

On FROST and Unconditional LDVR Security

Ian Goldberg¹, Chelsea Komlo², Stefano Tessaro³

¹ University of Waterloo

² BNY

³ University of Washington

Abstract. Inspired by the recent polynomial-time adaptive attacks on threshold Schnorr signatures [20], we contextualize the Low-Dimensional Vector Representation (LDVR) assumption and its impact on relevant threshold Schnorr signature schemes. Our analysis focuses in large part on FROST [4, 13, 16], a widely deployed threshold Schnorr signature scheme. However, our analysis is applicable to any relevant threshold Schnorr signature scheme, such as Lindell’s three-round threshold Schnorr protocol [17].

We show that for cases where FROST is used with 131 parties and fewer, and for parameter choices of 128-bit generic (discrete log) security, FROST is secure for any signing threshold. This bound increases to 260 parties for parameter choices of 256-bit security; our analysis similarly can be extended to other parameter choices. This bound assumes the *best possible* LDVR adversary, even more powerful than the results by Rodríguez García and Tessaro [20].

In other words, for real-world uses of FROST with parameter choices of 128-bit security, and with a small (131 and fewer) number of parties, the security of FROST is determined entirely by the hardness of the Algebraic One-More Discrete Logarithm (AOMDL) assumption, as LDVR is unconditionally hard for this parameter regime. It is an open research question whether the concrete attacks given by Rodríguez García and Tessaro [20] can be improved to approach the unconditional LDVR security bound considered in our analysis.

We then show for each additional party for 132 total number of signers and greater, again for parameter choices of 128-bit generic security, the resulting security of FROST is *at worst* reduced by approximately one bit (or less) of security for the worst-case choice of corruption threshold. This analysis can be used by practitioners to determine the *worst possible* security of even large-scale FROST deployments.

Finally, we give two variants to FROST that build upon prior masking techniques to allow for full adaptive security. The first variant allows for masking keys to be derived from Diffie-Hellman identity keys, allowing for constant memory overhead, at the tradeoff of increased computation. The second variant allows for a 50% reduction in memory overhead for storage of pairwise masking keys, by employing symmetric key predistribution techniques.

Table of Contents

1	Introduction	3
1.1	FROST and LDVR	4
1.2	Gap Between Concrete Attack and LDVR Unconditional Bound	5
1.3	Mitigation Techniques	7
1.4	Acknowledgements	9
2	Preliminaries	9
2.1	Assumptions	11
2.2	Adaptive Security	11
3	Low-Dimensional Vector Representation (LDVR) Assumption	11
3.1	Concrete Hardness of LDVR	13
4	Case Study: LDVR Implication for FROST	13
4.1	Modeling FROST Security with Unconditional LDVR Security	14
4.2	For What Threshold Parameters Does LDVR Impact Affected Schemes?	15
4.3	Measuring the Cost of Each Additional Party	16
5	FROST with DH-Derived Masking Keys	16
6	FROST with Compressed Masking Keys	18

1 Introduction

A threshold signature scheme allows a subset of t parties out of n total possible signers to produce a signature with respect to a joint public key, where $t \leq n$. Furthermore, a threshold signature scheme ensures unforgeability, even when t_c of these parties becoming corrupted, such that $t_c \leq t - 1$.

A threshold signature scheme can be proven secure using two different techniques: the *static* security approach, and the *adaptive* security approach. In the static security setting, the adversary in the unforgeability experiment must declare which parties it wishes to corrupt at the beginning of the experiment, receiving the corresponding secret key shares for those participants, and cannot corrupt any parties after that point. In the adaptive security setting, the adversary is allowed to corrupt parties throughout the lifetime of the protocol, and receives not only the corresponding corrupt parties secret key shares, but the state of each corrupted participant at that point in time in the protocol, across all open signing sessions.

In this note, we provide additional analysis, inspired by the recent attacks defined by Rodríguez García and Tessaro [20] which demonstrates polynomial algorithms solving the *low-dimensional vector representation* (LDVR) assumption. LDVR has recently been proposed as an assumption required to prove the full adaptive security of a wide class of threshold Schnorr signature schemes [2, 5, 12, 13, 14, 16, 17, 18, 21, 22].

We employ the unconditional lower bound on LDVR [13], and show that for parameter choices providing 128-bit security, that FROST maintains full adaptive security for any corruption threshold, up to $n \leq 131$ parties. Full adaptive security entails that the adversary can corrupt up to t_c parties of its choosing to extract their private key shares, for some $t_c \leq t - 1$. (If the adversary can extract the private key shares of t parties, adaptively or non-adaptively, all threshold schemes are of course automatically broken.) Our analysis extends to $n \leq 260$ parties for parameter choices providing 256-bit security. For deployments of FROST with parameters such that the unconditional LDVR bound is exceeded (as given in Section 3.1), we review in Section 1.3 a range of lightweight mitigation techniques.

The Low-Dimensional Vector Representation (LDVR) Assumption. Introduced by Crites et al. [12, 13], the Low-Dimensional Vector Representation (LDVR) assumption can be described as follows.

Let $\mathbf{v}_i$ be the Vandermonde vector $\mathbf{v}_i = (\mathbf{1}, \mathbf{i}, \mathbf{i}^2, \dots, \mathbf{i}^{t-1}) \in \mathbb{Z}_p^t$. Let $\mathcal{A}$ be the LDVR adversary. $\mathcal{A}$ accepts as input the number of signers n , the corruption threshold t_c , and the reconstruction threshold t . $\mathcal{A}$ outputs $n + 1$ coefficients $(\alpha_i)_{i=0}^n \in \mathbb{Z}_p^{n+1}$, and then receives a random challenge $c \leftarrow_s \mathbb{Z}_p$ sampled uniformly at random by the environment. $\mathcal{A}$ must then output a corruption set **corrupt** $\subseteq [n]$ such that $|\text{corrupt}| \leq t_c$. $\mathcal{A}$ wins the LDVR game if Equation 1 holds:

$$\begin{aligned} \mathbf{w} &\leftarrow c\mathbf{v}_0 + \sum_{i=0}^n \alpha_i \mathbf{v}_i, & \text{and} \\ \mathbf{w} &\in \text{span}\{\mathbf{v}_i\}_{i \in \text{corrupt}} \end{aligned} \tag{1}$$

LDVR and Adaptive Security of Threshold Schnorr Signatures. As proven by Crites et al. [13], LDVR is required for a wide class of “key-unique” threshold Schnorr signatures, where key-uniqueness for threshold signature schemes is as defined by Ciampi et al. [10].

In particular, LDVR is required for threshold Schnorr signature schemes where the conditions given in Equation 2 hold.

$$\begin{aligned} (\{sk_i\}_{i \in [n]}, pk, \{pk_i\}_{i \in [n]}) &\stackrel{s}{\leftarrow} \text{Shamir.IssueShares}(sk, n, t), \\ pk &= g^{sk}, \\ pk_i &= g^{sk_i} \end{aligned} \tag{2}$$

Examples of threshold Schnorr signature schemes that fall under this category include FROST [4, 13, 16], FROST2 [4], FROST3 [9], Classic Schnorr [14, 17, 18], SPRINT [5], HARTS [2], and others [15, 21, 22].

A Polynomial-Time Algorithm for LDVR. Rodríguez García and Tesaro [20] reduce LDVR to a structured subset-sum problem and exploit the structure of standard participant identifiers to solve LDVR in polynomial time when sufficiently many parties participate. Their attack applies to parameter choices with $n = \Theta(\log^2 p)$ parties and $t_c = t - 1 = \Theta(\log p)$ corruptions, where p is the group order and the signing threshold is t . Their optimized attack for Curve25519 requires 5,020 parties, with a signing threshold of 205, with up to 204 adaptively corrupted parties. In this setting, the authors’ attack succeeds with probability 1/194 per attempt. The authors also show that an attacker who controls participant identifier selection can introduce hidden structure that enables attacks with only 512 parties and 256 adaptive corruptions for a 256-bit prime group order. This work gives a natural separation between static and adaptive security, previously unknown in the literature.

