A preliminary version of this paper appears as part of the Crypto 2022 paper *Better than Advertised Security for Non-Interactive Threshold Signatures* by Bellare, Crites, Komlo, Maller, Tessaro and Zhu.

Stronger Security for Non-Interactive Threshold Signatures: BLS and FROST

Mihir Bellare[1] Stefano Tessaro[2] Chenzhi Zhu[3]

June 2022

Abstract

We give a unified syntax, and a hierarchy of definitions of security of increasing strength, for non-interactive threshold signature schemes. They cover both fully non-interactive schemes (these are ones that have a single-round signing protocol, the canonical example being threshold-BLS) and ones, like FROST, that have a prior round of message-independent pre-processing. The definitions in the upper echelon of our hierarchy ask for security that is well beyond any currently defined, let alone proven to be met by the just-mentioned schemes, yet natural, and important for modern applications like securing digital wallets. We prove that BLS and FROST are better than advertised, meeting some of these stronger definitions. Yet, they fall short of meeting our strongest definition, a gap we fill for FROST via a simple enhancement to the scheme. We also surface subtle differences in the security achieved by variants of FROST.

[1] Department of Computer Science & Engineering, University of California, San Diego. Email: `mihir@eng.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grant CNS-2154272 and a gift from Microsoft.

[2] Department of Computer Science, University of Washington. Email: `tessaro@cs.washington.edu`. URL: `https://homes.cs.washington.edu/~tessaro/`. Supported in part by NSF grants CNS-1930117 (CAREER), CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

[3] Department of Computer Science, University of Washington. Email: `zhucz20@cs.washington.edu`. URL: `https://homes.cs.washington.edu/~zhucz20/`. Supported in part by grants of second author.

Contents

1 Introduction

Threshold signatures, which originated in the late 1980s [18, 19], are seeing renewed attention, driven in particular by an interest in using them to secure digital wallets in the cryptocurrencies ecosystem [22]. Parallel IETF [31] and NIST [35] standardization efforts are evidence as to the speed at which the area is moving into practice.

Whether securing a user's digital wallet, or being used by a CA to create a certificate, forgery of a digital signature is costly. The rising tide of system breaches and phishing attacks makes exposure of a signing key too likely to ignore. The idea of a threshold signature scheme is to distribute the secret signing key across multiple parties who then interact to produce a signature, the intent being to retain security even in the face of compromise of up to a threshold number of these parties. Over the years, threshold versions of many schemes have been presented, including RSA [17, 26, 38], DSA/ECDSA [23, 25, 22, 10, 21, 33, 14], Schnorr signatures [39, 24, 30] and BLS signatures [9].

Today, we see interest converging on schemes that are non-interactive. The representative examples are (threshold) BLS [13, 9] and FROST [30]. BLS is a pairing-based scheme that is fully non-interactive scheme, meaning signing consists of a single round. FROST is non-pairing-based scheme that is partially non-interactive scheme, meaning signing additionally involves a message-independent pre-processing round.

Our path. Focusing on non-interactive threshold signatures, this paper has a simple and positive message. We contend that schemes like BLS and FROST are *better than advertised.* We show that they meet definitions of security that are *stronger* than ones that have been previously defined, or that these schemes have been shown to meet in existing literature. Furthermore, these definitions capture natural strengths of the schemes that may be valuable for applications. But also, our new fine-grained viewpoint will surface differences between schemes. In particular, we show that a recently proposed optimization of FROST [16] is less secure than the original proposal.

The classical development paradigm in theoretical cryptography is to ask what security we would like, define it, and then seek schemes that meet it. Yet if we look back, there has been another path alongside; canonical, reference schemes guided a choice of definitions that model them, and, once made, these definitions went on to be influential targets for future schemes. (The formal definition of trapdoor permutations [27], for example, was crafted to model RSA.) We are inspired by the latter path. BLS [12] yields a threshold scheme [9] so natural and simple that it is hard to not see it as canonical, and, within the space of Schnorr threshold schemes, FROST [30] has a similarly appealing minimality. Examining them, we see strengths not captured by current definitions or results. We step back to create corresponding abstractions, including a syntax and a hierarchy of definitions of security for non-interactive threshold signature schemes. We then return to ask where, in this hierarchy, we can fit the starting schemes, giving proofs that fit BLS and FROST as high as possible. In terms of the proofs this needs, and that we give, this turns out to be challenging, so that we offer also some content of technical interest.

Although inspired by specific schemes, our definitional development, once started, unfolds in a logical way, and yields definitions that even go beyond what BLS and FROST achieve. These make intriguing new targets. We show how to achieve them, with minimal modifications to the existing schemes.

Non-interactive threshold schemes. We consider schemes where the signing operations involve a leader and a set of ns nodes, which we refer to as servers, with server i holding a secret share sk_i of the secret signing key sk. Signing is done via an interactive protocol that begins with a leader request to some set of at least t number of servers and culminates with the leader holding the signature, where $t \leq \mathsf{ns}$, the threshold, is a protocol parameter.

In a *fully non-interactive* threshold signature scheme, this protocol is a simple, one-round one. The leader sends a leader request lr, which specifies a message M and possibly other things, to any server i, and obtains in response a *partial signature*, $psig_i$, that i computes as a function of sk_i and M. The leader can request partial signatures asynchronously, at any time, and independently for each server, and there is no server-to-server communication. Once it has enough partial signatures, the leader aggregates them into a signature sig of M under the verification key vk corresponding to sk. The canonical example is the threshold BLS scheme [9, 13], where $sk, sk_1, \ldots, sk_{\mathsf{ns}} \in \mathbb{Z}_p$ for a public prime p, and $psig_i \leftarrow \mathsf{h}(M)^{sk_i}$ where $\mathsf{h} : \{0,1\}^* \to \mathbb{G}$ is a public hash function with range a group $\mathbb{G}$ of order p. Aggregation produces sig as a weighted product of the partial signatures.

A *partially non-interactive* threshold signature scheme adds to the above a message-independent pre-processing round in which, pinged by the leader at any point, a server i returns a pre-processing token pp_i. The leader's request for a partial signatures will now depend on tokens it has received. The canonical example is FROST [30].

This understanding of a non-interactive scheme encompasses what FROST calls flexibility; obtaining $psig_i$ from *any* $\geq t$ servers will allow us to reconstruct a signature.

<u>Which forgeries are non-trivial?</u> For a regular (non-threshold) signature scheme, the first and most basic notion of security is un-forgeability (UF) [27]. The adversary (given access to a signing oracle) outputs a forgery consisting of a message M and a valid signature for it. To win, the forgery must be *non-trivial*, meaning not legitimately obtained. This is naturally captured, in this context, as meaning that M was not a signing query.

Turning to define un-forgeability for a non-interactive threshold signature scheme, we assume the adversary has corrupted the leader, and up to $t-1$ servers, where $1 \leq t \leq \mathsf{ns}$ is the threshold. Furthermore it has access to the honest servers. Again, it outputs a forgery consisting of a message M and valid signature for it, and, to win, the forgery must be *non-trivial*, meaning not legitimately obtained. Deciding what "non-trivial" means, however, is now a good deal more delicate, and interesting, than it was for regular signatures.

In this regard, we suggest that many of the prior works have set a low bar, being more generous than necessary in declaring a forgery trivial, leading to definitions that are weaker than one can desire, and weaker even than what their own schemes seem to meet. The definitions we formulate rectify this by giving a hierarchy of five non-triviality conditions of increasing stringency, yielding a corresponding hierarchy TS-UF-0 $\leftarrow$ TS-UF-1 $\leftarrow$ TS-UF-2 $\leftarrow$ TS-UF-3 $\leftarrow$ TS-UF-4 of five notions of un-forgeability of increasing strength. (Here an arrow B $\leftarrow$ A means A implies B: any scheme that is A-secure is also B-secure.) TS-UF-0, the lowest in the hierarchy, is the notion currently in the literature.

Returning to regular (non-threshold) signature schemes, strong un-forgeability (SUF) has the same template as UF, but makes the non-triviality condition more stringent, asking that there have been no signing query M that returned sig. We ask if SUF has any analogue in the threshold setting. For non-interactive schemes, we suggest it does and give a hierarchy of three definitions of strong unforgeability TS-SUF-2 $\leftarrow$ TS-SUF-3 $\leftarrow$ TS-SUF-4. The numbering reflects that TS-UF-i $\leftarrow$ TS-SUF-i for $i = 2, 3, 4$.

<u>The case of BLS.</u> Returning to threshold BLS, Boldyreva's analysis [9] adopts the formalism of Gennaro, Jarecki, Krawczyk, and Rabin [26, 23, 25]. The non-triviality condition here is that *no* server was asked to issue a partial signature on the forgery message M. This is TS-UF-0 in our hierarchy. But allowing asynchronous requests is a feature of this scheme and model. A corrupted leader could ask one honest server i for a partial signature. No other server would even be aware of this request, but the adversary would now have $psig_i$. Under TS-UF-0, the forgery is now trivial, and the adversary does not win. Yet (assuming a threshold $t \geq 2$), there is no reason possession

of just $psig_i$ should allow creation of a signature, and indeed for threshold BLS there is no attack that seems able to create such a signature, indicating the scheme is achieving more than TS-UF-0. This leads to the next level of our hierarchy, TS-UF-1, where the non-triviality condition is that a partial signature of M was requested from at most $t-1-c$ honest servers, where c is the number of corrupted servers. Does threshold BLS achieve this TS-UF-1 definition? As we will see, proving this presents challenges, but we will succeed in showing that the answer is yes, under a variant of the CDH assumption which we introduce (and prove to be hard in the generic group model). Yet, TS-UF-1 was not considered in the literature, and only TS-UF-0 is proved for many other non-interactive schemes [38, 29, 40, 11]. The only exceptions are the work of Libert, Joye, and Yung [32] and recent concurrent work by Groth [28], which comes to a similar conclusion/result on BLS. (We discuss the relation below.) We note that Shoup [38] implicitly tackles a similar technical challenge by dealing with differing corruption and reconstruction thresholds, but the resulting security notion is not TS-UF-1.

The distinction between TS-UF-1 and TS-UF-0 is not just academic. Implicit in applications of threshold signing in wallets is the fact that servers also perform well-formedness checks of what is being signed (typically, as part of a transaction). TS-UF-1 guarantees that every issued signature has been inspected by sufficiently many servers, but TS-UF-0 does not.

THE CASE OF FROST. Yet the hierarchy needs to go higher, and this becomes apparent when looking at partially non-interactive schemes like FROST [30]. Here, the discussion becomes more subtle, and interesting.

In more detail, a FROST pre-processing token takes the form of a pair $pp_i = (g^{r_i}, g^{s_i})$ of group elements for one-time use. (A server will ensure that the pre-processing token in its name in the leader request is one it has previously sent, and will never use it again.) An honest request lr includes, along with the message M to be signed, a sufficiently large server set $lr.\mathsf{SS} \subseteq [1..\mathsf{ns}]$, and, for each i in this set, a pre-processing token pp_i that i previously sent. Each server $i \in lr.\mathsf{SS}$ will then generate a signature share $psig_i = (R, z_i)$, where R is a value which can be computed (publicly) from the tokens included in lr, whereas z_i depends on the discrete logarithms of the server's token and its own key share sk_i. The z_i's can then be aggregated into a value z such that (R, z) is a valid Schnorr signature for M. Two variants of FROST are known, and differ in the way in which R is computed from lr. The original version, which we refer to as FROST1 [30], requires $|lr.\mathsf{SS}|$ exponentiations, whereas a more recent optimization, which we call FROST2 [16], only requires a single exponentiation. The current security analysis [16] group model [20] does not surface any security difference between the two variants, but only because it considers a non-triviality notion as in our TS-UF-0 notion.

In terms of our framework, we show that $\mathsf{FROST1}$ achieves TS-SUF-3 security. The proof is under the OMDL (One More Discrete Log) assumption of [5] in the RO (Random Oracle) model of [7]. This considers a signature trivial even if some of the honest servers in $lr.\mathsf{SS}$ do not respond to a (malicious) leader request, as long as the tokens associated with these servers are not honestly generated. In particular, the honest servers may not respond because they recognize these tokens as invalid, or because the malicious leader did not submit the request to them. We show that, while $\mathsf{FROST2}$ fails to achieve TS-SUF-3, it achieves the next step down in our hierarchy, TS-SUF-2. (Again the proof is under OMDL in the ROM.) This is still stronger than the notions lower in the hierarchy. Our proofs for $\mathsf{FROST1}$ and $\mathsf{FROST2}$ signing operations rely on the one-more discrete logarithm (OMDL) assumption and the random oracle model.

STRONGER GOALS. A stronger security goal (TS-UF-4 in our hierarchy) is to expect that the *only* way to obtain a signature for a message M is to follow the above blueprint, i.e., to issue the same honest leader request lr to all servers in $lr.\mathsf{SS}$. In fact, we may even ask for more, in terms of *strong* unforgeability — the value R is uniquely defined by lr, and, along with the message M, it defines a *unique* signature (although not efficiently computable given the verification key alone). An ideal goal, which corresponds to our strongest security goal, is to ensure that the *only* way to generate the signature associated with lr is to obtain a signature share for lr from every honest server whose tokens are included in lr. This is a notion we refer to as TS-SUF-4.

We will however show that neither $\mathsf{FROST1}$ nor $\mathsf{FROST2}$ meet TS-SUF-4. To overcome this, we will show a general transformation which can boost the security of a TS-SUF-3-secure scheme like $\mathsf{FROST1}$ to achieve TS-SUF-4. Our framework allows schemes more general than the FROST ones, and also leaves the question open of better and more efficient designs achieving the stronger notions. Moreover, we provide simple reference schemes for all of our notions, which, while inefficient, guide us in understanding the subtle differences among notions and baseline requirements. In particular, these schemes will enable us to separate the proposed notions.

A SUMMARY FOR OUR NOTIONS. In summary, our unforgeabilty notions declare a signature for a message M trivial in the following cases:

- TS-UF-0: A partial signature for the message M was generated by at least one honest server.
- TS-UF-1: A partial signature for the message M was generated by at least $t-c$ honest servers, where c is the number of corrupted servers.
- TS-UF-2: There exists a leader request lr for the message M which was answered by at least $t-c$ honest servers.
- TS-UF-3: There exists a leader request lr for the message M such that every honest server $i \in lr.\mathsf{SS}$ either answered lr or the token pp_i associated with i in lr is maliciously generated.
- TS-UF-4: There exists a leader request lr for the message M such that every honest server $i \in lr.\mathsf{SS}$ answered lr.

Analogous notions of strong unforgeability are obtained by further associating a request lr to a (unique) signature, in addition to a message M.

We stress that it is not clear which scenarios demand which notions in our hierarchy. This is especially true because we are still lacking formal analyses of full-fledged systems using threshold signatures, but it is not hard to envision a potential mismatch between natural expectations from such schemes and what they actually achieve. In both FROST variants, for example, it is natural to expect that a signature can only be generated by a sufficient number of honest servers answering the *same* request, a property which we show is actually achieved. Further, one may also expect that all honest servers that generated these honest tokens need to be involved in the generation of a valid signature, but this stronger property is actually not achieved by either of the FROST variants.

WHAT WE DO NOT DO. Some schemes like FROST come with a concrete *distributed key-generation* (DKG) protocol. Security proofs frequently (but not always) consider, monolithically, the composition of DKG and threshold signing. This lack of modular treatment is due to the fact that efficient DKG protocols like Pedersen's [36] are not secure [25] in the strongest possible sense *in isolation*, but it may still be possible to show security when they are used with a particular threshold signature scheme. Here, instead, we idealize DKG protocols, as the points we are trying to express are orthogonal to the concrete choice of a DKG. Our result would still guarantee security of the schemes when used with truly secure DKGs (such as the DKG from [25]), but further investigation is needed to extend our proofs to consider more efficient DKGs.

Our framework does not handle adaptive corruptions, i.e., we demand instead that the adversary declares its corruption set initially. We could extend our definitions to adaptive corruptions rather easily, but our concrete bounds would be impacted. In particular, we would resort to a generic reduction guessing the corrupted set beforehand, with a multiplicative loss of 2^{ns}, which is acceptable for the smaller values of the number ns of parties that we consider common in practice.

Our framework cannot cover recent protocols, like that of Canetti et al. [14], which combine a *multi-round* message-independent pre-processing phase with a final, message-dependent, round. (Conversely, their UC security analysis does not give definitions which help our fine-grained framework.)

Finally, many prior works also consider *robustness*, i.e., the guarantee that a signature is always produced. Here, we follow the same viewpoint as in FROST, and do not focus on robustness explicitly. This allows us to prevent imposing a small t (relative to ns) just for the sake of ensuring it. However, our schemes all implicitly give verification keys vk_i for each server, and it is not hard to verify individual partial signatures $psig_i$. Any t valid partial signatures will always aggregate into a valid signature.

RELATED AND CONCURRENT WORK. A recent preprint by Groth [28] presents a general definition for fully non-interactive schemes in a setting with a (non-interactive) DKG. His definition implies TS-UF-1, and he also provides a proof sketch that BLS (with his newly proposed non-interactive DKG) is secure under a variant of the OMCDH assumption, which is closely related to our VCDH assumption which we also show to be hard in the GGM. Groth's framework is not suitable for partially non-interactive schemes like FROST, which are the main focus of our work.

HISTORY OF THIS PAPER. This paper (BTZ) was submitted to Crypto 2022. The PC imposed a (hard) merge with the also-submitted work of CKM [15], resulting in the joint Crypto 2022 paper BCKMTZ [1]. The Crypto 2022 paper includes the materiel here, and, from CKM [15], the $\mathsf{FROST2}$ scheme together with an analysis of its security when used with a DKG, the latter being a variant of Pedersen's DKG introduced in conjunction with $\mathsf{FROST1}$ [30]. We thank Tibor Jager for his generous shepherding of the merge. Full versions of the BTZ and CKM papers have been kept separate, and we see each group as responsible for the proofs in their portion of the joint work.

2 Preliminaries

NOTATION. If $b \geq a \geq 1$ are positive integers, then $\mathbb{Z}_a$ denotes the set $\{0, \ldots, a-1\}$ and $[a..b]$ denotes the set $\{a, \ldots, b\}$. If $\boldsymbol{x}$ is a vector then $|\boldsymbol{x}|$ is its length (the number of its coordinates), $\boldsymbol{x}[i]$ is its i-th coordinate and $[\boldsymbol{x}] = \{\, \boldsymbol{x}[i] \,:\, 1 \leq i \leq |\boldsymbol{x}| \,\}$ is the set of all its coordinates. A string is identified with a vector over $\{0, 1\}$, so that if x is a string then $x[i]$ is its i-th bit and $|x|$ is its length. By ε we denote the empty vector or string. The size of a set S is denoted $|S|$. For sets D, R let $\mathrm{FNS}(D, R)$ denote the set of all functions $f : D \to R$.

Let S be a finite set. We let $x \leftarrow\!\!\$\; S$ denote sampling an element uniformly at random from S and assigning it to x. We let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; r)$ denote executing algorithm A on inputs $x_1, \ldots$ and coins r with access to oracles $\mathrm{O}_1, \ldots$ and letting y be the result. We let $y \leftarrow\!\!\$\; A^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; r)$. Algorithms are randomized unless otherwise indicated. Running time is worst case.

GAMES. We use the code-based game playing framework of [8]. (See Fig. 2 for an example.) Games have procedures, also called oracles. Among the oracles are INIT (Initialize) and FIN (Finalize). In executing an adversary $\mathcal{A}$ with a game Gm, the adversary may query the oracles at will, with the restriction that its first query must be to INIT (if present), its last to FIN, and it can query these oracles at most once. The value returned by the FIN procedure is taken as the game output. By $\mathrm{Gm}(\mathcal{A}) \Rightarrow y$ we denote the event that the execution of game Gm with adversary $\mathcal{A}$ results in output y. We write $\Pr[\mathrm{Gm}(\mathcal{A})]$ as shorthand for $\Pr[\mathrm{Gm}(\mathcal{A}) \Rightarrow \mathsf{true}]$, the probability that the game returns true.

In writing game or adversary pseudocode, it is assumed that Boolean variables are initialized to false, integer variables are initialized to 0 and set-valued variables are initialized to the empty set $\emptyset$.

<u>Groups.</u> Let $\mathbb{G}$ be a group of order p. We will use multiplicative notation for the group operation, and we let $1_{\mathbb{G}}$ denote the identity element of $\mathbb{G}$. We let $\mathbb{G}^* = \mathbb{G} \setminus \{1_{\mathbb{G}}\}$ denote the set of non-identity elements, which is the set of generators of $\mathbb{G}$ if the latter has prime order. If $g \in \mathbb{G}^*$ is a generator and $X \in \mathbb{G}$, the discrete logarithm base g of X is denoted $\mathsf{DL}_{\mathbb{G},g}(X)$, and it is in the set $\mathbb{Z}_{|\mathbb{G}|}$.

3 A Framework for Non-Interactive Threshold Signatures

We present our hierarchy of definitions of security for non-interactive threshold schemes, formalizing both unforgeability (UF) and strong unforgeability (SUF) in several ways. We provide relations between all notions considered.

3.1 Syntax and Correctness

<u>Maintaining state.</u> Parties as implemented in protocols would maintain state. When activated with some inputs (which include messages from other parties), they would apply some algorithm Alg to these and their current state to get outputs (including outgoing messages) and an updated state. To model this, we do not change our definition of algorithms, but make the state an explicit input and output that will, in definitions, be maintained by the overlying game. Thus, we would write something like $(\cdots, \mathsf{st}) \leftarrow\$ \mathsf{Alg}(\cdots, \mathsf{st})$.

<u>Syntax.</u> A non-interactive threshold signature scheme TS specifies a number $\mathsf{ns} \geq 1$ of servers, a reconstruction threshold t, a set HF of functions from which the random oracle is drawn, a key-generation algorithm Kg, a server pre-processing algorithm SPP, a leader pre-processing algorithm LPP, a leader signing-request algorithm LR, a server partial-signature algorithm PS, a leader partial-signature aggregation algorithm Agg and a verification algorithm Vf. If disambiguation is needed, we write $\mathsf{TS.ns}, \mathsf{TS}.t, \mathsf{TS.HF}, \mathsf{TS.Kg}, \mathsf{TS.SPP}, \mathsf{TS.LPP}, \mathsf{TS.LR}, \mathsf{TS.PS}, \mathsf{TS.Agg}, \mathsf{TS.Vf}$, respectively. We now explain the operation and use of these components, the understanding of which may be aided by already looking at the correctness game $\mathbf{G}^{\text{ts-cor}}_{\mathsf{TS}}$ of Figure 1.

Parties involved are a leader (numbered 0, implicit in some prior works, but made explicit here) and servers numbered $1, \ldots, \mathsf{ns}$, for a total of $\mathsf{ns} + 1$ parties. Algorithms have oracle access to a function h that is drawn at random from HF in games (line 1 Figure 1) and plays the role of the random oracle. Specifying HF as part of the scheme allows the domain and range of the random oracle to be scheme dependent.

The key-generation algorithm Kg, run once at the beginning (line 1 of Figure 1), creates a public signature-verification key vk, associated public auxiliary information aux and an individual secret signing key sk_i for each server $i \in [1..\mathsf{ns}]$. (Usually, $sk_1, \ldots, sk_{\mathsf{ns}}$ will be shares of a global secret key sk, but the definitions do not need to make sk explicit. The leader does not hold any secrets associated to vk.) While key-generation may in practice be performed by a distributed key-generation protocol, our syntax assumes it done by a trusted algorithm to allow a modular treatment. Keys are held by parties in their state, encoded into dedicated fields of the latter as shown at line 3 of Figure 1. For specific scheme, we will typically use aux to model additional information that can be leaked by key generation step without violating security (e.g., the values g^{sk_i} in most cases).

