POLITECNICO DI TORINO
Repository ISTITUZIONALE

Designs, Efficiency and Decentralised Applications of Threshold Schnorr Signatures

Original
Designs, Efficiency and Decentralised Applications of Threshold Schnorr Signatures / Cenk, M., Errati, L., Meneghetti, A.. - ELETTRONICO. - 8:(2025), pp. 99-117. (Financial Cryptography in Rome 2025 Roma (ITALIA) October 1st, 2025) [10.69091/koine/vol-8-P12].

Availability:
This version is available at: 11583/3005796 since: 2025-12-11T23:28:02Z

Publisher:
De Cifris Press

Published
DOI:10.69091/koine/vol-8-P12

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026

Designs, Efficiency and Decentralised Applications of Threshold Schnorr Signatures

Murat Cenk[1], Leonardo Errati[2,3], and Alessio Meneghetti[2,4]

[1] Ripple Labs Inc., Toronto, Canada mcenk@ripple.com
[2] Università di Trento, Trento, Italy
[3] Politecnico di Torino, Torino, Italy leonardo.errati@polito.it
[4] Università di Bari, Bari, Italy alessio.meneghetti@uniba.it

Abstract. This survey presents some threshold variants of the Schnorr signature scheme selected according to their novelty or theoretical relevance, and evaluates them on four metrics: security guarantees, computational complexity, memory complexity and bandwidth requirements.
We will then exploit them to provide a few comparisons in the case of real-world applications, such as blockchain and custodianship.

Keywords: Threshold signature · Schnorr signature · Complexity analysis

1 Introduction

Motivation. Decentralised data and asset management systems are becoming a cornerstone of modern computing infrastructures, from cloud-native services and IoT ecosystems to decentralised finance (DeFi) and blockchain networks. This resulted in an increasing demand for secure, efficient, and distributed cryptographic primitives for decentralisation of trust, fault tolerance, and robustness. Threshold signature schemes are a crucial primitive that addresses these challenges decentralising and simplifying key management. Introduced in 1987 by Desmedt [14], they saw a surge during the early 2000s, with foundational work on threshold RSA [29, 13, 23] and distributed key generation [18]. Recent interest focused on threshold variants of Schnorr, ECDSA and EdDSA.

Contributions and related work. Due to the linearity of the signing equation in Schnorr signatures, its t-out-of-n threshold variants proved rather efficient and compatible with existing infrastructures. We focus on a selection of these variants, chosen due to their novelty, efficiency, and security guarantees: FROST [19], FROST2 [11], FROST3 [26], MIRAGE [3], and SPARKLE+ [12]. A review of broader scope was performed in [28], where the authors conduct a meta-analysis of existing surveys

De Cifris Koine – FCiR25 ACTA https://doi.org/10.69091/koine/vol-8-P12

on threshold signatures. In contrast, our work provides a focused and detailed comparison of threshold Schnorr protocols, including symbolic cost estimates and application-specific analysis.

Scope and limitations. This work exclusively focuses on chosen threshold Schnorr variants. Further work might include other unselected or concurrently appeared protocols, like GLACIUS [2]. While closely related, multisignature schemes are omitted, as they follow different paradigms and security requisites.

Notation. In the following, $(\mathbb{F}_q, +, \cdot)$ is the finite field of q elements and $\mathcal{E}$ an elliptic curve. Schnorr-like algorithms operate on a group $(\mathbb{G}, \cdot)$ and a field $\mathbb{F}$. These might be, for instance, the point-group of $\mathbb{F}_q$-rational points of $\mathcal{E}$ and $\mathbb{F}_q$.
We focus on signature schemes $\Pi_{\mathsf{Sign}} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ with private key sk and public key pk. In a threshold setting, sk is split into shards $(\mathsf{sk}_i)_{i\in[n]}$ where the i-th key is only known to the i-th participant $\mathcal{P}_i$ and $[n]$ stands for the interval $\{1, \ldots, n\}$. Without loss of generality, messages m will be signed only and exactly by those parties $\mathcal{P}_1, \ldots, \mathcal{P}_t$ indexed by $J = [t]$.

Methodology. Each protocol will be evaluated on two metrics. First, the *computational complexity* of its key generation (η_{Keygen}), signature (η_{Sign}) and verification (η_{Verify}) components, broken down into basic operations: field additions ($\eta_{\mathbb{F}+}$), field multiplications ($\eta_{\mathbb{F}\times}$), field samples ($\eta_{\mathbb{F}\leftarrow\$}$), group multiplications ($\eta_{\mathbb{G}\times}$), group exponentiations ($\eta_{\mathbb{G}^\wedge}$), hash evaluations (η_{H}), calculations of a Lagrange coefficient (η_{Lag}). Second, the *bandwidth requirement* for the communications of $\mathcal{P}_i$ in terms of communication cost for group ($\beta_{\mathbb{G}}$) and field ($\beta_{\mathbb{F}}$) elements.
To achieve a fair comparison and avoid affecting security guarantees, all protocols are taken as-is, thus our analysis can be considered as a theoretical worst-case.

