((ist_{j,i}, ist'_{j,i}), (ips_{j,i}, ips'_{j,i}))$ or $(ips_{r,i}^{0}, \mathsf{ssid}^{0}, \mathcal{S}^{0})$ $:= ((ips_{r,i}, ips'_{r,i}), \bot, \bot).$
- $-\Pi^0$.Vf $((pk, \hat{pk}), \sigma, msg)$: Signatures from either Π^1 instance are valid. Return $\Pi^{1}.\mathsf{Vf}(pk,\sigma,msq) \lor (t < n \land \Pi^{1}.\mathsf{Vf}(pk,\sigma,msq)).$

Per the construction of the Π^0 . USign algorithms, honest user execution never takes advantage of the double execution behavior; it simply runs on the first Π^1 instance completely as normal. Therefore, the correctness and blindness of Π^0 are immediately implied from Π^1 .

 Π^0 is not OMUF-1 secure. Consider the OMUF-1 game with the parameters (n, t) such that $2 \le t \le n$ and an adversary \mathcal{A} who corrupts no issuers. \mathcal{A} can perform a double execution by interacting with t + 1 issuers. A obtains 2 valid signatures, but $\mathsf{allow}_1 = 1$.

 Π^0 is OMUF-0 secure. Consider any adversary \mathcal{A} . Partition all of \mathcal{A} 's ISign interactions into the set of those which run a single execution under pk and the set of those which run a double execution under pk. Similarly partition the signatures output by \mathcal{A} . If \mathcal{A} wins OMUF-0, then one (or both) of these sets of signatures must be larger than the corresponding set of issuer interactions. In either case, there is a reduction from the security of Π^1 :

- Suppose \mathcal{A} performs q pk-mode Π^0 . |Sign interactions and then outputs more than q pk-mode signatures. Our reduction performs q Π^1 . |Sign interactions and then outputs more than q signatures. This reduction clearly breaks the OMUF-1 (and even OMUF-0) security of Π^1 .
- Suppose \mathcal{A} performs $q \ \hat{pk}$ -mode Π^0 . ISign interactions and then outputs more _ than q p k-mode signatures. Our reduction performs $2q \Pi^1$. (Sign interactions and then outputs more than q signatures. Because \mathcal{A} corrupts at most t – 1 issuers but the Π^1 threshold is set to t + 1, our OMUF-1 reduction is allowed at most 1 signature per every 2 interactions with uncorrputed issuers: allow₁ $\leq q$. Therefore, this reduction breaks the OMUF-1 security of Π^1 . \Box

D.2 Proof of Lemma 6 – OMUF-3 Implies OMUF-SB

In this section we formally prove Lemma 6, which states that OMUF-3 security implies OMUF-SB security.

Proof. Any winning adversary against the OMUF-SB game is also a winning adversary against the OMUF-3 game. To prove this, we show that $\mathsf{allow}_3 \leq \mathsf{allow}_{SB}$:

$$\mathsf{allow}_3 = \left| \left\{ \mathsf{ssid} : \left(\exists \mathcal{S}_3 \in \mathcal{P}([n] \setminus \mathcal{C}) : \mathcal{S}_3 \neq \emptyset, \mathcal{S}_3 \subseteq \mathsf{El}_3[\mathsf{ssid}, \mathcal{S}_3] \right) \right\} \right|$$
(1)

$$\leq \sum_{\substack{\mathcal{S}_3 \in \mathcal{P}([n] \setminus \mathcal{C}) \\ \mathcal{S}_3 \neq \emptyset}} \left| \left\{ \mathsf{ssid} : \mathcal{S}_3 \subseteq \mathsf{El}_3[\mathsf{ssid}, \mathcal{S}_3] \right\} \right| \tag{2}$$

$$\leq \sum_{\substack{\mathcal{S}_{3} \in \mathcal{P}([n] \setminus \mathcal{C}) \\ \mathcal{S}_{3} \neq \emptyset}} \sum_{\substack{\mathcal{S} \in \mathcal{P}([n]) \\ \mathcal{S} \supseteq \mathcal{S}_{3}, \mathcal{S} \setminus \mathcal{S}_{3} \subseteq \mathcal{C}}} \left| \left\{ sid : (sid, \mathcal{S}) \in \mathsf{ES}_{\mathsf{SB}} \right\} \right|$$
(3)

$$= \sum_{\mathcal{S}\in\mathcal{P}([n])} \left| \left\{ sid : (sid, \mathcal{S}) \in \mathsf{ES}_{\mathsf{SB}} \right\} \right|$$
(4)

$$= \text{allow}_{\text{SB}}$$
 (5)

On line 1, we rewrite the definition of allow_3 into a single expression. Between lines 1 and 2, observe that every ssid counted in the former expression will be counted in the latter one or more times. Between lines 2 and 3, consider any one issuer $i \in S_3$. For each different ssid such that $i \in \mathsf{El}_3[\mathsf{ssid}, S_3]$, issuer i must run on a different sid and output some supposed signer set S such that $S \supseteq S_3$ and $S \setminus S_3 \subseteq C$. Between lines 3 and 4, observe that every valid S corresponds to exactly one S_3 .

E Unforgeability Relations in the Special Cases t = 1 and t = n

In Section 4, we describe the implication relations between the six one-more unforgeability relations defined in Section 3.2. Our security definitions allow for any parameters t, n such that $1 \le t \le n$. Nonetheless, it is also interesting to consider the restricted edge cases where t = 1 or t = n.

If we restrict the parameters to the cases $t \in \{1, n\}$, several of the results in Section 4 are affected. In particular, the proofs of Lemmas 2, 3, 7, and 8 are inapplicable, and those relations must be reconsidered.

Lemma 12. If $t \in \{1, n\}$, OMUF-0 security implies OMUF-1 security.

Proof. For t = 1, this implication is immediate from the security game definitions. In particular, $\mathsf{allow}_0 = \mathsf{allow}_1$. Therefore, any winning adversary against the OMUF-1 game is also a winning adversary against the OMUF-0 game.

For t = n, suppose there exists a winning OMUF-1 adversary \mathcal{A}_1 . Construct reduction \mathcal{R}_0 which picks a single issuer $i \leftarrow_{\$} [n]$ at random and corrupts $[n] \setminus \{i\}$. If i is the issuer queried by \mathcal{A}_1 the least, then $\mathsf{allow}_0 = \mathsf{allow}_1$ and \mathcal{R}_0 wins the OMUF-0 game. If not, then $\mathsf{allow}_0 > \mathsf{allow}_1$ and the reduction fails. Therefore, this reduction is tight up to a factor of n.

Lemma 13. If $t \in \{1, n\}$, OMUF-2 security implies OMUF-3 security.

Proof. This implication is immediate from the security game definitions. If t = 1, there is only one supposed signing set $S = \{i\}$ that any issuer *i* can output. If t = n, there is only one supposed signing set S = [n] that any issuer can output. In either case, allow₂ = allow₃. Therefore, any winning adversary against the OMUF-3 game is also a winning adversary against the OMUF-2 game.

Lemma 14. If t = 1, OMUF-0 security and OMUF-SB security imply one another.

Proof. These implications are immediate from the security game definitions. In particular, when t = 1, there is only one supposed signing set $S = \{i\}$ that any issuer *i* can output. Issuer *i* must use a different *sid* for each of its interactions. Therefore, allow₀ = allow_{SB}. Any winning adversary against the OMUF-0 or OMUF-SB game is also a winning adversary against the other.