1.1 FROST and LDVR

In this work, we focus on the impact of LDVR on FROST in particular, due to the widespread use of FROST in practice.

FROST. FROST [4, 13, 16] was the first two-round threshold Schnorr signature scheme that was proven to be concurrently secure. The static security of FROST was proven with respect to the Algebraic One-More Discrete Logarithm (AOMDL) problem, in the random oracle model [4]. However, the full *adaptive*

security of FROST was proven with respect to AOMDL in the Algebraic Group Model (AGM), and LDVR.

FROST has been defined as an informational RFC for the Crypto Forum Research Group (CFRG) within the IRTF [11]. There is additionally a team working to define a FROST submission for the NIST threshold signature scheme call for proposals [7, 8].

LDVR Crossover for FROST. Let n be the number of total parties for a threshold signature scheme, t_c be the number of tolerated participant corruptions, and t be the signing threshold. In this note, we wish to explore the values of (n, t_c, t) at which the hardness of LDVR becomes relevant for a particular scheme.

We show that for parameter choices of 128-bit generic security in elliptic-curve groups, the smallest value of n for which LDVR becomes relevant to the security of FROST is when $n = 132$, for $t_c = t - 1$ in the range $64 \leq t - 1 \leq 68$. Our analysis considers the best possible LDVR adversaries, even more powerful than the concrete attacks demonstrated by Rodríguez García and Tessaro [20]. Moreover, our analysis extends to other parameter regimes. For groups offering 256-bit generic security, $n \leq 260$ offers unconditional LDVR security.

We show a visualization of the hardness of LDVR with respect to 128-bit parameter sets in Figure 1. In Figure 2, we show the tolerance at which high thresholds allow for corrupted parties. As such, even at high thresholds, some tolerance for corrupted parties is allowed, but not the full intended corruption threshold.

Impact of Each Additional Signer. With the unconditional LDVR bounds in mind, we then identify the exact security cost incurred by each additional FROST signer. We show that for groups with 128-bit security, each additional signer costs at most one bit of security. Hence, assuming the best possible LDVR adversary, security in the adaptive corruption setting degrades predictably with the total number of signers.

1.2 Gap Between Concrete Attack and LDVR Unconditional Bound

The unconditional LDVR bound limits the success probability of every adversary, regardless of its computational power, but need not be attainable by an efficient attack. Consequently, when this bound no longer guarantees a desired security level, this does not by itself establish an attack at the corresponding cost. A substantial gap remains between this guarantee and the concrete attacks of Rodríguez García and Tessaro [20], whose optimized Curve25519 attack with canonical participant identifiers uses 5,020 parties. Their work also does not offer, at this stage, a precise analytical characterization of the parameters for which their attack template works, as this appears challenging.

Moreover, that work [20, Section 5] demonstrates limitations of the underlying subset-sum approach at smaller parameters. Specifically, for canonical identifiers, any 256-bit prime group order, $n = 256$, and $t_c = t - 1 = 128$, the corresponding subset-sum problem admits a solution for at most a $2^{-106.5}$ fraction of uniformly

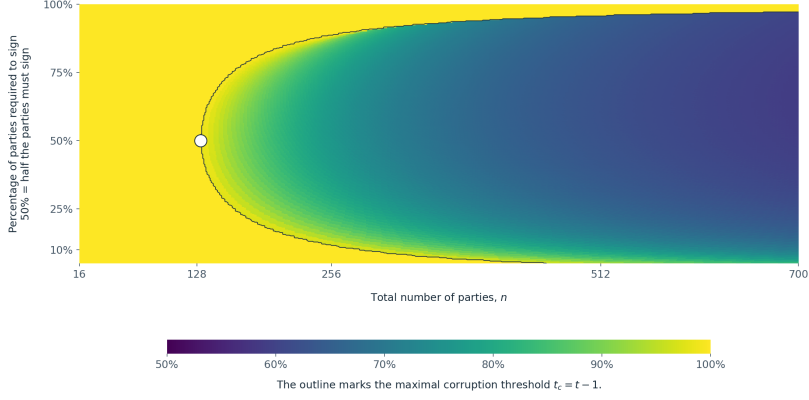


Fig. 1: Security of FROST relative to number of parties n , corruption threshold t_c , and reconstruction threshold t , with respect to the bound at which LDVR is unconditionally hard, for 128-bit security. The circle marks the boundary between $n = 131$ and $n = 132$ parties near a 50% signing threshold.

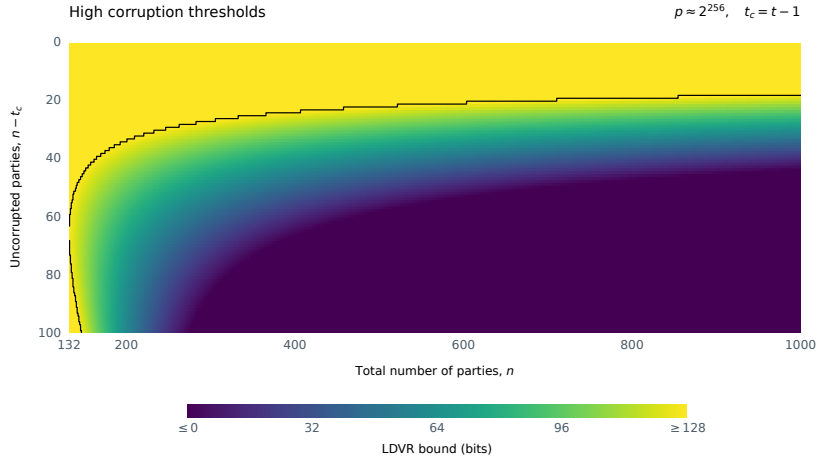


Fig. 2: High-threshold view of Figure 1 for $p \approx 2^{256}$, with $t_c = t - 1$. The yellow region meets the 128-bit LDVR bound. At $n = 1000$, this includes $t_c = t - 1 = 982$, with signing threshold 983.

random targets. Thus, even unlimited computation cannot give this attack template a higher success probability per challenge, although for these parameters the unconditional hardness result no longer holds.

We stress that this limitation concerns the subset-sum formulation used by the attacks and does not establish a comparable hardness bound for LDVR itself. How closely efficient attacks, including in particular different attacks not relying on the subset-sum problem, can approach the unconditional bound remains an open question.

1.3 Mitigation Techniques

Increasing Reconstruction Threshold. One straightforward mitigation for existing deployments of larger number of signers is to simply increase the reconstruction threshold t while keeping n and t_c fixed. For fixed n and t_c , increasing t decreases the ratio in Equation 6. When $t \geq 2t_c + 1$, the LDVR advantage is at most q_{ldvr}/p , although smaller increases can suffice to meet Equation 11. For example, for security parameter λ such that $p \approx 2^{2\lambda}$, any $t \geq 2t_c - \lambda/\lg n + 1$ suffices to provide LDVR advantage at most $q_{ldvr}/2^\lambda$; i.e., λ -bit security.

Hashing Participant Identifiers. Rodríguez García and Tessaro [20, Section 6.4] propose deriving evaluation points as $x_i = H(i)$, where H maps into $\mathbb{F}_p$. In the random oracle model, these points induce a random fixed-weight subset-sum instance, avoiding the structure exploited by their polynomial-time attack. However, their analysis still gives sub-exponential time attacks on random points when the number of parties is $n \geq 1048$ [20, Section 6.3], so deriving participant identifiers by hashing alone does not establish full adaptive security.

Masking. Recent work introduces a masking technique to FROST to achieve full adaptive security, in ms-FROST [1] and FaFROST [3]. Both schemes preserve the two-round structure of FROST, but mask each participant's second-round signature share. With this simple technique, these schemes allow adaptive-security proofs from AOMDL in the AGM and ROM without relying on LDVR.