2 Preliminaries

2.1 Threshold Digital Signatures

Definition 1. *An* r-round (t, n)-threshold digital signature scheme *is a tuple of algorithms* $\Pi_{\mathsf{TSign}} = (\mathsf{Setup}, \mathsf{Keygen}, \{\mathsf{Sign}_k\}_{k\in[r]}, \mathsf{Combine}, \mathsf{Verify})$ *where*

- $\mathsf{Setup}(1^\lambda)$ *is a probabilistic algorithm returning the public parameters* pp.
- $\mathsf{Keygen}(\mathsf{pp}, 1^\lambda, t, n)$ *is a probabilistic algorithm generating a single public key* pk *and the* n *shares* $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ *of a* t*-out-of-*n *shared private key* sk.
- *Each* $\mathsf{Sign}_k(\mathsf{pp}, \mathsf{sk}_i, \mathsf{input}_k)$, $k = 1, \ldots, r$, *is a probabilistic algorithm representing the* k*-th signing round, where* input_k *is the set of inputs to round* k. Sign_r *returns the partial signature* σ_i *of party* $\mathcal{P}_i$ *on message* m.

- $\mathsf{Combine}(\mathsf{pp}, \mathsf{m}, J, \{\sigma_j\}_J)$ *is a deterministic algorithm combining the partial signatures indexed by* $J \subseteq [n]$ *such that* $|J| \geq t$*, and either returning a combined signature* σ *for message* m *or the error symbol* $\perp$*.*
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}, \sigma)$ *is a deterministic algorithm returning* accept *or* reject*.*

Denoting by Sign *the aggregation of all* r *signing rounds,* Π_{TSign} *is* correct *if*

$$\mathbb{P}\left[\mathsf{Verify}\big(\mathsf{pp}, \mathsf{pk}, \mathsf{m}, \sigma\big) = \mathsf{accept} \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 1^\lambda) \\ J \subseteq [n],\ |J| \geq t \\ \sigma \leftarrow \mathsf{Combine}(\mathsf{m}, J, \{\mathsf{Sign}(\mathsf{sk}_j, \mathsf{m})\}_J) \end{array}\right] = 1$$

Dependence on pp is often implicit. Part of input_k might need to be saved for use in rounds $k' \neq k$; we will analyse the space μ required to store such values during execution of Π_{TSign}. We denote as $\eta_{\mathsf{Sign}k}$ and η_{Combine} the number of operations executed during the k-th signing round and the combining phase.

Constructions from classical schemes. A secure Π_{TSign} constructed from an existing scheme Π_{Sign} is preferrable in terms of compatibility and security. This requires the private key sk to be that of Π_{Sign} and signatures returned by Π_{TSign} to be indistinguishable from those returned by Π_{Sign} with the common sk.

Security of signature schemes. The security of any scheme is established within the behaviour of a PPT adversary $\mathcal{A}$. In classical signature schemes the prevailing security notion is *EUF-CMA*, with signature experiment $\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi_{\mathsf{TSign}}}(\lambda)$. $\mathsf{Keygen}(1^\lambda)$ is run and pk is given to $\mathcal{A}$, which has access to a signature oracle $\mathcal{O}_{\mathsf{Sign},\mathsf{sk}}$. After a polynomial number of queries $\mathcal{Q} = \{\mathsf{m}_1, \ldots, \mathsf{m}_{q(\lambda)}\}$, $\mathcal{A}$ succeeds if and only if it provides a valid signature to an unqueried message $\mathsf{m} \notin \mathcal{Q}$.
In the security of threshold schemes, synchrony assumptions play a critical role [30]; the adversary can corrupt a set corr of at most $t-1$ parties, and adversarial power needs to mind (e.g.) its *visibility* and *corrupting ability*:

- a *concurrent* $\mathcal{A}$ can open simultaneous signing sessions and decide whether to complete them or not, a *rushing* $\mathcal{A}$ can wait for the output of other participants before actively performing the protocol.
- a *static* $\mathcal{A}$ can corrupt a set corr of up to $t-1$ signers before the protocol starts and only then interacts with honest users hon, while an *adaptive* $\mathcal{A}$ can actively add new users to corr during protocol execution, thus actively adapting its behaviour and choices.

Security proofs usually model the environment according to the *Random Oracle Model (ROM)* [5] or the *Algebraic Group Model (AGM)* [17].