Lemma 15. If t = n, OMUF-1 security and OMUF-SB security imply one another.

Proof. To show that OMUF-1 security implies OMUF-SB security, observe that when t = n allow₁ is the minimum of all uncorrupted issuers' interaction counts. That minimal server must use a different *sid* for each of its interactions, so allow_{SB} must be at least as large. allow₁ \leq allow_{SB}. Thus, any winning OMUF-SB adversary is also a winning OMUF-1 adversary.

To show that OMUF-SB security implies OMUF-1 security, suppose there exists a winning OMUF-1 adversary \mathcal{A}_1 . Construct reduction \mathcal{R}_{SB} which picks a single issuer $i \leftarrow_{\$} [n]$ at random and corrupts $[n] \setminus \{i\}$. If i is the issuer queried by \mathcal{A}_1 the least, then $\mathsf{allow}_{SB} = \mathsf{allow}_1$ and \mathcal{R}_{SB} wins the OMUF-SB game. If not, then $\mathsf{allow}_{SB} > \mathsf{allow}_1$ and the reduction fails. Therefore, this reduction is tight up to a factor of n.

To summarize the cases $t \in \{1, n\}$: OMUF-0, OMUF-FC, OMUF-1, and OMUF-SB all imply one another, and OMUF-2 and OMUF-3 imply one another. By Lemma 1 the latter imply the former, and by Lemma 3 the former do not imply the latter.

Intuitively, it is not surprising that the one-more unforgeability relations simplify in this way. When t = 1 or t = n, signing set consistency is essentially a vacuous property. Therefore, in these cases the notions are separated only by whether or not they require ssid consistency. (The equivalence of OMUF-0 and OMUF-1 when t = n is not as obvious.)

F Proofs of Threshold Blind BLS Signatures

We present the proofs that were deferred in Section 5 here. Throughout the one-more unforgeaiblity proofs, we denote the Lagrange basis polynomials for a set S and $i \in S$ as $\Lambda_i^S(X) := \prod_{j \in S, j \neq i} \frac{X-j}{i-j}$. For simplicity, we denote $\Lambda_i^S(0)$ by dropping the input, Λ_i^S .

F.1 Proof of tBlindBLS Blindness

Observe that the three $\mathsf{tBlindBLS}$ variants differ only in their Kg and ISign_1 algorithms, which do not appear in the blindness game at all (Definition 11). Therefore, the blindness of all three variants can be analyzed at once $(\mathsf{USign}_1$ also differs in its use of interpolation coefficients, but this is not relevant to the proof of blindness).

Theorem 6 (tBlindBLS Blindness). For all ppt adversaries \mathcal{A} and $i \in \{1, 2, 3\}$, $\left|\Pr\left[\mathsf{Exp}_{\mathsf{tBlindBLS-}i,\mathcal{A}}^{\mathsf{blind}}(\lambda) = \mathsf{true}\right] - 1/2 \right| = q_H/q$ for the experiment from Figure 7.

Proof. The blindness game $\mathsf{Exp}_{\mathsf{tBlindBLS},\mathcal{A}}^{\mathsf{blind}}$ can be perfectly simulated without knowledge of the secret bit *b* unless the adversary finds a message msg such that $H(msg) = 1_{\mathbb{G}_2}$. We take *b* to be 0 and run the code of tBlindBLS earnestly.

Regardless of its inputs (and of b), the first oracle $\mathcal{O}^{\text{USign}_0}$ returns two uniformly random group elements unless $H(msg_b) = 1_{\mathbb{G}_2}$ for one of the challenge messages. We abort if the adversary finds such a message msg which only occurs with probability q_H/q in ROM for q_H being the number of H queries made by the adversary. When this case does not occur, $H(msg_b)^{r_b}$ is uniformly random for $b \in \{0, 1\}$, so $\mathcal{O}^{\text{USign}_0}$ oracle does not leak information about b. The second oracle $\mathcal{O}^{\text{USign}_1}$ is nearly as simple: it either returns two properly verified signatures (σ_0, σ_1) for messages (msg_0, msg_1) , or it returns (\bot, \bot) . Since BLS signatures are deterministic, there is only one possible (σ_0, σ_1) for any given (msg_0, msg_1) and pk. Thus, the view of \mathcal{A} in this simulated game is exactly identical to that in the real game unless we abort. \mathcal{A} can either provide honest inputs to $\mathcal{O}^{\text{USign}_1}$ and receive (σ_0, σ_1) or provide dishonest inputs and receive (\bot, \bot) .

Since the view of \mathcal{A} can be perfectly simulated when the game does not abort, it follows that $\left| \Pr \left[\mathsf{Exp}_{\mathsf{tBlind}\mathsf{BLS-}i,\mathcal{A}}^{\mathsf{blind}}(\lambda) = \mathsf{true} \right] - 1/2 \right| = q_H/q.$

F.2 Proof of Theorem 1 – OMUF-1 Security of tBlindBLS-1

Proof. Suppose there exists an efficient adversary \mathcal{A} such that the adversary's advantage $\Pr[\mathsf{Exp}_{\mathsf{tBlindBLS-1},\mathcal{A},n,t}^{\mathsf{OMUF-1}}(\lambda) = \mathsf{true}]$ is non-negligible. We construct the following reduction $\mathcal{R}(\mathcal{A})$, which wins $(t - |\mathcal{C}|, n, N)$ -T-OMDH with non-negligible probability:

Parameter Generation: Choose a random $c \leftarrow \{0, \ldots, t-1\}$ and receive $(\mathcal{BG}, g^{p(0)}, \hat{g}^{p(0)}, \hat{g}_1, \ldots, \hat{g}_N)$ from T-BOMDH challenger for the parameters (t - c, n, N). Set $pp := \mathcal{BG}$ and send (pp) to the adversary.

- **Key Generation:** Receive C from \mathcal{A} . If $|\mathcal{C}| \neq c$ which was chosen during the parameter generation, then abort. Otherwise, pick a random (t-1)-degree polynomial r over \mathbb{Z}_q . Define s, a (t-1)-degree polynomial over \mathbb{Z}_q , as $s(i) := r(i) + p(i) \cdot \prod_{i' \in \mathcal{C}} (i-i')$. At indexes in \mathcal{C} , s(i) = r(i), which is known to $\mathcal{R}(\mathcal{A})$; at other indexes, s(i) is unknown to $\mathcal{R}(\mathcal{A})$. Send $(g^{p(0)}, \{s(i)\}_{i \in \mathcal{C}})$ to \mathcal{A} as $(pk, \{sk_i\}_{i \in \mathcal{C}})$.
- Queries to $\mathcal{O}^H(msg_j)$: For the *j*'th fresh random oracle query msg_j , set $H(msg_j)$:= \hat{g}_j . Return $H(msg_j)$ accordingly.
- Signing Queries ($\mathcal{O}^{\mathsf{lSign}}(j=1, sid, i, \mathsf{ctx}, (a, S))$): Upon an oracle query for an uncorrupted issuer *i*, use the $a'_i := \mathsf{TBOMDH}_p(i, a)$ to compute $a^{s(i)} = a^{r(i)} \cdot (a'_i) \prod_{i' \in \mathcal{C}} (i-i')$. For corrupted issuers, $a^{s(i)}$ can be computed directly. Return response $b_i := a^{s(i)}$.