In ms-FROST, parties establish pairwise keys at the time of key generation. At the time of signing, let $\text{sid} = (pk, m, C, B)$ be the signing context, where C is the signing coalition and $B = \langle (D_i, E_i) \rangle_{i \in C}$ is the ordered commitment list. For each ordered pair $i, j \in C$ with $i \neq j$, let $\rho_{i,j} = \text{PRF}(\text{seed}_{i,j}, \text{sid}) \in \mathbb{Z}_p$. Each signer $i \in C$ then derives its mask as

$$\text{mask}_i = \sum_{j \in C \setminus \{i\}} (\rho_{i,j} - \rho_{j,i}),$$

and returns the masked response $\tilde{z}_i = z_i + \text{mask}_i$, where z_i is the FROST response. The aggregate masks cancel because

$$\sum_{i \in C} \text{mask}_i = \sum_{\substack{i, j \in C \\ i < j}} ((\rho_{i,j} - \rho_{j,i}) + (\rho_{j,i} - \rho_{i,j})) = 0.$$

Although signer i can compute the pairwise masking contribution it shares with signer j , this is only one part of signer j 's full mask:

$$\mathbf{mask}_j = (\rho_{j,i} - \rho_{i,j}) + \sum_{k \in C \setminus \{i,j\}} (\rho_{j,k} - \rho_{k,j}).$$

The remaining parts are derived from pairwise keys shared between j and the other signers in C , so signer i cannot recover $\mathbf{mask}_j$. The masks therefore form a one-time additive sharing of zero: each individual response is hidden, while the combined response is unchanged and verifies as in FROST. FaFROST uses a similar pairwise-blinding technique.

Intuitively, pairwise blinding prevents an adversary from inferring each party's public key shares, which are required to produce a Schnorr forgery using LDVR. Along with this blinding technique, ms-FROST and FaFROST also prohibit exposure of each party's public key share at the time of key generation. To achieve identifiable abort, it is possible for signers to engage in a multi-round interactive protocol, providing Chaum-Pedersen proofs, where the secret key and masks are all committed to via Pedersen commitments.

Note that these masking mitigation technique *cannot* be retroactively applied to existing FROST deployments without complete rotation of signing keys, due to the requirement that participants' public key shares remain secret prior to corruption. Key generation must be performed in a manner that public key shares are not leaked, which excludes common distributed key generation protocols such as Pedersen DKG, but includes trusted dealer key generation.

Generating Masking Keys Via Non-Interactive Key Exchange. In FaFROST and ms-FROST, static pairwise keys are assumed to be available to parties at the time of signing. However, at the point where these schemes begin to offer security over plain FROST, maintaining pairwise keys begins to incur memory cost linear in the number of signers.

For example, in the 128-bit security setting, each pairwise symmetric key occupies 16 bytes. With 132 parties (the point at which the security of ms-FROST and FaFROST improves over plain FROST), each party stores keys shared with the other 131 parties.⁴ Using one masking key per pair incurs the following memory overhead per party:

$$\text{FaFROST / ms-FROST: } (132 - 1) \times 16 = 2,096 \text{ bytes} \approx 2.05 \text{ KiB.}$$

To improve on this state, each party can instead derive pairwise symmetric keys using non-interactive key exchange. For example, Diffie-Hellman identity keys can be generated at the time of key generation, and then pairwise masks can be derived on the fly at the time of signing. In this setting, memory overhead for each party's secret signing key will remain constant, but computation for each

⁴ ms-FROST is written to employ two masking keys per pair, but can use one masking key as in FaFROST with minor changes to the security proof.

participant will increase. This tradeoff in memory overhead for computation may be acceptable in settings where one entity represents many signing keys, such as a custodian.

For example, when $n = 500$ and $t = 250$, we can compare the overhead of the two approaches to masking, as follows.

$$\begin{aligned} \text{FaFROST / ms-FROST: } & 249 \text{ hash evaluations,} \\ \text{DH-derived: } & 249 \text{ exponentiations} \\ & + 498 H_{\text{mask}} \text{ evaluations.} \end{aligned}$$

We show this construction in more detail in Section 5.

Compressing Pairwise Keys via Key Predistribution. Another direction for improved masking techniques is employing key predistribution techniques for pairwise keys, introduced by Paterson and Stinson [19]. In the construction by Paterson and Stinson, one endpoint of each pair derives its shared key by hashing a master seed with the participant identifiers. The other endpoint stores the derived key.

In the setting of masked FROST, this method approximately halves the key storage per party compared with storing every pairwise key. For settings of 128-bit generic security, the overhead of this approach improves to approximately 1 KiB per party, a 50% improvement over FaFROST and ms-FROST.

We show this variant in greater detail in Section 6.

1.4 Acknowledgements

We would like to thank Alex Rodríguez García for his helpful feedback. In particular, we are grateful for discussion of the gap between the unconditional LDVR bound and the concrete attacks [20].

Statement on AI Use. All work was reviewed and edited by the authors. Open AI’s GPT-6 Astra was used to summarize related work, generate Figure 1, derive overhead of existing masking techniques, propose bounds on the cost of each additional signing party, and to generate Figure 6. Additionally, this model suggested employing the technique by Paterson and Stinson [19] to achieve a 50% improvement in memory overhead of masking keys, and generated the figure of this variant.

2 Preliminaries

We represent a security parameter as $\lambda \in \mathbb{Z}$. We denote the assignment of an element y to the value x as $y \leftarrow x$, and sampling an element from some set S uniformly at random as $x \xleftarrow{\$} S$. For a randomized algorithm A , we write $x \xleftarrow{\$} A()$ to indicate the random variable x that is output from the execution of A . We

MAIN $\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\text{dl}}(\lambda)$	MAIN $\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-}(\text{a})\text{omdl}}(\lambda)$	$\mathcal{O}^{\text{dl}}(X, \alpha, (\beta_i)_{i=1}^{\ell})$
$(\mathbb{G}, p, g) \leftarrow \text{GrpGen}(1^\lambda)$ $x \leftarrow \mathbb{Z}_p; X \leftarrow g^x$ $x' \leftarrow \mathcal{A}((\mathbb{G}, p, g), X)$ if $x' = x$ return 1 return 0	$(\mathbb{G}, p, g) \leftarrow \text{GrpGen}(1^\lambda)$ $\text{ctr} \leftarrow 0$ for $i \in [\ell]$ do $x_i \leftarrow \mathbb{Z}_p; X_i \leftarrow g^{x_i}$ $\mathbf{x} \leftarrow (x_1, \dots, x_\ell)$ $\mathbf{X} \leftarrow (X_1, \dots, X_\ell)$ $\mathbf{x}' \leftarrow \mathcal{A}^{\mathcal{O}^{\text{dl}}}((\mathbb{G}, p, g), \mathbf{X})$ if $\mathbf{x}' = \mathbf{x}$ return 1 return 0	$// X = g^\alpha \prod_{i=1}^{\ell} X_i^{\beta_i}$ if $\text{ctr} = \ell - 1$, return $\perp$ $\text{ctr} \leftarrow \text{ctr} + 1$ $x \leftarrow \alpha + \sum_{i=1}^{\ell} \beta_i \cdot x_i$ return x $x \leftarrow \text{dlog}(X)$ return x

Fig. 3: Experiments defining DL, OMDL, and AOMDL. Boxes mark the differences for AOMDL; $\text{dlog}(X)$ returns the discrete logarithm of X to base g .

say that f is a negligible function of λ if it holds that $\lim_{\lambda \rightarrow \infty} f(\lambda)\lambda^c = 0$, for all $c > 0$ [6].

We use $[n]$ to represent the set $\{1, \dots, n\}$, and represent integer vectors in bold-face font. We represent the falling factorial as $(n)_k = n(n-1) \cdots (n-k+1)$ for integers $k > 0$, with $(n)_k = 1$ for $k \leq 0$.