The signing protocol can be seen as having two rounds, which we think as a pre-processing and online stage. In a pre-processing round, any server i can run $(pp, \mathsf{st}_i) \leftarrow\$ \mathsf{SPP}[\mathsf{h}](\mathsf{st}_i)$ to get

```
Game $\mathbf{G}^{\text{ts-cor}}_{\mathsf{TS}}$

INIT:
1  $\mathsf{h} \leftarrow\!\$\ \mathsf{TS.HF}$ ; $sk_0 \leftarrow \bot$ ; $(vk, aux, sk_1, \ldots, sk_{\mathsf{ns}}) \leftarrow\!\$\ \mathsf{Kg}[\mathsf{h}]$
2  For $i = 0, \ldots, \mathsf{ns}$ do  // Initialize party states with keys
3      $\mathsf{st}_i.\mathsf{sk} \leftarrow sk_i$ ; $\mathsf{st}_i.\mathsf{vk} \leftarrow vk$ ; $\mathsf{st}_i.\mathsf{aux} \leftarrow aux$
4  Return $vk, aux, sk_1, \ldots, sk_{\mathsf{ns}}$

PPO($i$):  // $i \in [1..\mathsf{ns}]$
5  $(pp, \mathsf{st}_i) \leftarrow\!\$\ \mathsf{SPP}[\mathsf{h}](\mathsf{st}_i)$ ; $\mathsf{st}_0 \leftarrow \mathsf{LPP}[\mathsf{h}](pp, \mathsf{st}_0)$
6  Require: $pp \neq \bot$
7  Return $pp$

SIGNO($M, SS$):
8  Require: $SS \subseteq [1..\mathsf{ns}]$ and $|SS| \geq t$  // Set of signers
9  $(lr, \mathsf{st}_0) \leftarrow\!\$\ \mathsf{LR}[\mathsf{h}](M, SS, \mathsf{st}_0)$
10 Require: $lr \neq \bot$  // Leader accepts request
11 If ($lr.\mathsf{msg} \neq M$ or $lr.\mathsf{SS} \neq SS$) then $\mathsf{win} \leftarrow \mathsf{true}$
12 For $i \in SS$ do
13     $(psig_i, \mathsf{st}_i) \leftarrow\!\$\ \mathsf{PS}[\mathsf{h}](lr, i, \mathsf{st}_i)$  // Servlet partial signatures
14 $(sig, \mathsf{st}_0) \leftarrow\!\$\ \mathsf{Agg}[\mathsf{h}](lr, \{psig_i\}_{i \in SS}, \mathsf{st}_0)$
15 If $\mathsf{Vf}[\mathsf{h}](vk, M, sig) = \mathsf{false}$ then $\mathsf{win} \leftarrow \mathsf{true}$

RO($x$):  // Random oracle
16 Return $\mathsf{h}(x)$

FIN:
17 Return $\mathsf{win}$
```

Figure 1: Game used to define correctness of threshold signature scheme TS with threshold t.

a *pre-processing token* pp which it sends to the leader. (Here st_i is the state of i.) Via $\mathsf{st}_0 \leftarrow \mathsf{LPP}[\mathsf{h}](pp, \mathsf{st}_0)$, the leader updates its state st_0 to incorporate token pp. (In Figure 1, this is reflected in lines 5–7.)

In a signing round the leader begins with a message and a choice of a signer set $SS \subseteq [1..\mathsf{ns}]$ of size at least t. Via $(lr, \mathsf{st}_0) \leftarrow\!\$\ \mathsf{LR}[\mathsf{h}](M, SS, \mathsf{st}_0)$ it generates a leader request lr that, through st_0, implicitly depends on a choice of pre-processing tokens. (Lines 8,9 of Figure 1.) The leader request is sent to each $i \in SS$, who, via $(psig_i, \mathsf{st}_i) \leftarrow\!\$\ \mathsf{PS}[\mathsf{h}](lr, i, \mathsf{st}_i)$, computes a partial signature $psig_i$ and returns it to the leader. Via $(sig, \mathsf{st}_0) \leftarrow\!\$\ \mathsf{Agg}[\mathsf{h}](lr, \{psig_i\}_{i \in SS}, \mathsf{st}_0)$, the leader aggregates the partial signatures into a signature sig of M, the desired output of the protocol. (Lines 12–14 of Figure 1.)

The verification algorithm, like in a standard signature scheme, takes vk, a message M and a candidate signature, and returns a boolean validity decision.

ECHO SCHEMES. We define a sub-class of non-interactive threshold schemes that we call *echo schemes*. Recall that a leader request lr is mandated to specify a message $lr.\mathsf{msg}$ and a set $lr.\mathsf{SS} \subseteq [1..\mathsf{ns}]$ of servers from whom partial signatures are being requested. In an echo scheme, lr additionally specifies a function $lr.\mathsf{PP} : lr.\mathsf{SS} \to \{0,1\}^*$. If the leader is honest, $lr.\mathsf{PP}(i)$ is a token pp that i had previously sent to the leader. That is, the leader is echoing tokens back to the servers, whence the name. In considering security, of course, $lr.\mathsf{PP}(i)$ is picked by the adversary and may not be a prior token. As we will discuss in Section 5.1, FROST is a typical example of an echo scheme.

CORRECTNESS OF A TS SCHEME. The game of Figure 1 defines correctness, and serves also to detail the above. Recall that TS specifies a threshold $t \in [1..\mathsf{ns}]$. The adversary will make the leader's pre-processing requests, via oracle PPO. It will likewise make signing requests via oracle SIGNO. If any condition listed under Require: fails the adversary is understood as losing, the game automatically returning false. We let $\mathbf{Adv}^{\text{ts-corr}}_{\mathsf{TS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ts-cor}}_{\mathsf{TS}}]$ be the advantage of an adversary $\mathcal{A}$. The default requirement is perfect correctness, which means that $\mathbf{Adv}^{\text{ts-corr}}_{\mathsf{TS}}(\mathcal{A}) = 0$ for all $\mathcal{A}$, regardless of computing time and number of oracle queries, but this can be relaxed, as may be necessary for lattice-based protocols.

The way in which we are supposed to interpret the correctness definition is that a request *lr* is associated with a set *SS* and a message M, and if such a request is issued successfully by the leader (i.e., $lr \neq \perp$), then the servers in *SS* would all accept *lr* producing partial signatures which aggregate into a valid signature for M. We note that this definition assumes that we submit requests to all servers in the same order. One can give a stronger (but more complex) definition which ensures correctness even when servers process requests in different orders, but note that for all schemes we discuss below they will be equivalent, and we hence omit the more cumbersome game to define it.

3.2 Unforgeability and Strong Unfogeability

UNFORGEABILITY. Unforgeability as usual asks that the adversary be unable to produce a valid signature *sig* on some message M of its choice except in a trivial way. The question is what "trivial" means. For regular signatures, it means that the adversary did not obtain a signature of M from the signing oracle [27]. For threshold signatures, it is more subtle. We will give several definitions.

Fig. 2 simultaneously describes several games, $\mathbf{G}^{\text{ts-uf-}i}_{\mathsf{TS}}$ for $i = 0, 1, 2, 3, 4$, where $\mathbf{G}^{\text{ts-uf-3}}_{\mathsf{TS}}$ is only defined if TS is an echo scheme. (We will get to the second set of games later.) They are almost the same, differing only at line 20. The corresponding advantages of an adversary $\mathcal{A}$ are $\mathbf{Adv}^{\text{ts-uf-i}}_{\mathsf{TS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ts-uf-}i}_{\mathsf{TS}}(\mathcal{A})]$. The adversary calls INIT with a choice of a set of servers to corrupt. It is also viewed as having corrupted the leader. Playing the leader role, it can request pre-processing tokens via oracle PPO. It can provide a server with a leader-request *lr* of its choice to obtain a partial signature *psig*. At the end, it outputs to FIN its forgery message M and signature *sig*. If the signature is not valid, line 18 ensures that the adversary does not win. Now, to win, the signature must be non-trivial. It is in how this is defined that the games differ. Associated to i is a *trivial forgery* predicate $\mathtt{tf}_i$ that is invoked at line 20. The choices for these predicates are shown in the table in Figure 3, and the notion corresponding to game $\mathtt{tf}_i$ is denoted TS-UF-i. When $i = 0$ we have the usual notion from the literature, used in particular in [9, 23, 25]. As i increases, we get more stringent (less generous) in declaring a forgery trivial, and the notion gets stronger.

Concretely, TS-UF-0 considers a signature for a message M trivial if a request *lr* with $lr.\mathsf{msg}$ was answered by server with a partial signature. Moving on, TS-UF-1 strengthens this by declaring a signature trivial only if at least $t - |CS|$ servers have responded to some request for message M, where these requests could have been different. In turn, TS-UF-2 strengthens this even further by requiring that there was a single prior request *lr* for M which was answered by $t - |CS|$ servers.

The notions TS-UF-3 only deals with echo schemes. Recall that for these schemes, a request *lr* contains a map $lr.\mathsf{PP} : lr.\mathsf{SS} \to \{0,1\}^*$, where $lr.\mathsf{PP}(i)$ is meant to be a token issued by server i. Here, we consider a signature for message M trivial if there exists a request *lr* for M which is answered by all honest servers i for which $lr.\mathsf{PP}(i)$ is a valid token previously output by i, and this set consists of at least $t - |CS|$ servers. Finally, our strongest notion, TS-UF-4 simply considers a signature trivial if there exists a request *lr* for M which is answered by all honest servers in $i \in lr.\mathsf{SS}$.

```
Games $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-}i}$ $(i = 0, 1, 2, 3, 4)$ and $\mathbf{G}_{\mathsf{TS}}^{\text{ts-suf-}i}$ $(i = 2, 3, 4)$

INIT($CS$):
 1 Require: $CS \subseteq [1..\mathsf{ns}]$ and $|CS| < t$ // Set of corrupted parties
 2 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{TS.HF}$ ; $(vk, aux, sk_1, \ldots, sk_{\mathsf{ns}}) \leftarrow\!\!\$\ \mathsf{Kg}[\mathsf{h}]$
 3 $HS \leftarrow [1..\mathsf{ns}] \setminus CS$ // Set of honest parties
 4 For $i \in HS$ do
 5     $\mathsf{st}_i.\mathsf{sk} \leftarrow sk_i$ ; $\mathsf{st}_i.\mathsf{vk} \leftarrow vk$ ; $\mathsf{st}_i.\mathsf{aux} \leftarrow aux$
 6 Return $vk, aux, \{sk_i\}_{i \in CS}$

PPO($i$):
 7 Require: $i \in HS$
 8 $(pp, \mathsf{st}_i) \leftarrow\!\!\$\ \mathsf{SPP}[\mathsf{h}](\mathsf{st}_i)$ ; $\mathrm{PP}_i \leftarrow \mathrm{PP}_i \cup \{pp\}$ ; Return $pp$

PSIGNO($i, lr$):
 9 $M \leftarrow lr.\mathsf{msg}$
10 Require: $lr.\mathsf{SS} \subseteq [1..\mathsf{ns}]$ and $M \in \{0,1\}^*$ and $i \in HS$
11 $\mathrm{L} \leftarrow \mathrm{L} \cup \{lr\}$ ; $(psig, \mathsf{st}_i) \leftarrow\!\!\$\ \mathsf{PS}[\mathsf{h}](lr, i, \mathsf{st}_i)$
12 If $(psig \neq \bot)$ then
13     $\mathrm{S}_1(M) \leftarrow \mathrm{S}_1(M) \cup \{i\}$ ; $\mathrm{S}_2(lr) \leftarrow \mathrm{S}_2(lr) \cup \{i\}$
14 Return $psig$

RO($x$): // Random oracle
15 Return $\mathsf{h}(x)$

FIN($M, sig$):
16 For all $lr \in \mathrm{L}$ do
17     $\mathrm{S}_3(lr) \leftarrow \{\, i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i \,\}$ ; $\mathrm{S}_4(lr) \leftarrow HS \cap lr.\mathsf{SS}$
18 If (not $\mathsf{Vf}[\mathsf{h}](vk, M, sig)$) then return false
19 Return (not $\mathtt{tf}_i(M)$) // Game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-}i}$ for $i = 0, 1$
20 Return (not $\exists\, lr$ ( $lr.\mathsf{msg} = M$ and $\mathtt{tf}_i(lr)$ )) // Game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-}i}$ for $i = 2, 3, 4$
21 Return (not $\exists\, lr$ ( $lr.\mathsf{msg} = M$ and $\mathtt{tsf}_i(lr, vk, sig)$ )) // Game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-suf-}i}$
```

Figure 2: Games used to define TS-UF-i and TS-SUF-i unforgeability of threshold signature scheme TS. Line 20 is included only in game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-}i}$ and line 21 only in game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-suf-}i}$. These lines refer to the trivial-forgery predicates $\mathtt{tf}_i(lr)$ and trivial strong-forgery predicates $\mathtt{tsf}_i(lr, vk, sig)$ from Figure 3. In particular, the set $\mathrm{S}_3(lr)$ and, thus, TS-UF-3 and TS-SUF-3 unforgeability are defined only if TS is an echo scheme.

It is natural to expect TS-UF-3 and TS-UF-4 to be similar, but as we will see below, they are actually not equivalent. (Although we will give a transformation that boosts an TS-UF-3-secure scheme into an TS-UF-4-secure one.)

STRONG UNFORGEABILITY. For standard signatures, strong unforgeability asks, in addition to unforgeability, that the adversary be unable to produce a new signature on any message, where new means different from any obtained legitimately for that message. We ask, does this have any counterpart in threshold signatures? In fact, FROST seems to have such a property. We now provide formalisms to capture such properties.

It turns out that giving a general definition of strong unforgeability is rather complex, and we will restrict ourselves to a natural subclass of schemes (which includes FROST). Concretely, we ask that there is an algorithm SVf, called a *strong verification algorithm*, that takes a public key vk, a

$\mathtt{tf}_0(M)$	:	$\mathrm{S}_1(M) \neq \emptyset$
$\mathtt{tf}_1(M)$	:	$\lvert \mathrm{S}_1(M)\rvert \geq t - \lvert CS\rvert$
$\mathtt{tf}_2(lr)$	:	$\lvert \mathrm{S}_2(lr)\rvert \geq t - \lvert CS\rvert$
$\mathtt{tf}_3(lr)$	:	$\mathtt{tf}_2(lr)$ and $\mathrm{S}_2(lr) = \mathrm{S}_3(lr)$
$\mathtt{tf}_4(lr)$	:	$\mathtt{tf}_2(lr)$ and $\mathrm{S}_2(lr) = \mathrm{S}_4(lr)$

$\mathtt{tsf}_2(lr, vk, sig)$	:	$\mathtt{tf}_2(lr)$ and $\mathsf{SVf}[\mathsf{h}](vk, lr, sig)$
$\mathtt{tsf}_3(lr, vk, sig)$	:	$\mathtt{tf}_3(lr)$ and $\mathsf{SVf}[\mathsf{h}](vk, lr, sig)$
$\mathtt{tsf}_4(lr, vk, sig)$	:	$\mathtt{tf}_4(lr)$ and $\mathsf{SVf}[\mathsf{h}](vk, lr, sig)$

$$\begin{tikzcd}
\text{TS-UF-0} \arrow[r, bend right] & \text{TS-UF-1} \arrow[l] \arrow[r, dotted, bend right] & \text{TS-UF-2} \arrow[l] & \text{TS-UF-3} \arrow[l] \arrow[r, dashed, bend left] & \text{TS-UF-4} \arrow[l] \arrow[dll, dotted] \\
 & & \text{TS-SUF-2} \arrow[u] \arrow[ur, dotted] & \text{TS-SUF-3} \arrow[l] \arrow[u] \arrow[ur, dotted] \arrow[r, dashed, bend right] & \text{TS-SUF-4} \arrow[l] \arrow[u]
\end{tikzcd}$$

Figure 3: **Top:** Trivial-forgery conditions $\mathtt{tf}_i(lr)$ ($i = 0, 1, 2, 3, 4$) and trivial-strong-forgery conditions $\mathtt{tsf}_i(lr, vk, sig)$ ($i = 1, 2, 3, 4$) used to define TS-SUF-i and TS-SUF-i security in games $\mathbf{G}^{\text{ts-uf-}i}_{\mathsf{TS}}$ and $\mathbf{G}^{\text{ts-suf-}i}_{\mathsf{TS}}$, respectively. **Bottom:** Relations between notions of security.

leader request lr, and a signature sig as inputs and outputs true or false. We require that for any vk, lr there exists at most one signature sig such that $\mathsf{SVf}(vk, lr, sig) = \mathsf{true}$. Also, TS is asked to satisfy a strong correctness property which is defined using the same game as $\mathbf{G}^{\text{ts-cor}}_{\mathsf{TS}}$ except the condition $\mathsf{Vf}[\mathsf{h}](vk, M, sig) = \mathsf{false}$ in line 15 is replaced with $\mathsf{SVf}[\mathsf{h}](vk, lr, sig) = \mathsf{false}$.

For a scheme TS with a strong verification algorithm, we consider the $\mathbf{G}^{\text{ts-suf-}i}_{\mathsf{TS}}$ ($i = 2, 3, 4$) games in Figure 2, where $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{TS}}$ is only defined if TS additionaly is an echo scheme. The differences (across the different values of i) are only in the trivial strong forgery predicates $\mathtt{tsf}_i$ used at line 21, and the choices are again shown in the table in Figure 3. The corresponding advantage of an adversary $\mathcal{A}$ is $\mathbf{Adv}^{\text{ts-suf-i}}_{\mathsf{TS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ts-suf-}i}_{\mathsf{TS}}(\mathcal{A})]$. The ensuing notion is called TS-SUF-i.

3.3 Relations and Transformations

Relations between notions. Figure 3 shows relations between the notions of unforgeability and strong unforgeabilty that we have defined. A (blue, non-dotted) arrow A $\rightarrow$ B is an implication, saying that A implies B: any scheme that is A-secure is also B-secure. Now see the nodes as forming a graph with edges the blue, non-dotted arrows. The thin arrow from TS-UF-0 to TS-UF-1 indicates us that the implication only holds under a quantatively loose reduction. (We prove this in Theorem 3.1.) We claim that in this graph, if there is no path from a notion B to a notion A, they are separate or distinct: there exists a scheme that is B-secure but not A-secure. The dotted arrows are separations that we explicitly prove. These, together with the full arrows, prove the claim just made. The thick dottet arrows indicate the existence of a generic transformation lifting security of a scheme to achieve a stronger notion. (We establish this below as part of Theorem 3.2.)

Reference schemes and proofs of relations. In Appedndix A, we give a set of (fully) non-interactive threshold schemes that we call reference schemes. They represent simple, canonical ways to achieve the different notions. They may not be of practical interest, because they have key and signature sizes proportional to ns, but the point is to embody notions in a representative way. A few things emanate from these schemes. One is that we use them, in the same Appendix, to establish the separations given by the dotted lines in Figure 3, thereby showing that any notions between which there is no path, in the graph given by the full arrows, are indeed separate. Second, we get a scheme that achieves our strongest notion, TS-SUF-4, which neither FROST nor BLS achieve. (Although we can get such a scheme by applying our transformation from Theorem 3.2 to FROST1.) Finally, reference schemes, as canonical examples, are ways to understand the notions.

From TS-UF-0 to TS-UF-1, loosely The following theorem shows TS-UF-1 security is implied by TS-UF-0 security, although with an exponential loss in t, which is acceptable in settings where t is expected to be constant.

Theorem 3.1 *Let* TS *be a threshold signature scheme. For any TS-UF-1 adverary* $\mathcal{A}$ *there exists a TS-UF-0 adversary* $\mathcal{B}$ *such that* $\mathbf{Adv}^{\text{ts-uf-1}}_{\mathsf{TS}}(\mathcal{A}) \leq \binom{\mathsf{ns}}{t-1} \cdot \mathbf{Adv}^{\text{ts-uf-0}}_{\mathsf{TS}}(\mathcal{B})$ *. Moreover,* $\mathcal{B}$ *runs in time roughly equal that of* $\mathcal{A}$*, and the number of* $\mathcal{B}$*'s queries to each oracle is at most that of* $\mathcal{A}$*.*

If the adversary always corrupts $t-1$ parties, it is clear that TS-UF-0 and TS-UF-1 are equivalent. Otherwise, in general, for an adversary that breaks TS-UF-1 security and corrupts a subset CS of servers with size less than $t-1$, if the adversary wins the game $\mathbf{G}^{\text{ts-uf-1}}_{\mathsf{TS}}$ by outputting (M^*, sig^*), we know $|\mathrm{S}_1(M^*)| < t - |CS|$. Therefore, we can modify the adversary to initially guess a subset $ECS \subseteq [1..\mathsf{ns}] \setminus CS$ with size $t - |CS| - 1$ and corrupt all parties in ECS. If ECS happens to contain $\mathrm{S}_1(M^*)$, the adversary actually wins. It is not hard to see that the probability that this is true is $1/\binom{\mathsf{ns}-|CS|}{t-|CS|-1} \geq 1/\binom{\mathsf{ns}}{t-1}$. We give a formal proof in Appedndix B.

From TS-(S)UF-3 to TS-(S)UF-4. Fig. 4 gives a general transformation from TS-(S)UF-3 security to TS-(S)UF-4 security. Concretely, we give a construction ATS from any TS-(S)UF-3-secure echo scheme TS and a digital signature scheme DS. The size of signatures produced by ATS and the verification algorithm Vf are exactly the same as TS. The main idea is to use signatures to authenticate each token contained in a leader request lr from TS, so that an honest server only answers the request if all the authentications are valid. The rest of the protocol remains the same.

In the game $\mathbf{G}^{\text{ts-(s)uf-4}}_{\mathsf{ATS}}$, we can show that as long as the adversary does not break the strong unforgeability of DS, for any leader request lr such that $\mathrm{S}_2(lr) > 0$, it holds that $\mathrm{S}_3(lr) = \mathrm{S}_4(lr)$, which implies the conditions $\mathtt{tf}_3$ and $\mathtt{tf}_4$ are equivalent. Therefore, we can reduce TS-(S)UF-4 security of ATS to TS-(S)UF-3 security of TS and SUF-CMA security of DS. (The latter notion is formally defined via the game in Fig. 5.) This is captured by the the following theorem. (The proof is in Appedndix C.)

Theorem 3.2 *Let* $XX \in \{SUF, UF\}$*. Let* TS *be an echo scheme and* DS *be a digital signature scheme. For any TS-XX-4 adversary* $\mathcal{A}$ *there exists a TS-XX-3 adversary* $\mathcal{B}$ *and a SUF-CMA adversary* $\mathcal{C}$ *such that*

$$\mathbf{Adv}^{\text{ts-xx-4}}_{\mathsf{ATS[TS,DS]}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ts-xx-3}}_{\mathsf{TS}}(\mathcal{B}) + \mathsf{ns} \cdot \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{DS}}(\mathcal{C}) \ .$$

Moreover, $\mathcal{B}$ *and* $\mathcal{C}$ *run in time roughly equal that of* $\mathcal{A}$*. The number of* $\mathcal{B}$*'s queries to each oracle is at most that of* $\mathcal{A}$*. The number of* $\mathcal{C}$*'s* SignO *queries is at most the number of* PPO *queries made by* $\mathcal{A}$*.*

```
Protocol ATS[TS, DS]

Kg[h]:
1  $vk, taux, \{tsk_i\}_{i\in[1..ns]} \leftarrow$ TS.Kg
2  For $i \in [1..ns]$ do
3      $(svk_i, ssk_i) \leftarrow\$$ DS.Kg
4      $sk_i \leftarrow (tsk_i, ssk_i)$
5  $aux \leftarrow (taux, svk_1, \ldots, svk_{ns})$
6  Return $vk, aux, \{sk_i\}_{i\in[1..ns]}$

SPP[h](st_i):
7  $(tpp, st_i) \leftarrow\$$ SPP[h]$(st_i)$
8  $(tsk_i, ssk_i) \leftarrow st_i.sk$
9  $tsig \leftarrow\$$ DS.Sig$(ssk_i, tpp)$
10 Return $((tpp, tsig), st_i)$

LPP[h](i, pp, st_0):
11 $(tpp, tsig) \leftarrow pp$
12 $st_0.SigMap(i, tpp) \leftarrow tsig$
13 Return TS.LPP[h]$(i, tpp, st_0)$

OriginLR(lr):
14 For $i \in lr.SS$ do
15     $(tpp, tsig) \leftarrow lr.PP(i)$
16     $lr.PP(i) \leftarrow tpp$
17 Return $lr$

LR[h](M, SS, st_0):
18 $(lr, st_0) \leftarrow$ TS.LR[h]$(M, SS, st_0)$
19 For $i \in SS$ do
20     $tpp_i \leftarrow lr.PP(i)$
21     $lr.PP(i) \leftarrow (tpp_i, st_0.SigMap(i, tpp_i))$
22 Return $(lr, st_0)$

PS[h](lr, i, st_i):
23 $(taux, svk_1, \ldots, svk_{ns}) \leftarrow st_i.aux$
24 For $i \in lr.SS$ do
25     $(tpp_i, tsig_i) \leftarrow lr.PP(i)$
26     If DS.Vf$(svk_i, tpp_i, tsig_i)$ = false then
27         Return $\perp$
28 Return TS.PS[h](OriginLR$(lr), i, st_i)$

Agg[h](PS, st_0):
29 Return TS.Agg[h]$(PS, st_0)$

Vf[h](vk, M, sig):
30 Return TS.Vf[h]$(vk, M, sig)$

SVf[h](vk, lr, sig):
31 Return TS.SVf[h]$(vk$, OriginLR$(lr), sig)$
```

Figure 4: The threshold signature ATS[TS, DS] constructed from an echo scheme TS and a digital signature scheme DS such that ATS.ns = TS.ns and ATS.t = TS.t. The algorithm OriginLR transforms a well-formed leader request lr for ATS to a well-formed leader request in TS. st_0.SigMap is a table that stores the signature corresponding to each token generated by honest servers, which is initially set to empty. PS denotes a set of partial signatures.

```
Games $\mathbf{G}^{\text{suf-cma}}_{DS}$

INIT:
1  $(vk, sk) \leftarrow\$$ DS.Kg
2  Return $vk$

SIGNO(M):
3  $sig \leftarrow\$$ DS.Sig$(sk, M)$
4  $Q \leftarrow Q \cup \{(M, sig)\}$
5  Return $sig$

FIN(M, sig):
6  If DS.Vf$(vk, M, sig)$ and $(M, sig) \notin Q$ then
7      Return true
8  Return false
```

Figure 5: The game $\mathbf{G}^{\text{suf-cma}}_{DS}$, where DS is a diginal signature scheme.