Ultimately, if $\mathcal{R}(\mathcal{A})$ does not abort during the key generation, it receives ℓ message-signature pairs $\{(msg_k^*, \sigma_k^*)\}_{k \in [\ell]}$ from \mathcal{A} . If they all verify and $\ell > \mathsf{count}_{t-|\mathcal{C}|}(q_1, \ldots, q_n)$, then $\mathcal{R}(\mathcal{A})$ can win T-BOMDH. Assume (without loss of generality) that every msg^* was queried to H, and (for all $k \in [\ell]$) define $j(k) \in [N]$ to be the index such that $msg_{j(k)} = msg_k^*$. For all $k \in [\ell]$, compute $(\hat{g}_{j(k)})^{p(0)} = (\sigma_k^* \cdot (\hat{g}_{j(k)})^{r(0)})^{1/\prod_{i' \in \mathcal{C}}(-i')}$. Output pairs $\{(j(k), (\hat{g}_{j(k)})^{p(0)})\}_{k \in [\ell]}$. $\mathcal{R}(\mathcal{A})$ does not abort with the probability 1/t. Thus, we finally conclude that

$$\Pr \Big[\mathsf{Exp}^{\mathsf{OMUF-1}}_{\mathsf{tBlindBLS-1},\mathcal{A},n,t}(\lambda) = \mathsf{true} \Big] \leq t \cdot \epsilon_{\mathsf{T}\text{-}\mathsf{BOMDH}}$$

F.3 Proof of Theorem 2 – OMUF-2 Security of tBlindBLS-2

Proof. As our proof sketch pointed out, we first aim to change binding factors $Z_i(\operatorname{ctx}, a) := \tilde{H}(\operatorname{ctx}, a)^{z_i}$ to the truly random values. We do this by a hybrid argument as follows. We define a series of variations $\mathcal{G}^0_{\mathcal{A}}, \ldots, \mathcal{G}^{t-1}_{\mathcal{A}}$. $\mathcal{G}^0_{\mathcal{A}}$ has the identical output to the $\operatorname{Exp}_{\operatorname{HindBLS-2,\mathcal{A},n,t}(\lambda)$, but changes how binding factors are computed. In particular, the binding factors of all issuers are computed by interpolating them from a specific set of issuers D so that the later games can directly aim changing the binding factors of the issuers in D. The following games change the binding factors of the issuers in D one by one by relying on the DDH assumption.

Replacing Binding Factors with Random Values. We denote the binding factor of the issuer *i* on ctx and *a* as $Z_i(\text{ctx}, a) := \tilde{H}(\text{ctx}, a)$. We define $\mathcal{G}^0_{\mathcal{A}}$ as a game that differs from $\mathsf{Exp}^{\mathsf{OMUF-2}}_{\mathsf{tBlindBLS-2},\mathcal{A},n,t}$ in the following ways:

- After receiving C from A, select an arbitrary set of issuers $D \subseteq [n]$ such that $C \subseteq D$ and |D| = t 1.
- For all $i \notin D$, compute $Z_i(\mathsf{ctx}, a)$ by interpolation from the binding factors of the issuers in set D. Specifically, $Z_i(\mathsf{ctx}, a) := \prod_{i' \in D} (Z_{i'}(\mathsf{ctx}, a))^{\Lambda_{i'}^{D'}(i)}$, where $D' := D \cup \{0\}$.

The views of \mathcal{A} in $\mathsf{Exp}^{\mathsf{OMUF-2}}_{\mathsf{tBlindBLS-2},\mathcal{A},n,t}$ and $\mathcal{G}^0_{\mathcal{A}}$ are identical. Therefore,

$$\Pr\left[\mathcal{G}^{0}_{\mathcal{A}}(\lambda) = \mathsf{true}\right] = \Pr\left[\mathsf{Exp}^{\mathsf{OMUF-2}}_{\mathsf{tBlindBLS-2},\mathcal{A},n,t}(\lambda) = \mathsf{true}\right]$$

Next, for all $t' \in \{1, \ldots, t-1\}$, we define $\mathcal{G}_{\mathcal{A}}^{t'}$ as a game that differs from $\mathcal{G}_{\mathcal{A}}^{0}$ in the following way:

- For the first $(\min\{t', t-1-|\mathcal{C}|\})$ issuers *i* in set $D \setminus \mathcal{C}$, set all binding factors at random. Specifically, $Z_i(\mathsf{ctx}, a) \leftarrow_{\$} \mathbb{G}_2$.

If $t' > t - 1 - |\mathcal{C}|$, then $\mathcal{G}_{\mathcal{A}}^{t'-1}$ and $\mathcal{G}_{\mathcal{A}}^{t'}$ are identical. If $t' \leq t - 1 - |\mathcal{C}|$, then a DDH reduction proves that \mathcal{A} 's probabilities of winning in $\mathcal{G}_{\mathcal{A}}^{t'-1}$ and $\mathcal{G}_{\mathcal{A}}^{t'}$ can only differ negligibly (for all $t' \in \{1, \ldots, t - 1\}$). Using such a distinguisher we build a DDH reduction as a variant of $\mathcal{G}_{\mathcal{A}}^{t'-1}$ that differs in the following ways:

- Receive DDH challenge tuple $(g_1, g'_1, g_2, g'_2) \in \mathbb{G}_2^4$.
- Label the random oracle queries to \tilde{H} as $(\mathsf{ctx}_1, a_1), (\mathsf{ctx}_2, a_2), \ldots$. For all j, pick $\delta_{1,j}, \delta_{2,j} \leftarrow_{\$} \mathbb{Z}_q$ and program $\tilde{H}(\mathsf{ctx}_j, a_j) := (g_1)^{\delta_{1,j}} \cdot (g_2)^{\delta_{2,j}}$.
- Denote by i' the t'-th issuer in set $D \setminus C$. Set $Z_{i'}(\mathsf{ctx}_j, a_j) := (g'_1)^{\delta_{1,j}} \cdot (g'_2)^{\delta_{2,j}}$. (Without loss of generality, we assume that \mathcal{A} queries \tilde{H} on every (ctx, a) sent to $\mathcal{O}^{\mathsf{lSign}}$.)

This reduction uses the well-known "DDH randomized self-reduction" technique [Sta96, NR04] to generate many pairs of correlated group elements from the original challenge tuple. If that original tuple is a Diffie-Hellman tuple, then all pairs $(\tilde{H}(\mathsf{ctx}, a), Z_{i'}(\mathsf{ctx}, a))$ are related by the same secret exponent (effectively, $z_{i'}$). In that case, the game is identical to $\mathcal{G}_{\mathcal{A}}^{t'-1}$. If the original tuple is a random tuple, then all the blinding factors $Z_{i'}(\mathsf{ctx}, a)$ used by issuer i' are independently random. In that case, the game is identical to $\mathcal{G}_{\mathcal{A}}^{t'-1}$. Therefore, the advantage ϵ_{DDH} of this distinguisher must be negligible. For all $t' \in \{1, \ldots, t-1\}$,

$$\Pr\left[\mathcal{G}_{\mathcal{A}}^{t'-1}(\lambda) = \mathsf{true}\right] \le \Pr\left[\mathcal{G}_{\mathcal{A}}^{t'}(\lambda) = \mathsf{true}\right] + \epsilon_{\mathsf{DDH}}$$

Overall,

$$\Pr\big[\mathcal{G}^0_{\mathcal{A}}(\lambda) = \mathsf{true}\big] \leq \Pr\big[\mathcal{G}^{t-1}_{\mathcal{A}}(\lambda) = \mathsf{true}\big] + (t-1) \cdot \epsilon_{\mathsf{DDH}}$$

Reduction from BOMDH Assumption. Finally, we construct reduction $\mathcal{R}(\mathcal{A})$, which wins (N)-BOMDH with non-negligible probability. We begin by introducing some key notations that will be used throughout this proof to improve the readability. We introduce the following notation for the readability: $S_i(\mathsf{ctx}, a)$ denotes the signature share of the issuer *i* on ctx and *a*. In other words, $S_i(\mathsf{ctx}, a) = \mathcal{O}^{\mathsf{lSign}}(j = 1, sid, i, \mathsf{ctx}, (a, S))$.