Polynomial Interpolation. A polynomial $f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1}$ of degree $t-1$ over a field $\mathbb{F}$ can be interpolated by t points. Let η be a set of t distinct integers corresponding to the x -coordinates $x_i \in \mathbb{F}, i \in \eta$, of these points. Then the Lagrange polynomial $L_i(x)$ has the form:

$$L_i(x) = \prod_{j \in \eta; j \neq i} \frac{x - x_j}{x_i - x_j} \quad (3)$$

Given a set of t points $\{(x_i, f(x_i))\}_{i \in \eta}$, any point $f(x_\ell)$ on the polynomial f can be determined by Lagrange interpolation as follows:

$$f(x_\ell) = \sum_{k \in \eta} f(x_k) \cdot L_k(x_\ell)$$

Group Generation. Let $\mathbb{G}$ be a cyclic group of prime order p , and $\mathbb{Z}_p$ be the field of integers modulo p . Let g be a generator of $\mathbb{G}$. **GrpGen** is then a probabilistic polynomial-time algorithm that takes as input a security parameter 1^λ and outputs a group description $(\mathbb{G}, p, g)$ consisting of a group $\mathbb{G}$ of order p , where p is a 2λ -bit prime, and a generator g of $\mathbb{G}$.

2.1 Assumptions

Assumption 1 (Discrete Logarithm (DL) Assumption). *Define*

$$\text{Adv}_{\mathcal{A}, \text{GrpGen}}^{\text{dl}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\text{dl}}(\lambda) = 1],$$

where $\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\text{dl}}(\lambda)$ is defined in Figure 3. The DL assumption holds relative to GrpGen if $\text{Adv}_{\mathcal{A}, \text{GrpGen}}^{\text{dl}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

Assumption 2 (One-More Discrete Logarithm (OMDL) Assumption). *Define*

$$\text{Adv}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-omdl}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-omdl}}(\lambda) = 1],$$

where $\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-omdl}}(\lambda)$ is defined in Figure 3. The ℓ -OMDL assumption holds relative to GrpGen if $\text{Adv}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-omdl}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

Assumption 3 (Algebraic One-More Discrete Logarithm (AOMDL) Assumption). *Define*

$$\text{Adv}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-aomdl}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-aomdl}}(\lambda) = 1],$$

where $\text{Expt}_{\mathcal{A}, \text{GrpGen}}^{\ell\text{-aomdl}}(\lambda)$ is defined in Figure 3. The ℓ -AOMDL assumption holds relative to GrpGen if $\text{Adv}_{\mathcal{A}}^{\ell\text{-aomdl}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

2.2 Adaptive Security

Definition 1 (Adaptive Security). For an adversary $\mathcal{A}$ playing the adaptive security game against threshold signature scheme TS as shown in Figure 4, define

$$\text{Adv}_{\mathcal{A}, \text{TS}}^{\text{adp-TS-UF}}(\lambda, n, t, t_c) = \Pr[\text{Expt}_{\mathcal{A}, \text{TS}}^{\text{adp-TS-UF}}(\lambda, n, t, t_c) = 1].$$

TS is (n, t, t_c) -adaptively secure if for all PPT adversaries $\mathcal{A}$, the advantage $\text{Adv}_{\mathcal{A}, \text{TS}}^{\text{adp-TS-UF}}(\lambda, n, t, t_c)$ is negligible. TS has full adaptive security if it is $(n, t, t-1)$ -adaptively secure for any $t \leq n$.

3 Low-Dimensional Vector Representation (LDVR) Assumption

The Low-Dimensional Vector Representation (LDVR) Assumption was first introduced by Crites et al. [13].

The problem can be given as follows, for parameters (t_c, t, n) , where $t_c < t \leq n$. For a threshold scheme, t_c is the number of allowed corrupted parties, t is the reconstruction threshold, and n is the total number of possible parties. Let $\mathbf{v}_i$ be the Vandermonde vector $\mathbf{v}_i = (\mathbf{1}, \mathbf{i}, \mathbf{i}^2, \dots, \mathbf{i}^{t-1}) \in \mathbb{Z}_p^t$. Let q_{ldvr} be the number of queries the LDVR adversary can make to $\mathcal{O}_{\text{ldvr}}$; without loss of generality, we assume an adversary that makes exactly q_{ldvr} number of queries. For the description below, $0 \leq t_c \leq t-1 < n < p$.

The LDVR game is defined in Figure 5. Informally:

MAIN $\text{Expt}_{\mathcal{A}, \text{TS}}^{\text{adp-TS-UF}}(\lambda, n, t, t_c)$	$\text{Sign}_1^*(k)$
$\text{par} \leftarrow \text{Setup}(1^\lambda)$ $Q_{\text{st}} \leftarrow \emptyset, Q_m \leftarrow \emptyset$ $\text{corrupt} \leftarrow \emptyset, \text{honest} \leftarrow [n]$ $(pk, \{(pk_i, sk_i)\}_{i \in [n]}) \leftarrow \text{KeyGen}(n, t)$ $\text{input} \leftarrow (\text{par}, pk, \{pk_i\}_{i \in [n]})$ for $i \in [n]$ do $\text{ctr}_i \leftarrow 0$ $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}_1^*, \text{Sign}_2^*, \text{corrupt}}(\text{input})$ return 1 if $m^* \notin Q_m$ $\quad \wedge \text{Verify}(pk, m^*, \sigma^*) = 1$ return 0	return $\perp$ if $k \notin \text{honest}$ $\text{ctr}_k \leftarrow \text{ctr}_k + 1$ $\text{eid} \leftarrow \text{ctr}_k$ $(\text{pm}_{k, \text{eid}, 1}, \text{st}_{k, \text{eid}, 1}) \leftarrow \text{Sign}_1(k, sk_k)$ $Q_{\text{st}}[k, \text{eid}, 1] \leftarrow \text{st}_{k, \text{eid}, 1}$ return $(\text{eid}, \text{pm}_{k, \text{eid}, 1})$
$\text{corrupt}(k)$	$\text{Sign}_2^*(k, \text{eid}, C, m, \{\text{pm}_{i, \text{eid}, 1}\}_{i \in C})$
return $\perp$ if $(k \notin \text{honest}) \vee (\text{corrupt} \geq t_c)$ $\text{corrupt} \leftarrow \text{corrupt} \cup \{k\}$ $\text{honest} \leftarrow \text{honest} \setminus \{k\}$ $\text{state}_k \leftarrow Q_{\text{st}}[k, \cdot, \cdot] \quad // \text{ all state for party } k$ return (sk_k, state_k)	return $\perp$ if $k \notin \text{honest} \vee Q_{\text{st}}[k, \text{eid}, 1] = \perp$ return $\perp$ if $Q_{\text{st}}[k, \text{eid}, 2] \neq \perp$ parse $\text{st}_{k, \text{eid}, 1} \leftarrow Q_{\text{st}}[k, \text{eid}, 1]$ $Q_m \leftarrow Q_m \cup \{m\}$ $\text{pm}_{k, \text{eid}, 2}$ $\quad \leftarrow \text{Sign}_2(k, \text{st}_{k, \text{eid}, 1}, C, m, \{\text{pm}_{i, \text{eid}, 1}\}_{i \in C})$ $Q_{\text{st}}[k, \text{eid}, 2] \leftarrow \text{pm}_{k, \text{eid}, 2}$ return $\text{pm}_{k, \text{eid}, 2}$

Fig. 4: Experiment for defining adaptive unforgeability. The public parameters par are implicitly given as input to all algorithms. Here, $t \leq n$ is the reconstruction threshold and $t_c \leq t - 1$ is the maximum number of signers $\mathcal{A}$ can corrupt.