4 The Security of Threshold BLS Signatures

We revisit the BLS signature scheme [9, 13] within our definitional framework. While a proof of TS-UF-0 security basically recasts similar guarantees to those proved by [9], the more interesting

```
Protocol $\mathsf{BLS}[\mathbb{G}, \mathbb{G}_\mathsf{T}]$

$\mathsf{Kg}[\mathsf{h}]$:
For $i \in [0..t-1]$ do
    $a_i \leftarrow\$ \mathbb{Z}_p$
For $i \in [1..\mathsf{ns}]$ do
    $sk_i \leftarrow\$ \sum_{j=0}^{t-1} i^j \cdot a_j$ ; $vk_i \leftarrow g^{sk_i}$
$vk \leftarrow g^{a_0}$
$aux \leftarrow (vk_1, \ldots, vk_{\mathsf{ns}})$
Return $vk, aux, \{sk_i\}_{i \in [1..\mathsf{ns}]}$

$\mathsf{SPP}[\mathsf{h}](\mathsf{st}_i)$:
Return $(\perp, \mathsf{st}_i)$

$\mathsf{LPP}[\mathsf{h}](i, pp, \mathsf{st}_0)$:
Return $\mathsf{st}_0$

$\mathsf{PS}[\mathsf{h}](lr, i, \mathsf{st}_i)$:
If $i \in lr.\mathsf{SS}$ then
    $psig \leftarrow \mathsf{h}(lr.\mathsf{msg})^{\lambda_i^{lr.\mathsf{SS}} \cdot \mathsf{st}_i.sk}$
Else $psig \leftarrow \perp$
Return $(psig, \mathsf{st}_i)$

$\mathsf{Agg}[\mathsf{h}](\mathrm{PS}, \mathsf{st}_0)$:
$sig \leftarrow 1_\mathbb{G}$
For $psig \in \mathrm{PS}$ do
    $sig \leftarrow sig \cdot psig$
Return $(sig, \mathsf{st}_0)$

$\mathsf{LR}[\mathsf{h}](M, SS, \mathsf{st}_0)$:
$lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow SS$
Return $(lr, \mathsf{st}_0)$

$\mathsf{Vf}[\mathsf{h}](vk, M, sig)$:
Return $\mathsf{e}(vk, \mathsf{h}(M)) = \mathsf{e}(g, sig)$
```

Figure 6: The protocol $\mathsf{BLS}[\mathbb{G}, \mathbb{G}_\mathsf{T}]$, where $\mathbb{G}$ is and $\mathbb{G}_\mathsf{T}$ are cyclic groups with prime order p and g is a generator of $\mathbb{G}$. Further, $\mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathsf{T}$ is a bilinear map. Moreover, ns is the number of parties, and t is the threshold parameter. The scheme is defined for any choice of $t \leq \mathsf{ns} \leq p-1$. Further, $\mathsf{h} : \{0,1\}^* \to \mathbb{G}$.

contribution is our proof of TS-UF-1 security. We note that TS-UF-1 security follows already from TS-UF-0 security with a loss of ns^{t-1} by Theorem 3.1. Here, however, we give a *tighter* reduction to a stronger assumption, the Vector CDH assumption, which we prove to hold in the GGM, with concrete hardness matching that of the original CDH assumption.

The scheme itself is described in Figure 6. As BLS is fully non-interactive, some of the algorithms in the scheme description are trivial. We rely on an efficiently computable bilinear map $\mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathsf{T}$, where $\mathbb{G}, \mathbb{G}_\mathsf{T}$ are both groups of order p, and, for a generator $g \in \mathbb{G}$ we have $\mathsf{e}(g^a, g^b) = \mathsf{e}(g, g)^{ab}$. (As in the original BLS proof [13], one can generalize these results to asymmetric case $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathsf{T}$ whenever an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ exists.) We remark that the security of BLS with aggregation is enhanced when the message is pre-pended with the public key prior to hashing [13, 4]. We accordingly recommend this for implementations but for simplicity have omitted it here.

THE VECTOR CDH ASSUMPTION. The *t-Vector Computational Diffie-Hellman* (t-VCDH) assumption is parameterized by an integer $t \geq 1$, and a group $\mathbb{G}$ of order p, with a generator g. It is described by the game $\mathbf{G}_\mathbb{G}^{t\text{-vcdh}}$ in Figure 7. It considers an adversary that is given multiple random group elements $\boldsymbol{X}[i] = g^{\boldsymbol{x}[i]}$ for $i \in [1..t]$, as well as a group element Y. The adversary can then issue queries $\textsc{Eval}(\boldsymbol{\alpha})$ for $\boldsymbol{\alpha} \in Z_p^t$ to obtain $Y^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$ for $\boldsymbol{\alpha} \in \mathbb{Z}_p^t$ of its choice, where $\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle = \sum_{i=1}^t \boldsymbol{\alpha}[i] \cdot \boldsymbol{\beta}[i]$ for any $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^t$. (Here, all operations are mod p.) The adversary wins if it outputs $\boldsymbol{\alpha}$ and $Z = Y^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$ for some $\boldsymbol{\alpha}$ which is not in the span of the prior queries. We denote by $\mathbf{Adv}_\mathbb{G}^{t\text{-vcdh}}(\mathcal{A})$ the corresponding advantage metric, which measures the probability of the game returning true. We also introduce the (conventional) CDH assumption in Figure 7, with the associated $\mathbf{Adv}_\mathbb{G}^{\text{cdh}}(\mathcal{A})$ advantage metric.

We study the t-VCDH assumption in the generic-group model (GGM) [37, 34], and show in Appedndix D that it is as hard as the Discrete Logarithm and the CDH problems in a prime order group, i.e., any attack succeeding with constant probability requires $\Omega(\sqrt{p})$ operations. In

```
Game $\mathbf{G}^{t\text{-vcdh}}_{\mathbb{G}}$

Init:
  $Y \leftarrow\!\!\$\ \mathbb{G}$; $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^t$
  For $i \in [1, t]$ do
      $\boldsymbol{X}[i] \leftarrow g^{\boldsymbol{x}[i]}$
  $\mathsf{VecSet} \leftarrow \emptyset$
  Return $(\boldsymbol{X}, Y)$
```

```
Eval($\boldsymbol{\alpha}$): // $\boldsymbol{\alpha} \in \mathbb{Z}_p^t$
  $\mathsf{VecSet} \leftarrow \mathsf{VecSet} \cup \{\boldsymbol{\alpha}\}$
  Return $Y^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$

Fin($Z, \boldsymbol{\alpha}$):
  Return $\boldsymbol{\alpha} \notin \mathsf{span}(\mathsf{VecSet})$
          $\wedge\ Z = Y^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$
```

```
Game $\mathbf{G}^{\mathrm{cdh}}_{\mathbb{G}}$

Init:
  $x \leftarrow\!\!\$\ \mathbb{Z}_p$; $X \leftarrow g^x$; $Y \leftarrow\!\!\$\ \mathbb{G}$
  Return $(X, Y)$

Fin($Z$):
  Return $Z = Y^x$
```

Figure 7: Games used to define the t-VCDH assumption (left and center columns), and the standard CDH assumption (right column). Here, $\mathbb{G}$ is a cyclic group of order p with generator g.

Appedndix F, we show that t-VCDH is also implied by CDH, in the standard model, for restricted classes of adversaries. Our GGM analysis suggests that this loose reduction is pessimistic.

TS-UF-1 Security of BLS. We then show the following theorem, which we prove in Appedndix E.

Theorem 4.1 (TS-UF-1 security of BLS) *For any TS-UF-1 adversary $\mathcal{A}$ making at most q_s queries to* PSignO *and at most q_h queries to* RO*, there exists a t-VCDH adversary $\mathcal{B}$, making at most $t-1$ queries to* Eval *such that*

$$\mathbf{Adv}^{\text{ts-uf-1}}_{\mathsf{BLS}[\mathbb{G},\mathbb{G}_T]}(\mathcal{A}) \leq (q_h + q_s) \cdot \mathbf{Adv}^{t\text{-vcdh}}_{\mathbb{G}}(\mathcal{B}) \ .$$

Moreover, $\mathcal{B}$ runs in time roughly equal that of $\mathcal{A}$, plus the time to perform at most $\mathsf{ns}^2 + (3 + q_s + q_h)\mathsf{ns}$ exponentiations and group operations.

For completeness, in Appedndix G, we give a simpler proof of TS-UF-0 security (which mirrors the analysis of [9]), which in turn gives us a looser version of the above theorem based on CDH alone. Alternatively, one can use a Lemma in Appedndix F to obtain a similar result. We also note that BLS does not achieve TS-UF-2 security (and, thus, any stronger notion), since the only part of a partial signature depending on SS is the Lagrange coefficient in the exponent, which is easily altered. This allows a malicious leader to combine partial signatures from distinct requests $lr \neq lr'$ with $lr.\mathsf{SS} \neq lr'.\mathsf{SS}$ but $lr.\mathsf{msg} = lr'.\mathsf{msg} = M$ to give a signature for M.

5 The Security of FROST

5.1 The FROST1 and FROST2 Schemes

Scheme descriptions. This section revisits the security of FROST, first proposed in [30] by Komlo and Goldberg, as a (partially) non-interactive threshold signature scheme. We consider both the original scheme, which we refer to as FROST1, as well as an optimized version, FROST2, from a recent follow-up work [16]. We give a detailed description of both schemes in Figure 8. The leader state st_0 contains a set curPP_i for each server i representing the set of tokens generated by server i that has not yet been used in a signing request. The state st_i for server i contains a function mapPP that maps each token pp to the randomness that is used to generate pp and $\mathsf{st}_i.\mathrm{mapPP}(pp) = \perp$ if pp is not generated by server i yet or has already been used in a signing request. The coefficient $\lambda_i^{lr.\mathsf{SS}}$ in line 39 is the Lagrange coefficient for the set $lr.\mathsf{SS}$, which is defined

```
Protocol $\boxed{\mathsf{FROST1}}$, $\boxed{\mathsf{FROST2}}[\mathbb{G}]$

$\mathsf{Kg}[\mathsf{h}]$:
For $i \in [0..t-1]$ do
    $a_i \leftarrow\!\!\$\ \mathbb{Z}_p$
For $i \in [1..\mathsf{ns}]$ do
    $sk_i \leftarrow\!\!\$\ \sum_{j=0}^{t-1} i^j \cdot a_j$ ; $vk_i \leftarrow g^{sk_i}$
$vk \leftarrow g^{a_0}$
$aux \leftarrow (vk_1, \ldots, vk_{\mathsf{ns}})$
Return $vk, aux, \{sk_i\}_{i\in[1..\mathsf{ns}]}$

$\mathsf{SPP}[\mathsf{h}](\mathsf{st}_i)$:
$r \leftarrow \mathbb{Z}_p$ ; $s \leftarrow \mathbb{Z}_p$
$pp \leftarrow (g^r, g^s)$
$\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (r, s)$
Return $(pp, \mathsf{st}_i)$

$\mathsf{LPP}[\mathsf{h}](i, pp, \mathsf{st}_0)$:
$\mathsf{st}_0.\mathrm{curPP}_i \leftarrow \mathsf{st}_0.\mathrm{curPP}_i \cup \{pp\}$
Return $\mathsf{st}_0$

$\mathsf{LR}[\mathsf{h}](M, SS, \mathsf{st}_0)$:
If $\exists\, i \in SS : \mathsf{st}_0.\mathrm{curPP}_i = \emptyset$ then
    Return $\bot$
$lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow SS$
For $i \in SS$ do
    Pick $pp_i$ from $\mathsf{st}_0.\mathrm{curPP}_i$
    $lr.\mathsf{PP}(i) \leftarrow pp_i$
    $\mathsf{st}_0.\mathrm{curPP}_i \leftarrow \mathsf{st}_0.\mathrm{curPP}_i \setminus \{pp_i\}$
Return $(lr, \mathsf{st}_0)$

$\mathsf{Vf}[\mathsf{h}](vk, M, sig)$:
$(R, z) \leftarrow sig$
$c \leftarrow \mathsf{h}_2(vk, M, R)$
Return $(g^z = R \cdot vk^c)$

$\mathsf{CompPar}[\mathsf{h}](vk, lr)$:
$M \leftarrow lr.\mathsf{msg}$
For $i \in lr.\mathsf{SS}$ do
    $\boxed{d_i \leftarrow \mathsf{h}_1(vk, lr, i)}$
    $\boxed{d_i \leftarrow \mathsf{h}_1(vk, lr)}$
    $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$
$R \leftarrow \prod_{i\in lr.\mathsf{SS}} R_i S_i^{d_i}$
$c \leftarrow \mathsf{h}_2(vk, M, R)$
Return $(R, c, \{d_i\}_{i\in lr.\mathsf{SS}})$

$\mathsf{PS}[\mathsf{h}](lr, i, \mathsf{st}_i)$:
$pp_i \leftarrow lr.\mathsf{PP}(i)$
If $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \bot$ then
    Return $(\bot, \mathsf{st}_i)$
$(r_i, s_i) \leftarrow \mathsf{st}_i.\mathrm{mapPP}(pp_i)$
$\mathsf{st}_i.\mathrm{mapPP}(pp_i) \leftarrow \bot$
$(R, c, \{d_j\}_{j\in lr.\mathsf{SS}})$
    $\leftarrow \mathsf{CompPar}[\mathsf{h}](\mathsf{st}_i.\mathsf{vk}, lr)$
$z_i \leftarrow r_i + d_i \cdot s_i + c \cdot \lambda_i^{lr.\mathsf{SS}} \cdot \mathsf{st}_i.\mathsf{sk}$
Return $((R, z_i), \mathsf{st}_i)$

$\mathsf{Agg}[\mathsf{h}](\mathrm{PS}, \mathsf{st}_0)$:
$R \leftarrow \bot$ ; $z \leftarrow 0$
For $(R', z') \in \mathrm{PS}$ do
    If $R = \bot$ then $R \leftarrow R'$
    If $R \neq R'$ then return $(\bot, \mathsf{st}_0)$
    $z \leftarrow z + z'$
Return $((R, z), \mathsf{st}_0)$

$\mathsf{SVf}[\mathsf{h}](vk, lr, sig)$:
$(R^*, z^*) \leftarrow sig$
$(R, c, \{d_j\}_{j\in lr.\mathsf{SS}})$
    $\leftarrow \mathsf{CompPar}[\mathsf{h}](vk, lr)$
Return $(R = R^*) \wedge (g^{z^*} = R \cdot vk^c)$
```

Figure 8: The protocol $\mathsf{FROST1}[\mathbb{G}]$ and $\mathsf{FROST2}[\mathbb{G}]$, where $\mathbb{G}$ is a cyclic group with prime order p and generator g. Further, ns is the number of parties, and t is the threshold of the schemes. We require $t \leq \mathsf{ns} \leq p-1$. The protocol $\mathsf{FROST1}$ contains all but the dashed box, and the protocol $\mathsf{FROST2}$ contains all but the solid box. The function $\mathsf{h}_i(\cdot)$ is computed as $\mathsf{h}(i, \cdot)$ for $i = 1, 2$. PS denotes a set of partial signatures.

(for any set $S \subseteq [1..\mathsf{ns}]$) as

$$\lambda_i^S := \prod_{j\in S, i\neq j} \frac{j}{j-i} .$$

The algorithm $\mathsf{CompPar}$ is a helper algorithm that computes the parameters $R, c, \{d_i\}_{i\in lr.\mathsf{SS}}$ used during signing. We stress that the only difference between $\mathsf{FROST1}$ and $\mathsf{FROST2}$ is the way d_i is computed in $\mathsf{CompPar}$. In $\mathsf{FROST1}$, each d_i is a different hash value for each server i, while in $\mathsf{FROST2}$, d_i's are the same hash value for all servers.

```
Games $\mathbf{G}^{\mathrm{omdl}}_{\mathbb{G}}$

INIT:
  $\mathrm{cid} \leftarrow 0;\ \ell \leftarrow 0;\ T \leftarrow ()$

CHAL():
  $\mathrm{cid} \leftarrow \mathrm{cid} + 1;\ x_{\mathrm{cid}} \leftarrow\!\!\$\ \mathbb{Z}_p$
  Return $g^{x_{\mathrm{cid}}}$

DLOG($X$):
  If $T(X) \neq \perp$ then return $T(X)$
  $\ell \leftarrow \ell + 1;\ T(X) \leftarrow \mathsf{DL}_{\mathbb{G},g}(X)$
  Return $T(X)$

FIN($\{y_i\}_{i\in[\mathrm{cid}]}$):
  If $\ell \geq \mathrm{cid}$ then return false
  If $\forall\ i \in [\mathrm{cid}] : y_i = x_i$ then
      Return true
  Return false
```

Figure 9: The OMDL game, where $\mathbb{G}$ is a cyclic group with prime order p and generator g.

It is not hard to verify that both schemes satisfy perfect correctness.

Overview of our results. Crites, Komlo, and Maller [16] argue that FROST2 improves the signing efficiency of FROST1 as the number of exponentiations for computing the nonce R is reduced from at least t to one, but they only consider TS-UF-0 security of FROST2. In this section, we strengthen their results (however, in a setting without distributed key generation) by showing FROST2 is actually TS-SUF-2-secure (under OMDL), but we also show it is not TS-UF-3 secure. In contrast, we show FROST1 is TS-SUF-3-secure but not TS-UF-4-secure. Theoretically, our results imply the separations between TS-(S)UF-2 and TS-(S)UF-3 and between TS-(S)UF-3 and TS-(S)UF-4. Practically speaking, our results indicate a separation between the security of FROST1 and FROST2. To complete the picture, a TS-SUF-4 secure variant of FROST1 can be obtained via the general transformation from Theorem 3.2, although it is an interesting open question whether a more efficient variant exists.

5.2 TS-SUF-2 Security of FROST2

We first show that FROST2 is TS-SUF-2-secure in the ROM under the OMDL assumption. The OMDL assumption, introduced in [5], is formally defined in Figure 9. Formally, we show the following theorem.

Theorem 5.1 *For any TS-SUF-2 adversary $\mathcal{A}$ making at most q_s queries to* PPO *and at most q_h queries to* RO*, there exists an OMDL adversary $\mathcal{B}$ making at most $2q_s + \mathsf{ns}$ queries to* CHAL *such that*

$$\mathbf{Adv}^{\mathrm{ts\text{-}suf\text{-}2}}_{\mathsf{FROST2}[\mathbb{G}]}(\mathcal{A}) \leq \sqrt{q \cdot (\mathbf{Adv}^{\mathrm{omdl}}_{\mathbb{G}}(\mathcal{B}) + 3q^2/p)}\ ,$$

where $q = q_s + q_h + 1$. Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$, plus the time to perform at most $(4\mathsf{ns} + 2) \cdot q + 2q_s + 2\mathsf{ns}^2$ exponentiations and group operations.

The previous analysis of FROST2 [16] can be seen as implying TS-SUF-0 security, either in the AGM or under non-standard assumptions (which are, in turn, validated in the AGM). Our result here proves stronger security, without relying on the AGM, but also without considering FROST's DKG. (We believe our analysis should extend, at least in the AGM, but we omit the added complexity of the DKG in this paper.) The core of the proof is a reduction from OMDL, which will need to use rewinding (via a variant of the Forking Lemma). The main challenge is to ensure that the reduction can simulate properly with a number of queries to DLOG which is smaller than the number of DL challenges. Further below, we are going to show that FROST2 is not TS-UF-3 secure, thus showing the above result is optimal with respect to our hierarchy.

```
$\mathsf{Fork}^{\mathcal{A}}(x)$:
Pick the random coin $\rho$ of $\mathcal{A}$ at random
$h_1, h'_1, \ldots, h_q, h'_q \leftarrow H$
$(I, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$
If $I = \perp$ then return $\perp$
$(I', \mathrm{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$
If $I \neq I'$ then return $\perp$
Return $(I, \mathrm{Out}, \mathrm{Out}')$
```

Figure 10: The forking algorithm build from $\mathcal{A}$.

Proof of of Theorem 5.1: Let $\mathcal{A}$ be an adversary as described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(M^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\mathsf{h}_2(vk, M^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\mathsf{h}_1(vk, lr)$ prior to the query $\mathrm{PSIGNO}(i, lr)$ for some i and lr. (This adds up to q_s additional RO queries, and we let $q = q_h + q_s + 1$.) Denote lr^* as the leader query such that $\mathsf{h}_1(vk, lr^*)$ is the first query prior to the query $\mathsf{h}_2(vk, M^*, R^*)$ satisfying $\mathsf{SVf}[\mathsf{h}](vk, lr^*, sig^*) = \mathsf{true}$. If such lr^* does not exists, lr^* is set to $\perp$. Denote the event E_1 as

$$\mathsf{Vf}[\mathsf{h}](vk, M^*, sig^*) \ \wedge\ (lr^* = \perp \ \vee\ \mathrm{S}_2(lr^*) < t - |CS|) .$$

It is clear that if $\mathcal{A}$ wins the game $\mathbf{G}^{\text{ts-suf-2}}_{\mathsf{FROST2}}$, then E_1 must occur, which implies $\Pr[E_1] \geq \mathbf{Adv}^{\text{ts-suf-2}}_{\mathsf{FROST2}[\mathbb{G}]}(\mathcal{A})$. Therefore, the theorem will follow from the following lemma. (We isolate this statement as its own lemma also because it will be helpful in the proof of Theorem 5.4 below.) ▌

Lemma 5.2 *There exists an OMDL adversary $\mathcal{B}$ making at most $2q_s + t$ queries to* CHAL *such that*

$$\Pr[E_1] \leq \sqrt{q \cdot (\mathbf{Adv}^{\mathrm{omdl}}_{\mathbb{G}}(\mathcal{B}) + 3q^2/p)} .$$

Moreover, $\mathcal{B}$ runs in time roughly twice that of $\mathcal{A}$, plus the time to perform at most $(4\mathsf{ns} + 2) \cdot q + 2q_s + 2\mathsf{ns}^2$ exponentiations and group operations.

Before turning to the proof of Lemma 5.2, we first introduce the following variant of the forking lemma that will be used within its proof.

Lemma 5.3 *Let $q \geq 1$ be an integer, $S \subseteq [1..q]$ be a set, and H be a set. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \ldots, h_q$ outputs a pair (I, Out), where $I \in \{\perp\} \cup S$ and* Out *is a side output. Let* IG *be a randomized algorithm that generates x. The accepting probability of $\mathcal{A}$ is defined as*

$$\mathrm{acc}(\mathcal{A}) = \Pr_{x \leftarrow\$ \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[(I, \mathrm{Out}) \leftarrow\$ \mathcal{A}(x, h_1, \ldots, h_q) \ :\ I \neq \perp] .$$

Consider algorithm $\mathsf{Fork}^{\mathcal{A}}$ described in Figure 10. The accepting probability of $\mathsf{Fork}^{\mathcal{A}}$ is defined as

$$\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) = \Pr_{x \leftarrow\$ \mathsf{IG}}[\alpha \leftarrow\$ \mathsf{Fork}^{\mathcal{A}}(x) \ :\ \alpha \neq \perp] .$$

Then, $\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) \geq \mathrm{acc}(\mathcal{A})^2/|S|$.

The above lemma slightly extends the generalized Forking Lemma of Bellare and Neven [6] in the sense that if $\mathcal{A}$ can only output index I within a given set $S \subseteq [1..q]$, then the final bound on $\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}})$ depends only on $|S|$ instead of q. The property allows us to get better bounds in our analysis. The proof (which is very similar to that of the Forking Lemma of [6]) is deferred to Appedndix H.1.