Parameter Generation: Receive $(\mathcal{BG}, g^{p(0)}, \hat{g}^{p(0)}, \hat{g}_1, \dots, \hat{g}_N)$ from BOMDH challenger and set $pp := \mathcal{BG}$.

- **Key Generation:** Receive C from A. For all $i \in C$, pick $k_i, z_i \leftarrow_{\$} \mathbb{Z}_q$. Send $(g^p, \{(k_i, z_i)\}_{i \in C})$ to A as $(pk, \{sk_i\}_{i \in C})$.
- **Random Oracle Queries** ($\mathcal{O}^H(msg_j)$): For the j'th fresh random oracle query msg_j , set $H(msg_j) := \hat{g}_j$. Return $H(msg_j)$ accordingly.

Random Oracle Queries $(\mathcal{O}^{\tilde{H}}(x))$: Respond honestly.

- Signing Queries ($\mathcal{O}^{\mathsf{lSign}}(j = 1, sid, i, \mathsf{ctx}, (a, S))$): Answer signing queries as follows.
 - For corrupted issuers $i \in \mathcal{C}$, $S_i(\mathsf{ctx}, a)$ can be computed directly. Specifically, $S_i(\mathsf{ctx}, a) := a^{k_i} \cdot \tilde{H}(\mathsf{ctx}, a)^{z_i}$.
 - For the first $(t 1 |\mathcal{C}|)$ uncorrupted issues $i \notin \mathcal{C}$ queried with some (ctx, a) , respond randomly. Specifically, $S_i(\mathsf{ctx}, a) \leftarrow \mathbb{G}_2$.
 - For the $(t |\mathcal{C}|)$ -th and later uncorrupted issuers $i \notin \mathcal{C}$ queried with some (ctx, a) , use the BOMDH oracle to learn $a^p = \mathsf{BOMDH}_p(a)$. Interpolate response $S_i(\mathsf{ctx}, a) := (a^p)^{A_0^{E'}} \cdot \prod_{i' \in \mathcal{C} \cup E} (S_{i'}(\mathsf{ctx}, a))^{A_{i'}^{E'}(i)}$, where E is the set of indices of the first $(t 1 |\mathcal{C}|)$ uncorrupted issuers to be queried with (ctx, a) and $E' := \mathcal{C} \cup E \cup \{0\}$.

The view of \mathcal{A} in $\mathcal{R}(\mathcal{A})$ is identical to its view in $\mathcal{G}_{\mathcal{A}}^{t-1}$. Ultimately, $\mathcal{R}(\mathcal{A})$ receives ℓ message-signature pairs $\{(msg_k^*, \sigma_k^*)\}_{k \in [\ell]}$ from \mathcal{A} . If they all verify and ℓ is greater than or equal to the number of (ctx, a) inputs that were queried on $t - |\mathcal{C}|$ or more uncorrupted issuers, then $\mathcal{R}(\mathcal{A})$ can win BOMDH. Assume (without loss of generality) that every msg^* was queried to H, and (for all $k \in [\ell]$) define $j(k) \in [N]$ to be the index such that $msg_{j(k)} = msg_k^*$. Then $\mathcal{R}(\mathcal{A})$ outputs pairs $\{(j(k), \sigma_k^*)\}_{k \in [\ell]}$.

We conclude that

$$\Pr\left[\mathsf{Exp}_{\mathsf{tBlindBLS-2},\mathcal{A},n,t}^{\mathsf{OMUF-2}}(\lambda) = \mathsf{true}\right] \le \epsilon_{\mathsf{BOMDH}} + (t-1) \cdot \epsilon_{\mathsf{DDH}}$$

F.4 Proof of Theorem 3 – OMUF-3 Security of tBlindBLS-3

Proof. Similar to the proof of Theorem 2, this proof will contain two steps: replacing binding factors with random values and then having a reduction from BOMDH assumption. For this scheme, we rely on the pseudorandomness property of F while replacing the binding factors.

Replacing Binding Factors with Random Values. We define $\mathcal{G}_{\mathcal{A}}$ as a game that differs from $\mathsf{Exp}^{\mathsf{OMUF-3}}_{\mathsf{tBlindBLS-3},\mathcal{A},n,t}$ in the following way:

- For each pair of uncorrupted servers $i, j \in [n] \setminus C$, do not pick a pairwise key $w_{i,j} = w_{j,i}$. Instead, sample a random function $f_{i,j} : \{0,1\}^* \to \mathbb{G}$. In $\mathcal{O}^{\mathsf{lSign}}$ queries to issuers *i* and *j*, use $f_{i,j}(\cdot)$ in place of $F_{w_{i,j}}(\cdot)$.

Game $\mathcal{G}_{\mathcal{A}}$ is related to $\mathsf{Exp}_{\mathsf{tBlindBLS-3},\mathcal{A},n,t}^{\mathsf{OMUF-3}}$ by a series of intermediate hybrids, each of which successively replaces one more PRF instance with a random function.

There are $(n - |\mathcal{C}|)(n - |\mathcal{C}| - 1)/2 < n^2/2$ PRFs to be replaced. Each hybrid can be used to construct a distinguisher for the pseudorandomness game by using the provided oracle in the place of the function in question. Therefore, \mathcal{A} 's probabilities of winning in the $\mathsf{Exp}_{\mathsf{tBlindBLS-3},\mathcal{A},n,t}$ and $\mathcal{G}_{\mathcal{A}}$ games can only differ negligibly. In particular,

$$\Pr\left[\mathsf{Exp}_{\mathsf{tBlindBLS-3},\mathcal{A},n,t}^{\mathsf{OMUF-3}}(\lambda) = \mathsf{true}\right] \leq \Pr[\mathcal{G}_{\mathcal{A}}(\lambda) = \mathsf{true}] + (n^2/2) \cdot \epsilon_{\mathsf{PRF}}$$

where ϵ_{PRF} is the maximum advantage of the PRF distinguishers.

Reduction from BOMDH Assumption. Next, we construct reduction $\mathcal{R}(\mathcal{A})$, which wins (N)-OMDH with non-negligible probability.

We introduce the following notation for the readability: $S_i(\mathsf{ctx}, a, \mathcal{S})$ denotes the signature share of the issuer *i* on ctx , *a*, and \mathcal{S} . In other words, $S_i(\mathsf{ctx}, a, \mathcal{S}) = \mathcal{O}^{\mathsf{lSign}}(j = 1, sid, i, \mathsf{ctx}, (a, \mathcal{S}))$.