Security assumptions. Consider a group $\mathbb{G}$ and an element $g \in \mathbb{G}$, where g is often a generator of $\mathbb{G}$. We will consider the following security assumptions:

- *Discrete Logarithm assumption (DL)* [22]: given a second element $h \in \mathbb{G}$, it is computationally infeasible to determine an x such that $h = g^x$ holds.
- *Decisional Diffie-Hellman assumption (DDH)* [7]: it is computationally infeasible to distinguish between the triples $(g^{x_1}, g^{x_2}, g^{x_3})$ and $(g^{x_1}, g^{x_2}, g^{x_1 x_2})$.
- *Algebraic One-More Discrete Logarithm assumption (AOMDL)* [24]: given $t-1$ pairs (h_i, x_i) where $h_i = g^{x_i}$, solving DL on a set of t different group elements $h'_1, \ldots, h'_t$ is computationally infeasible.

2.2 Commitment Schemes

Definition 2. *A* commitment scheme *is a tuple of algorithms*

- $\mathsf{PGen}(1^\lambda)$, *a probabilistic algorithm returning the public parameters* pp.
- $\mathsf{Com}(\mathsf{pp}, \mathsf{m})$, *a probabilistic algorithm returning* (c, d), *where* c *is the commitment string and* d *the decommitment (or opening) material.*
- $\mathsf{Open}(\mathsf{pp}, \mathsf{m}, \mathsf{c}, \mathsf{d})$, *a deterministic algorithm returning* accept *or* reject.

such that scheme $\Pi_{\mathsf{Com}} = (\mathsf{PGen}, \mathsf{Com}, \mathsf{Open})$ *is* correct, *i.e.*

$$\mathbb{P}\left[\mathsf{Open}\big(\mathsf{pp}, \mathsf{c}, \mathsf{d}\big) = \mathsf{accept} \;\middle|\; (\mathsf{pp}, \mathsf{m}, \mathsf{c}, \mathsf{d}) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{m})\right] = 1$$

Public parameters pp are often implicit. The minimal security requirement for schemes Π_{Com} is to satisfy the computational formulations of the following:

- *(computationally) hiding*: any PPT adversary $\mathcal{A}$ wins the hiding game $\mathsf{Hide}(\lambda)$ with negligible probability. The adversary chooses $\mathsf{m}_0, \mathsf{m}_1$ and a challenger selects a random m_b, committing and returning c_b; then $\mathcal{A}$ needs to guess b.
- *(perfectly) hiding*: for every pair $(\mathsf{m}_0, \mathsf{m}_1)$ the distributions of c_0 and c_1 are indistinguishable, where $(\mathsf{c}_0, \mathsf{d}_0) \leftarrow \mathsf{Com}(\mathsf{m}_0)$ and $(\mathsf{c}_1, \mathsf{d}_1) \leftarrow \mathsf{Com}(\mathsf{m}_1)$.
- *(computationally) binding*: for any PPT adversary $\mathcal{A}$,

$$\mathbb{P}\left[\begin{array}{c|c} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda), & \mathsf{m}_0 \neq \mathsf{m}_1, \\ (\mathsf{c}, \mathsf{m}_0, \mathsf{d}_0, \mathsf{m}_1, \mathsf{d}_1) \leftarrow \mathcal{A}(\mathsf{pp}) & \begin{array}{c}\mathsf{Open}(\mathsf{m}_0, \mathsf{c}, \mathsf{d}_0) = \mathsf{accept}, \\ \mathsf{Open}(\mathsf{m}_1, \mathsf{c}, \mathsf{d}_1) = \mathsf{accept}\end{array} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- *(perfectly) binding*: if the above probability is null.

Hash-based commitments. If $\mathsf{Com}(\mathsf{m}) = (\mathsf{H}(\mathsf{m}), \mathsf{m})$ and $\mathsf{Open}(\mathsf{m}, \mathsf{c}, \mathsf{d})$ is $\mathsf{H}(\mathsf{d}) \stackrel{?}{=} \mathsf{c}$ for some hash function H, the scheme Π_{Com} is said to be *hash-based*. We assume this is the case, thus evaluating Com and Open costs exactly as hashing. Security-wise, Π_{Com} is hiding and binding if H is a cryptographic hash function. We will do the

same to evaluate the cost of schemes requiring *homomorphic commitments*. This is often done when schemes do not require simulation-based or UC security. For simplicity, we shall compare the bandwidth cost of sending c to that of sending a group element, and the cost of sending d to be that of a field element. Remark how this holds if we commit on exponents e, usually $e \leftarrow \mathsf{H}(r\|\mathsf{m})$, by sending group elements as g^e in the signature phase.

3 The Classical Schnorr Signature

The Schnorr signature scheme ([27], pseudocode in Fig. 1) was introduced in 1990, and can be constructed applying the Fiat-Shamir transform to the Schnorr ZKID protocol. Its performance can be improved via clever tweaks, such as sending $k - e \cdot \mathsf{sk}$ instead of $k + e \cdot \mathsf{sk}$ during Sign to avoid performing an inversion in Verify. These do not affect its security.