Proof of of Lemma 5.2: We first construct an algorithm $\mathcal{C}$ compatible with the syntax in Lemma 5.3. The input of $\mathcal{C}$ consists of $(2q_s+t)$ uniformly random group elements $A_0,\ldots,A_{t-1}$, U_1, V_1, $\ldots,U_{q_s},V_{q_s} \in \mathbb{G}$ and uniformly random integers $h_1,\ldots,h_{2q} \in \mathbb{Z}_p$. Also, $\mathcal{C}$ can access an oracle $\textsc{Dlog}$, which on input $X \in \mathbb{G}$ outputs $\mathsf{DL}_{\mathbb{G},g}(X)$. (We can think of this oracle as part of $\mathcal{C}$ in the context of the Forking Lemma, as $\mathcal{C}$ does not need to be efficient.) To start with, $\mathcal{C}$ initializes all the states $\mathsf{st}_0,\ldots,\mathsf{st}_{\mathsf{ns}}$. In addition, it initializes counters ctr_s,ctr_h to 0 and a function dt to an empty table, which are used to record the $\textsc{Dlog}$ query related to each (U_j,V_j). $\mathcal{C}$ also initializes $\mathrm{curLR} \leftarrow \emptyset$ to record all leader requests that appears during the game and initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. We also use a flag BadPPO to denote whether a bad event occurs, which are initially set to false. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\textsc{Init}}$, $\widetilde{\mathrm{PPO}}$, $\widetilde{\mathrm{P}\textsc{Sign}\mathrm{O}}$, $\widetilde{\mathrm{RO}}$, which are simulated as follows.

$\widetilde{\textsc{Init}}(CS)$**:** $\mathcal{C}$ initializes h to an empty table and sets $vk \leftarrow A_0$, $vk_i = \prod_{j=0}^{t-1} A_j^{i^j}$ for $i \in [1..\mathsf{ns}]$, and $sk_i = \textsc{Dlog}(vk_i)$ for $i \in CS$. Finally, $\mathcal{C}$ returns vk,$aux = (vk_1,\ldots,vk_{\mathsf{ns}})$, $\{sk_i\}_{i\in CS}$.

$\widetilde{\mathrm{RO}}$ **query** $\mathsf{h}_1(x)$**:** If $\mathsf{h}_1(x) \neq \bot$, $\mathcal{C}$ returns $\mathsf{h}_1(x)$. Otherwise, parse x as $(\widetilde{vk}, lr)$. If the parsing fails or $\widetilde{vk} \neq vk$, $\mathcal{C}$ sets $\mathsf{h}_1(x) \leftarrow\!\!\$\ \mathbb{Z}_p$ and returns $\mathsf{h}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1, sets $\mathsf{h}_1(x) \leftarrow h_{2\mathrm{ctr}_h-1}$, and adds lr to curLR. Also, $\mathcal{C}$ computes $R \leftarrow \prod_{i\in lr.\mathsf{SS}} R_i S_i^{h_{2\mathrm{ctr}_h-1}}$, where $(R_i,S_i) \leftarrow lr.\mathsf{PP}(i)$. If $\mathsf{h}_2(vk, lr.\mathsf{msg}, R) = \bot$, $\mathcal{C}$ sets $\mathsf{h}_2(vk, lr.\mathsf{msg}, R) = h_{2\mathrm{ctr}_h}$. In addition, define $\mathrm{mapLR}(\mathrm{ctr}_h) := lr$ and set $\mathrm{curLR} \leftarrow \mathrm{curLR} \cup \{lr\}$. Finally, $\mathcal{C}$ returns $\mathsf{h}_1(x)$.

$\widetilde{\mathrm{RO}}$ **query** $\mathsf{h}_2(x)$**:** If $\mathsf{h}_2(x) \neq \bot$, $\mathcal{C}$ returns $\mathsf{h}_2(x)$. Otherwise, parse x as $(\widetilde{vk}, M, R)$. If the parsing fails or $\widetilde{vk} \neq vk$, $\mathcal{C}$ sets $\mathsf{h}_2(x) \leftarrow\!\!\$\ \mathbb{Z}_p$ and returns $\mathsf{h}_2(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathsf{h}_2(x) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathsf{h}_2(x)$.

$\widetilde{\mathrm{PPO}}(i)$ **query:** Same as in the game $\mathbf{G}^{\text{ts-suf-2}}_{\mathsf{FROST2}}$, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$, $\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (0,0)$, and $\mathrm{ctrPP}(i,pp) \leftarrow \mathrm{ctr}_s$. In addition, BadPPO is set to true if there exists $lr \in \mathrm{curLR}$ such that $lr.\mathsf{PP}(i) = (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$.

$\widetilde{\mathrm{P}\textsc{Sign}\mathrm{O}}(i, lr)$ **query:** Same as in the game $\mathbf{G}^{\text{ts-suf-2}}_{\mathsf{FROST2}}$, except in the simulation of algorithm PS, if $\mathsf{st}_i.\mathrm{mapPP}(pp) \neq \bot$, $\mathcal{C}$ sets $z_i \leftarrow \textsc{Dlog}\left(U_j V_j^{d_i} vk_i^{c\lambda_i^{lr.\mathsf{SS}}}\right)$, where $j \leftarrow \mathrm{ctrPP}(i, lr.\mathsf{PP}(i))$. In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (i, k, d_i, c\lambda_i^{lr.\mathsf{SS}}, z_i)$, where k denotes the index such that $\mathsf{h}_1(vk, lr)$ is set to h_{2k-1} during the simulation.

After receiving the output $(M^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $\bot$ if $\mathsf{BadPPO} = \mathsf{true}$ or E_1 does not occur. Otherwise, $\mathcal{C}$ finds the index I such that $\mathsf{h}_2(vk, M^*, R^*)$ is set to h_I during the simulation. By our assumption of $\mathcal{A}$, we know such I must exist. Then, $\mathcal{C}$ returns (I, Out), where Out consists of all variables received or generated by $\mathcal{C}$.

<u>ANALYSIS OF $\mathcal{C}$</u> To use Lemma 5.3, we define $S := \{2k\}_{k\in[1..q]}$ and IG as the algorithm that samples $2q_s + t$ group elements uniformly from $\mathbb{G}$ and outputs them. From the simulation, we know the output index I of $\mathcal{C}$ is always in S. Also, it is clear that $\mathcal{C}$ simulates the game $\mathbf{G}^{\text{ts-suf-2}}_{\mathsf{FROST2}}$ perfectly when all the inputs of $\mathcal{C}$ are uniformly sampled from their domain, which implies $\mathrm{acc}(\mathcal{C}) \geq \Pr[E_1] - \Pr[\mathsf{BadPPO}]$, where $\Pr[E_1]$ refers to the probability in the original $\mathbf{G}^{\text{ts-suf-2}}_{\mathsf{FROST2}}$ game with $\mathcal{A}$ (as in

the lemma statement), whereas $\Pr[\mathsf{BadPPO}]$ is the probability that $\mathsf{BadPPO} = \mathsf{true}$ at the end of $\mathcal{C}$'s execution. Since every pair U_j, V_j is sampled uniformly from $\mathbb{G}$, for each $\mathrm{PPO}(i)$ query, the probability BadPPO is set to true is less than $|\mathrm{curLR}|/|G| \le q_h/p$. Therefore, we have $\Pr[\mathsf{BadPPO}] \le q_s q_h/p$. By Lemma 5.3,

$$\begin{aligned}\mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) &\ge (\Pr[E_1] - q_s q_h/p)^2/q \ge \Pr[E_1]^2/q - 2\Pr[E_1] q_s q_h/(p \cdot q)\\ &\ge \Pr[E_1]^2/q - 2q_s/p\ .\end{aligned}$$

<u>CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$</u> We now give a construct of the OMDL adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$, and the available DLOG oracle. To start with, $\mathcal{B}$ queries INIT and queries CHAL oracle $2q_s + t$ times to generate $A_0, \ldots, A_{t-1}, U_1, V_1, \ldots, U_{q_s}, V_{q_s}$ as the input of $\mathsf{Fork}^{\mathcal{C}}$ and runs $\mathsf{Fork}^{\mathcal{C}}$. Without loss of generality, we can assume all the OMDL challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. All DLOG queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$ to DLOG oracle of the game $\mathbf{G}^{\mathrm{omdl}}_{\mathbb{G}}$. Denote the event $\mathsf{BadHash}$ as any two of the scalars $h_1, h'_1, \ldots, h_q, h'_q$ generated in the execution of $\mathsf{Fork}^{\mathcal{C}}$ are same. Since $h_1, h'_1, \ldots, h_q, h'_q$ are sampled uniformly from $\mathbb{Z}_p$, we know $\Pr[\mathsf{BadHash}] \le 2q^2/p$.

It is left to show that if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, \mathrm{Out}, \mathrm{Out}')$ and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ can win the game $\mathbf{G}^{\mathrm{omdl}}_{\mathbb{G}}$, which implies

$$\mathbf{Adv}^{\mathrm{omdl}}_{\mathbb{G}}(\mathcal{B}) \ge \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) - \Pr[\mathsf{BadHash}] \ge \Pr[E_1]^2/q - 3q^2/p\ .$$

We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out'. By the execution of $\mathsf{Fork}^{\mathcal{C}}$, we know $(vk, M^*, R^*) = (vk', M^{*\prime}, R^{*\prime})$ and $vk = A_0$. Since $I \in S$, let $k^* = I/2$. It is not hard to see that $\mathrm{mapLR}(k^*) = lr^*$. (If $\mathrm{mapLR}(k^*) = \perp$, lr^* is also $\perp$.)

We first show how to compute the discrete log of $A_0, \ldots, A_{t-1}$. Denote the discrete log of $A_0, \ldots, A_{t-1}$ as $a_0, \ldots, a_{t-1}$ and define a polynomial $f(x) := \sum_{i=0}^{t-1} a_i x^i$. Since $\mathsf{BadHash}$ does not occur, we have $\mathsf{h}_2(vk, M^*, R^*) = h_I \ne h'_I = \mathsf{h}'_2(vk, M^*, R^*)$. Since $g^{z^*} = R^x A_0^{h_I}$, $g^{z^{*\prime}} = R^x A_0^{h'_I}$, $\mathcal{B}$ computes $f(0) = a_0 = \frac{z^* - z^{*\prime}}{h_I - h'_I}$. Define $T_{\mathsf{dt}} := \{\ j\ :\ (i, k, d, c, z) \leftarrow \mathsf{dt}(j), k = k^*\ \}$. For each $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, let $(i, k, d, c, z) \leftarrow \mathsf{dt}(j)$ and $(i', k', d', c', z') \leftarrow \mathsf{dt}'(j)$, and we have $g^z = U_j V_j^d vk_i^c$, $g^{z'} = U_j V_j^{d'} vk_{i'}^{c'}$. Since $\mathsf{BadPPO} = \mathsf{false}$ during both execution of $\mathcal{C}$, we know (U_j, V_j) is returned by a query $\mathrm{PPO}(i)$ prior to the query $\mathsf{h}_2(vk, M^*, R^*)$ during the first execution of $\mathcal{C}$. Since the two executions of $\mathcal{C}$ are exactly the same prior to the query $\mathsf{h}_2(vk, M^*, R^*)$, we know $i' = i$. Also, we know $d = h_k = h_{k^*} = h_{k'} = d'$. Therefore, $\mathcal{B}$ can compute $f(i) = \mathsf{DL}_{\mathbb{G},g}(vk_i) = \frac{z - z'}{c - c'}$. Denote $D := \{i\}_{j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}, (i,k,d,c,z) \leftarrow \mathsf{dt}(j)}$. Since E_1 occurs in the first execution of $\mathcal{C}$, we know $|T_{\mathsf{dt}}| = |\mathrm{S}_2(lr^*)| < t - |CS|$. Therefore, we know $|D| = |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| < t - |CS|$. Therefore, $\mathcal{B}$ can pick an arbitrary set $D' \in HS \setminus D$ with size $(t - |CS| - |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| - 1)$ and for each $i \in D'$, $\mathcal{B}$ queries DLOG oracle on vk_i. Therefore, $\mathcal{B}$ knows the value of $f(i)$ for $i \in CS \cup D \cup D' \cup \{0\}$. Since $|CS \cup D \cup D' \cup \{0\}| = t$, $\mathcal{B}$ can compute the value of $a_0, \ldots, a_{t-1}$ using Lagrange interpolation.

We now show how to compute the discrete log of $U_1, V_1, \ldots, U_{q_s}, V_{q_s}$. Denote their discrete log as $u_1, v_1, \ldots, u_{q_s}, v_{q_s}$. From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (i, k, d, c, z) \ne \perp$ if and only if $\mathcal{C}$ queries DLOG on $U_j V_j^d vk_i^c$. Therefore, denote $\mathrm{DLOG}(U_j V_j^d vk_i^c)$ as the DLOG query associated with $\mathsf{dt}(j)$. For each $j \in q_s$, there are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\perp$. In this case, $\mathcal{B}$ computes u_j, v_j by directly querying oracle $\mathrm{DLOG}(U_j)$ and $\mathrm{DLOG}(V_j)$.

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\perp$. Without loss of generality, assume $\mathsf{dt}(j) =$

(i, k, d, c, z), which implies $g^z = U_j V_j^d vk_i^c$. $\mathcal{B}$ computes v_j by directly querying oracle DLOG(V_j) and computes $u_j = z - d \cdot v_j - c \cdot f(i)$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\perp$ and we denote $(i, k, d, c, z) \leftarrow \mathsf{dt}(j)$ and $(i', k', d', c', z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $k \neq k'$ or $k = k' > k^*$. In this case, we know $d = h_k \neq h'_{k'} = d'$ and $g^z = U_j V_j^d vk_i^c$, $g^{z'} = U_j V_j^{d'} vk_{i'}^{c'}$. Therefore, $\mathcal{B}$ computes $v_j = \frac{z - c \cdot f(i) - z' + c' \cdot f(i')}{d - d'}$, $u_j = z - d \cdot v_j - c \cdot f(i)$.

Case 3: $k = k' = k^*$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1.

Case 4: $k = k' < k^*$. $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, in this case, we have $d = d'$ and $c = c'$. Therefore, $\mathcal{B}$ queries DLOG oracle once in order to simulate the DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$.

We now count the number of DLOG queries made by $\mathcal{B}$.

- $\mathcal{B}$ queries DLOG oracle $|CS|$ times queries for simulating query DLOG(vk_i) made by $\mathcal{C}$ for each $i \in CS$.
- $\mathcal{B}$ queries DLOG oracle $|D'|$ times queries for computing $a_0, \ldots, a_{t-1}$.
- For each $j \in q_s$, $\mathcal{B}$ queries DLOG twice for simulating query associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j, v_j in case 0, 1, 2, 4 and queries 3 times in case 3.

Since the condition of case 3 is equivalent to $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, the total number of DLOG queries made by $\mathcal{B}$ is equal to $2q_s + |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| + |CS| + |D'| = 2q_s + t - 1$. Therefore, $\mathcal{B}$ wins the game $\mathbf{G}_{\mathbb{G}}^{\mathrm{omdl}}$. ▌

5.3 Security of FROST1

In this section, we show that FROST1 is TS-SUF-3-secure in the ROM under the OMDL assumption. Formally, we show the following theorem.

Theorem 5.4 *For any TS-SUF-3 adversary $\mathcal{A}$ making at most q_s queries to* PPO *and at most q_h queries to* RO*, there exists an OMDL adversary $\mathcal{B}$ making at most $2q_s + t$ queries to* CHAL *such that*

$$\mathbf{Adv}^{\text{ts-suf-3}}_{\mathsf{FROST1}[\mathbb{G}]}(\mathcal{A}) \leq 4\mathsf{ns} \cdot q \cdot \sqrt{\mathbf{Adv}^{\mathrm{omdl}}_{\mathbb{G}}(\mathcal{B}) + 6q/p}\ ,$$

where $q = q_s + q_h + 1$. Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$, plus the time to perform at most $6\mathsf{ns} \cdot q + 4q_s + 2\mathsf{ns}^2$ exponentiations and group operations.

The proof here follows a similar pattern than that of Theorem 5.1, but will be more complex. In particular, the lesser tight bound is due to the fact that we need to consider an additional bad event, which we upper bound via a different reduction from OMDL. As we explain in detail below, this reduction will make use of a looser Forking Lemma, which is a variant of the "Local Forking Lemma" [3], which only resamples a single random oracle output when rewinding. The extra looseness is due to needing to ensure an extra condition when rewinding.

Proof of Theorem 5.4: Let $\mathcal{A}$ be the adversary described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(M^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\mathsf{h}_2(vk, M^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\mathsf{h}_1(vk, lr, i)$ prior to the query PSIGNO(i, lr) for some i and lr. (This adds up to q_s additional RO queries, and we let $q = q_h + q_s + 1$.) Denote lr^* as the leader query such that $\mathsf{h}_1(vk, lr^*, i)$ is the first RO query prior to the $\mathsf{h}_2(vk, M^*, R^*)$ query for some i satisfying $\mathsf{SVf}[\mathsf{h}](vk, lr^*, sig^*) = \mathsf{true}$. If such lr^* does not exist, lr^* is set to $\perp$. Denote the event E_1 as

$$\mathsf{Vf}[\mathsf{h}](vk, M^*, sig^*) \ \wedge \ (lr^* = \perp \ \vee \ \mathrm{S}_2(lr^*) < t - |CS|) \ .$$

Denote the event E_2 as

$$\mathsf{Vf}[\mathsf{h}](vk, M^*, sig^*) \ \wedge \ lr^* \neq \perp \ \wedge \ \mathrm{S}_2(lr^*) \neq \mathrm{S}_3(lr^*) \ .$$

If $\mathcal{A}$ wins the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST2}}$ and $lr^* \neq \perp$, we know either $\mathrm{S}_2(lr^*) < t - |CS|$ or $\mathrm{S}_2(lr^*) \neq \mathrm{S}_3(lr^*)$. Therefore, if $\mathcal{A}$ wins the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST2}}$, then either E_1 or E_2 occurs, which implies

$$\mathbf{Adv}^{\text{ts-suf-3}}_{\mathsf{FROST1}[\mathbb{G}]}(\mathcal{A}) \leq \Pr[E_1] + \Pr[E_2] \leq 2 \max\{\Pr[E_1], \Pr[E_2]\} \ .$$

Thus, we conclude the theorem with the following two lemmas.

Lemma 5.5 *There exists an OMDL adversary $\mathcal{B}$ making at most $2q_s + t$ queries to* Chal *such that*

$$\Pr[E_1] \leq \sqrt{q \cdot (\mathbf{Adv}^{\mathrm{omdl}}_{\mathbb{G}}(\mathcal{B}) + 3q^2(\mathsf{ns}+1)^2/p)} \ ,$$

Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$, plus the time to perform at most $6\mathsf{ns} \cdot q + 4q_s + 2\mathsf{ns}^2$ exponentiations and group operations.

Lemma 5.6 *There exists an OMDL adversary $\mathcal{B}$ making at most $2q_s$ queries to* Chal *such that*

$$\Pr[E_2] \leq \mathsf{ns} \cdot q\sqrt{2(\mathbf{Adv}^{\mathrm{omdl}}_{\mathbb{G}}(\mathcal{B}) + 1/p)} \ .$$

Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$, plus the time to perform at most $6\mathsf{ns} \cdot q + 4q_s + 2\mathsf{ns}^2$ exponentiations and group operations.

This completes the proof of the theorem, subject to proofs of the lemmas that we discuss next. ▌

The proof of Lemma 5.5 is almost the same as Lemma 5.2, so we omit the full proof. The only difference is that $\mathcal{C}$ takes as input $h_1, \ldots, h_{(\mathsf{ns}+1)q}$ in order to simulate all RO queries. For a RO query $\mathsf{h}_1(vk, lr, i)$, $\mathcal{C}$ first enumerates all $i' \in [\mathsf{ns}]$ and assigns $h_{(\mathrm{ctr}_h - 1)(\mathsf{ns}+1)+i'}$ to $\mathsf{h}_1(vk, lr, i')$. Then, $\mathcal{C}$ computes the nonce R for lr and assigns $h_{\mathrm{ctr}_h(\mathsf{ns}+1)}$ to $\mathsf{h}_2(vk, lr.\mathsf{msg}, R)$ if it is not assigned any value yet. Similarly, for a new RO query $\mathsf{h}_1(vk, M, R)$, its value is set to $h_{\mathrm{ctr}_h(\mathsf{ns}+1)}$. The rest follows by similar analysis.

To prove Lemma 5.6, we need the following variant of the forking lemma, which extends the local forking lemma of [3]. The difference is that forking is happening on two indices I, J (leading to I', J' in the forking) while in [3] there is a single I (and corresponding I' in the forking). The difference between a local forking ([3] and Lemma 9) versus classical ([6] and Lemma 5) is that in the former only one point is resampled in forking while in the latter it is all points following the fork. The proof is in Appedndix H.2.

Lemma 5.7 *Let $q \geq 1$ be an integer and H and Q be two sets. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \ldots, h_q$ outputs a tuple (I, J, Out), where $I \in \{\perp\} \cup [1..q]$, $J \in Q$, and* Out *is a side output. Let IG be a randomized algorithm that generates x. The accepting probability of $\mathcal{A}$ is defined as*

$$\mathrm{acc}(\mathcal{A}) := \Pr_{x \leftarrow\$ \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[(I, \mathrm{Out}) \leftarrow\$ \mathcal{A}(x, h_1, \ldots, h_q) \ : \ I \neq \perp] \ .$$

```
$\mathsf{Fork}_2^{\mathcal{A}}(x)$:
1 Pick the random coin $\rho$ of $\mathcal{A}$ at random
2 $h_1, \ldots, h_q \leftarrow H$
3 $(I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$
4 If $I = \perp$ then return $\perp$
5 $h'_I \leftarrow H$
6 $(I', J', \mathrm{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{I-1}, h'_I, h_{I+1}, \ldots, h_q; \rho)$
7 If $I \neq I'$ or $J \neq J'$ then return $\perp$
8 Return $(I, J, \mathrm{Out}, \mathrm{Out}')$
```

Figure 11: The forking algorithm build from $\mathcal{A}$.

Consider algorithm $\mathsf{Fork}_2^{\mathcal{A}}$ *described in Figure 11. The accepting probability of* $\mathsf{Fork}_2^{\mathcal{A}}$ *is defined as*

$$\mathrm{acc}(\mathsf{Fork}_2^{\mathcal{A}}) := \Pr_{x \leftarrow\$ \,\mathsf{IG}}[\alpha \leftarrow\$ \,\mathsf{Fork}^{\mathcal{A}}(x) \ : \ \alpha \neq \perp] \, .$$

Then, $\mathrm{acc}(\mathsf{Fork}_2^{\mathcal{A}}) \geq \mathrm{acc}(\mathcal{A})^2/(q \cdot |Q|)$.

Proof of of Lemma 5.6: We first construct an algorithm $\mathcal{C}$ following the syntax of the algorithm described in Lemma 5.7. The input of $\mathcal{C}$ consists of $2q_s$ uniformly random group elements $U_1, V_1, \ldots, U_{q_s}, V_{q_s} \in \mathbb{G}$ and uniformly random vectors $h_1, \ldots, h_{\mathsf{ns}\cdot q} \in (\mathbb{Z}_p)$. Similarly to the proof of Lemma 5.2, $\mathcal{C}$ can access $\textsc{Dlog}$ oracle and at the beginning, initializes all the states $\mathsf{st}_0, \ldots, \mathsf{st}_{\mathsf{ns}}$ as in the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST1}}$, and initializes the counters ctr_s,ctr_h to 0 and the function dt to an empty table. $\mathcal{C}$ also initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\textsc{Init}}, \widetilde{\mathrm{PPO}}, \widetilde{\mathrm{P}\textsc{Sign}\mathrm{O}}, \widetilde{\mathrm{RO}}$, which are simulated as follows. In the following description, we use i to denote the index of parties, j to denote the index of $U_1, V_1, \ldots, U_{q_s}, V_{q_s}$, and k to denote the index of $h_1, \ldots, h_{\mathsf{ns}\cdot q}$.

$\widetilde{\textsc{Init}}(CS)$**:** $\mathcal{C}$ initializes h to an empty table and samples $a_0, \ldots, a_{t-1}$ uniformly from $\mathbb{Z}_p$. Define $f(x) := \sum_{i=0}^{t-1} a_i x^i$. Then, $\mathcal{C}$ sets $vk \leftarrow g^{f(0)}$, $vk_i = g^{f(i)}$ for $i \in [1..\mathsf{ns}]$, and $sk_i \leftarrow f(i)$ for $i \in CS$. Finally, $\mathcal{C}$ returns vk,$aux = (vk_1, \ldots, vk_{\mathsf{ns}})$, $\{sk_i\}_{i \in CS}$.