- **Parameter Generation:** Receive $(\mathcal{BG}, g^{p(0)}, \hat{g}^{p(0)}, \hat{g}_1, \dots, \hat{g}_N)$ from BOMDH challenger and set $pp := \mathcal{BG}$.
- **Key Generation:** Receive C from \mathcal{A} . For all $i \in C$, pick $k_i \leftarrow_{\$} \mathbb{Z}_q$. For all $i, j \in [n]$ such that i < j and $i \in C$ or $j \in C$, pick $w_{i,j} \leftarrow_{\$} \mathcal{K}$ and define $w_{j,i} := w_{i,j}$. Send $(g^p, \{(k_i, \{w_{i,j}\}_{j \in [n] \setminus \{i\}})\}_{i \in C})$ to \mathcal{A} as $(pk, \{sk_i\}_{i \in C})$.
- **Random Oracle Queries** $(\mathcal{O}^H(msg_j))$: For the *j*'th fresh random oracle query msg_j , set $H(msg_j) := \hat{g}_j$. Return $H(msg_j)$ accordingly.
- Signing Queries ($\mathcal{O}^{\mathsf{lSign}}(j = 1, sid, i, \mathsf{ctx}, (a, S))$): Answer signing queries as follows.
 - For corrupted issuers $i \in \mathcal{C}$, $S_i(\mathsf{ctx}, a, \mathcal{S})$ can be computed directly.
 - For an uncorrupted issuer $i \notin C$, if at least one other uncorrupted issuer in S has not yet been queried with (ctx, a, S) , then respond randomly. Specifically, $S_i(\mathsf{ctx}, a, S) \leftarrow_{\$} \mathbb{G}_2$.
 - For an uncorrupted issuer $i \notin C$, if all other uncorrupted issuers in S have already been queried with (ctx, a, S) , then use the BOMDH oracle to learn $a^p = \mathsf{BOMDH}_p(a)$. Compute response $S_i(\mathsf{ctx}, a, S) := a^p \cdot \prod_{i' \in S \setminus \{i\}} S_{i'}(\mathsf{ctx}, a, S)^{-1}$.

The view of \mathcal{A} in $\mathcal{R}(\mathcal{A})$ is identical to its view in $\mathcal{G}_{\mathcal{A}}$. Ultimately, $\mathcal{R}(\mathcal{A})$ receives ℓ message-signature pairs $\{(msg_k^*, \sigma_k^*)\}_{k \in [\ell]}$ from \mathcal{A} . If they all verify and ℓ is greater than the number of $(\mathsf{ctx}, a, \mathcal{S})$ inputs that were queried on all uncorrupted issuers in \mathcal{S} , then $\mathcal{R}(\mathcal{A})$ can win BOMDH. Assume (without loss of generality) that every msg^* was queried to H, and (for all $k \in [\ell]$) define $j(k) \in [N]$ to be the index such that $msg_{j(k)} = msg_k^*$. Output pairs $\{(j(k), \sigma_k^*)\}_{k \in [\ell]}$. Reduction $\mathcal{R}(\mathcal{A})$ wins the BOMDH game with probability equal to that of \mathcal{A} winning the \mathcal{G} game. Thus, we conclude that,

$$\Pr\left[\mathsf{Exp}_{\mathsf{tBlindBLS-3},\mathcal{A},n,t}^{\mathsf{OMUF-3}}(\lambda) = \mathsf{true}\right] \le \epsilon_{\mathsf{BOMDH}} + (n^2/2) \cdot \epsilon_{\mathsf{PRF}}$$

G Snowblind Construction

For simplicity, we presented the Snowblind construction in Section 6 in its original form, not fully adapted to our syntax. As our SB+ construction relies on a Π_{SB} scheme according to our definition, we also restate SB in our notation:

Construction 1 (Snowblind Threshold Blind Signature (SB)) Let DS := (Pg, Kg, Sign, Vf) be a signature scheme. SB is defined as follows.

- Pg(1^λ): Sample a group description (G, g, q) ← GGen(λ), a generator h ← G, and and public parameters for the signature pp_{sig} ← DS.Pg(λ). Select hash functions H_{cm}, H_{sig} : {0,1}* → Z_q. Return $pp := ((G, q, g, h), H_{cm}, H_{sig}, pp_{sig})$.
- $\mathsf{Kg}(n,t)$: Sample the secret key $sk \leftarrow \mathbb{Z}_q$ and generate secret key shares $(sk'_1, ..., sk'_n) \leftarrow \mathsf{Share}(sk, n, t)$. Sample signature key pairs $(\hat{sk}_j, \hat{pk}_j) \leftarrow \mathsf{DS.Kg}()$ for $j \in [n]$. Set the public key $pk := g^{sk}$ and $aux := \{\hat{pk}_j\}_{j \in [n]}$. Return $(pk, \{(sk'_j, \hat{sk}_j)\}_{j \in [n]}, aux)$.

USign₀(pk, aux, msg, S): Return ($ust_0 := (pk, msg)$, $ups_0 := S$).

ISign₁(*i*, *sk_i*, *aux*, ctx, *ups*₀): Parse *ups*₀ := S. If *i* ∉ S, then abort. Otherwise, Sample $a_i, b_i \leftarrow \mathbb{Z}_q$ and $y_i \leftarrow \mathbb{Z}_q^*$. Set $A_i := g^{a_i}, B_i := g^{b_i}h^{y_i}$, and cm_{*i*} := H_{cm}(sid, *i*, *y_i*). Return (*ist*_{1,*i*} := (*sk_i*, *aux*, S, *a_i*, *b_i*, *y_i*), *ips*_{1,*i*} := (*A_i*, *B_i*, cm_{*i*})).