1. The adversary is allowed to query the LDVR challenge oracle up to q_{ldvr} number of times, such that $\text{sid} \in [q_{\text{ldvr}}]$, with $(\alpha_{i, \text{sid}})_{i=0}^n$ as input.
2. For each query, the adversary chooses coefficients $(\alpha_{i, \text{sid}}) \in \mathbb{Z}_p^{n+1}$, and sends this as input to the LDVR oracle.
3. In return, the adversary receives the random challenge $c_{\text{sid}} \in \mathbb{Z}_p$ from the LDVR oracle.
4. The adversary computes $\mathbf{w}_{\text{sid}} \in \mathbb{Z}_p^t$, as given in Equation 4:

$$\mathbf{w}_{\text{sid}} = c_{\text{sid}} \mathbf{V}_0 + \sum_{i=0}^n \alpha_{i, \text{sid}} \mathbf{V}_i \quad (4)$$

5. The adversary then outputs the tuple $(\text{corrupt}, \text{sid})$, where $\text{sid} \in [q_{\text{ldvr}}]$, and such that $|\text{corrupt}| \leq t_c$, and $\text{corrupt} \subseteq [n]$.
6. The adversary wins if $\text{sid} \in [q_{\text{ldvr}}]$, and the following relation holds:

$$\mathbf{w}_{\text{sid}} \in \text{span}\{\mathbf{v}_i\}_{i \in \text{corrupt}} \quad (5)$$

MAIN $\text{Expt}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda)$	$\mathcal{O}_{\text{ldvr}}(\text{sid}, (\alpha_{i, \text{sid}})_{i=0}^n)$
for $i \in (0, \dots, n)$ do	if $\text{sid} \notin [q_{\text{ldvr}}] \vee Q[\text{sid}] = 1$
$\mathbf{v}_i \leftarrow (\mathbf{1}, i, i^2, \dots, i^{t-1}) \in \mathbb{Z}_p^t$	return $\perp$
for $\text{sid} \in [q_{\text{ldvr}}]$ do	if $(\alpha_{i, \text{sid}})_{i=0}^n \notin \mathbb{Z}_p^{n+1}$
$Q[\text{sid}] \leftarrow 0$	return $\perp$
$(\text{corrupt}, \text{sid}) \leftarrow \mathcal{C}^{\mathcal{O}_{\text{ldvr}}}(t_c, t, n)$	$Q[\text{sid}] \leftarrow 1$
if $Q[\text{sid}] = 0 \vee \text{sid} \notin [q_{\text{ldvr}}]$	$c_{\text{sid}} \leftarrow \mathbb{Z}_p$
return 0	$\mathbf{w}_{\text{sid}} \leftarrow c_{\text{sid}} \mathbf{v}_0 + \sum_{i=0}^n \alpha_{i, \text{sid}} \mathbf{v}_i$
if $ \text{corrupt} > t_c \vee \text{corrupt} \not\subseteq [n]$	return c_{sid}
return 0	
if $\mathbf{w}_{\text{sid}} \in \text{span}\{\mathbf{v}_i\}_{i \in \text{corrupt}}$	
return 1	
return 0	

Fig. 5: Experiment for defining the LDVR assumption.

Assumption 4 (Low Dimensional Vector Representation (LDVR) Assumption.).
Define

$$\text{Adv}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda) = \Pr[\text{Expt}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda) = 1],$$

where $\text{Expt}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda)$ is defined in Figure 5. The (t_c, t, n) -LDVR assumption holds if $\text{Adv}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{C}$.

3.1 Concrete Hardness of LDVR

Crites et al. [13] showed that the advantage for *any* LDVR adversary $\mathcal{C}$ can be concretely upper-bounded. In the context of threshold signatures, an LDVR adversary is bounded by the total number of parties n , the signing threshold t , the corruption threshold t_c , as well as the number of allowed oracle queries q_{ldvr} , and the order of the group p (determined by the security parameter λ). We show the generalized parameter here, for reference.

We show this concrete bound in Equation 6:

$$\text{Adv}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda) \leq q_{\text{ldvr}} \frac{(n)_{2t_c - t + 1}}{p(t_c)_{2t_c - t + 1}} \quad (6)$$

where $(n)_k$ is the falling factorial as defined in Section 2.

4 Case Study: LDVR Implication for FROST

While the LDVR assumption is required for the adaptive security of many threshold Schnorr signature schemes, as discussed in Section 1, the main question

is how the assumption impacts the underlying security of the scheme, in light of the recent results by Rodríguez García and Tessaro [20].

Here, we look at the impact of the LDVR assumption with one particular case study, FROST [16]. In particular, we describe how even if the hardness of the LDVR assumption is no better than the bound given by Crites et al. [13], (going even beyond the concrete attacks by Rodríguez García and Tessaro [20]), how the resulting security of FROST is impacted, depending on the choices of the number of parties n and corruption threshold t_c . Our analysis is for FROST specifically, but likewise applies to other impacted schemes [2, 5, 14, 17, 18, 21, 22].

As such, we upper-bound the strongest possible advantage for an LDVR adversary, by the following:

$$\mathbf{Adv}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda) \leq q_{\text{ldvr}} \frac{(n)_{2t_c-t+1}}{p(t_c)_{2t_c-t+1}} \quad (7)$$

Next, we recall the statement of (full) adaptive security of FROST, as given by Crites et al. [13], in Theorem 1.

Theorem 1 (Full Adaptive Security of FROST [13]). *FROST is adaptively unforgeable in the AGM and ROM under the AOMDL and LDVR assumptions. Concretely, for any algebraic PPT adversary $\mathcal{A}$ making at most q_s queries to Sign_1 and at most q_h random oracle queries, there exists a PPT adversary $\mathcal{B}$ and a PPT adversary $\mathcal{C}$, who makes at most q queries to its oracle, such that*

$$\begin{aligned} \mathbf{Adv}_{\mathcal{A}, \text{FROST}}^{\text{adp-TS-UF}}(\lambda, n, t, t_c) &\leq \mathbf{Adv}_{\mathcal{B}, \text{GrpGen}}^{(2q_s+t)\text{-aomdl}}(\lambda) \\ &\quad + \mathbf{Adv}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda) + \frac{2(nq)^2}{p} \end{aligned} \quad (8)$$

where $q = q_s + q_h + 1$, and where $q_{\text{ldvr}} \leq q$. The running times of $\mathcal{B}$ and $\mathcal{C}$ are roughly that of $\mathcal{A}$. $\mathcal{B}$ makes at most $2q_s + t - 1$ discrete-logarithm queries.

In summary, the advantage of an adversary $\mathcal{A}$ against the unforgeability of FROST is upper-bounded by the advantage of two different adversaries; (1) the advantage of an adversary $\mathcal{B}$ against the AOMDL experiment, and (2) the advantage of an adversary $\mathcal{C}$ against the LDVR experiment.

4.1 Modeling FROST Security with Unconditional LDVR Security

In light of the recent results by Rodríguez García and Tessaro [20], it is important to understand the resulting impact on FROST and related schemes. Given the unconditional LDVR hardness bound shown in Equation 6, we can likewise model the security of FROST with respect to this unconditional hardness bound.

We show the bound on the advantage of an adversary against the unforgeability of FROST, assuming the unconditional LDVR bound, in Equation 9.

$$\begin{aligned} \mathbf{Adv}_{\mathcal{A}, \text{FROST}}^{\text{adp-TS-UF}}(\lambda, n, t, t_c) &\leq \mathbf{Adv}_{\mathcal{B}, \text{GrpGen}}^{(2q_s + t)\text{-aomdl}}(\lambda) \\ &\quad + \frac{q(n)_{2t_c-t+1}}{p(t_c)_{2t_c-t+1}} + \frac{2(nq)^2}{p}. \end{aligned} \quad (9)$$

4.2 For What Threshold Parameters Does LDVR Impact Affected Schemes?

We revisit the setting in which the *best possible* LDVR attacker imposes additional security risks to relevant threshold Schnorr signature schemes [2, 5, 14, 16, 17, 18, 21, 22], assuming the unconditional bound. We assume a group of order $p \approx 2^{256}$, providing approximately 128 bits of generic (discrete log) security.