$\widetilde{\mathrm{RO}}$ **query** $\mathsf{h}_1(x)$**:** If $\mathsf{h}_1(x) \neq \perp$, $\mathcal{C}$ returns $\mathsf{h}_1(x)$. Otherwise, $\mathcal{C}$ parses x as $(\widetilde{vk}, lr, \tilde{i})$ for some $\tilde{i} \in [1..\mathsf{ns}]$. If the parsing fails or $\widetilde{vk} \neq vk$, $\mathcal{C}$ sets $\mathsf{h}_1(x) \leftarrow\$ \,\mathbb{Z}_p$ and returns $\mathsf{h}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathsf{h}_1(vk, lr, i) \leftarrow h_{\mathsf{ns}(\mathrm{ctr}_h - 1)+i}$ for each $i \in [1..\mathsf{ns}]$. In addition, define $\mathrm{mapLR}(\mathrm{ctr}_h) := lr$. Then, $\mathcal{C}$ computes $R \leftarrow \prod_{i \in lr.\mathsf{SS}} R_i S_i^{d_i}$, where $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$ and $d_i = \mathsf{h}_1(vk, lr, i)$. If $\mathsf{h}_2(vk, lr.\mathsf{msg}, R) = \perp$, $\mathcal{C}$ sets $\mathsf{h}_2(vk, lr.\mathsf{msg}, R) \leftarrow\$ \,\mathbb{Z}_p$. Finally, $\mathcal{C}$ returns $\mathsf{h}_1(x)$.

$\widetilde{\mathrm{RO}}$ **query** $\mathsf{h}_2(x)$**:** If $\mathsf{h}_2(x) \neq \perp$, $\mathcal{C}$ returns $\mathsf{h}_2(x)$. Otherwise, $\mathcal{C}$ sets $\mathsf{h}_2(x) \leftarrow\$ \,\mathbb{Z}_p$ and returns $\mathsf{h}_2(x)$.

$\widetilde{\mathrm{PPO}}(i)$ **query:** Same as in the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST1}}$, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$, $\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (0, 0)$, and $\mathrm{ctrPP}(i, pp) \leftarrow \mathrm{ctr}_s$.

$\widetilde{\mathrm{P}\textsc{Sign}\mathrm{O}}(i, lr)$ **query:** Same as in the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST1}}$, except in the simulation of algorithm PS, if $\mathsf{st}_i.\mathrm{mapPP}(pp) \neq \perp$, $\mathcal{C}$ computes $z_i \leftarrow \textsc{Dlog}\left(U_j V_j^{d_i}\right) + c\lambda_i^{lr.SS} \cdot f(i)$, where $j \leftarrow \mathrm{ctrPP}(i, pp)$.

In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (k, d_i, z_i - c\lambda_i^{lr.\mathsf{SS}} \cdot f(i))$, where k denotes the index such that $\mathsf{h}_1(vk, lr, i)$ is set to h_k during the simulation.

After receiving the output $(M^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $(\bot, \bot, \bot)$ if E_2 does not occur. Otherwise, we know $\mathrm{S}_2(lr^*) > 0$ and $\mathrm{S}_2(lr^*) \neq \mathrm{S}_3(lr^*)$. Therefore, there exists k^* and i^* such that $\mathrm{mapLR}(k^*) = lr^*$ and $i^* \in \mathrm{S}_3(lr^*) \setminus \mathrm{S}_2(lr^*)$. (Since $\mathrm{S}_2(lr^*) \subseteq \mathrm{S}_3(lr^*)$, we must have $\mathrm{S}_3(lr^*) \setminus \mathrm{S}_2(lr^*) \neq \emptyset$.) Since $i^* \in \mathrm{S}_3(lr^*)$, there exists $j^* \in [1..q_s]$ such that $lr^*.\mathsf{PP}(i^*) = (U_{j^*}, V_{j^*})$. If $\mathsf{dt}(j^*) = \bot$, $\mathcal{C}$ sets $J \leftarrow \bot$. Otherwise, let $(k, d, z) \leftarrow \mathsf{dt}(j^*)$ and $\mathcal{C}$ sets $J = k$. Then, $\mathcal{C}$ returns $(\mathsf{ns}(k^* - 1) + i^*, J, \mathrm{Out})$, where Out consists of all variables received or generated by $\mathcal{C}$, including i^*, j^*, k^*, lr^*.

<u>ANALYSIS OF $\mathcal{C}$</u> To use Lemma 5.7, we define IG as the algorithm that samples $2q_s$ group elements uniformly from $\mathbb{G}$ and outputs them. The output J is either $\bot$ or in $[1..(\mathsf{ns} \cdot q)]$. It is not hard to see that $\mathcal{C}$ simulates the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST1}}$ perfectly when all the inputs of $\mathcal{C}$ are uniformly sampled from their domain, which implies $\mathrm{acc}(\mathcal{C}) \geq \Pr[E_2]$, where $\Pr[E_2]$ refers to the probability in the original $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{FROST1}}$ game with $\mathcal{A}$ (as in the lemma statement). By Lemma 5.7,

$$\mathrm{acc}(\mathsf{Fork}_2^{\mathcal{C}}) \geq \frac{\Pr[E_2]^2}{\mathsf{ns} \cdot q(\mathsf{ns} \cdot q + 1)} \leq \frac{\Pr[E_2]^2}{2\mathsf{ns}^2 q^2} \ .$$

<u>CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$</u> We now give a construct of the OMDL adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$. To start with, $\mathcal{B}$ queries INIT and queries CHAL oracle $2q_s$ times to generate U_1,V_1,...,U_{q_s},V_{q_s} as the input of $\mathsf{Fork}^{\mathcal{C}}$ and runs $\mathsf{Fork}^{\mathcal{C}}$. Without loss of generality, we can assume all the OMDL challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. All DLOG queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$ to DLOG oracle of the game $\mathbf{G}^{\text{omdl}}_{\mathbb{G}}$. Denote the event $\mathsf{BadHash}$ as $h_I \neq h'_I$, where I are outputted by the first execution of $\mathcal{C}$. Since h_I, I are independent of h'_I, we know $\Pr[\mathsf{BadHash}] \leq 1/p$.

It is left to show that if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, J, \mathrm{Out}, \mathrm{Out}')$ and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ can win the game $\mathbf{G}^{\text{omdl}}_{\mathbb{G}}$, which implies

$$\mathbf{Adv}^{\text{omdl}}_{\mathbb{G}}(\mathcal{B}) \geq \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) - \Pr[\mathsf{BadHash}] \geq \frac{\Pr[E_2]^2}{2\mathsf{ns}^2 q^2} - 1/p \ .$$

We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out'. We first show how to compute the discrete log of U_j, V_j for $j \neq j^*$. Denote u_j, v_j as the discrete log of U_j, V_j. There are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\bot$. In this case, $\mathcal{B}$ computes u_j, v_j by directly querying oracle DLOG(U_j) and DLOG(V_j).

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\bot$. Without loss of generality, assume $\mathsf{dt}(j) = (k, d, z)$, which implies $g^z = U_j V_j^d$. $\mathcal{B}$ computes v_j by directly querying oracle DLOG(V_j) and computes $u_j = z - d \cdot v_j$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\bot$ and we denote $(k, d, z) \leftarrow \mathsf{dt}(j)$ and $(k', d', z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $d \neq d'$. In this case, $\mathcal{B}$ computes $v_j = \frac{z - z'}{d - d'}$, $u_j = z - d \cdot v_j$.

Case 3: $d = d'$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, since $d = d'$, $\mathcal{B}$ queries DLOG oracle only once in order to answer queries DLOG$(U_j V_j^d)$ and DLOG$(U_j V_j^{d'})$ from $\mathsf{Fork}^{\mathcal{C}}$.

```
Adversary $\mathcal{A}^{\text{INIT},\text{PPO},\text{PSIGNO},\text{RO}}$:
$CS \leftarrow \{3,4\}$ ; $(vk, aux, \{sk_3, sk_4\}) \leftarrow\$ \text{INIT}(CS)$
$(R_1, S_1) \leftarrow\$ \text{PPO}(1)$ ; $(R_2, S_2) \leftarrow\$ \text{PPO}(2)$ ; $\gamma \leftarrow \lambda_1^{\{1,3,4\}}/\lambda_1^{\{1,2,3\}}$
$lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow \{1,2,3\}$
$lr.\mathsf{PP}(1) \leftarrow (R_1, S_1)$ ; $lr.\mathsf{PP}(2) \leftarrow (R_2, S_2)$
$lr.\mathsf{PP}(3) \leftarrow (R_1^{\gamma-1}R_2^{-1}, S_1^{\gamma-1}S_2^{-1})$
$z_1 \leftarrow \text{PSIGNO}(1, lr)$
$d \leftarrow \text{RO}(1, vk, lr)$ ; $R \leftarrow R_1^{\gamma}S_1^{\gamma\cdot d}$ ; $c \leftarrow \text{RO}(2, vk, R, M)$
$z \leftarrow \gamma \cdot z_1 + c(\lambda_3^{\{1,3,4\}} \cdot sk_3 + \lambda_4^{\{1,3,4\}} \cdot sk_4)$
Return $(M, (R, z))$
```

Figure 12: Adversary $\mathcal{A}$ that wins the game $\mathbf{G}^{\text{ts-uf-3}}_{\mathsf{FROST2}}$, where M is a fixed message.

From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (k, d, z) \neq \perp$ if and only if $\mathcal{C}$ queries DLOG on $(U_j V_j^d)$. Therefore, denote $\text{DLOG}(U_j V_j^d)$ as the DLOG query associated with $\mathsf{dt}(j)$. For all the above cases, $\mathcal{B}$ queries DLOG oracle twice for simulating DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j, v_j.

We now show how to compute u_{j^*} and v_{j^*}. From the execution of $\mathsf{Fork}_2^{\mathcal{C}}$, we know $vk = vk'$ and $\text{mapLR}(k) = \text{mapLR}'(k)$ for all $k \leq I$, which implies $lr^* = \text{mapLR}(I) = \text{mapLR}'(I) = lr^{*\prime}$. Since E_2 occurs in both executions of $\mathcal{C}$, we know $\mathsf{SVf}(vk, lr^*, (R^*, z^*)) = \mathsf{true}$ and $\mathsf{SVf}(vk, lr^*, (R^{*\prime}, z^{*\prime})) = \mathsf{true}$ are valid. Therefore, $g^{z^*} = R^* g^{a_0 c}$, $R^* = \sum_{i \in lr^*.\mathsf{SS}} R_i S_i^{d_i}$, $g^{z^{*\prime}} = R^{*\prime} g^{a_0 c'}$, $R^{*\prime} = \sum_{i \in lr^*.\mathsf{SS}} R_i S_i^{d'_i}$, where $(R_i, S_i) = lr.\mathsf{PP}(i)$, $c = \mathsf{h}_2(vk, M^*, R^*)$, $c' = \mathsf{h}'_2(vk, M^*, R^{*\prime})$, and $d_i = \mathsf{h}_1(vk, lr^*, i)$, $d'_i = \mathsf{h}'_1(vk, lr^*, i)$. Since for each $i \neq i^*$ we have $d_i = h_{\mathsf{ns}(k^*-1)+i} = d'_i$, we have $g^{z^* - z^{*\prime}} = \frac{R^*}{R^{*\prime}} g^{a_0(c-c')} = S_{i^*}^{d_{i^*} - d'_{i^*}} g^{a_0(c-c')}$. Therefore, $\mathcal{C}$ can compute $v_{j^*} = \frac{z^* - z^{*\prime} - a_0(c-c')}{d_{i^*} - d'_{i^*}}$. If $J = \perp$, $\mathcal{B}$ computes u_{j^*} by querying $\text{DLOG}(U_{j^*})$ directly. In this case, $\mathcal{B}$ queries DLOG only once to compute u_{j^*} and v_{j^*}. If $J \neq \perp$, let $(k, d, z) \leftarrow \mathsf{dt}(j^*) =$ and $(k', d', z') \leftarrow \mathsf{dt}(j^*)$. Then, $\mathcal{B}$ computes $u_{j^*} = z - d \cdot v_{j^*}$. Since $i^* \notin S_2(lr^*)$, we know $k \neq I$.(Otherwise, suppose $k = I$. Since $I = \mathsf{ns}(k^* - 1) + i^*$ and $\text{mapLR}(k^*) = lr^*$, we know a $\text{PSIGNO}(i^*, lr^*)$ is made and does not return $\perp$ during the simulation, which implies $i^* \in S_2(lr^*)$.) Thus, we have $k' = J = k \neq I$ and $d = h_J = d'$, which means $\mathcal{B}$ only needs to query DLOG once to simulate the DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$. Therefore, the total number of DLOG queries made by $\mathcal{B}$ is equal to $2q_s - 1$, which implies $\mathcal{B}$ wins the game $\mathbf{G}^{\text{omdl}}_{\mathbb{G}}$. ▌

5.4 Attacks for FROST1 and FROST2

<u>FROST2 IS NOT TS-UF-3 SECURE</u> Consider the setting where $\mathsf{ns} = 4$ and $t = 3$ and the adversary $\mathcal{A}$ for the game $\mathbf{G}^{\text{ts-uf-3}}_{\mathsf{FROST2}}$ described in Figure 12. We now show that $\mathbf{Adv}^{\text{ts-uf-3}}_{\mathsf{FROST2}}(\mathcal{A}) = 1$. From the execution of PSIGNO, we know $g^{z_1} = R_1 S_1^d vk_1^{\lambda_1^{\{1,2,3\}} \cdot c}$. Therefore,

$$g^z = R_1^{\gamma} S_1^{d \cdot \gamma} vk_1^{\gamma \cdot \lambda_1^{\{1,2,3\}} \cdot c} vk_3^{\lambda_3^{\{1,3,4\}} \cdot c} vk_4^{\lambda_4^{\{1,3,4\}} \cdot c}$$

$$= R g^{c \cdot \sum_{i \in \{1,3,4\}} \lambda_i^{\{1,3,4\}} \cdot sk_i} = R \cdot vk^c \,,$$

which implies $(M, (R, z))$ is valid for vk. Also, it is clear that $\text{S}_2(lr) = \{1\}$ and $\text{S}_3(lr) = \{1, 2\}$, which implies the condition $\mathtt{tf}_3(lr)$ does not hold. Therefore, $\mathcal{A}$ wins the game $\mathbf{G}^{\text{ts-uf-3}}_{\mathsf{FROST2}}$ with

```
Adversary $\mathcal{A}^{\text{INIT},\text{PPO},\text{PSIGNO},\text{RO}}$:
$CS \leftarrow \{5,10\}$ ; $(vk, aux, \{sk_5, sk_{10}\}) \leftarrow\$ \text{INIT}(CS)$
$(R_1, S_1) \leftarrow\$ \text{PPO}(11)$ ; $s_2, r_2, s_3, r_3 \leftarrow\$ \mathbb{Z}_p$
$lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow \{11, 15, 20\}$
$lr.\mathsf{PP}(11) \leftarrow (R_1, S_1)$ ; $lr.\mathsf{PP}(15) \leftarrow (g^{r_2}, g^{s_2})$ ; $lr.\mathsf{PP}(20) \leftarrow (g^{r_3}, g^{s_3})$
$z_1 \leftarrow \text{PSIGNO}(11, lr)$
For $i \in \{11, 15, 20\}$ do $d_i \leftarrow \text{RO}(1, vk, lr, i)$
$R \leftarrow R_1 S_1^{d_{11}} g^{r_2+r_3+s_2 \cdot d_{15}+s_3 \cdot d_{20}}$ ; $c \leftarrow \text{RO}(2, vk, R, M)$
$z \leftarrow z_1 + r_2 + r_3 + s_2 \cdot d_{15} + s_3 \cdot d_{20} + c(\lambda_5^{\{5,10,11\}} \cdot sk_5 + \lambda_{10}^{\{5,10,11\}} \cdot sk_{10})$
Return $(M, (R, z))$
```

Figure 13: Adversary $\mathcal{A}$ that wins the game $\mathbf{G}^{\text{ts-uf-4}}_{\mathsf{FROST1}}$, where M is a fixed message.

probability 1.

<u>FROST1 IS NOT TS-UF-4 SECURE</u> Consider the setting where $\mathsf{ns} = 20$ and $t = 3$ and the adversary $\mathcal{A}$ for the game $\mathbf{G}^{\text{ts-uf-4}}_{\mathsf{FROST1}}$ described in Figure 13. We now show that $\mathbf{Adv}^{\text{ts-uf-4}}_{\mathsf{FROST1}}(\mathcal{A}) = 1$. From the execution of PSIGNO, we know $g^{z_1} = R_1 S_1^{d_{11}} vk_{11}^{\lambda_{11}^{\{11,15,20\}} \cdot c}$. The key observation here is that $\lambda_{11}^{\{11,15,20\}} = \frac{15 \cdot 20}{(15-11)(20-11)} = \frac{25}{3} = \frac{5 \cdot 10}{(5-11)(10-11)} = \lambda_{11}^{\{5,10,11\}}$. Therefore,

$$\begin{aligned} g^z &= R_1 S_1^{d_{11}} g^{r_2+r_3+s_2 \cdot d_{15}+s_3 \cdot d_{20}} vk_{11}^{\lambda_{11}^{\{11,15,20\}} \cdot c} vk_5^{\lambda_5^{\{5,10,11\}} \cdot c} vk_{10}^{\lambda_{10}^{\{5,10,11\}} \cdot c} \\ &= R g^{c \cdot \sum_{i \in \{5,10,11\}} \lambda_i^{\{5,10,11\}} \cdot sk_i} = R \cdot vk^c \ , \end{aligned}$$

which implies $(M, (R, z))$ is valid for vk. Also, it is clear that $\mathrm{S}_2(lr) = \{11\}$ and $\mathrm{S}_4(lr) = \{11, 15, 20\}$, which implies the condition $\mathtt{tf}_4(lr)$ does not hold. Therefore, $\mathcal{A}$ wins the game $\mathbf{G}^{\text{ts-uf-4}}_{\mathsf{FROST1}}$ with probability 1.

The reason why the attack is possible for $\mathsf{FROST1}$ is because the honest server 11 replies to the leader request lr with tokens $lr.\mathsf{PP}(15)$ and $lr.\mathsf{PP}(20)$ not generated by the honest servers 15 and 20 but by the adversary instead. Therefore, the attack is prevented by the general transformation from TS-SUF-3 security to TS-SUF-4 security described in Fig. 4 since after the transformation an honest server replies to a leader request only when all the tokens within the request are authenticated by the corresponding servers, and it is not possible for the adversary to generate authenticated tokens on behalf of honest servers.

Acknowledgments

We thank the Crypto 2022 referees for their detailed and constructive feedback on the paper, which we have implemented to the best of our ability.

References

[1] M. Bellare, E. Crites, C. Komlo, M. Maller, S. Tessaro, and C. Zhu. Better than advertised security for non-interactive threshold signatures. In Y. Dodis and T. Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, 2022, Proceedings*, volume ? of *Lecture Notes in Computer Science*, page ? Springer, 2022. 7

[2] M. Bellare and W. Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In K. Bhargavan, E. Oswald, and M. Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 529–552. Springer, Heidelberg, Dec. 2020. 36

[3] M. Bellare, W. Dai, and L. Li. The local forking lemma and its application to deterministic encryption. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 607–636. Springer, Heidelberg, Dec. 2019. 22, 23

[4] M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Heidelberg, July 2007. 15

[5] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. 5, 18

[6] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, Oct. / Nov. 2006. 19, 20, 23

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 5

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 7, 38

[9] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, Jan. 2003. 3, 4, 10, 14, 16

[10] D. Boneh, R. Gennaro, and S. Goldfeder. Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In T. Lange and O. Dunkelman, editors, *LATINCRYPT 2017*, volume 11368 of *LNCS*, pages 352–377. Springer, Heidelberg, Sept. 2017. 3

[11] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, Aug. 2018. 5

[12] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, Dec. 2001. 3

[13] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, Sept. 2004. 3, 4, 14, 15

[14] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, Nov. 2020. 3, 7

[15] E. Crites, C. Komlo, and M. Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021. https://eprint.iacr.org/2021/1375. 7

[16] E. Crites, C. Komlo, and M. Maller. How to prove schnorr assuming schnorr: Security of multi-and threshold signatures. *Cryptology ePrint Archive*, 2021. 3, 5, 16, 18

[17] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994. 3

[18] Y. Desmedt. Society and group oriented cryptography: A new concept. In C. Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer, Heidelberg, Aug. 1988. 3

[19] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, Aug. 1990. 3

[20] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, Aug. 2018. 5

[21] R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 1179–1194. ACM Press, Oct. 2018. 3

[22] R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 156–174. Springer, Heidelberg, June 2016. 3

[23] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 354–371. Springer, Heidelberg, May 1996. 3, 4, 10

[24] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure applications of Pedersen's distributed key generation protocol. In M. Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 373–390. Springer, Heidelberg, Apr. 2003. 3

[25] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, Jan. 2007. 3, 4, 6, 10

[26] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and efficient sharing of RSA functions. *Journal of Cryptology*, 13(2):273–300, Mar. 2000. 3, 4

[27] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988. 3, 4, 10

[28] J. Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339, 2021. https://eprint.iacr.org/2021/339. 5, 7

[29] J. Katz and M. Yung. Threshold cryptosystems based on factoring. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 192–205. Springer, Heidelberg, Dec. 2002. 5

[30] C. Komlo and I. Goldberg. Frost: flexible round-optimized schnorr threshold signatures. In *International Conference on Selected Areas in Cryptography*, pages 34–65. Springer, 2020. 3, 4, 5, 7, 16

[31] C. Komlo, I. Goldberg, and T. Wilson-Brown. Two-Round Threshold Signatures with FROST. Internet-Draft draft-irtf-cfrg-frost-01, Internet Engineering Task Force, Aug. 2021. Work in Progress. 3

[32] B. Libert, M. Joye, and M. Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In M. M. Halldórsson and S. Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014. 5

[33] Y. Lindell, A. Nof, and S. Ranellucci. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. Cryptology ePrint Archive, Report 2018/987, 2018. https://eprint.iacr.org/2018/987. 3

[34] U. M. Maurer. Abstract models of computation in cryptography (invited paper). In N. P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, Dec. 2005. 15, 36

[35] National Institute of Standards and Technology. Multi-Party Threshold Cryptography, 2018–Present. https://csrc.nist.gov/Projects/threshold-cryptography. 3

[36] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, Aug. 1992. 6

[37] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 15, 36

[38] V. Shoup. Practical threshold signatures. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, Heidelberg, May 2000. 3, 5

[39] D. R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In V. Varadharajan and Y. Mu, editors, *ACISP 01*, volume 2119 of *LNCS*, pages 417–434. Springer, Heidelberg, July 2001. 3

[40] H. Wee. Threshold and revocation cryptosystems via extractable hash proofs. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 589–609. Springer, Heidelberg, May 2011. 5

[41] A. Yun. Generic hardness of the multiple discrete logarithm problem. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 817–836. Springer, Heidelberg, Apr. 2015. 36

A Reference schemes and proofs of relations between notions

Fix t, ns such that $2 \leq t < \mathsf{ns}$. The reference schemes are shown in Figure 14.

Proposition A.1 Suppose DS is a signature scheme and $2 \leq t < \mathsf{ns}$. Let $\mathsf{RTS}_\ell[\mathsf{DS}]$ $(\ell = 1, 2, 3, 4)$ be the reference threshold schemes defined in Figure 14. Then: **(1)** For all $\ell \in \{1, 2, 3, 4\}$, if DS is UF-CMA-secure then $\mathsf{RTS}_\ell[\mathsf{DS}]$ is TS-UF-ℓ-secure, and **(2)** For all $\ell \in \{2, 3, 4\}$, if DS is unique and SUF-CMA-secure then $\mathsf{RTS}_\ell[\mathsf{DS}]$ is TS-SUF-ℓ-secure.

The proof of the above proposition is rather straightforward. We now use these schemes to prove the separations claimed by the dotted arrows in Figure 3.