Keygen()	Sign(m)	Verify(m, σ)
1 : $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_q$	1 : $k \leftarrow\!\!\$\ \mathbb{Z}_q, r \leftarrow g^k$	1 : $e_v \leftarrow \mathsf{H}(r\|\mathsf{m})$
2 : $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$	2 : $e \leftarrow \mathsf{H}(r\|\mathsf{m})$	2 : $r_v \leftarrow g^s \mathsf{pk}^{e_v}$
3 : **return** $(\mathsf{pk}, \mathsf{sk})$	3 : $s \leftarrow k - e \cdot \mathsf{sk}$	3 : **if** $r_v = r$ **return** accept
	4 : **return** $\sigma = (r, s)$	4 : **else return** reject

Fig. 1. Pseudocode of Schnorr's digital signature scheme.

Complexity of the Schnorr *scheme.* With the notation introduced in Section 1,

$$\begin{aligned} \eta_{\mathsf{Keygen}} &= \eta_{\mathbb{F}\leftarrow\!\!\$} + \eta_{\mathbb{G}^\wedge} \\ \eta_{\mathsf{Sign}} &= \eta_{\mathbb{F}\leftarrow\!\!\$} + \eta_{\mathbb{F}+} + \eta_{\mathbb{F}\times} + \eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} \\ \eta_{\mathsf{Verify}} &= \eta_{\mathbb{G}\times} + 2\eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} \end{aligned} \tag{1}$$

Usually, $\eta_{\mathbb{G}^\wedge}$ and η_{H} are considered the costliest operations. Network complexity is only meaningful in settings where synchrony is relevant.

Building a threshold variant. Threshold schemes typically split Sign into r rounds: early rounds establish trusted shared randomness (e.g., via commitments) to generate common parameters, while later ones produce partial signatures σ_i for aggregation. Remark how Sign_1 can be pre-computed, meaning that calculations are performed beforehand but only used when t users agree on a message m to be signed and the protocol is executed.

4 A comparison of existing schemes

Protocol selection. We consider (t, n)-threshold signatures, thus omitting (n, n) schemes such as TwoSchnorr [25], MSDL [8] and MuSig [21], and focus on schemes chosen due to their efficiency, novelty and security, sorted by year. This is detailed in Table 1. The three related schemes FROST, FROST2, FROST3 were selected due to their novelty. We also included, in chronological order, SPARKLE+ and MIRAGE due to their improvements in security guarantees over previous schemes. OLAF, a variation of FROST3, was omitted since it only varies its key generation and we focus on signatures. A remarkable variant, GLACIUS, provides adaptive security under DDH with constant key size and identifiable aborts. However, since it appeared concurrently with this work and introduces a novel security definition, it was not included in our comparison.

Algorithm	**St.**	**Ad.**	**Main Features**	**Selected**
FROST [19]	✓	✗	efficient, IETF standard	✓
FROST2 [11]	✓	✗	improvement over FROST	✓
FROST3 [26]	✓	✗	improvement over FROST2	✓
MIRAGE [4]	✓	✗	dealerless key generation	✓
OLAF [10]	✓	✗	SimplePedPop + FROST3	✗
SPARKLE+ [12]	✓	✓	security, auditability	✓
GLACIUS [2]	✓	✓	signing keys of constant size	✗

Table 1. Some relevant threshold Schnorr schemes: static (St.) or adaptive (Ad.) security, an extremely brief overview, and whether they were selected or not.

Methodology. Since for all schemes the verification algorithm (Verify) is that of Schnorr and since Keygen is performed sporadically, we can focus on Sign_k and Combine. Each k-th signing round Sign_k ends when party $\mathcal{P}_i$ communicates with a bandwidth cost of $\beta_{\mathsf{Sign}k}$ and a communication delay modelled by the delay parameter δ.

Most schemes allow Sign_1 to be pre-computed, meaning that calculations are performed beforehand, so that they are used only when the protocol is executed (once t users have agreed on the message to be signed); this does not affect $\beta_{\mathsf{Sign}1}$.

Finally, remark how for some specific t and n, the performance may vary wildly, thus an asymptotic view will better capture our results.

4.1 FROST

In [19] Komlo and Goldberg introduce FROST (Flexible Round-Optimized Schnorr Threshold signature, Fig. 2) and prove its security in the Random Oracle Model against rushing and concurrent adversaries under the AOMDL assumption. This is, to our knowledge, the first full-threshold Schnorr variant.