Concretely using Equation 7, we seek to establish parameters $0 \leq t_c \leq t-1 < n$ such that

$$\begin{aligned} \mathbf{Adv}_{\mathcal{C}}^{(t_c, t, n)\text{-ldvr}}(\lambda) &\leq q_{ldvr} \frac{\binom{n}{2t_c-t+1}}{p(t_c)_{2t_c-t+1}} \\ &\leq \frac{q_{ldvr}}{2^{128}} \quad // \text{ Intended security bound} \end{aligned} \quad (10)$$

For $q_{ldvr} > 0$, cancelling q_{ldvr} gives

$$\frac{\binom{n}{2t_c-t+1}}{(t_c)_{2t_c-t+1}} \leq \frac{p}{2^{128}}. \quad (11)$$

For all permitted reconstruction and corruption thresholds,

$$\frac{\binom{n}{2t_c-t+1}}{(t_c)_{2t_c-t+1}} \leq \binom{n}{t_c} \leq \binom{n}{\lfloor n/2 \rfloor}, \quad (12)$$

where the inequalities are tight for the worst case of $t_c = t-1 = \lfloor n/2 \rfloor$.

For Equation 11 to hold for every threshold in the full adaptive setting, it is therefore sufficient that Equation 13 holds:

$$\binom{n}{\lfloor n/2 \rfloor} \leq \frac{p}{2^{128}}. \quad (13)$$

For $p \approx 2^{256}$, the right-hand side is approximately 2^{128} . We have

$$\begin{aligned} \log_2 \binom{131}{65} &\approx 127.1493225247, \\ \log_2 \binom{132}{66} &\approx 128.1493225247. \end{aligned} \quad (14)$$

Equation 13 holds for 128-bit secure groups, for every threshold in the full adaptive setting, when $n \leq 131$. At $n = 132$, with full-corruption choices

$t - 1 = t_c = 64, \dots, 68$, we see that $\binom{n}{t-1} > 2^{128}$, with signing thresholds $65, \dots, 69$, meaning that the security of FROST drops (slightly) below the bound of generic (discrete-log) security for 128-bit secure groups. We show in Figure 1 and Figure 2 a range of threshold parameters and their corresponding security, assuming the unconditional LDVR bound.

4.3 Measuring the Cost of Each Additional Party

By employing the upper bound on the advantage of *any* LDVR adversary as given in Equation 7, going beyond even the attack by Rodríguez García and Tessaro [20], we can model the resulting security loss incurred by each additional signer, in the full corruption setting, as follows. From Equation 11 and the tightness of Equation 12 in the worst-case setting $t_c = t - 1 = \lfloor n/2 \rfloor$, LDVR provides $\log_2 p - \log_2 \binom{n}{\lfloor n/2 \rfloor}$ bits of security in that setting. Each additional signer therefore reduces the security level by $\Delta_{n \rightarrow n+1} = \log_2 \binom{n+1}{\lfloor (n+1)/2 \rfloor} - \log_2 \binom{n}{\lfloor n/2 \rfloor}$ bits.

For $n = 131$, $n + 1 = 132$, $\Delta_{n \rightarrow n+1} = \Delta_{131 \rightarrow 132} = 1$ bit exactly. Further analysis shows that for each additional signer, security is similarly reduced by at most one bit.

Specifically, we can bound $\Delta_{n \rightarrow n+1}$ as follows:

$$\begin{aligned}
0 \leq \Delta_{n \rightarrow n+1} &= \log_2 \binom{n+1}{\lfloor (n+1)/2 \rfloor} - \log_2 \binom{n}{\lfloor n/2 \rfloor} \\
&= \log_2 \left(\frac{\binom{n+1}{\lfloor (n+1)/2 \rfloor}}{\binom{n}{\lfloor n/2 \rfloor}} \right) \quad (\text{logarithm quotient rule}) \\
&\leq \log_2 \left(\frac{2 \binom{n}{\lfloor n/2 \rfloor}}{\binom{n}{\lfloor n/2 \rfloor}} \right) \quad (\text{by Pascal's identity}) \\
&= \log_2 2 = 1.
\end{aligned} \tag{15}$$

5 FROST with DH-Derived Masking Keys

For settings where higher numbers of parties are required, using masking is a simple technique to allow for full adaptive security. Here, we show a simple variant to FROST using pairwise masking, building on prior masking techniques [1, 3], but where pairwise keys are derived on the fly from identity keys, allowing for constant-sized memory overhead as the number of signers grows.

In more detail, Figure 6 adapts FROST to use pairwise masking as in ms-FROST [1] and FaFROST [3], but where pairwise keys are derived on the fly from Diffie-Hellman (DH) identity keys. At the time of key generation, each party i will additionally derive an independent DH identity key pair (sk_i^I, pk_i^I) , where $pk_i^I = g^{sk_i^I}$. Additionally, signing-key commitments $g^{sk_i^S}$ are *not* published.

Setup (1^λ) $(\mathbb{G}, p, g) \leftarrow \text{GrpGen}(1^\lambda)$ $H_{\text{non}}, H_{\text{sig}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ $H_{\text{mask}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ $\text{par} \leftarrow ((\mathbb{G}, p, g), H_{\text{non}}, H_{\text{sig}}, H_{\text{mask}})$ return par	Sign ₁ ($pk, \{pk_i^I\}_{i \in [n]}, k, sk_k^S, \boxed{sk_k^I}$) $(r_k, s_k) \leftarrow \mathbb{Z}_p^2$ $R_k \leftarrow g^{r_k}; \quad S_k \leftarrow g^{s_k}$ $\text{state}_{k,1} \leftarrow (R_k, r_k, S_k, s_k, sk_k^S, \boxed{sk_k^I},$ $\quad pk, \{pk_i^I\}_{i \in [n]})$ return $((R_k, S_k), \text{state}_{k,1})$
KeyGen (n, t) $sk \leftarrow \mathbb{Z}_p; \quad pk \leftarrow g^{sk}$ $\{(i, sk_i^S)\}_{i \in [n]}$ $\leftarrow \text{Shamir.IssueShares}(sk, n, t)$ for $i \in [n]$ do $sk_i^I \leftarrow \mathbb{Z}_p \setminus \{0\}; \quad pk_i^I \leftarrow g^{sk_i^I}$ return $(pk, \{pk_i^I\}_{i \in [n]},$ $\quad \{(sk_i^S, \boxed{sk_i^I})\}_{i \in [n]})$	Sign ₂ ($k, \text{state}_{k,1}, C, m, \mathcal{PM}_1$) return $\perp$ if $C \not\subseteq [n] \vee C < t \vee k \notin C$ parse $(R'_k, r_k, S'_k, s_k, sk_k^S, \boxed{sk_k^I},$ $\quad pk, \{pk_i^I\}_{i \in [n]}) \leftarrow \text{state}_{k,1}$ parse $\{(i, R_i, S_i)\}_{i \in C} \leftarrow \mathcal{PM}_1$ return $\perp$ if $(R'_k, S'_k) \neq (R_k, S_k)$ for $i \in C$ do $a_i \leftarrow H_{\text{non}}(i, pk, C, m, \mathcal{PM}_1)$ $\tilde{R} \leftarrow \prod_{i \in C} R_i S_i^{a_i}$ $c \leftarrow H_{\text{sig}}(\tilde{R}, pk, m)$
Combine ($C, m, \mathcal{PM}_1, \mathcal{PM}_2$) parse $\{\tilde{z}_i\}_{i \in C} \leftarrow \mathcal{PM}_2$ parse $\{(i, R_i, S_i)\}_{i \in C} \leftarrow \mathcal{PM}_1$ for $i \in C$ do $a_i \leftarrow H_{\text{non}}(i, pk, C, m, \mathcal{PM}_1)$ $\tilde{R} \leftarrow \prod_{i \in C} R_i S_i^{a_i}$ $z \leftarrow \sum_{i \in C} \tilde{z}_i$ return $(\tilde{R}, z)$	sid $\leftarrow (pk, m, C, \mathcal{PM}_1)$ for $j \in C \setminus \{k\}$ do $K_{k,j} \leftarrow (pk_j^I)^{sk_k^I}$ $\rho_{k,j} \leftarrow H_{\text{mask}}(pk, k, pk_k^I, j, pk_j^I, K_{k,j}, \text{sid})$ $\rho_{j,k} \leftarrow H_{\text{mask}}(pk, j, pk_j^I, k, pk_k^I, K_{k,j}, \text{sid})$ $\text{mask}_k \leftarrow \sum_{j \in C \setminus \{k\}} (\rho_{k,j} - \rho_{j,k})$ $\tilde{z}_k \leftarrow r_k + s_k a_k + c \lambda_k^C sk_k^S + \text{mask}_k$ return $\tilde{z}_k$
Verify (pk, m, σ) parse $(\tilde{R}, z) \leftarrow \sigma$ $c \leftarrow H_{\text{sig}}(\tilde{R}, pk, m)$ return $(g^z = \tilde{R} \cdot pk^c)$	

Fig. 6: FROST with pairwise masking keys derived by a DH NIKE in round two. Boxes mark changes from plain FROST. The public parameters par , n , t are implicitly given as input to all algorithms. Here $\lambda_k^C = \prod_{j \in C \setminus \{k\}} j/(j-k)$. Lists are ordered by participant identifier.