TS-UF-1 $\not\Rightarrow$ TS-UF-2. We need to exhibit a scheme TS that is TS-UF-1-secure but not TS-UF-2-secure. Let $\mathsf{TS} = \mathsf{RTS}_1[\mathsf{DS}]$ (Figure 14) where DS is a UF-CMA-secure standard signature scheme. Proposition A.1 says this achieves TS-UF-1. We now give an attack showing it fails TS-UF-2. The idea is that the adversary can make partial-signing requests to t servers, all with the same message but with $lr.\mathsf{SS}$, and thus lr itself, varying across the requests, so that $S_2(lr)$ stays small for any particular lr, and non-triviality under $\mathtt{tf}_2$ is maintained. Proceeding to the details of the attack, fix a message M, and consider the following adversary $\mathcal{A}$ for game $\mathbf{G}^{\text{ts-uf-2}}_{\mathsf{TS},t}$:

```
Adversary $\mathcal{A}$
1. $CS \leftarrow \emptyset$ ; $(vk, aux, \emptyset) \leftarrow\$ \mathsf{Init}(CS)$
2. $lr_1.\mathsf{msg} \leftarrow M$ ; $lr_1.\mathsf{SS} \leftarrow [1..t]$ ; $lr_2.\mathsf{msg} \leftarrow M$ ; $lr_2.\mathsf{SS} \leftarrow [2..t+1]$
3. For $i \in [1..t-1]$ do $psig_i \leftarrow\$ \text{PSIGNO}(i, lr_1)$
4. $psig_t \leftarrow\$ \text{PSIGNO}(t, lr_2)$
5. $sig \leftarrow (lr_1, [1..t], \{psig_i\}_{i\in[1..t]})$ ; $\text{FIN}(M, sig)$
```

We claim that $\mathbf{Adv}^{\text{ts-uf-2}}_{\mathsf{TS},t}(\mathcal{A}) = 1$. The adversary has gathered t valid signatures, so the condition at line 18 of Figure 14 is met, and $\mathsf{TS.Vf}(vk, M, sig) = \mathsf{true}$. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tf}_2(lr_1) = \mathtt{tf}_2(lr_2) = \mathsf{false}$. Indeed, we have $|\mathrm{S}_2(lr_1)| = t - 1 < t - |CS| = t$ and $|\mathrm{S}_2(lr_1)| = 1 < t - |CS| = t$.

TS-SUF-2 $\not\Rightarrow$ TS-UF-3. We need to exhibit a scheme TS that is TS-SUF-2-secure but not TS-UF-3-secure. Let $\mathsf{TS} = \mathsf{RTS}_2[\mathsf{DS}]$ where DS is a unique SUF-CMA-secure standard signature scheme. Proposition A.1 says this achieves TS-SUF-2. We now give an attack showing it fails TS-UF-3. The idea is that the adversary can set $lr.\mathsf{PP}$ values as it wishes, and in particular different from the honest ones, so that $\mathrm{S}_3(lr)$ is empty. Proceeding to the details of the attack, fix a message M, and consider the following adversary $\mathcal{A}$ for game $\mathbf{G}^{\text{ts-uf-3}}_{\mathsf{TS},t}$:

```
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{Kg}:}$
1  For $i = 1, \ldots, \mathsf{ns}$ do $(vk_i, sk_i) \leftarrow\!\!\$\ \mathsf{DS.Kg}$
2  $vk \leftarrow (vk_1, \ldots, vk_{\mathsf{ns}})$ ; Return $(vk, \varepsilon, sk_1, \ldots, sk_{\mathsf{ns}})$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{SPP}(\mathsf{st}):}$
3  Return $(\varepsilon, \mathsf{st})$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{LPP}(pp, \mathsf{st}_0):}$
4  Return $\mathsf{st}_0$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{LR}(M, SS, \mathsf{st}_0):}$
5  $lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow SS$
6  For $i \in lr.\mathsf{SS}$ do $lr.\mathsf{PP}(i) \leftarrow \varepsilon$  // $\ell = 2, 3, 4$
7  Return $(lr, \mathsf{st}_0)$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{PS}(lr, \mathsf{st}):}$
8  If $(\mathsf{st.me} \notin lr.\mathsf{SS}$ or $|lr.\mathsf{SS}| < t)$ then return $(\bot, \mathsf{st})$
9  If $(lr.\mathsf{PP}(\mathsf{st.me}) \neq \varepsilon)$ then return $(\bot, \mathsf{st})$  // $\ell = 3, 4$
10 $psig \leftarrow\!\!\$\ \mathsf{DS.Sig}(\mathsf{st.sk}, lr.\mathsf{msg})$  // $\ell = 1$
11 $psig \leftarrow\!\!\$\ \mathsf{DS.Sig}(\mathsf{st.sk}, lr)$  // $\ell = 2, 3, 4$
12 Return $(psig, \mathsf{st})$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{Agg}(lr, \{psig_i\}_{i \in lr.\mathsf{SS}}, \mathsf{st}_0):}$
13 $sig \leftarrow (lr, lr.\mathsf{SS}, \{psig_i\}_{i \in lr.\mathsf{SS}})$ ; Return $(sig, \mathsf{st}_0)$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{Vf}(vk, M, sig):}$
14 $(vk_1, \ldots, vk_{\mathsf{ns}}) \leftarrow vk$ ; $(lr, F, \{psig_i\}_{i \in F}) \leftarrow sig$
15 If $(lr.\mathsf{msg} \neq M$ or $F \not\subseteq lr.\mathsf{SS})$ then return $\mathsf{false}$
16 $T \leftarrow \{\, i \in F \,:\, \mathsf{DS.Vf}(vk_i, M, psig_i) \,\}$  // $\ell = 1$
17 $T \leftarrow \{\, i \in F \,:\, \mathsf{DS.Vf}(vk_i, lr, psig_i) \,\}$  // $\ell = 2, 3, 4$
18 Return $(T = F$ and $|T| \geq t)$  // $\ell = 1, 2$
19 $E \leftarrow \{\, i \in lr.\mathsf{SS} \,:\, lr.\mathsf{PP}(i) = \varepsilon \,\}$ ; Return $(T = E = F$ and $|T| \geq t)$  // $\ell = 3$
20 Return $(T = lr.\mathsf{SS} = F$ and $|T| \geq t)$  // $\ell = 4$
$\underline{\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{SVf}(vk, lr, sig):}$  // $\ell = 2, 3, 4$
21 $(lr', F, \{psig_i\}_{i \in F}) \leftarrow sig$
22 Return $(\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{Vf}(vk, lr.\mathsf{msg}, sig)$ and $lr = lr')$
```

Figure 14: Reference threshold signature schemes $\mathsf{RTS}_\ell[\mathsf{DS}]$ associated to signature scheme DS for $\ell = 1, 2, 3, 4$.

<u>Adversary $\mathcal{A}$</u>

```
1. $CS \leftarrow \emptyset$ ; $(vk, aux, \emptyset) \leftarrow\!\!\$\ \mathsf{Init}(CS)$
2. $lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow [1..t]$ ; For $i \in [1..t]$ do $lr.\mathsf{PP}(i) \leftarrow 0$
3. For $i \in [1..t]$ do $psig_i \leftarrow\!\!\$\ \textsc{PSignO}(i, lr)$
4. $sig \leftarrow (lr, [1..t], \{psig_i\}_{i \in [1..t]})$ ; $\textsc{Fin}(M, sig)$
```

We claim that $\mathbf{Adv}^{\text{ts-uf-3}}_{\mathsf{TS},t}(\mathcal{A}) = 1$. The adversary has gathered t valid signatures, so the condition at line 18 of Figure 14 is met, and $\mathsf{TS.Vf}(vk, M, sig) = \mathsf{true}$. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tf}_3(lr) = \mathsf{false}$. Indeed, we have $\mathrm{S}_2(lr) = [1..t]$ but $\mathrm{S}_3(lr) = \emptyset$.

<u>TS-SUF-3 $\not\Rightarrow$ TS-UF-4.</u> We need to exhibit a scheme TS that is TS-SUF-3-secure but not TS-UF-4-secure. Let $\mathsf{TS} = \mathsf{RTS}_3[\mathsf{DS}]$ where DS is a unique SUF-CMA-secure standard signature scheme. Proposition A.1 says this achieves TS-SUF-3. We now give an attack showing it fails TS-UF-4.

Letting E be the set of all $i \in \mathit{lr}.\mathsf{SS}$ such that $\mathit{lr}.\mathsf{PP}(i)$ is correct, meaning equals ε, the idea is that the adversary can let E be a strict subset of $\mathit{lr}.\mathsf{SS}$ and then pass verification using only signatures from E. Proceeding to the details of the attack, fix a message M, and consider the following adversary $\mathcal{A}$ for game $\mathbf{G}^{\text{ts-uf-4}}_{\mathsf{TS},t}$:

```
Adversary $\mathcal{A}$
1. $CS \leftarrow \emptyset$ ; $(vk, aux, \emptyset) \leftarrow\$ \mathsf{Init}(CS)$
2. $lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow [1..t+1]$
3. $lr.\mathsf{PP}(t+1) \leftarrow 0$ ; For $i \in [1..t]$ do $lr.\mathsf{PP}(i) \leftarrow \varepsilon$
4. For $i \in [1..t]$ do $psig_i \leftarrow\$ \text{PSIGNO}(i, lr)$
5. $sig \leftarrow (lr, [1..t], \{psig_i\}_{i\in[1..t]})$ ; $\text{FIN}(M, sig)$
```

We claim that $\mathbf{Adv}^{\text{ts-uf-4}}_{\mathsf{TS},t}(\mathcal{A}) = 1$. At line 19 of Figure 14 we have $E = [1..t]$ and thus line 19 returns true, so $\mathsf{TS.Vf}(vk, M, sig) = \mathsf{true}$. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tf}_4(\mathit{lr}) = \mathsf{false}$. Indeed, we have $\mathrm{S}_2(\mathit{lr}) = [1..t]$ but $\mathrm{S}_4(\mathit{lr}) = [1..t+1]$.

TS-UF-4 $\not\Rightarrow$ TS-SUF-2. We need to exhibit a scheme TS that is TS-UF-4-secure but not TS-SUF-2-secure. First, if $x \in \{0,1\}^*$, it is convenient to let $\mathrm{pre}(x)$ be the first bit of x and $\mathrm{suff}(x)$ the rest. Now, let DS^* be a unique SUF-CMA-secure standard signature scheme, and modify it to a scheme DS as follows: Let $\mathsf{DS.Sign}(\cdot,\cdot) \leftarrow 0\|\mathsf{DS}^*.\mathsf{Sign}(\cdot,\cdot)$ and let $\mathsf{DS.Vf}(\cdot,\cdot,\mathit{psig}) \leftarrow \mathsf{DS}^*.\mathsf{Vf}(\cdot,\cdot,\mathrm{suff}(\mathit{psig}))$. Then DS is UF-CMA-secure, but, since flipping the first bit of psig does not affect its validity, not SUF-CMA-secure. Now, consider $\mathsf{RTS}_4[\mathsf{DS}]$, and, for it, an alternative verification algorithm $\mathsf{RTS}_4[\mathsf{DS}].\mathsf{Vf}'$ that is as in Figure 14 except that line 17 is changed to $T \leftarrow \{\, i \in F : \mathsf{DS.Vf}(\mathit{vk}_i, \mathit{lr}, \mathit{psig}_i) \text{ and } \mathrm{pre}(\mathit{psig}_i) = 0 \,\}$. Let TS be the same as $\mathsf{RTS}_4[\mathsf{DS}]$ except that we change SVf as follows: at line 22 of Figure 14, invoke $\mathsf{RTS}_4[\mathsf{DS}].\mathsf{Vf}'$ rather than $\mathsf{RTS}_4[\mathsf{DS}].\mathsf{Vf}$. (Note that $\mathsf{TS.Vf}$ stays as in $\mathsf{RTS}_4[\mathsf{DS}]$ as shown in Figure 14. The modified verification algorithm is only used by $\mathsf{TS.SVf}$. Also note the latter meets the required condition of accepting at most one signature per key and message, due to the uniqueness of DS^*.) Proposition A.1 says $\mathsf{RTS}_4[\mathsf{DS}]$ is TS-UF-4-secure, and hence so is TS, because the only difference between these two is in SVf and TS-UF-4 does not depend on this. We now give an attack showing TS fails TS-SUF-2. The idea is to exploit lack of SUF-CMA-security of the base scheme DS. Proceeding to the details of the attack, fix a message M, and consider the following adversary $\mathcal{A}$ for game $\mathbf{G}^{\text{ts-suf-2}}_{\mathsf{TS},t}$, where $\mathrm{flip1}(x)$ returns string x with its first bit flipped:

```
Adversary $\mathcal{A}$
1. $CS \leftarrow \emptyset$ ; $(vk, aux, \emptyset) \leftarrow\$ \mathsf{Init}(CS)$
2. $lr.\mathsf{msg} \leftarrow M$ ; $lr.\mathsf{SS} \leftarrow [1..t]$ ; For $i \in [1..t]$ do $lr.\mathsf{PP}(i) \leftarrow \varepsilon$
3. For $i \in [1..t]$ do $psig_i^* \leftarrow\$ \text{PSIGNO}(i, lr)$ ; $psig_i \leftarrow \mathrm{flip1}(psig_i^*)$
4. $sig \leftarrow (lr, [1..t], \{psig_i\}_{i\in[1..t]})$ ; $\text{FIN}(M, sig)$
```

We claim that $\mathbf{Adv}^{\text{ts-suf-2}}_{\mathsf{TS},t}(\mathcal{A}) = 1$. Line 18 of Figure 14 returns true, so $\mathsf{TS.Vf}(vk, M, sig) = \mathsf{true}$. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tsf}_2(\mathit{lr}, \mathit{vk}, \mathit{sig}) = \mathsf{false}$. Indeed, $\mathsf{TS.SVf}(\mathit{vk}, \mathit{lr}, \mathit{sig}) = \mathsf{false}$ because the first bit of psig_i is 1 for all $i \in [1..t]$.

B Proof of Theorem 3.1

Proof of of Theorem 3.1: We describe a construction of the adversary $\mathcal{B}$ as follows. $\mathcal{B}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}, \widetilde{\text{PPO}}, \widetilde{\text{PSIGNO}}, \widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$**:** $\mathcal{B}$ randomly samples a set $ECS \in [1..\mathsf{ns}] \setminus CS$ of size $t - |CS| - 1$ and makes an oracle query $\text{INIT}(CS \cup ECS)$ in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$. After receiving $vk, aux, \{sk_i\}_{i \in CS \cup ECS}$, $\mathcal{B}$ sets $\mathsf{st}_i.\mathsf{sk} \leftarrow sk_i$, $\mathsf{st}_i.\mathsf{vk} \leftarrow vk$, and $\mathsf{st}_i.\mathsf{aux} \leftarrow aux$ for all $i \in ECS$. Finally, $\mathcal{B}$ returns $vk, aux, \{sk_i\}_{i \in CS}$.

$\widetilde{\text{PPO}}(i)$ **query:** Same as in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-1}}$, except when $i \in [1..\mathsf{ns}] \setminus (CS \cup ECS)$, $\mathcal{B}$ directly relays the query to oracle PPO in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$.

$\widetilde{\text{PSIGNO}}(i, lr)$ **query:** Same as in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-1}}$, except when $i \in [1..\mathsf{ns}] \setminus (CS \cup ECS)$ directly relays the query to oracle PSIGNO in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$. In addition, denote $\widetilde{\text{L}}$ and $\widetilde{\text{S}}_1$ as L and S_1 defined in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-1}}$. $\mathcal{B}$ also updates the set $\widetilde{\text{L}}$ and $\widetilde{\text{S}}_1(lr.\mathsf{msg})$ the same as in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-1}}$.

$\widetilde{\text{RO}}(x)$ **query:** $\mathcal{B}$ directly relays the query to oracle RO in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$.

After receiving the output (M^*, sig^*) from $\mathcal{A}$, denote the event $\mathsf{GoodECS}$ as $\widetilde{\text{S}}_1(M^*) \subseteq ECS$. If $\mathcal{A}$ wins the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-1}}$ and $\mathsf{GoodECS}$ occurs, $\mathcal{B}$ returns (M^*, sig^*). Otherwise, $\mathcal{B}$ aborts.

Denote the event WIN as $\mathcal{A}$ wins the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-1}}$ simulated by $\mathcal{B}$. We first show $\mathcal{B}$ wins the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$ if $\mathsf{WIN} \wedge \mathsf{GoodECS}$ occurs. From the simulation, we know $\text{S}_1(M^*) = \widetilde{\text{S}}_1(M^*) \setminus ECS$, where $\text{S}_1(M^*)$ is defined in the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$. Since $\mathsf{WIN} \wedge \mathsf{GoodECS}$ occurs, we know (M^*, sig^*) is valid for the public key vk and $\widetilde{\text{S}}_1(M^*) = \emptyset$, which implies $\mathcal{B}$ wins the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-uf-0}}$.

Therefore, it is left to show that $\Pr[\mathsf{WIN} \wedge \mathsf{GoodECS}] \geq \frac{1}{\binom{\mathsf{ns}}{t-1}} \mathbf{Adv}_{\mathsf{TS}}^{\text{ts-uf-1}}(\mathcal{A})$. We first fix a set $S \in [1..\mathsf{ns}]$ with size less than t and consider the case when $CS = S$. If WIN occurs, we know $|\widetilde{\text{S}}_1(M^*)| < t - |CS|$. Since $\mathcal{B}$ perfectly simulates the game $\mathbf{G}_{\mathsf{TS}}^{\text{ts-suf-0}}$ no matter which ECS is picked, we know the set $\widetilde{\text{S}}_1(M^*)$ is independent of the choice of ECS, which implies

$$\begin{aligned}\Pr\left[\,\mathsf{GoodECS} \mid \mathsf{WIN} \wedge CS = S\,\right] &= \Pr\left[\,ECS \in \widetilde{\text{S}}_1(M^*) \mid \mathsf{WIN} \wedge CS = S\,\right] \\ &\geq \frac{1}{\binom{\mathsf{ns}-|S|}{t-1-|S|}} \geq \frac{1}{\binom{\mathsf{ns}}{t-1}} \,.\end{aligned}$$

Therefore,

$$\begin{aligned}\Pr[\mathsf{GoodECS}] &= \sum_{\substack{S \subseteq [1..\mathsf{ns}],\\ |S|<t}} \Pr\left[\,\mathsf{GoodECS} \mid \mathsf{WIN} \wedge CS = S\,\right] \cdot \Pr\left[\,\mathsf{WIN} \mid CS = S\,\right] \\ &\geq \sum_{\substack{S \subseteq [1..\mathsf{ns}],\\ |S|<t}} \frac{1}{\binom{\mathsf{ns}}{t-1}} \Pr\left[\,\mathsf{WIN} \mid CS = S\,\right] \\ &= \frac{1}{\binom{\mathsf{ns}}{t-1}} \Pr[\mathsf{WIN}] = \frac{1}{\binom{\mathsf{ns}}{t-1}} \mathbf{Adv}_{\mathsf{TS}}^{\text{ts-uf-1}}(\mathcal{A}) \,.\end{aligned}$$

▌

C Proof of Theorem 3.2

Proof: This proof only deals with TS-SUF-4 security, but a similar proof also works for TS-UF-4 security.

```
$\underline{\mathcal{B}^{\text{INIT},\text{PPO},\text{PSIGNO},\text{RO}}():}$
 1 For $i \in [1..\mathsf{ns}]$ do
 2     $(svk_i, ssk_i) \leftarrow \mathsf{DS.Kg}$
 3 $(M, sig) \leftarrow \mathcal{A}^{\widetilde{\text{INIT}},\widetilde{\text{PPO}},\widetilde{\text{PSIGNO}},\widetilde{\text{RO}}}()$
 4 If $\mathsf{WIN} \wedge (\neg\mathsf{BadLR})$ occurs then
 5     Return $(M, sig)$
 6 Else abort

$\underline{\widetilde{\text{INIT}}(CS):}$
 7 $vk, taux, \{tsk_i\}_{i \in CS} \leftarrow\$ \text{INIT}(CS)$
 8 For $i \in CS$ do
 9     $sk_i \leftarrow (tsk_i, ssk_i)$
10 $aux \leftarrow (taux, svk_1, \ldots, svk_{\mathsf{ns}})$
11 Return $vk, aux, \{sk_i\}_{i \in CS}$

$\underline{\widetilde{\text{PPO}}(i):}$
12 $tpp \leftarrow \text{PPO}(i)$
13 $tsig \leftarrow\$ \mathsf{DS.Sig}(ssk_i, pp)$
14 Return $(tpp, tsig)$

$\underline{\widetilde{\text{PSIGNO}}(i, lr):}$
15 For $i \in lr.\mathsf{SS}$ do
16     $(pp_i, tsig_i) \leftarrow lr.\mathsf{PP}(i)$
17     If $\mathsf{DS.Vf}(svk_i, pp_i, tsig_i) = \mathsf{false}$ then
18         Return $\perp$
19 Return $\text{PSIGNO}(i, \mathsf{OriginLR}(lr))$

$\underline{\widetilde{\text{RO}}(x):}$
20 Return $\text{RO}(x)$
```

Figure 15: Adversary $\mathcal{B}$ for the proof of Lemma C.1. $\mathcal{B}$ also compute the sets L, PP, and $\text{S}_2(lr)$, $\text{S}_3(lr)$, $\text{S}_4(lr)$ for each $lr \in \text{L}$ following the same logic as in the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$ and thus can check whether the event $\mathsf{WIN} \wedge (\neg\mathsf{BadLR})$ occurs.

Let $\mathcal{A}$ be the adversary described in the theorem. After $\mathcal{A}$ returns, denote the event BadLR as there exists lr such that $\text{S}_2(lr) > 0$ and $\text{S}_3(lr) \neq \text{S}_4(lr)$. Denote the event WIN as $\mathcal{A}$ wins the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$. Then, we have

$$\mathbf{Adv}^{\text{ts-suf-4}}_{\mathsf{ATS}}(\mathcal{A}) \leq \Pr[\mathsf{WIN} \wedge (\neg\mathsf{BadLR})] + \Pr[\mathsf{BadLR}] \ .$$

Thus, we can conclude the theorem with the following two lemmas.

Lemma C.1 *There exists a TS-XX-3 adversary $\mathcal{B}$ making at most q_{s1} queries to* PPO*, at most q_{s2} queries to* PSIGNO*, and at most q_h queries to* RO *such that*

$$\Pr[\mathsf{WIN} \wedge (\neg\mathsf{BadLR})] \leq \mathbf{Adv}^{\text{ts-suf-3}}_{\mathsf{TS}}(\mathcal{B}) \ .$$

Moreover, $\mathcal{B}$ runs in time roughly equal that of $\mathcal{A}$

Lemma C.2 *There exists a SUF-CMA adversary $\mathcal{C}$ making at most q_{s1} queries to* SIGNO *such that*

$$\Pr[\mathsf{BadLR}] \leq \mathsf{ns} \cdot \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{DS}}(\mathcal{C}) \ ,$$

Moreover, $\mathcal{C}$ runs in time roughly equal that of $\mathcal{A}$

■

Proof of of Lemma C.1: We give a construction of the adversary $\mathcal{B}$ in Fig. 15, where $\mathcal{B}$ runs $\mathcal{A}$ by simulating the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$. The simulation is done simply by relaying all queries from $\mathcal{A}$ to the oracles in the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{TS}}$ and doing the extra authentication parts by $\mathcal{B}$ itself. It is clear that $\mathcal{B}$ simulates the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$ perfectly, which implies the probability that $\mathcal{B}$ does not abort is equal to $\Pr[\mathsf{WIN} \wedge (\neg\mathsf{BadLR})]$.