- $\begin{aligned} \mathsf{USign}_1(ust_0, \{ips_{1,j}\}_{j\in\mathcal{S}}) \colon \text{Parse } ust_0 &:= (pk, msg) \text{ and } ips_{1,j} := (A_j, B_j, \mathsf{cm}_j) \\ \text{for } j \in \mathcal{S}. \text{ Sample } \alpha, \beta \leftarrow \mathbb{Z}_q^* \text{ and } r \leftarrow \mathbb{Z}_q. \text{ Set } A &:= \prod_{j\in\mathcal{S}} A_j, B := \\ \prod_{j\in\mathcal{S}} B_j, \ \bar{R} &:= g^r A^{\alpha/\beta} B^{\alpha}, \ \bar{c} &:= \mathsf{H}_{sig}(pk, msg, \bar{R}), \text{ and } c := \bar{c} \cdot \beta. \text{ Return} \\ (ust_1 := (pk, msg, r, c, \alpha, \beta), ups_1 := (c, \{\mathsf{cm}_j\}_{j\in\mathcal{S}})). \end{aligned}$
- $\begin{aligned} \mathsf{ISign}_2(i, ist_{1,i}, ups_1): & \mathsf{Parse}\ ist_{1,i} := (sk_i, aux, \mathcal{S}, a_i, b_i, y_i), \ sk_i := (sk'_i, sk_i) \ \text{and} \\ & ups_1 := (c, \{\mathsf{cm}_j\}_{j \in \mathcal{S}}). \ \mathsf{Set}\ sesst := (sid, \mathcal{S}, c, \{\mathsf{cm}_j\}_{j \in \mathcal{S}}) \ \mathsf{and}\ \mathsf{compute}\ ds_i \leftarrow \\ & \mathsf{DS.Sign}(\hat{sk}_i, sesst). \ \mathsf{Return}\ (ist_{2,i} := (sk'_i, aux, \mathcal{S}, a_i, sesst, \{\mathsf{cm}_j\}_{j \in \mathcal{S}}), ips_{2,i} \\ & := (b_i, y_i, ds_i)). \end{aligned}$
- $\begin{array}{l} \mathsf{USign}_2(ust_1, \{ips_{2,j}\}_{j\in\mathcal{S}}) \text{: Parse } ust_1 := (pk, msg, r, c, \alpha, \beta) \text{ and } ips_{2,j} := (b_j, y_j, ds_j) \text{ for } j \in \mathcal{S}. \text{ Return } (ust_2 := (pk, msg, r, c, \alpha, \beta, \{(b_j, y_j)\}_{j\in\mathcal{S}}), ups_2 := \{(ds_j, y_j)\}_{j\in\mathcal{S}}). \end{array}$
- $$\begin{split} \mathsf{ISign}_3(i, ist_{2,i}, ups_2) &: \mathsf{Parse}\ ist_{2,i} := (sk'_i, aux, \mathcal{S}, a_i, sesst, \{\mathsf{cm}_j\}_{j \in \mathcal{S}}),\ aux := \{pk_j\}_{j \in [n]} \ \text{and}\ ups_2 := \{(ds_j, y_j)\}_{j \in \mathcal{S}} \ \text{for}\ j \in \mathcal{S}. \ \text{Check if there is a}\ j \in \mathcal{S} \ \text{such} \ \text{that}\ \mathsf{cm}_j \neq \mathsf{H}_{cm}(sid, j, y_j) \ \text{or}\ \mathsf{DS}.\mathsf{Vf}(pk_j, sesst, ds_j) = \mathsf{false}. \ \text{Abort if there is} \ \text{such}\ j. \ \text{Otherwise, set}\ \mathcal{T} := \sum_{j \in \mathcal{S}} y_j \ \text{and}\ z_i := a_i + c \cdot \mathcal{T} \cdot \Lambda_i^{\mathcal{S}} \cdot sk'_i. \ \text{Return} \ (\mathcal{S}, \mathsf{ssid} := \bot, ips_{3,i} := z_i). \end{split}$$
- $\begin{aligned} \mathsf{USign}_3(ust_2, \{ips_{3,j}\}_{j\in\mathcal{S}}) \colon \text{Parse } ust_2 &:= (pk, msg, r, c, \alpha, \beta, \{(b_j, y_j)\}_{j\in\mathcal{S}}) \text{ and} \\ ips_{3,j} &:= z_j \text{ for } j \in \mathcal{S}. \text{ Set } \Upsilon := \sum_{j\in\mathcal{S}} y_j, z := \sum_{j\in\mathcal{S}} z_j, \text{ and } b := \sum_{j\in\mathcal{S}} b_j. \text{ If} \\ B \neq g^b h^y \text{ or } g^z \neq Apk^{c\cdot y}, \text{ then return } \bot. \text{ Otherwise, set } \bar{z} := r + \alpha \cdot z/\beta + \alpha \cdot b, \\ \bar{y} &:= \alpha \cdot y, \text{ and } \sigma := (\bar{R}, \bar{z}, \bar{y}). \text{ If } \mathsf{Vf}(pk, msg, \sigma) \text{ holds, then return } \sigma. \text{ Otherwise, return } \bot. \end{aligned}$

 $Vf(pk, msg, \sigma)$: Parse $\sigma := (\bar{R}, \bar{z}, \bar{y})$ and set $\bar{c} := H_{sig}(pk, msg, \bar{R})$. Return

$$\bar{y} \neq 0 \land \bar{R} \cdot pk^{\bar{c} \cdot \bar{y}} = g^{\bar{z}}h^{\bar{y}}$$

H Attack Against OMUF-3 Security of SB

We build an efficient adversary \mathcal{A} against the OMUF-3 security of Π^{SB} for the parameters (n := 3, t := 2) as follows. \mathcal{A} gets pp and calls OMUF-3 game for $\mathcal{C} := \{3\}$. The main strategy of \mathcal{A} is as follows. \mathcal{A} runs a signing session between the issuers 1 and 2. \mathcal{A} aims to use the issuer 1's outputs for the first two signing round to create a valid third round signing request ups for the issuer 3 for this session. Then \mathcal{A} uses the issuer 3's final signature share and sk_3 to forge a signature. As there is only one 3rd round signing query in this scenario, which corresponds to the set of issuers $\{1, 2\}$, allow₃ = 0, so \mathcal{A} can win by outputting 1 valid signature.

Now we describe \mathcal{A} 's behavior in detail.

- \mathcal{A} makes a signing query to both issuer 1 and 2 for an arbitrary *sid* with the oracle calls:

$$(A_1, B_1, \mathsf{cm}_1) \leftarrow \mathcal{O}^{\mathsf{ISign}_1}(sid, 1, ups_0) := (\mathcal{S} := \{1, 2\}))$$
$$(A_2, B_2, \mathsf{cm}_2) \leftarrow \mathcal{O}^{\mathsf{ISign}_1}(sid, 2, ups_0) := (\mathcal{S} := \{1, 2\}))$$

- \mathcal{A} chooses and arbitrary message msg^* and sets $ups_1 := (c, \{\mathsf{cm}_1, \mathsf{cm}_2\})$ for $c := \mathsf{H}_{sig}(pk, \bar{R}, msg^*), \ \bar{R} := A_2^{\Lambda_2^{S'}/\Lambda_2^S} \cdot B_1 \cdot B_2$, and $\mathcal{S}' := \{2,3\}$. \mathcal{A} does not include A_1 in \bar{R} so that it does not need the 3rd round signature share of the issuer 1. \mathcal{A} includes both B_1 and B_2 in \bar{R} because the issuer 2 will output its 3rd round signature share for $\Upsilon = y_1 + y_2$. Finally, we arrange A_2 with the exponents to be able to forge a 3rd round signature share of the issuer 2 for the issuer set \mathcal{S}' . \mathcal{A} makes the second round queries,

$$(b_1, y_1, ds_1) \leftarrow \mathcal{O}^{\mathsf{lSign}_2}(sid, 1, ups_1) \quad (b_2, y_2, ds_2) \leftarrow \mathcal{O}^{\mathsf{lSign}_2}(sid, 2, ups_1)$$

 $-\mathcal{A}$ simply makes the final round signing query for the issuer 3 as

$$z_2 \leftarrow \mathcal{O}^{\mathsf{ISign}_3}(sid, 2, ups_2 := \{(ds_j, y_j)\}_{j \in \{1, 2\}})$$