$\mathsf{Sign}_1(J)$

1 : $k_i \leftarrow\$ \mathbb{Z}_q, k'_i \leftarrow\$ \mathbb{Z}_q$
2 : $r_i \leftarrow g^{k_i}, r'_i \leftarrow g^{k'_i}$
3 : $\mathsf{send}(r_i, r'_i)$

$\mathsf{Sign}_2(\mathsf{m}, J, \{r_j, r'_j\}_J)$

1 : $r_i \leftarrow g^{k_i}, r'_i \leftarrow g^{k'_i}$
2 : **for** $j \in J$ **do**
3 : $\quad e'_j \leftarrow \mathsf{H}(j \| \mathsf{pk} \| \mathsf{m} \| \{r_k, r'_k\}_{k \in J})$
4 : $r \leftarrow \prod_{j \in J} r_j (r'_j)^{e'_j}$
5 : $e \leftarrow \mathsf{H}(\mathsf{pk} \| \mathsf{m} \| r)$
6 : $\lambda_i \leftarrow \mathsf{Lagrange}(J, i)$
7 : $s_i \leftarrow k_i + e'_i \cdot k'_i + \lambda_i \cdot e \cdot \mathsf{sk}_i$
8 : $\mathsf{send}(s_i)$

$\mathsf{Combine}(\mathsf{m}, J, \{s_j\}_J)$

1 : **for** $j \in J$ **do**
2 : $\quad e'_j \leftarrow \mathsf{H}(j \| \mathsf{pk} \| \mathsf{m} \| \{r_k, r'_k\}_{k \in J})$
3 : $s \leftarrow \sum_{j \in J} s_j$
4 : $r \leftarrow \prod_{j \in J} r_j (r'_j)^{e'_j}$
5 : **return** (r, s)

Fig. 2. The pseudocode of FROST.

Intuitively, Sign_1 builds up shared randomness and Sign_2 builds upon it to produce the partial signature σ_i. This is a recurring structure in such schemes.

Computational complexity. Considering the pseudocode of FROST in Fig. 2,

$$\begin{aligned} \eta_{\mathsf{Sign1}} &= 2\eta_{\mathbb{F} \leftarrow\$} + 2\eta_{\mathbb{G}^\wedge} \\ \eta_{\mathsf{Sign2}} &= 2\eta_{\mathbb{F}+} + 3\eta_{\mathbb{F}\times} + (2t-2)\eta_{\mathbb{G}\times} + (t+2)\eta_{\mathbb{G}^\wedge} + (t+1)\eta_{\mathsf{H}} + \eta_{\mathsf{Lag}} + \delta \\ \eta_{\mathsf{Combine}} &= (t-1)\eta_{\mathbb{F}+} + (2t-2)\eta_{\mathbb{G}\times} + t\eta_{\mathbb{G}^\wedge} + t\eta_{\mathsf{H}} + \delta \end{aligned} \tag{2}$$

Bandwidth complexity. To achieve a fair comparison we take the pseudocode as-is and consider the values sent by other parties $\mathcal{P}_j$ once they are received. The memory required by the rounds of FROST is listed alongside the corresponding bandwidth requirements, reflecting the network strain during each phase.

$$\beta_{\mathsf{Sign1}} = 2\beta_{\mathbb{G}} \qquad \beta_{\mathsf{Sign2}} = \beta_{\mathbb{F}} \qquad \beta_{\mathsf{Combine}} = \beta_{\mathbb{F}} + \beta_{\mathbb{G}} \tag{3}$$

4.2 FROST2

In [11], Crites, Komlo, and Maller describe a framework for proving the security of threshold variants of Schnorr and introduce FROST2 while applying it to FROST. This new variant optimises some redundant operations and provides the same security level of FROST. The pseudocode of FROST2 is in Fig. 3.

$\mathsf{Sign}_1(J)$

1 : $k_i \leftarrow\!\$\ \mathbb{Z}_q, k_i' \leftarrow\!\$\ \mathbb{Z}_q$
2 : $r_i \leftarrow g^{k_i}, r_i' \leftarrow g^{k_i'}$
3 : $\mathsf{send}(r_i, r_i')$

$\mathsf{Sign}_2(\mathsf{m}, J, \{r_j, r_j'\}_{j\in J})$

1 : $r_i \leftarrow g^{k_i}, r_i' \leftarrow g^{k_i'}$
2 : $e' \leftarrow \mathsf{H}(\mathsf{pk}\|\mathsf{m}\|\left\{r_k, r_k'\right\}_{k\in J})$
3 : $r \leftarrow \prod_{j\in J} r_j (r_j')^{e'}$
4 : $e \leftarrow \mathsf{H}(\mathsf{pk}\|\mathsf{m}\|r)$
5 : $\lambda_i \leftarrow \mathsf{Lagrange}(J, i)$
6 : $s_i \leftarrow k_i + e' \cdot k_i' + \lambda_i \cdot e \cdot \mathsf{sk}_i$
7 : $\mathsf{send}(s_i)$

$\mathsf{Combine}(\mathsf{m}, J, \{s_j\}_{j\in J})$

1 : $e' \leftarrow \mathsf{H}(\mathsf{pk}\|\mathsf{m}\|\left\{r_k, r_k'\right\}_{k\in J})$
2 : $s \leftarrow \sum_{j\in J} s_j$
3 : $r \leftarrow \prod_{j\in J} r_j (r_j')^{e'}$
4 : **return** (r, s)

Fig. 3. The pseudocode of FROST2.