In round two, in addition to the plain FROST signing operations, party k computes the DH shared secret $K_{k,j} = (pk_j^I)^{sk_k^I}$ for each other signer j . Both parties obtain the same DH shared secret, $K_{k,j} = K_{j,k} = g^{sk_k^I sk_j^I}$, and derive directional masks $\rho_{k,j}$ and $\rho_{j,k}$ by hashing these values with their (ordered) identities, pk , and the session identifier sid .

For a derived session identifier sid , the masks satisfy

$$\sum_{k \in C} \text{mask}_k = \sum_{k \in C} \sum_{j \in C \setminus \{k\}} (\rho_{k,j} - \rho_{j,k}) = 0, \quad (16)$$

so correctness for plain Schnorr signatures holds.

6 FROST with Compressed Masking Keys

We next give a variant of masked FROST using hash-based key predistribution to lower the memory overhead required by masking keys by 50%. We show this variant in Figure 7.

We build upon the technique of key predistribution by Paterson and Stinson [19]. In their construction, one endpoint of each pair derives its shared key by hashing a master seed with the participant identifiers, while the other endpoint stores the derived key. Balancing these assignments approximately halves the key storage per party compared with storing every pairwise key.

Figure 7 gives a variant of FROST using this key predistribution technique. Each party stores one master seed and approximately half the pairwise keys; the remaining keys are derived at the time of signing. Parties are assigned to either be an “owner” of a key or a “receiver”; the owner derives its copy of the key, while the receiver stores its copy.

For identifiers $i, j \in [n]$, Equation 17 gives a deterministic function to determine which of these parties is the owner of a given pairwise key: Importantly, Equation 17 satisfies $\text{owner}(i, j) = \text{owner}(j, i)$ and assigns at most $\lceil (n-1)/2 \rceil$ keys for each party to store.

$$\text{owner}(i, j) = \begin{cases} i, & 1 \leq (j - i) \bmod (2\lfloor n/2 \rfloor + 1) \leq \lfloor n/2 \rfloor, \\ j, & \text{otherwise.} \end{cases} \quad (17)$$

We define Figure 7 with a trusted dealer, who samples a master seed $\text{seed}_i \leftarrow \$\{0, 1\}^\lambda$ for each party i . When $\text{owner}(i, j) = i$, the dealer computes $\text{seed}_{i,j} = H_{\text{seed}}(\text{seed}_i, pk, i, j)$ and stores it in $\text{MaskingKeys}_j[i]$. The dealer securely distributes $(sk_i^S, \text{seed}_i, \text{MaskingKeys}_i)$ to each party i . However, in a distributed setting where parties perform key generation in a distributed manner, owners can generate their masking keys directly, and send the derived pairwise keys to the recipients using a private and authenticated channel.

Let $\text{MaskingKeys}_i[j]$ denote a key stored by receiver i for owner j . Both endpoints use the same mask $\rho_{i,j} = H_{\text{mask}}(\text{seed}_{i,j}, \text{sid})$ with opposite signs: $\Delta_{i,j} =$

Setup (1^λ) $(\mathbb{G}, p, g) \leftarrow \text{GrpGen}(1^\lambda)$ $H_{\text{non}}, H_{\text{sig}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ $H_{\text{seed}} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ $H_{\text{mask}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ $\text{par} \leftarrow ((\mathbb{G}, p, g), H_{\text{non}}, H_{\text{sig}}, H_{\text{seed}}, H_{\text{mask}})$ return par KeyGen (n, t) $sk \leftarrow \mathbb{Z}_p; \quad pk \leftarrow g^{sk}$ $\{(i, sk_i^S)\}_{i \in [n]}$ $\leftarrow \text{Shamir.IssueShares}(sk, n, t)$ for $i \in [n]$ do $\text{seed}_i \leftarrow \mathbb{Z}_p$ $\text{MaskingKeys}_i \leftarrow \emptyset$ for $i, j \in [n], i \neq j$ do if $\text{owner}(i, j) = i$ do $\text{seed}_{i,j} \leftarrow H_{\text{seed}}(\text{seed}_i, pk, i, j)$ $\text{MaskingKeys}_j[i] \leftarrow \text{seed}_{i,j}$ return $(pk, \{(sk_i^S, \text{seed}_i, \text{MaskingKeys}_i)\}_{i \in [n]})$ Combine ($C, m, \mathcal{PM}_1, \mathcal{PM}_2$) parse $\{\tilde{z}_i\}_{i \in C} \leftarrow \mathcal{PM}_2$ parse $\{(i, R_i, S_i)\}_{i \in C} \leftarrow \mathcal{PM}_1$ for $i \in C$ do $a_i \leftarrow H_{\text{non}}(i, pk, C, m, \mathcal{PM}_1)$ $\tilde{R} \leftarrow \prod_{i \in C} R_i S_i^{a_i}$ $z \leftarrow \sum_{i \in C} \tilde{z}_i$ return $(\tilde{R}, z)$	Sign ₁ ($pk, k, sk_k^S, \text{seed}_k, \text{MaskingKeys}_k$) $(r_k, s_k) \leftarrow \mathbb{Z}_p^2$ $R_k \leftarrow g^{r_k}; \quad S_k \leftarrow g^{s_k}$ $\text{state}_{k,1} \leftarrow (R_k, r_k, S_k, s_k, sk_k^S, pk, \text{seed}_k, \text{MaskingKeys}_k)$ return $((R_k, S_k), \text{state}_{k,1})$ Sign ₂ ($k, \text{state}_{k,1}, C, m, \mathcal{PM}_1$) return $\perp$ if $C \not\subseteq [n] \vee C < t \vee k \notin C$ parse $(R'_k, r_k, S'_k, s_k, sk_k^S, pk, \text{seed}_k, \text{MaskingKeys}_k) \leftarrow \text{state}_{k,1}$ parse $\{(i, R_i, S_i)\}_{i \in C} \leftarrow \mathcal{PM}_1$ return $\perp$ if $(R'_k, S'_k) \neq (R_k, S_k)$ for $i \in C$ do $a_i \leftarrow H_{\text{non}}(i, pk, C, m, \mathcal{PM}_1)$ $\tilde{R} \leftarrow \prod_{i \in C} R_i S_i^{a_i}$ $c \leftarrow H_{\text{sig}}(\tilde{R}, pk, m)$ $\text{sid} \leftarrow (pk, m, C, \mathcal{PM}_1)$ for $j \in C \setminus \{k\}$ do if $\text{owner}(k, j) = k$ do $\text{seed}_{k,j} \leftarrow H_{\text{seed}}(\text{seed}_k, pk, k, j)$ else $\text{seed}_{k,j} \leftarrow \text{MaskingKeys}_k[j]$ $\rho_{k,j} \leftarrow H_{\text{mask}}(\text{seed}_{k,j}, \text{sid})$ $\text{mask}_k \leftarrow \sum_{j \in C \setminus \{k\}} \Delta_{k,j} \rho_{k,j}$ $\tilde{z}_k \leftarrow r_k + s_k a_k + c \lambda_k^C sk_k^S + \text{mask}_k$ return $\tilde{z}_k$ Verify (pk, m, σ) parse $(\tilde{R}, z) \leftarrow \sigma$ $c \leftarrow H_{\text{sig}}(\tilde{R}, pk, m)$ return $(g^z = \tilde{R} \cdot pk^c)$
--	---

Fig. 7: FROST with compressed masking keys. Boxes mark additions to plain FROST. A trusted dealer runs **KeyGen** and distributes each party's secret outputs over authenticated, confidential channels. Parameters par , n , t are implicit. The function **owner** is defined in Equation 17. Here $\Delta_{i,j} = 1$ if $i < j$ and -1 otherwise, and $\lambda_k^C = \prod_{j \in C \setminus \{k\}} j/(j-k)$. In a distributed setting, owners can generate their masking keys themselves and send the derived pairwise masking key to the receiver over an authenticated and confidential channel.