- Observe that $A_2 \cdot pk_j^{\Lambda_2^{\mathcal{S}} \cdot f(c,\Upsilon)} = g^{z_2}$, so $A_2^{\Lambda_2^{\mathcal{S}'}/\Lambda_2^{\mathcal{S}}} \cdot pk_j^{\Lambda_2^{\mathcal{S}'} \cdot f(c,\Upsilon)} = g^{z'_2}$ for $z'_2 := z_2 \cdot \Lambda_2^{\mathcal{S}'}/\Lambda_2^{\mathcal{S}}$.
- \mathcal{A} computes the signature share for sk_3 as $z_3 := c \cdot \Lambda_3^{S'} \cdot sk_3 \Upsilon$ and sets the final forgery $\sigma^* := (\bar{R}, \bar{y}, \bar{z})$ for

$$\bar{y} := \Upsilon \qquad \qquad \bar{z} := z_2' + z_3 + b_1 + b_2$$

and \mathcal{A} outputs $\ell := 1, \{(\sigma^*, msg^*)\}.$

It is easy to observe that \mathcal{A} computes valid signing queries to get a valid signature share of the issuer 2. Furthremore, \mathcal{A} forgery verifies since

$$\begin{split} g^{\bar{z}} \cdot h^{\bar{y}} &= g^{\bar{z}} \cdot h^{\Upsilon} \\ &= g^{\bar{z}} \cdot h^{\Upsilon} \\ &= A_2^{\Lambda_2^{S'}/\Lambda_2^S} \cdot pk_j^{\Lambda_2^{S'} \cdot f(c,\Upsilon)} \cdot g^{z_3 + b_1 + b_2} \cdot h^{\Upsilon} \\ &= A_2^{\Lambda_2^{S'}/\Lambda_2^S} \cdot B_1 \cdot B_2 \cdot pk_j^{\Lambda_2^{S'} \cdot f(c,\Upsilon)} \cdot g^{z_3} \\ &= \bar{R} \cdot pk_j^{\Lambda_2^{S'} \cdot f(c,\Upsilon)} \cdot g^{z_3} \\ &= \bar{R} \cdot pk_j^{\Lambda_2^{S'} \cdot f(c,\Upsilon)} \cdot pk_3^{\Lambda_3^{S'} \cdot f(c,\Upsilon)} \\ &= \bar{R} \cdot pk_j^{f(c,\Upsilon)} \end{split}$$

I Proof of Theorem 4 – **OMUF-3** Security of Π^3

Proof. Following the proof strategy of Theorem 3, we first have a step to replace binding factors with random values. We rely on the pseudorandomness of F for this step. Next we have a reduction from the properties of Π^{SB} .

Replacing Binding Factors with Random Values. We define $\mathcal{G}_{\mathcal{A}}$ as a game that differs from $\mathsf{Exp}_{\Pi^3,\mathcal{A},n,t}^{\mathsf{OMUF-3}}$ in the following way:

- For each pair of uncorrupted servers $i, j \in [n] \setminus C$, do not pick a pairwise key $w_{i,j} = w_{j,i}$. Instead, sample a random function $f_{i,j} : \{0,1\}^* \to \{0,1\}^l$. In $\mathcal{O}^{\mathsf{lSign}_{r+1}^3}$ queries to issuers i and j, use $f_{i,j}(\cdot)$ in place of $F_{w_{i,j}}(\cdot)$.

Game $\mathcal{G}_{\mathcal{A}}$ is related to $\mathsf{Exp}_{\Pi^3,\mathcal{A},n,t}^{\mathsf{OMUF}-3}$ by a series of intermediate hybrids, each of which successively replaces one more PRF instance with a random function. There are $(n - |\mathcal{C}|)(n - |\mathcal{C}| - 1)/2 < n^2/2$ PRFs to be replaced. Each hybrid can be used to construct a distinguisher for the pseudorandomness game by using the provided oracle in the place of the function in question. Therefore, \mathcal{A} 's probabilities of winning in the $\mathsf{Exp}_{\Pi^3,\mathcal{A},n,t}^{\mathsf{OMUF}-3}$ and $\mathcal{G}_{\mathcal{A}}$ games can only differ negligibly. In particular,

$$\Pr\Big[\mathsf{Exp}^{\mathsf{OMUF-3}}_{\varPi^3,\mathcal{A},n,t}(\lambda) = \mathsf{true}\Big] \leq \Pr[\mathcal{G}_{\mathcal{A}}(\lambda) = \mathsf{true}] + (n^2/2) \cdot \epsilon_{\mathsf{PRF}}$$

where ϵ_{PRF} is the maximum advantage of the PRF distinguishers.

Reduction From Π^{SB} . Next, we construct reduction $\mathcal{R}(\mathcal{A})$, which wins the OMUF-SB game against Π^{SB} with non-negligible probability:

Key Generation: Receive C from A and forward it to $\mathsf{Exp}_{\Pi^{\mathsf{SB}},n,t}^{\mathsf{OMUF}-\mathsf{SB}}$. Receive $(pk, \{sk_i^{\mathsf{SB}}\}_{i \in \mathcal{C}}, aux)$ from $\mathsf{Exp}_{\Pi^{\mathsf{SB}},n,t}^{\mathsf{OMUF}-\mathsf{SB}}$. For all $i, j \in [n]$ such that i < j and

 $i \in \mathcal{C}$ or $j \in \mathcal{C}$, pick $w_{i,j} \leftarrow \mathcal{K}$ and define $w_{j,i} := w_{i,j}$. For all $i \in \mathcal{C}$, define $sk_i^3 := (sk_i^{\text{SB}}, \{w_{i,j}\}_{j \in [n] \setminus \{i\}})$. Send $(pk, \{sk_i^3\}_{i \in \mathcal{C}}, aux)$ to \mathcal{A} . Set a table Twhich initially is assigned to \emptyset in all positions.

- **1st Round Signing Queries:** Sample nonce_i $\leftarrow \{0,1\}^{\lambda}$. If the issuer *i* used nonce_i in any previous $\mathcal{O}_3^{|\text{Sign}_1}$ queries, set **BadNonce** and abort. 2nd Round Signing Queries: Check if nonce'_i = nonce_i as the original al-
- gorithm and abort if not. If there is no abort, set ssid := $\{\text{nonce}_j\}_{j \in S}$, and instead of running the Π^{SB} . ISign₁ algorithm, make a $\mathcal{O}_{SB}^{|Sign_1}$ query for $sid^{SB} := ssid^3$ and by using the (ups'_0, S) from the issuer state $ist^3_{1,i}$. *j*'th Round Signing Queries $(j \in \{3, ..., r\})$: For each round, parse the in-
- puts as the original algorithm and make the corresponding $\mathcal{O}_{\mathsf{SB}}^{\mathsf{lSign}_{j-1}}$ query for $sid^{\mathsf{SB}} := \mathsf{ssid}^3$ that was fixed in the second round. Set the outpus as the orginal algorithm and output.
- r+1'st Round Signing Queries $\mathcal{O}_3^{\mathsf{lSign}}(j=r+1, sid, i, ist_{r,i}^3, ups_r^3)$: Parse ups_r^3 $:= ups_{r-1}^{\mathsf{SB}} \text{ and } ist_{r,i}^3 := (ist_{r-1,i}^{\mathsf{SB}}, \mathsf{ssid}^3, \{w_{i,j}\}_{j \in [n] \setminus \{i\}}).$ - For corrupted issuers $i \in \mathcal{C}$, the response can be computed directly.