Therefore, it is left to show that if $\mathcal{B}$ does not abort, then $\mathcal{B}$ wins the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{TS}}$. Suppose $\mathsf{WIN} \wedge (\neg\mathsf{BadLR})$ occurs in the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$ simulated by $\mathcal{B}$. We use $\widetilde{\text{L}}$, $\widetilde{\text{PP}}$, $\widetilde{\text{S}}_2$, $\widetilde{\text{S}}_3$, and $\widetilde{\text{S}}_4$ to denote the variables L, PP, S_2, S_3, and S_4 in the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{TS}}$. From the simulation, we know $\widetilde{\mathrm{L}} = \{\mathsf{OriginLR}(lr)\}_{lr\in\mathrm{L}}$. For any $lr \in \mathrm{L}$, denote $\widetilde{lr} = \mathsf{OriginLR}(lr)$, and we have $lr.\mathsf{msg} = \widetilde{lr}.\mathsf{msg}$ and $\mathsf{TS.SVf}[\mathsf{h}](vk, \widetilde{lr}, sig) = \mathsf{true}$ if and only if $\mathsf{ATS.SVf}[\mathsf{h}](vk, lr, sig) = \mathsf{true}$. Since the output (M, sig) must be valid for the public key vk due to WIN occurs, to show $\mathcal{B}$ wins the game $\mathbf{G}^{\text{ts-suf-3}}_{\mathsf{TS}}$, we just need to show for any $lr \in \mathrm{L}$ such that $lr.\mathsf{msg} = M$ and $\mathsf{ATS.SVf}[\mathsf{h}](vk, lr, sig) = \mathsf{true}$, it holds that

$$(|\widetilde{S}_2(\widetilde{lr})| < t - |CS|) \ \vee\ (\widetilde{S}_2(\widetilde{lr}) \neq \widetilde{S}_3(\widetilde{lr}))\ . \tag{1}$$

Since WIN occurs, we know either $|S_2(lr)| < t - |CS|$ or $S_2(lr) \neq S_4(lr)$. If $|S_2(lr)| < t - |CS|$, from the simulation, we know $|\widetilde{S}_2(\widetilde{lr})| = |S_2(lr)| < t - |CS|$, which implies (1) holds. Otherwise, we have $|S_2(lr)| \geq t - |CS| > 0$ and $S_2(lr) \neq S_4(lr)$. Since BadLR does not occur, we have $S_3(lr) = S_4(lr) = HS \cap lr.\mathsf{SS}$. Therefore, for any $i \in HS \cap lr.\mathsf{SS}$, it holds that $lr.\mathsf{PP}(i) \in \mathrm{PP}_i$, which implies $\widetilde{lr}.\mathsf{PP}(i) \in \widetilde{\mathrm{PP}}_i$. Since $\widetilde{lr}.\mathsf{SS} = lr.\mathsf{SS}$, we have $\widetilde{S}_3(\widetilde{lr}) = HS \cap lr.\mathsf{SS}$. Therefore, we have $\widetilde{S}_2(\widetilde{lr}) = S_2(lr) \neq S_4(lr) = HS \cap lr.\mathsf{SS} = \widetilde{S}_3(\widetilde{lr})$, which implies (1) holds. ▮

Proof of of Lemma C.2: We describe a construction of the adversary $\mathcal{C}$ as follows. To start with, $\mathcal{C}$ queries $\textsc{Init}()$ oracle and receives svk^*. Also, $\mathcal{C}$ initializes all the states $\mathsf{st}_0, \ldots, \mathsf{st}_{\mathsf{ns}}$. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\textsc{Init}}, \widetilde{\mathrm{PPO}}, \widetilde{\mathrm{P}\textsc{Sign}\mathrm{O}}, \widetilde{\mathrm{RO}}$, which are simulated as follows.

$\widetilde{\textsc{Init}}(CS)$**:** Same as in the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$, except $\mathcal{C}$ additionally randomly picks an index $i^* \in HS$ and sets $(svk_{i^*}, ssk_{i^*}) \leftarrow\$ (svk^*, \bot)$ instead of generating them by $\mathsf{DS.Kg}$. Also, $\mathcal{C}$ initializes h to an empty table.

$\widetilde{\mathrm{PPO}}(i)$ **query:** Same as in the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$, except when $i = i^*$, in the execution of $\mathsf{SPP}[\mathsf{h}](\mathsf{st}_{i^*})$, $\mathcal{C}$ computes $tsig \leftarrow\$ \textsc{Sign}\mathrm{O}(tpp)$ instead of generating it by $\mathsf{DS.Sig}$.

$\widetilde{\mathrm{P}\textsc{Sign}\mathrm{O}}(i, lr)$ **query:** Same as in the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$.

$\widetilde{\mathrm{RO}}(x)$ **query:** If $\mathsf{h}(x) \neq \bot$, $\mathcal{C}$ returns $\mathsf{h}(x)$. Otherwise, $\mathcal{C}$ sets $\mathsf{h}(x) \leftarrow\$ \mathbb{Z}_p$ and returns $\mathsf{h}(x)$.

After receiving the output from $\mathcal{A}$, denote the event $\mathsf{GoodInd}$ as there exists $lr^* \in \mathrm{L}$ such that $S_2(lr^*) > 0$ and $i^* \in S_4(lr^*) \setminus S_3(lr^*)$. If $\mathsf{GoodInd}$ does not occur, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $(tpp_{i^*}, tsig_{i^*})$, where $(tpp_{i^*}, tsig_{i^*}) \leftarrow lr^*.\mathsf{PP}(i^*)$.

We first show that $\mathcal{B}$ wins the game $\mathbf{G}^{\text{suf-cma}}_{\mathsf{DS}}$ if $\mathsf{GoodInd}$ occurs. Since $S_2(lr^*) > 0$, we know for all $i \in lr^*.\mathsf{SS}$, $\mathsf{DS.Vf}(svk_i, tpp_i, tsig_i) = \mathsf{true}$ where $(tpp_i, tsig_i) \leftarrow lr^*.\mathsf{PP}(i)$. Since $i^* \in S_4(lr^*) \setminus S_3(lr^*)$, we have $(tpp_{i^*}, tsig_{i^*}) \notin \mathrm{PP}_{i^*}$. From the simulation, we know $\mathrm{PP}_{i^*} = Q$, where Q is defined in the game $\mathbf{G}^{\text{suf-cma}}_{\mathsf{DS}}$. Therefore, we know $(tpp_{i^*}, tsig_{i^*})$ is valid for $svk^* = svk_{i^*}$ and $(tpp_{i^*}, tsig_{i^*}) \notin Q$, which implies $\mathcal{B}$ wins the game $\mathbf{G}^{\text{suf-cma}}_{\mathsf{DS}}$.

It is left show that $\Pr[\mathsf{GoodInd}] \geq \frac{1}{\mathsf{ns}} \Pr[\mathsf{BadLR}]$. We first fix a set $S \in [1..\mathsf{ns}]$ with size less than t and consider the case when $CS = S$. If BadLR occurs, then there exists $\widetilde{lr} \in \mathrm{L}$ such that $S_2(\widetilde{lr}) > 0$ and $S_4(\widetilde{lr}) \neq S_3(\widetilde{lr})$, which implies $S_4(\widetilde{lr}) \setminus S_3(\widetilde{lr}) \neq \emptyset$. [1] Since $\mathcal{C}$ perfectly simulates the game $\mathbf{G}^{\text{ts-suf-4}}_{\mathsf{ATS}}$ no matter which i^* is picked, we know the set $S_3(\widetilde{lr}) \setminus S_2(\widetilde{lr})$ is independent of the choice of i^*, which implies

$$\begin{aligned}\Pr\left[\,\mathsf{GoodInd} \,|\, \mathsf{BadLR}\ \wedge\ CS = S\,\right] &= \Pr\left[\,i^* \in S_3(\widetilde{lr}) \setminus S_2(\widetilde{lr}) \,|\, \mathsf{BadLR}\ \wedge\ CS = S\,\right] \\ &\geq \frac{1}{\mathsf{ns} - |S|} \geq \frac{1}{\mathsf{ns}}\ .\end{aligned}$$

[1]Since $S_3(\widetilde{lr}) \subseteq S_4(\widetilde{lr})$, we must have $S_3(\widetilde{lr}) \setminus S_2(\widetilde{lr}) \neq \emptyset$ if $S_4(\widetilde{lr}) \neq S_3(\widetilde{lr})$.

```
Game $\mathbf{G}^{\text{gg-}t\text{-vcdh}}_G$   // Set $G$ is the range of the encoding, $|G|$ is prime.

INIT():
  $p \leftarrow |G|$ ; $\mathrm{E} \leftarrow\$ \text{Bijections}(\mathbb{Z}_p, G)$   // Think " $\mathrm{E}(x) = g^x$ "
  $\mathbb{1} \leftarrow \mathrm{E}(0)$ ; $g \leftarrow \mathrm{E}(1)$   // Think $\mathbb{1}$ the identity and $g$ generator
  $y \leftarrow\$ \mathbb{Z}_p$ ; $Y \leftarrow \mathrm{E}(y)$
  For $i = 1, \ldots, t$ do $\boldsymbol{x}[i] \leftarrow\$ \mathbb{Z}_p$ ; $\boldsymbol{X}[i] \leftarrow \mathrm{E}(\boldsymbol{x}[i])$
  $\mathrm{GL} \leftarrow \{\mathbb{1}, g, Y, \boldsymbol{X}[1], \ldots, \boldsymbol{X}[t]\}$
  Return $\mathbb{1}, g, Y, \boldsymbol{X}[1], \ldots, \boldsymbol{X}[t]$

OP($A, B, \mathsf{sgn}$):   // $A, B \in G$ and $\mathsf{sgn} \in \{+, -\}$
  If ($A \notin \mathrm{GL}$ or $B \notin \mathrm{GL}$) then return $\perp$
  $c \leftarrow (\mathrm{E}^{-1}(A)\ \mathsf{sgn}\ \mathrm{E}^{-1}(B)) \bmod p$ ; $C \leftarrow \mathrm{E}(c)$ ; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{C\}$
  Return $C$

EVAL($\boldsymbol{\alpha}$):   // $\boldsymbol{\alpha} \in \mathbb{Z}_p^t$
  $\mathsf{VecSet} \leftarrow \mathsf{VecSet} \cup \{\boldsymbol{\alpha}\}$
  $Z \leftarrow \mathrm{E}(y \cdot \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle)$ ; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{Z\}$
  Return $Z$

FIN($Z, \boldsymbol{\alpha}$):
  If ($\boldsymbol{\alpha} \in \mathsf{span}(\mathsf{VecSet})$) then return $\mathsf{false}$
  If ($Z \notin \mathrm{GL}$) then return $\mathsf{false}$
  Return ($Z = \mathrm{E}(y \cdot \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle)$)
```

Figure 16: Game defining t-VCDH problem in the generic group model.

Therefore, we have

$$\begin{aligned}\Pr[\mathsf{GoodInd}] &= \sum_{\substack{S \subseteq [1..\mathsf{ns}],\\ |S|<t}} \Pr\left[\,\mathsf{GoodInd} \mid \mathsf{BadLR}\ \wedge\ CS = S\,\right] \cdot \Pr\left[\,\mathsf{BadLR} \mid CS = S\,\right] \\ &\geq \sum_{\substack{S \subseteq [1..\mathsf{ns}],\\ |S|<t}} \frac{1}{\mathsf{ns}} \Pr\left[\,\mathsf{BadLR} \mid CS = S\,\right] = \frac{1}{\mathsf{ns}} \Pr[\mathsf{BadLR}]\ .\end{aligned}$$

▌

D Security proof for VCDH in the GGM

In this section, we prove that the t-VCDH assumption holds in the generic group model (GGM) [37, 34]. Our GGM framework and proof follow [2].

GGM DEFINITIONS. Suppose G is a set whose size $p = |G|$ is a prime, and $\mathrm{E} : \mathbb{Z}_p \to G$ is a bijection, called the encoding function. For $A, B \in G$, define $A\ \mathsf{op}_\mathrm{E}\ B = \mathrm{E}(\mathrm{E}^{-1}(A) + \mathrm{E}^{-1}(B))$. Then G is a group under the operation op_E [41], with identity element $\mathrm{E}(0)$, and the encoding function becomes a group homomorphism: $\mathrm{E}(a + b) = \mathrm{E}(a)\ \mathsf{op}_\mathrm{E}\ \mathrm{E}(b)$ for all $a, b \in \mathbb{Z}_p$. The element $g = \mathrm{E}(1) \in G$ is a generator of this group, and $\mathrm{E}^{-1}(A)$ is then the discrete logarithm of $A \in G$ relative to g. We call op_E the group operation on G induced by E.

In the GGM, the encoding function E is picked at random and the adversary is given an oracle for the group operation op_E induced on G by E. Game $\mathbf{G}^{\text{gg-}t\text{-vcdh}}_G$ in Fig. 16 defines, in this way, the

```
Games $\boxed{\mathrm{Gm}_0}$, $\mathrm{Gm}_1$

INIT():
  $V \leftarrow \emptyset$; $p \leftarrow |G|$ ; $y \leftarrow\!\!\$\ \mathbb{Z}_p$
  For $i = 1, \ldots, t$ do $\boldsymbol{x}[i] \leftarrow\!\!\$\ \mathbb{Z}_p$
  $\mathbb{1} \leftarrow \mathrm{VE}(0)$; $g \leftarrow \mathrm{VE}(1)$; $Y \leftarrow \mathrm{VE}(\mathsf{Y})$
  For $i = 1, \ldots, t$ do $\boldsymbol{X}[i] \leftarrow \mathrm{VE}(\mathsf{X}_i)$
  Return $\mathbb{1}, g, Y, \boldsymbol{X}[1], \ldots, \boldsymbol{X}[t]$

VE($\mathsf{p}$): // Here $\mathsf{p} \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}]$.
  If ( $\mathrm{TV}[\mathsf{p}] \neq \bot$ ) then return $\mathrm{TV}[\mathsf{p}]$
  $v \leftarrow \mathsf{p}(\boldsymbol{x}, y)$; $C \leftarrow\!\!\$\ G \setminus V$
  If $\mathrm{TE}[v] \neq \bot$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C \leftarrow \mathrm{TE}[v]}$
  $V \leftarrow V \cup \{C\}$; $\mathrm{TE}[v] \leftarrow C$
  $\mathrm{TV}[\mathsf{p}] \leftarrow C$; $\mathrm{TI}[C] \leftarrow \mathsf{p}$ ; Return $\mathrm{TV}[\mathsf{p}]$

VE$^{-1}(C)$: // Here $\mathrm{TI}[C] \neq \bot$.
  Return $\mathrm{TI}[C]$

OP($A, B, \mathsf{sgn}$): // Here $\mathrm{TI}[A], \mathrm{TI}[B] \neq \bot$ and $\mathsf{sgn} \in \{+, -\}$
  $\mathsf{p} \leftarrow \mathrm{VE}^{-1}(A)\ \mathsf{sgn}\ \mathrm{VE}^{-1}(B)$ ; $C \leftarrow \mathrm{VE}(\mathsf{p})$ ; Return $C$

EVAL($\boldsymbol{\alpha}$): // Here $\boldsymbol{\alpha} \in \mathbb{Z}_p^t$
  $\mathsf{VecSet} \leftarrow \mathsf{VecSet} \cup \{\boldsymbol{\alpha}\}$
  $\mathsf{p} \leftarrow \sum_{i=1}^{t} \boldsymbol{\alpha}[i] \cdot \mathsf{X}_i\mathsf{Y}$ ; $Z \leftarrow \mathrm{VE}(\mathsf{p})$
  Return $Z$

FIN($Z, \boldsymbol{\alpha}$): // Here $\mathrm{TI}[Z] \neq \bot$.
  If ($\boldsymbol{\alpha} \in \mathsf{span}(\mathsf{VecSet})$) then return $\mathsf{false}$
  $\mathsf{p} \leftarrow \sum_{i=1}^{t} \boldsymbol{\alpha}[i] \cdot \mathsf{X}_i\mathsf{Y}$
  Return ($Z = \mathrm{VE}(\mathsf{p})$)
```

Figure 17: Games Gm_0 and Gm_1 for the proof of Theorem D.1.

t-VCDH problem. The set G parameterizes the game, and the random choice of encoding function $\mathrm{E} : \mathbb{Z}_p \to G$ is shown at line 1. Procedure OP then implements either the group operation op_{E} on G induced by E (when sgn is $+$) or its inverse (when sgn is $-$). Set GL holds all group elements generated so far. Oracle EVAL takes $\boldsymbol{\alpha} \in \mathbb{Z}_p^t$ to return what in the generic group corresponds to Y raised to the power $\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$. We let $\mathbf{Adv}_{G}^{\text{gg-t-vcdh}}(\mathcal{A}) = \Pr[\mathbf{G}_{G}^{\text{gg-}t\text{-vcdh}}(\mathcal{A})]$ be its ggm-vcdh-advantage.

<u>SECURITY OF t-VCDH IN THE GGM.</u> The following upper bounds the ggm-vcdh-advantage of an adversary $\mathcal{A}$ as a function of t and the number of its OP and EVAL queries.

Theorem D.1 *Let G be a set whose size $p = |G|$ is a prime. Let $t \geq 1$ be an integer. Let $\mathcal{A}$ be an adversary making $\mathrm{Q}_{\mathcal{A}}^{\mathrm{OP}}$ queries to its* OP *oracle and $\mathrm{Q}_{\mathcal{A}}^{\mathrm{EVAL}}$ queries to its* EVAL *oracle. Let $q = \mathrm{Q}_{\mathcal{A}}^{\mathrm{OP}} + \mathrm{Q}_{\mathcal{A}}^{\mathrm{EVAL}} + t + 4$. Then*

$$\mathbf{Adv}_{G}^{\text{gg-t-vcdh}}(\mathcal{A}) \leq \frac{q(q-1)}{p} \ . \tag{2}$$

We note that the bound tells us that the hardness of the t-VCDH problem in the GGM is comparable to that of the discrete logarithm and the CDH problems.

Proof of Theorem D.1: The proof will consider two games Gm_0 and Gm_1. Beyond the procedures of game $\mathbf{G}_G^{\text{gg-}t\text{-vcdh}}$, the games also define procedures VE and VE^{-1}, the *polynomial encoding* and its inverse. These procedures are not exported, meaning can be called only by other game procedures, not by the adversary.

Throughout, we assume the adversary $\mathcal{A}$ makes no trivial queries. By this we mean that the checks at lines 7 and 14 of game $\mathbf{G}_G^{\text{gg-}t\text{-vcdh}}$ are not triggered. In our games the consequence is that we assume $\mathrm{TI}[A], \mathrm{TI}[B] \neq \perp$ in any $\mathrm{O{\scriptsize P}}(A, B, \mathsf{sgn})$ query and $\mathrm{TI}[Z] \neq \perp$ in the $\mathrm{F{\scriptsize IN}}(Z, \boldsymbol{\alpha})$ query.

We start with game Gm_0 of Figure 17, claiming that

$$\mathbf{Adv}_G^{\text{gg-t-vcdh}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \; . \tag{3}$$

The game picks $y, \boldsymbol{x}[1], \ldots, \boldsymbol{x}[t]$ in the same way as game $\mathbf{G}_G^{\text{gg-}t\text{-vcdh}}$. However, the encoding function E is generated implicitly and lazily, via the table TE. Moreover, this table is set indirectly by calling the procedure VE, which we call the polynomial-encoding function, on the indicated polynomial arguments. In particular, VE takes as input a $(t+1)$-variate polynomial[2] $\mathsf{p} \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}]$ and returns $\mathrm{E}(\mathsf{p}(\boldsymbol{x}, y))$. By induction, one can easily see that we always query VE with the correct polynomials, so that running either of $\mathbf{G}_G^{\text{gg-}t\text{-vcdh}}$ and Gm_0q with the same E, the same adversary's randomness, and the same $\boldsymbol{x}, y$, would generate the same values to the adversary. Therefore, $\mathbf{G}_G^{\text{gg-}t\text{-vcdh}}$ and Gm_0 behave identically and, Equation (3) holds.

A bit more precisely, to implement VE, the game maintains two tables $\mathrm{TV} : \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}] \to G \cup \{\perp\}$ and $\mathrm{TI} : G \to \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}] \cup \{\perp\}$ (the "I" stands for "inverse"). The only subtle point is that it is possible that $\mathrm{TV}[\mathsf{p}] = \mathrm{TV}[\mathsf{p}'] = C$ for two distinct polynomials $\mathsf{p} \neq \mathsf{p}'$, and $\mathrm{TI}[C] = \mathsf{p}$, i.e., only one inverse is defined. However, when this is the case, $v = \mathsf{p}(\boldsymbol{x}, y) = \mathsf{p}'(\boldsymbol{x}, y)$ must hold, and thus either polynomial can be used later when $\mathrm{TI}[C] = \mathsf{p}$, as *only* the evaluation on $\boldsymbol{x}, y$ matter here.

Moving on, the second game Gm_1 is also depicted in Figure 17, and is identical to Gm_0, except for the minor difference that the encodings are no longer a function of the *evaluation* of the polynomial, but *of the polynomial itself*. Therefore, now, even when $\mathsf{p}(\boldsymbol{x}, y) = \mathsf{p}'(\boldsymbol{x}, y)$, the outputs of $\mathrm{VE}(\mathsf{p})$ and $\mathrm{VE}(\mathsf{p}')$ are distinct (and uniform). As bad is set exactly the frist time that we encounter two such polynomials„ Gm_0 and Gm_1 are equivalent-until-bad, and thus, by the Fundamental Lemma [8],

$$\Pr[\mathrm{Gm}_0(\mathcal{A})] \leq \Pr[\mathrm{Gm}_1(\mathcal{A})] + \Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \; . \tag{4}$$

In Gm_1, the outputs of all VE calls are independent of $\boldsymbol{x}$ and y, and the latter are only used to set bad, which also does not affect the behavior. Therefore, we can equivalently sample $\boldsymbol{x}$ and y *at the end* of the execution, and check whether any of the VE queries would have provoked $\mathsf{bad} \leftarrow \mathsf{true}$. For this to happen, there must exist two queries $\mathrm{VE}(\mathsf{p}_i)$ and $\mathrm{VE}(\mathsf{p}_j)$ such that $\mathsf{p}_i \neq \mathsf{p}_j$, and

$$\mathsf{p}_i(\boldsymbol{x}, y) - \mathsf{p}_j(\boldsymbol{x}, y) = 0 \; . \tag{5}$$

Because every polynomial p that is queried to VE has degree at most 2, the same is true for $\mathsf{p}_i(\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}) - \mathsf{p}_j(\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y})$. Hence, the probability that Equation (5) holds is at most $2/p$ by the Schwartz-Zippel lemma. Further, as VE is invoked at most q times, there are most $\binom{q}{2}$ such pairs $\mathsf{p}_i, \mathsf{p}_j$, and thus, by the union bound,

$$\Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \binom{q}{2} \cdot \frac{2}{p} = \frac{q(q-1)}{p} \; . \tag{6}$$

Finally, to conclude the proof, we argue that $\Pr[\mathrm{Gm}_1(\mathcal{A})] = 0$. This is because by our assumption

[2]Here $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}]$ is the set of $(t+1)$-variate polynomials with coefficients in $\mathbb{Z}_p$ and (formal) variables $\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}$.

```
$\underline{\mathcal{B}^{\text{INIT},\text{EVAL}}():}$
$(\boldsymbol{X}, Y) \leftarrow_{\$} \text{INIT}()$
$i^* \leftarrow_{\$} [1..q]$; $M^* \leftarrow \bot$
$(M, sig) \leftarrow_{\$} \mathcal{A}^{\widetilde{\text{INIT}},\widetilde{\text{PPO}},\widetilde{\text{PSIGNO}},\widetilde{\text{RO}}}()$
If $M = M^*$ then
    $Z = sig / \prod_{j=1}^{k} Y^{\lambda_j(0) \cdot sk_j}$
    Return $(\boldsymbol{\alpha}^{(0)}, Z)$
Else abort

$\underline{\widetilde{\text{INIT}}(CS):}$
Require: $CS = [1..k]$, $k < t$
$HS \leftarrow [1..\mathsf{ns}] \setminus CS$
For $i \in [1..k]$ do
    $sk_i \leftarrow_{\$} \mathbb{Z}_p$; $vk_i \leftarrow g^{vk_i}$
For $i \in [k+1..t]$ do $vk_i \leftarrow \boldsymbol{X}[i]$
For $i \in [t+1..\mathsf{ns}]$ do
    $vk_i \leftarrow \prod_{j=1}^{t} vk_j^{\lambda_j(i)}$
$vk \leftarrow \prod_{j=1}^{t} vk_j^{\lambda_j(0)}$
$aux = (vk_1, \ldots, vk_{\mathsf{ns}})$
Return $vk, aux, \{sk_i\}_{i \in CS}$

$\underline{\widetilde{\text{PPO}}(i):}$
Return $\bot$

$\underline{\widetilde{\text{PSIGNO}}(i, lr):}$
$M \leftarrow lr.\mathsf{msg}$
$\text{S}_1(M) \leftarrow \text{S}_1(M) \cup \{i\}$
If $M \neq M^*$ then
    $psig \leftarrow vk_i^{\lambda_i^{lr}.\mathsf{ss} \cdot D[M]}$
Else
    $A \leftarrow \text{EVAL}(\boldsymbol{\alpha}^{(i)})$
    $psig \leftarrow \left(\prod_{j=1}^{k} Y^{\lambda_j(i) \cdot sk_j} \cdot A\right)^{\lambda_i^{lr}.\mathsf{ss}}$
    If $|\text{S}_1(M^*)| \geq t - k$ then abort
Return $psig$

$\underline{\widetilde{\text{RO}}(M):}$ // Random oracle
$\text{cnt} \leftarrow \text{cnt} + 1$
If $\text{cnt} = i^*$ then
    $M^* \leftarrow M$
    $T[M] \leftarrow Y$; $D[M] \leftarrow \bot$
Else
    $y \leftarrow_{\$} \mathbb{Z}_p$; $T[M] \leftarrow g^y$; $D[M] \leftarrow y$
Return $T[M]$
```