Computational complexity. The theoretical performance of FROST2 is

$$\begin{aligned} \eta_{\mathsf{Sign1}} &= 2\eta_{\mathbb{F}\leftarrow\!\$} + 2\eta_{\mathbb{G}^\wedge} \\ \eta_{\mathsf{Sign2}} &= 2\eta_{\mathbb{F}+} + 3\eta_{\mathbb{F}\times} + (2t-2)\eta_{\mathbb{G}\times} + (t+2)\eta_{\mathbb{G}^\wedge} + 2\eta_{\mathsf{H}} + \eta_{\mathsf{Lag}} + \delta \\ \eta_{\mathsf{Combine}} &= (t-1)\eta_{\mathbb{F}+} + (2t-2)\eta_{\mathbb{G}\times} + t\eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} + \delta \end{aligned} \tag{4}$$

which is a net improvement over that of FROST (Equation (3)). Notice how Sign_2 in FROST2 only needs one digest e', while FROST needed t digests $e_1', \ldots, e_t'$.

Bandwidth complexity. The bandwidth requirements are the same as in FROST.

4.3 FROST3

A final modification of FROST, FROST3 was introduced by Ruffing, Ronge, Jin, Schneier-Bensch and Schröder [26] applying their ROAST (Robust Asynchronous Schnorr Threshold Signatures) wrapper to FROST. Its security assumptions are unmodified, and a further efficiency improvement makes the scheme suitable for most applications.
In [10] its key generation is substituted with SimplPedPoP, introducing distributed key generation and proving it secure without the algebraic group model. The authors denote SimplePedPop + FROST3 as Olaf.

$\mathsf{Sign}_1(J)$

1 : $k_i \leftarrow\!\!\$\ \mathbb{Z}_q, k'_i \leftarrow\!\!\$\ \mathbb{Z}_q$
2 : $r_i \leftarrow g^{k_i}, r'_i \leftarrow g^{k'_i}$
3 : $\mathsf{send}(r_i, r'_i)$

$\mathsf{Sign}_2(\mathsf{m}, J, \{r_j, r'_j\}_{j\in J})$

1 : $\hat{r} \leftarrow \prod_{j\in J} r_j, \hat{r}' \leftarrow \prod_{j\in J} r'_j$
2 : $e' \leftarrow \mathsf{H}(\mathsf{pk}\|\mathsf{m}\|\hat{r}\|\hat{r}')$
3 : $r \leftarrow \hat{r}(\hat{r}')^{e'}$
4 : $e \leftarrow \mathsf{H}(\mathsf{pk}\|\mathsf{m}\|r)$
5 : $\lambda_i \leftarrow \mathsf{Lagrange}(J, i)$
6 : $s_i \leftarrow k_i + e' \cdot k'_i + \lambda_i \cdot e \cdot \mathsf{sk}_i$
7 : $\mathsf{send}(s_i)$

$\mathsf{Combine}(\mathsf{m}, J, \{s_j\}_{j\in J})$

1 : $e' \leftarrow \mathsf{H}(\mathsf{pk}\|\mathsf{m}\|\hat{r}\|\hat{r}')$
2 : $\hat{r} \leftarrow \prod_{j\in J} r_j, \hat{r}' \leftarrow \prod_{j\in J} r'_j$
3 : $s \leftarrow \sum_{j\in J} s_j$
4 : $r \leftarrow \hat{r}(\hat{r}')^{e'}$
5 : **return** (r, s)

Fig. 4. The pseudocode of FROST3.

Computational complexity. Exploiting the intermediate values $\hat{r}$ and $\hat{r}'$, FROST3 (Fig. 4) calculates r with one exponentiation instead of t. This was perhaps already intended by the authors of FROST2.

$$\begin{aligned}
\eta_{\mathsf{Sign1}} &= 2\eta_{\mathbb{F}\leftarrow\!\!\$} + 2\eta_{\mathbb{G}^\wedge} \\
\eta_{\mathsf{Sign2}} &= 2\eta_{\mathbb{F}+} + 3\eta_{\mathbb{F}\times} + (2t+1)\eta_{\mathbb{G}\times} + \eta_{\mathbb{G}^\wedge} + 2\eta_{\mathsf{H}} + \eta_{\mathsf{Lag}} + \delta \\
\eta_{\mathsf{Combine}} &= (t-1)\eta_{\mathbb{F}+} + (2t+1)\eta_{\mathbb{G}\times} + \eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} + \delta
\end{aligned} \tag{5}$$

Bandwidth complexity. The bandwidth requirement are the same as in FROST.

4.4 MIRAGE

First introduced by Battagliola *et al.* [3] in its $(2,3)$-threshold formulation, MIRAGE was properly generalised to a (t,n)-threshold scheme by Battagliola, Longo and Meneghetti [4]. It was built from their concept of *extensible decentralised secret sharing*, meaning that its key generation is decentralised and allows to increase the number n of parties without modifying existing shares. It is statically secure in the Random Oracle Model against rushing and concurrent adversaries under the DDH assumption.