 - For an uncorrupted issuer $i \notin \mathcal{C}$, lookup $Q := T[ssid^3, \mathcal{S}, ups_r^3]$.
 - If $|Q| < |S \setminus C| 1$, then respond randomly by sampling $t_{j,i} \leftarrow \{0,1\}^l$ and setting

$$\overline{ips}_{r,j,i}^{\mathsf{SB}} := t_{j,i} \bigoplus_{k \in \mathcal{S} \cap \mathcal{C}} F_{w_{i,k}}(\mathsf{ssid}^3, j, ups_{r-1}^{\mathsf{SB}}, \mathcal{S})$$

for $j \in \mathcal{S}$ and $ips_{r+1,i}^3 := \{\overline{ips_{r,j,i}}\}_{j \in \mathcal{S}}$. Also update $T[ssid^3, \mathcal{S}, ups_r^3] :=$ $\begin{aligned} & Q \cup \{ips_{r+1,i}^3\}. \\ & - \text{ If } |Q| = |\mathcal{S} \setminus \mathcal{C}| - 1 \text{, then make a} \end{aligned}$

$$(ips_{r,j}^{\mathsf{SB}},\mathsf{ssid}^{\mathsf{SB}},\mathcal{S}^{\mathsf{SB}}) \leftarrow \mathcal{O}_{\mathsf{SB}}^{\mathsf{lSign}_r}(sid^{\mathsf{SB}} := \mathsf{ssid}^3, j, ups_{r-1}^{\mathsf{SB}})$$

query for $j \in \mathcal{S} \setminus \mathcal{C}$. Parse $ips_{r+1,k} \in Q$ as $ips_{r+1,k} := \{\overline{ips_{r,j,k}}\}_{j \in \mathcal{S}}$. Then for $j \in \mathcal{S} \setminus \mathcal{C}$, set

$$\overline{ips}_{r,j,i}^{\mathsf{SB}} := ips_{r,j}^{\mathsf{SB}} \bigoplus_{k \in \mathcal{S} \setminus (\mathcal{C} \cup \{i\})} \overline{ips}_{r,j,k}^{\mathsf{SB}} \bigoplus_{k \in \mathcal{S} \cap \mathcal{C}} F_{w_{i,k}}(\mathsf{ssid}^3, j, ups_{r-1}^{\mathsf{SB}}, \mathcal{S})$$

Finally, set $ips_{r+1,i}^3 := \{\overline{ips}_{r,j,i}^{\mathsf{SB}}\}_{j \in \mathcal{S}}$, and output $(ips_{r+1}^3, \mathsf{ssid}^3, \mathcal{S})$.

Ultimately, receive $\ell + 1$ message-signature pairs $\{(msg_k^*, \sigma_k^*)\}_{k \in [\ell+1]}$ from \mathcal{A} and output them identically to $\mathsf{Exp}_{\Pi^{\mathsf{SB}},n,t}^{\mathsf{OMUF}\mathsf{-SB}}$

We first argue that the simulation of $\mathcal{R}(\mathcal{A})$ is indistinguishable from $\mathcal{G}_{\mathcal{A}}$. It is straightforward to observe that they are indistinguishable for key generation, and signing queries $\mathcal{O}_3^{|\text{Sign}_j|}$ for $j \in \{2, ..., r\}$. For the first signing round, $\mathcal{R}(\mathcal{A})$'s simulation is indistinguishable from $\mathcal{G}_{\mathcal{A}}$ as long as $\mathcal{R}(\mathcal{A})$ does not abort due to a **BadNonce** event which could only occur by the probability $q_{S_1}^2/2^{\lambda}$ where q_{S_1} is the number of first round signing queries.

For the last signing round, we argue that they are indistinguiashable as well. $\mathcal{G}_{\mathcal{A}}$ sets the functions $F_{w_{i,j}}$ for $i, j \notin \mathcal{C}$ as truly random functions. $\mathcal{R}(\mathcal{A})$ simulates this behavior by simply setting a random value $t_{i,j}$ which corresponds to the contributions of $F_{w_{i,j}}$ functions and $ips_{r,i}^{SB}$ when $|\mathcal{Q}| < |\mathcal{S} \setminus \mathcal{C}| - 1$. The contribution of $F_{w_{i,j}}$'s for a corrupted j is simulated truly. Given that **BadNonce** does not occur, this behavior sets the truly random functions of $\mathcal{G}_{\mathcal{A}}$ by lazy sampling which is perfectly indistinguishable. Finally, when $|\mathcal{Q}| = |\mathcal{S} \setminus \mathcal{C}| - 1$, we simply set the contribution of $F_{w_{i,j}}$ for $i, j \notin \mathcal{C}$ consistently to the previously set random $t_{i,j}$ values. Thus, the simulation of $\mathcal{R}(\mathcal{A})$ is indistinguishable from $\mathcal{G}_{\mathcal{A}}$.

random $t_{i,j}$ values. Thus, the simulation of $\mathcal{R}(\mathcal{A})$ is indistinguishable from $\mathcal{G}_{\mathcal{A}}$. Next we argue that if \mathcal{A} wins $\operatorname{Exp}_{\Pi^3}^{OMUF-3}$, then $\mathcal{R}(\mathcal{A})$ wins $\operatorname{Exp}_{\Pi^{SB}}^{OMUF-SB}$. For that, we need to show that allow₃ >= allow_{SB}. Observe that allow₃ only gets increased when all honest issuers in an issuer set outputs the same ssid. Following that, $\mathcal{R}(\mathcal{A})$ does not make any final round signing query to $\operatorname{Exp}_{\Pi^{SB}}^{OMUF-SB}$ until \mathcal{A} queries the $\mathcal{R}(\mathcal{A})$ for all honest issuers in the issuer set. When $\mathcal{R}(\mathcal{A})$ gets queries for all honest issuers in the issuer set, it makes $|\mathcal{S} \setminus \mathcal{C}|$ queries to $\operatorname{Exp}_{\Pi^{SB}}^{OMUF-SB}$ at once. As $\mathcal{R}(\mathcal{A})$ sets the session identifier of all these queries to $sid^{SB} := ssid^3$ they all run in the same session of $\operatorname{Exp}_{\Pi^{SB}}^{OMUF-SB}$. Furthermore, by the issuer-set consistency property of Π^{SB} , they all output the identical issuer set \mathcal{S}^{SB} . Thus, these $|\mathcal{S} \setminus \mathcal{C}|$ queries to $\operatorname{Exp}_{\Pi^{SB}}^{OMUF-SB}$ increases allow_{SB} exactly by 1. This shows that allow₃ = allow_{SB} and $\mathcal{R}(\mathcal{A})$ wins when \mathcal{A} wins and **BadNonce** event does not occur. Thus, we conclude that

$$\Pr[\mathcal{G}_{\mathcal{A}}(\lambda) = \mathsf{true}] \leq \Pr\Big[\mathsf{Exp}_{\Pi^{\mathsf{SB}}, \mathcal{R}(\mathcal{A}), n, t}^{\mathsf{OMUF-SB}}(\lambda) = \mathsf{true}\Big] + \Pr[\mathbf{BadNonce}]$$

and

$$\Pr\left[\mathsf{Exp}_{\Pi^{3},\mathcal{A},n,t}^{\mathsf{OMUF-3}}(\lambda) = \mathsf{true}\right] \leq \Pr\left[\mathsf{Exp}_{\Pi^{\mathsf{SB}},\mathcal{R}(\mathcal{A}),n,t}^{\mathsf{OMUF-SB}}(\lambda) = \mathsf{true}\right] + (n^{2}/2) \cdot \epsilon_{\mathsf{PRF}} + q_{S_{1}}^{2}/2^{\lambda}$$