1 for $i < j$ and -1 otherwise. If C is the set of signing parties, and $\text{mask}_i \leftarrow \sum_{j \in C \setminus \{i\}} \Delta_{i,j} \rho_{i,j}$, then $\sum_{i \in C} \text{mask}_i = 0$.

For $\lambda = 128$, the maximum masking-key storage per party is

$$\begin{aligned} n = 132 : \quad & \left(1 + \left\lceil \frac{131}{2} \right\rceil\right) 16 = 1,072 \text{ bytes} \approx 1.05 \text{ KiB}, \\ n = 500 : \quad & \left(1 + \left\lceil \frac{499}{2} \right\rceil\right) 16 = 4,016 \text{ bytes} \approx 3.92 \text{ KiB}. \end{aligned} \tag{18}$$

This is approximately half the memory overhead required for ms-FROST and FaFROST. When all 132 parties sign, there are $\binom{132}{2}$ key derivations across the signing set, averaging 65.5 per signer.

References

- [1] R. Bacho, Y. Chen, J. Loss, S. Tessaro, and C. Zhu. *Adaptively Secure Partially Non-Interactive Threshold Schnorr Signatures in the AGM*. Cryptology ePrint Archive, Paper 2025/1953. 2025. URL: <https://eprint.iacr.org/2025/1953>.
- [2] R. Bacho, J. Loss, G. Stern, and B. Wagner. *HARTS: High-Threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures*. Available at <https://eprint.iacr.org/2024/280>. 2024.
- [3] R. Baecker, P. Gerhart, D. Li Calsi, L. Russo, D. Schröder, and A. Yerukhimovich. *Fully Adaptive FROST with Identifiable Aborts from AOMDL*. Cryptology ePrint Archive, Paper 2025/1950. 2025. URL: <https://eprint.iacr.org/2025/1950>.
- [4] M. Bellare, E. Crites, C. Komlo, M. Maller, S. Tessaro, and C. Zhu. “Better than advertised security for non-interactive threshold signatures”. In: *Crypto 2022*. Vol. 13510. LNCS. Springer, 2022, pp. 517–550. DOI: 10.1007/978-3-031-15985-5_18.
- [5] F. Benhamouda, S. Halevi, H. Krawczyk, Y. Ma, and T. Rabin. “SPRINT: High-Throughput Robust Distributed Schnorr Signatures”. In: *Advances in Cryptology—Eurocrypt 2024, Part V*. Ed. by M. Joye and G. Leander. Vol. 14655. LNCS. Springer, 2024, pp. 62–91. URL: https://doi.org/10.1007/978-3-031-58740-5_3.
- [6] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography*. 2020. URL: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_5.pdf.
- [7] L. Brandão and M. Davidson. *Notes on Threshold EdDSA/Schnorr Signatures*. <https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8214B.ipd.pdf>. 2022.
- [8] L. Brandão and R. Peralta. *NIST First Call for Multi-Party Threshold Schemes*. <https://nvlpubs.nist.gov/nistpubs/ir/2026/NIST.IR.8214C.pdf>. 2026.
- [9] H. Chu, P. Gerhart, T. Ruffing, and D. Schröder. “Practical Schnorr Threshold Signatures Without the Algebraic Group Model”. In: *Crypto 2023*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14081. LNCS. Springer, 2023, pp. 743–773.
- [10] M. Ciampi, E. Crites, C. Komlo, and M. Maller. *On the Adaptive Security of Key-Unique Threshold Signatures*. Cryptology ePrint Archive, Paper 2025/943. 2025. URL: <https://eprint.iacr.org/2025/943>.

- [11] D. Connolly, C. Komlo, I. Goldberg, and C. A. Wood. *The Flexible Round-Optimized Schnorr Threshold (FROST) Protocol for Two-Round Schnorr Signatures*. RFC 9591. Available at <https://www.rfc-editor.org/rfc/rfc9591>. 2024.
- [12] E. Crites and A. Stewart. *A Plausible Attack on the Adaptive Security of Threshold Schnorr Signatures*. Cryptology ePrint Archive, Paper 2025/1001. 2025. URL: <https://eprint.iacr.org/2025/1001>.
- [13] E. C. Crites, J. Katz, C. Komlo, S. Tessaro, and C. Zhu. “On the Adaptive Security of FROST”. In: *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VI*. Ed. by Y. T. Kalai and S. F. Kamara. Vol. 16005. Lecture Notes in Computer Science. Springer, 2025, pp. 480–511. DOI: 10.1007/978-3-032-01887-8_16. URL: https://doi.org/10.1007/978-3-032-01887-8_16.
- [14] E. C. Crites, C. Komlo, and M. Maller. “Fully Adaptive Schnorr Threshold Signatures”. In: *Crypto 2023*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14081. LNCS. Springer, 2023, pp. 678–709. URL: https://doi.org/10.1007/978-3-031-38557-5_22.
- [15] F. Garillot, Y. Kondi, P. Mohassel, and V. Nikolaenko. “Threshold Schnorr with Stateless Deterministic Signing from Standard Assumptions”. In: *CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Ed. by T. Malkin and C. Peikert. Vol. 12825. LNCS. Springer, 2021, pp. 127–156. DOI: 10.1007/978-3-030-84242-0_6. URL: https://doi.org/10.1007/978-3-030-84242-0_6.
- [16] C. Komlo and I. Goldberg. “FROST: Flexible Round-Optimized Schnorr Threshold Signatures”. In: *SAC 2020*. Ed. by O. Dunkelman, M. J. Jacobson Jr., and C. O’Flynn. Vol. 12804. LNCS. Springer, 2021, pp. 34–65.
- [17] Y. Lindell. “Simple Three-Round Multiparty Schnorr Signing with Full Simulatability”. In: *IACR Communications in Cryptology* 1.1 (2024).
- [18] N. Makriyannis. *On the Classic Protocol for MPC Schnorr Signatures*. Cryptology ePrint Archive. Available at <https://eprint.iacr.org/2022/1332>. 2022.
- [19] M. B. Paterson and D. R. Stinson. “Optimal Constructions for ID-Based One-Way-Function Key Predistribution Schemes Realizing Specified Communication Graphs”. In: *Journal of Mathematical Cryptology* 9.4 (2015), pp. 215–225. DOI: 10.1515/jmc-2014-0031.
- [20] Àlex Rodríguez García and S. Tessaro. “Cryptanalysis of LDVR: Polynomial-Time Adaptive Attacks against Threshold Schnorr Signatures”. Cryptology ePrint Archive, Paper 2026/2126. 2026. URL: <https://eprint.iacr.org/2026/2126>.
- [21] T. Ruffing, V. Ronge, E. Jin, J. Schneider-Bensch, and D. Schröder. “ROAST: Robust Asynchronous Schnorr Threshold Signatures”. In: *CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers, and E. Shi. ACM, 2022, pp. 2551–2564.
- [22] D. R. Stinson and R. Stroh. “Provably Secure Distributed Schnorr Signatures and a (t, n) -Threshold Scheme for Implicit Certificates”. In: *ACISP 2001, Sydney, Australia, July 11-13, 2001*. Ed. by V. Varadharajan and Y. Mu. Vol. 2119. LNCS. Springer, 2001, pp. 417–434.