$\mathsf{Sign}_1(J)$

1 : $k_i \leftarrow\!\!\$\ \mathbb{Z}_q$
2 : $r_i \leftarrow g^{k_i}$
3 : $(\mathsf{c}_i^{r_i}, \mathsf{d}_i^{r_i}) \leftarrow \mathsf{Com}(r_i)$
4 : $\mathsf{send}(\mathsf{c}_i^{r_i})$

$\mathsf{Sign}_2(J, \{\mathsf{c}_j^{r_j}\}_{j\in J}))$

1 : $\mathsf{send}(\mathsf{d}_i^{r_i})$

$\mathsf{Sign}_3(\mathsf{m}, J, \{\mathsf{d}_j^{r_j}\}_{j\in J})$

1 : **for** j **in** J
2 : $\quad \mathsf{Open}(\mathsf{c}_j^{r_j}, \mathsf{d}_j^{r_j})$
3 : $r \leftarrow \prod_j r_j$
4 : $\lambda_i \leftarrow \mathsf{Lagrange}(J, i)$
5 : $e \leftarrow \mathsf{H}(r \| \mathsf{m})$
6 : $s_i \leftarrow k_i - \lambda_i \cdot e \cdot \mathsf{sk}_i$
7 : $(\mathsf{c}_i^{s_i}, \mathsf{d}_i^{s_i}) \leftarrow \mathsf{Com}(s_i)$
8 : $\mathsf{send}(\mathsf{c}_i^{s_i})$

$\mathsf{Sign}_4(\mathsf{m}, J, \{\mathsf{c}_j^{s_j}\}_{j\in J})$

1 : $\mathsf{send}(\mathsf{d}_i^{s_i})$

$\mathsf{Combine}(\mathsf{m}, J, \{\mathsf{d}_j^{s_j}\}_{j\in J})$

1 : **for** j **in** J **do**
2 : $\quad \mathsf{Open}(\mathsf{c}_j^{s_j}, \mathsf{d}_j^{s_j})$
3 : $s \leftarrow \sum_{j\in J} s_j$
4 : $r \leftarrow g^s \cdot \mathsf{pk}^e$
5 : **if** $\mathsf{H}(r\|\mathsf{m}) \neq e$
6 : $\quad$ **then abort**
7 : **return** (r, s)

Fig. 5. The pseudocode of MIRAGE.

Computational complexity. The four-round protocol MIRAGE is described in Fig. 5. The first two rounds are relatively light, only sharing common and trusted randomness. The third round reconstructs said randomness to commit to the partial signature, which is decommitted in round four.

$$\begin{aligned} \eta_{\mathsf{Sign1}} &= \eta_{\mathbb{F}\leftarrow\!\!\$} + \eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} \\ \eta_{\mathsf{Sign2}} &= \eta_{\mathsf{Sign4}} = \delta \\ \eta_{\mathsf{Sign3}} &= \eta_{\mathbb{F}+} + 2\eta_{\mathbb{F}\times} + (t-1)\eta_{\mathbb{G}\times} + \eta_{\mathsf{Lag}} + (t+2)\eta_{\mathsf{H}} + \delta \\ \eta_{\mathsf{Combine}} &= (t-1)\eta_{\mathbb{F}+} + \eta_{\mathbb{G}\times} + 2\eta_{\mathbb{G}^\wedge} + (t+1)\eta_{\mathsf{H}} + \delta \end{aligned} \tag{6}$$

Bandwidth complexity. While commitments replacing standard communications result in an increased number of rounds, the total network requirements are roughly the same as in FROST variants.

$$\begin{aligned} \beta_{\mathsf{Sign1}} &= \beta_{\mathbb{G}} & \beta_{\mathsf{Sign3}} &= \beta_{\mathbb{G}} \\ \beta_{\mathsf{Sign2}} &= \beta_{\mathsf{Sign4}} = \beta_{\mathbb{F}} & \beta_{\mathsf{Combine}} &= \beta_{\mathbb{F}} + \beta_{\mathbb{G}} \end{aligned} \tag{7}$$

4.5 SPARKLE+

One of the first schemes applying the notion of adaptive security, SPARKLE+ was introduced by Crites, Komlo and Maller [12] and proved to be adaptively secure up to t corruptions under the AOMDL security assumption.
It is statically secure in the Random Oracle Model (ROM) against rushing, concurrent adversaries under the DL assumption. It is also secure in the ROM and AGM models against rushing, adaptive and concurrent adversaries under the AOMDL assumption. To the best of our knowledge, SPARKLE+ achieves one of the tightest security reductions among all Schnorr variants.