Figure 18: Adversary $\mathcal{B}$ for the proof of Theorem 4.1.

that $\mathcal{A}$ makes no trivial queries, upon invoking $\text{FIN}(\boldsymbol{\alpha}^*, Z^*)$, there must exist a polynomial p such that $\text{TV}[\mathsf{p}] = Z^*$. Further, by construction, this polynomial can only be in the linear span L of the polynomials $1, \mathsf{X}_1, \ldots, \mathsf{X}_t$ and the polynomials $\sum_{i=1}^{t} \boldsymbol{\alpha}[i] \cdot \mathsf{X}_i \mathsf{Y}$ for each $\boldsymbol{\alpha} \in \mathsf{VecSet}$. If $\mathcal{A}$ indeed wins, because $\boldsymbol{\alpha}^* \notin \mathsf{span}(\mathsf{VecSet})$, we necessarily have that $\mathsf{p} = \sum_{i=1}^{t} \boldsymbol{\alpha}^*[i] \cdot \mathsf{X}_i \mathsf{Y} \notin L$. Therefore, $\text{VE}(\mathsf{p})$ is a fresh query to VE, which returns a value different from Z^*, and thus $\text{Gm}_1(\mathcal{A})$ returns false. ▌

E Proof of Theorem 4.1

Proof: We give a construction of the adversary $\mathcal{B}$ in Figure 18. Whenever we say that $\mathcal{B}$ aborts, we mean that $\mathcal{B}$ stops returning a default output. For notational convenience, we do not make calls to FIN explicit, rather we let $\mathcal{A}$ and $\mathcal{B}$ produce an output, which is (implicitly) processed by FIN. Morover, to reduce notational overhead, we assume without loss of generality that $\mathcal{A}$ asks for $CS = [1..k]$, where $k < t$. (This can always be achieved by permuting the servers indices accordingly.) Furthermore, we also assume that $\mathcal{A}$ queries RO on M prior to any query $\text{PSIGNO}(i, lr)$ such that $lr.\mathsf{msg} = M$. (This adds up to q_s additional RO queries, and we let $q = q_h + q_s$ be the total number of RO queries of the resulting adversary.) In the description of $\mathcal{B}$, for any $j \in [1..k]$, we also define the j-th Lagrange polynomials as

$$\lambda_j(\mathsf{X}) = \prod_{j' \in [1..t] \setminus \{j\}} \frac{\mathsf{X} - j'}{j - j'} .$$

Recall in particular that $p(\mathsf{X}) = \sum_{i=1}^{t} \lambda_i(\mathsf{X}) \cdot y_i$ is the unique degree $t-1$ polynomial such that $p(i) = y_i$ for all $i \in [1..t]$. We also use $\boldsymbol{\alpha}^{(x)} \in \mathbb{Z}_p^t$ for $x \in \mathbb{Z}_p$ to denote the vector such that

$$\boldsymbol{\alpha}^{(x)}[j] = \begin{cases} 0 & \text{if } j \in [1..k], \\ \lambda_j(x) & \text{if } j \in [k+1..t] \; . \end{cases}$$

Let us define $V = \{\boldsymbol{\alpha}^{(x)} \; : \; x \in [0..\mathsf{ns}] \setminus [1..k]\}$, and we note that any $t-k$ vectors in V are linearly independent. To see this, consider the $((t-k) \times (t-k))$-dimensional matrix whose $t-k$ rows consist of the vectors $\boldsymbol{\alpha}^{(x_i)}[k+1..t]$, i.e., the last $t-k$ components of $\boldsymbol{\alpha}^{(x_i)}$, for some distinct $x_1, \ldots, x_{t-k} \notin [1..k]$. Then, this matrix has full rank $t-k$, as the columns are the evaluations of the Lagrange polynomials $\lambda_{k+1}, \ldots, \lambda_t$ at $x_1, \ldots, x_{t-k}$, and as we argue next, they are linearly independent. (Which in turn implies that the rows of the matrix are linearly independent.) Indeed, if they were not independent, we would have $\eta_{k+1}, \ldots, \eta_t \in \mathbb{Z}_p$, not all of them equal zero, such that

$$\sum_{j=k+1}^{t} \eta_j \cdot \lambda_j(x_i) = \sum_{j=1}^{t} \eta_j \cdot \lambda_j(x_i) = 0$$

for all $i \in [1..t-k]$, where we have set $\eta_1 = \cdots = \eta_k = 0$. This would mean in turn that there exist a non-zero degree $t-1$ polynomial which is zero at all t points $[1..k] \cup \{x_1, \ldots, x_{t-k}\}$, which is a contradiction with the fact that such a polynomial can have at most $t-1$ zeros.

For now, let us ignore the abort condition within $\widetilde{\text{PS}\textsc{ign}\text{O}}(i, lr)$. Then, we claim that the simulation of $\mathcal{A}$'s execution within $\mathcal{B}$ is perfect when $\boldsymbol{X}, Y$ are a proper t-VCDH instance, i.e., when $\boldsymbol{X}$ and Y are both uniform. In particular, all generated keys have the right distribution, i.e., $vk_i = g^{sk_i}$, where $sk_i = p(i)$ for a random polynomial p of degree $t-1$. Further, the output of every $\widetilde{\text{RO}}$ query is uniform and independent.

Most importantly, a call to $\widetilde{\text{PS}\textsc{ign}\text{O}}(i, lr)$ always returns $psig = H^{\lambda_i^{lr.\mathsf{SS}} \cdot sk_i}$, where H is the response for a $\widetilde{\text{RO}}$ query on input $M = lr.\mathsf{msg}$. This is easy to see when $M \neq M^*$, in which case we do know the discrete logarithm $D[M]$ of H, and thus $H^{sk_i} = g^{D[M] \cdot sk_i} = vk_i^{D[M]}$. Instead, if $M = M^*$, then we have $H = Y$, and

$$Y^{sk_i} = Y^{\sum_{j=1}^{t} sk_j \cdot \lambda_j(i)} = Y^{\sum_{j=1}^{k} sk_j \cdot \lambda_j(i)} \cdot Y^{\langle \boldsymbol{\alpha}^{(i)}, \boldsymbol{x} \rangle} = \prod_{j=1}^{k} Y^{sk_j \cdot \lambda_j(i)} \cdot A \; ,$$

where $A \leftarrow \textsc{Eval}(\boldsymbol{\alpha}^{(i)})$.

Finally, assume that $\mathcal{A}$ indeed breaks TS-UF-1 security, i.e., it outputs M, sig such that $sig = H^{sk}$, where $sk = \mathsf{DL}_{\mathbb{G},g}(vk) = \sum_{j=1}^{t} \lambda_j(0) \cdot sk_i$. Moreover, $|\mathrm{S}_1(M^*)| < t-k$. Then, if $M = M^*$, we also have $H = Y$, and therefore $Z = Y^{\langle \boldsymbol{\alpha}^{(0)}, \boldsymbol{x} \rangle}$. Moreover, at most $t-k-1$ queries have been made to $\widetilde{\text{PS}\textsc{ign}\text{O}}$ for M^*, which in turn means that at most $t-k-1$ $\textsc{Eval}$ queries have been made by $\mathcal{B}$. By the linear independence properties we established above, we have $\boldsymbol{\alpha}^{(0)}$ is not in the span of the prior $\textsc{Eval}$ queries, and thus $\mathcal{B}$ breaks t-VCDH. Therefore, for $\mathcal{B}$ to succeed, we need that (1) $\mathcal{A}$'s succeeds and (2) $M = M^*$ for the final forgery output by $\mathcal{A}$. The probability that both happen is exactly

$$\cdot \mathbf{Adv}_{\mathbb{G}}^{t\text{-vcdh}}(\mathcal{B}) \geq 1/q \cdot \mathbf{Adv}_{\mathsf{BLS}[\mathbb{G},\mathbb{G}_\mathsf{T},\mathsf{ns},t],t}^{\text{ts-uf-1}}(\mathcal{A}) \; .$$

The additional abort condition ensures that $\mathcal{B}$ always makes at most $t-k-1 \leq t-1$ queries to $\textsc{Eval}$, and does not affect its success probability. ▌

F CDH (loosely) implies t-VCDH

The following lemma suffices in some applications to reduce the hardness of t-VCDH to that of CDH. Its use is not really necessary in our results (although it can, with some care, give an alternative flow to obtain a loose reduction to CDH for TS-UF-1 security of BLS).

Lemma F.1 *Let $\mathbb{G}$ be a cyclic group with generator g and prime order p. Let $1 \leq q \leq t-1$. Further, let $V \subseteq \mathbb{Z}_p^t$ be such that every $q+1$ vectors in V are linearly independent. Let $\mathcal{A}$ be a t-VCDH adversary which makes $q \leq t-1$ EVAL queries such that all $\boldsymbol{\alpha} \in \mathbb{Z}_p^t$ input to EVAL and FIN are in V. Then, there exists a CDH adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}_{\mathbb{G}}^{t\text{-vcdh}}(\mathcal{A}) \leq |V| \cdot \binom{|V|-1}{q} \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathrm{cdh}}(\mathcal{B}) \ .$$

Moreover, $\mathcal{B}$ runs in time equals that of running $\mathcal{A}$, plus, roughly, the time to compute a matrix inverse in $\mathbb{Z}_p^{t\times t}$, and the time to compute $3t$ exponentiations in $\mathbb{G}$.

For example, if V consists of all t unit vectors in $\mathbb{Z}_p^t$, and $q = t-1$, the loss in the above bound is exactly t—this corresponds to the naïve reduction guessing the coordinate i for which the adversary computes Y^{x_i}. **Proof:** The CDH adversary $\mathcal{B}$ is given $X = g^x, Y \in \mathbb{G}$, and needs to compute $Z = Y^x$. We give the adversary in Figure 19. In the description, we use a function Extend which, on input a sequence of linearly independent vectors $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{q+1})$ from $\mathbb{Z}_p^t$ (where $q \leq t-1$), it returns $(\boldsymbol{\alpha}_{q+2}, \ldots, \boldsymbol{\alpha}_t) \leftarrow \mathsf{Extend}(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{q+1})$ such that $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_t\}$ is a basis of $\mathbb{Z}_p^t$. Here, we think of the vector $\boldsymbol{\alpha}$ as a column vector, with $\boldsymbol{\alpha}^T$ being its transpose. In $\mathcal{B}$'s description, $\binom{S}{r}$ denotes all size-r subsets of a finite set S. We allow the adversary $\mathcal{B}$ to *abort*, with an implicit understanding that it terminates by outputting a fixed group element in this case. (The specific choice is irrelevant.)

To analyze how well $\mathcal{B}$ succeeds, define first $\boldsymbol{x} \in \mathbb{Z}_p^t$ such that $\boldsymbol{x}[i] = \mathsf{DL}_{\mathbb{G},g}(\boldsymbol{X}[i])$. Also, let us extend $\tilde{\boldsymbol{x}}$ with $\tilde{\boldsymbol{x}}[q+1] = \mathsf{DL}_{\mathbb{G},g}(X)$. It is easy to see that $\boldsymbol{x}$ is uniform over $\mathbb{Z}_p^t$, because $\boldsymbol{M}$ is full rank, and thus the simulated $\boldsymbol{X}$ has the correct distribution. Furthermore, as long as no abort occurs, the answers to EVAL queries are correct. This is because

$$\langle \boldsymbol{\alpha}_i, \boldsymbol{x} \rangle = \boldsymbol{\alpha}_i^T \boldsymbol{x} = \boldsymbol{\alpha}_i^T \boldsymbol{M} \tilde{\boldsymbol{x}} = \boldsymbol{e}_i^T \tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}[i] \ .$$

For a similar reason, if $\mathcal{A}$ indeed outputs $Z = Y^{\langle \boldsymbol{\alpha}_{q+1}, \boldsymbol{x} \rangle}$, i.e., both $Z = Y^{\langle \boldsymbol{\alpha}^*, \boldsymbol{x} \rangle}$ and $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}_{q+1}$ hold, then $Z = Y^{\tilde{\boldsymbol{x}}[q+1]} = Y^{\mathsf{DL}_{\mathbb{G},g}(X)}$.

Therefore, with E being the event that $\mathcal{B}$ does not abort,

$$\mathbf{Adv}_{\mathbb{G}}^{\mathrm{cdh}}(\mathcal{B}) \geq \Pr\left[E \ \wedge \ Z = Y^{\langle \boldsymbol{\alpha}^*, \boldsymbol{x} \rangle}\right] \ ,$$

where the greater-equal takes into account the fact that when aborting, the default output may also be, occasionally, correct. To compute the probability on the RHS, we can consider a different experiment where EVAL responds correctly with $Y^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$ even if $\boldsymbol{\alpha} \notin \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q\}$. (Clearly, this cannot be done efficiently, as in general this will require knowledge of $\mathsf{DL}_{\mathbb{G},g}(X)$.) Moreover, the event E is only defined at the end of the experiment, when $\mathcal{A}$ outputs $(\boldsymbol{\alpha}^*, Z)$, and occurs if $\boldsymbol{\alpha}^* \neq \boldsymbol{\alpha}_{q+1}$ or one the EVAL queries is not in $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q\}$.

The probability $\Pr\left[E \ \wedge \ Z = Y^{\langle \boldsymbol{\alpha}^*, \boldsymbol{x} \rangle}\right]$ has not changed in this experiment, yet the view of $\mathcal{A}$ is now independent of the choice of $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{q+1}$ and thus of the event E. Therefore,

$$\Pr\left[E \ \wedge \ Z = Y^{\langle \boldsymbol{\alpha}^*, \boldsymbol{x} \rangle}\right] = \Pr[E] \cdot \mathbf{Adv}_{\mathbb{G}}^{t\text{-vcdh}}(\mathcal{A})$$

```
$\underline{\mathcal{B}():}$
$(X, Y) \leftarrow\$ \textsc{Init}()$
$\boldsymbol{\alpha}_{q+1} \leftarrow\$ V$; $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q\} \leftarrow\$ \binom{V \setminus \{\boldsymbol{\alpha}_{q+1}\}}{q}$
$(\boldsymbol{\alpha}_{q+2}, \ldots, \boldsymbol{\alpha}_t) \leftarrow \mathsf{Extend}(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{q+1})$
$\boldsymbol{A} \leftarrow \begin{bmatrix} \boldsymbol{\alpha}_1^T \\ \vdots \\ \boldsymbol{\alpha}_t^T \end{bmatrix}$; $\boldsymbol{M} \leftarrow \boldsymbol{A}^{-1}$ // $\boldsymbol{A}, \boldsymbol{M} \in \mathbb{Z}_p^{t \times t}$
For $i \in [1..t] \setminus \{q+1\}$ do $\tilde{\boldsymbol{x}}[i] \leftarrow\$ \mathbb{Z}_p$; $\tilde{\boldsymbol{X}}[i] \leftarrow g^{\tilde{\boldsymbol{x}}[i]}$
$\tilde{\boldsymbol{X}}[q+1] \leftarrow X$
For $i \in [1..t]$ do
    $\boldsymbol{X}[i] \leftarrow \prod_{j=1}^{t} \tilde{\boldsymbol{X}}[j]^{\boldsymbol{M}[i,j]}$
$(\boldsymbol{\alpha}^*, Z) \leftarrow\$ \mathcal{A}^{\widetilde{\textsc{Init}}, \widetilde{\textsc{Eval}}}()$
If $\boldsymbol{\alpha}^* \neq \boldsymbol{\alpha}_{q+1}$ then abort
Else return $Z$

$\underline{\widetilde{\textsc{Init}}():}$
Return $(\boldsymbol{X}, Y)$

$\underline{\widetilde{\textsc{Eval}}(\boldsymbol{\alpha}):}$
If $\exists i \in [1..q] : \boldsymbol{\alpha} = \boldsymbol{\alpha}_i$ then
    Return $Y^{\tilde{\boldsymbol{x}}[i]}$
Else abort
```

Figure 19: Adversary $\mathcal{B}$ for the proof of Lemma F.1.

whereas

$$\Pr[E] = \frac{1}{|V|} \cdot \frac{1}{\binom{|V|-1}{q}} .$$

This concludes the proof. ▮

G TS-UF-0 Security of BLS

For completeness, the following theorem establishes the TS-UF-0 security of BLS based on the CDH assumption. The proof is simpler to the one given for Theorem 4.1, but very similar in spirit, and for this reason, we only give a proof sketch.

Theorem G.1 (TS-UF-0 security of BLS) *For any TS-UF-0 adversary $\mathcal{A}$ making at most q_s queries to* PSignO *and at most q_h queries to* RO*, there exists a CDH adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{ts-uf-0}}_{\mathsf{BLS}[\mathbb{G},\mathbb{G}_\mathsf{T}]}(\mathcal{A}) \leq (q_h + q_s) \cdot \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(\mathcal{B}) . \tag{7}$$

Moreover, $\mathcal{B}$ runs in time roughly equal that of $\mathcal{A}$, plus the time to perform at most $2\mathsf{ns} + q_s + q_h$ exponentiations and group operations.

We can then use Theorem 3.1 to obtain a bound for TS-UF-1 based on CDH alone, but this will incur a multiplicative loss of ns^{t-1}. This may be acceptable in scenarios with few servers (e.g., $\mathsf{ns} = 10$, $t = 3$.)

Proof of Sketch: Let $\mathcal{A}$ be the given TS-UF-0 adversary. We observe without loss of generality that we can assume $\mathcal{A}$ corrupts $t-1$ parties. (This is unlike TS-UF-1.) Furthermore, we assume that $CS = [1..t-1]$. Then, the construction of $\mathcal{B}$ is given in Figure 20, using some of the notational machinery from the proof of Theorem 4.1. As we did there, we also assume that $\mathcal{A}$ queries RO on M prior to any query PSignO(i, lr) such that $lr.\mathsf{msg} = M$. (This adds up to q_s additional RO queries, and we let $q = q_h + q_s$.) Then, it is not hard to argue that $\mathcal{B}$ satisfies Equation (7). ▮

```
$\mathcal{B}^{\text{INIT},\text{EVAL}}()$:
$i^* \leftarrow_\$ [1..q]$; $M^* \leftarrow \bot$
$(M, sig) \leftarrow_\$ \mathcal{A}^{\widetilde{\text{INIT}},\widetilde{\text{PPO}},\widetilde{\text{PSIGNO}},\widetilde{\text{RO}}}()$
If $M = M^*$ then
    Return $Z$
Else abort

$\widetilde{\text{INIT}}(CS)$:
Require: $CS = [1..t-1]$
$HS \leftarrow [1..\mathsf{ns}] \setminus CS$
$vk \leftarrow X$
For $i \in [1..t-1]$ do
    $sk_i \leftarrow_\$ \mathbb{Z}_p$; $vk_i \leftarrow g^{vk_i}$
For $i \in [t..\mathsf{ns}]$ do
    $vk_i \leftarrow vk^{\lambda_0(i)} \prod_{j=1}^{t-1} vk_j^{\lambda_j(i)}$
$aux = (vk_1, \ldots, vk_{\mathsf{ns}})$
Return $vk, aux, \{sk_i\}_{i \in CS}$

$\widetilde{\text{PPO}}(i, ctx)$:
Return $\bot$

$\widetilde{\text{PSIGNO}}(i, lr)$:
$M \leftarrow lr.\mathsf{msg}$
If $M \neq M^*$ then
    $psig \leftarrow \mathsf{vk}_i^{\lambda_i^{lr}.\mathsf{SS} \cdot D[M]}$
Else abort
Return $psig$

$\widetilde{\text{RO}}(M)$: // Random oracle
cnt $\leftarrow$ cnt + 1
If cnt $= i^*$ then
    $M^* \leftarrow M$
    $T[M] \leftarrow Y$; $D[M] \leftarrow \bot$
Else
    $y \leftarrow_\$ \mathbb{Z}_p$; $T[M] \leftarrow g^y$; $D[M] \leftarrow y$
Return $T[M]$
```

Figure 20: Adversary $\mathcal{B}$ for the proof of Theorem G.1.

H Proofs of forking lemmas

H.1 Proof of Lemma 5.3

Proof: For any $i \in S$, $h_1, \ldots, h_{i-1} \in H$, and input x, define

$$Y_i(x, h_1, \ldots, h_{i-1}) := \Pr_{h_i,\ldots,h_q \leftarrow\$ H}[I = i : (I, \text{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] .$$

Then, we have

$$\begin{aligned}\text{acc}(\mathcal{A}) &= \sum_{i \in S} \Pr_{x \leftarrow \mathsf{IG}, h_1,\ldots,h_q \leftarrow\$ H}[I = i : (I, \text{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \\ &= \sum_{i \in S} \mathbf{E}_{x \leftarrow \mathsf{IG}, h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_i(x, h_1, \ldots, h_{i-1})] .\end{aligned}$$

Thus, we have

$$\begin{aligned}\text{acc}(\mathsf{Fork}^{\mathcal{A}}) &= \sum_{i \in S} \Pr_{x \leftarrow \mathsf{IG}, h_1,\ldots,h_n, h'_i,\ldots,h'_n \leftarrow\$ H}[I = I' = i : \qquad (8) \\ &\quad (I, \text{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q), \\ &\quad (I', \text{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_n)] \\ &= \sum_{i \in S} \mathbf{E}_{x \leftarrow \mathsf{IG}, h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_i(x, h_1, \ldots, h_{i-1})^2] \qquad (9) \\ &\geq \sum_{i \in S} \left(\mathbf{E}_{x \leftarrow \mathsf{IG}, h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_{i,j}(x, h_1, \ldots, h_{i-1})]\right)^2 \qquad (10) \\ &\geq \frac{1}{|S|} \cdot \left(\sum_{i \in S} \mathsf{E}_{x \leftarrow \mathsf{IG}, h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_i(x, h_1, \ldots, h_{i-1})]\right)^2 \qquad (11)\end{aligned}$$

$$= \frac{\mathrm{acc}(\mathcal{A})^2}{|S|} \ , \tag{12}$$

where (10) is due to the fact that $\mathbf{E}[X^2] \geq (\mathbf{E}[X])^2$ and (11) is due to the fact that $\sum_{i=1}^n a_i^2 \geq \frac{1}{n}\left(\sum_{i=1}^n a_i\right)^2$. ▌

H.2 Proof of Lemma 5.7

Proof: Denote $\hat{h}_i = (h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_q) \in H^{q-1}$. For any $i \in [1..q]$, $j \in Q$, $\hat{h}_i \in H^{q-1}$, and input x, define

$$Y_{i,j}(x, \hat{h}_i) := \Pr_{h_i \leftarrow\$ H}[I = i, J = j : (I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \ .$$

Then, we have

$$\begin{aligned}
\mathrm{acc}(\mathcal{A}) &= \sum_{i=1}^{q} \sum_{j \in Q} \Pr_{x \leftarrow \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[I = i, J = j : (I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \\
&= \sum_{i=1}^{q} \sum_{j \in Q} \mathbf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \leftarrow\$ H^{q-1}}[Y_{i,j}(x, \hat{h}_i)] \ .
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) &= \sum_{i=1}^{q} \sum_{j \in Q} \Pr_{x \leftarrow \mathsf{IG}, h_1, \ldots, h_n, h_i' \leftarrow\$ H}[I = I' = i, J = J' = j : \qquad (13) \\
&\qquad (I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q), \\
&\qquad (I', J', \mathrm{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{i-1}, h_i', h_{i+1}, h_q)] \\
&= \sum_{i=1}^{q} \sum_{j \in Q} \mathbf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \leftarrow\$ H^{q-1}}[Y_{i,j}(x, \hat{h}_i)^2] \qquad (14) \\
&\geq \sum_{i=1}^{q} \sum_{j \in Q} \left(\mathbf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \leftarrow\$ H^{q-1}}[Y_{i,j}(x, \hat{h}_i)]\right)^2 \qquad (15) \\
&\geq \frac{1}{q \cdot |Q|} \cdot \left(\sum_{i=1}^{q} \sum_{j \in Q} \mathsf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \leftarrow\$ H^{q-1}}[Y_{i,j}(x, \hat{h}_i)]\right)^2 \qquad (16) \\
&= \frac{\mathrm{acc}(\mathcal{A})^2}{q \cdot |Q|} \ , \qquad (17)
\end{aligned}$$

where (15) is due to the fact that $\mathbf{E}[X^2] \geq (\mathbf{E}[X])^2$ and (16) is due to the fact that $\sum_{i=1}^n a_i^2 \geq \frac{1}{n}\left(\sum_{i=1}^n a_i\right)^2$. ▌