$\mathsf{Sign}_1(J)$

1 : $k_i \leftarrow\!\$\ \mathbb{Z}_q$
2 : $r_i \leftarrow g^{k_i}$
3 : $(\mathsf{c}_i, \mathsf{d}_i) \leftarrow \mathsf{Com}(i, r_i)$
4 : $\mathsf{send}(\mathsf{c}_i)$

$\mathsf{Sign}_2(J, \{\mathsf{c}_j\}_{j\in J})$

1 : $\sigma_i^d \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \mathsf{d}_i)$
2 : $\mathsf{send}(\sigma_i^d, \mathsf{d}_i)$

$\mathsf{Sign}_3(\mathsf{m}, J, \{\mathsf{d}_j, \sigma_j^d\}_{j\in J})$

1 : **for** $j \in J$ **do**
2 : $\quad \mathsf{Verify}(\mathsf{pk}_j, \mathsf{m}, \sigma_j^d)$
3 : **for** $j \in J \setminus \{i\}$ **do**
4 : $\quad \mathsf{Open}((j, r_j), \mathsf{c}_j, \mathsf{d}_j)$
5 : $r \leftarrow \prod_{j\in J} r_j$
6 : $\lambda_i \leftarrow \mathsf{Lagrange}(J, i)$
7 : $e \leftarrow \mathsf{H}(\mathsf{pk}\|r\|\mathsf{m})$
8 : $s_i \leftarrow k_i + \lambda_i \cdot e \cdot \mathsf{sk}_i$
9 : $\mathsf{send}(s_i)$

$\mathsf{Combine}(\mathsf{m}, J, \{s_j\}_{j\in J})$

1 : $s \leftarrow \sum_{j\in J} s_j$
2 : $r \leftarrow \prod_{j\in J} r_j$
3 : **return** (r, s)

Fig. 6. The pseudocode of SPARKLE+. Each i-th user must execute an instance $(\mathsf{Sign}, \mathsf{Verify})$ of classical Schnorr with its private sk_i and public $\mathsf{pk}_i \leftarrow g^{\mathsf{sk}_i}$.

Computational complexity. The security proof of SPARKLE+ (Section 4.5) requires an instance of classical Schnorr with complexity η_{Sign} and η_{Verify}. Like MIRAGE, it

exploits commitments to build trusted shared randomness.

$$\begin{aligned}
\eta_{\mathsf{Sign1}} &= \eta_{\mathbb{F}\leftarrow\$} + \eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} \\
\eta_{\mathsf{Sign2}} &= \eta_{\mathsf{Sign}} + \delta \\
&= \eta_{\mathbb{F}\leftarrow\$} + \eta_{\mathbb{F}+} + \eta_{\mathbb{F}\times} + \eta_{\mathbb{G}^\wedge} + \eta_{\mathsf{H}} + \delta \\
\eta_{\mathsf{Sign3}} &= \eta_{\mathbb{F}+} + 2\eta_{\mathbb{F}\times} + (t-1)\eta_{\mathbb{G}\times} + (t+1)\eta_{\mathsf{H}} + \eta_{\mathsf{Lag}} + t\eta_{\mathsf{Verify}} + \delta \\
&= \eta_{\mathbb{F}+} + 2\eta_{\mathbb{F}\times} + (2t-1)\eta_{\mathbb{G}\times} + 2t\eta_{\mathbb{G}\times} + (2t+1)\eta_{\mathsf{H}} + \eta_{\mathsf{Lag}} + \delta \\
\eta_{\mathsf{Combine}} &= (t-1)\eta_{\mathbb{F}+} + (t-1)\eta_{\mathbb{G}\times} + \delta
\end{aligned} \tag{8}$$

Bandwidth complexity. The same reasoning we applied to MIRAGE holds here. Its bandwidth requirements are partially affected by the use of Schnorr signatures.

$$\begin{aligned}
\beta_{\mathsf{Sign1}} &= \beta_{\mathbb{G}} & \beta_{\mathsf{Sign3}} &= \beta_{\mathbb{F}} \\
\beta_{\mathsf{Sign2}} &= 2\beta_{\mathbb{F}} + \beta_{\mathbb{G}} & \beta_{\mathsf{Combine}} &= \beta_{\mathbb{F}} + \beta_{\mathbb{G}}
\end{aligned} \tag{9}$$

4.6 Comparative Analysis

Table 2 contains a summary of our comparisons. As we discussed, its content might depend on the given implementation, and is only to be considered a theoretical comparison. We can perform a slightly informal analysis.

	FROST	FROST2	FROST3	MIRAGE	SPARKLE+
Security					
Static	$\mathsf{AOMDL}^{\mathsf{ROM}}$	$\mathsf{AOMDL}^{\mathsf{ROM}}$	$\mathsf{AOMDL}^{\mathsf{ROM}}$	$\mathsf{DDH}^{\mathsf{ROM}}$	$\mathsf{DL}^{\mathsf{ROM}}$
Adaptive	–	–	–	–	$\mathsf{AOMDL}^{\mathsf{ROM}}_{\mathsf{AGM}}$
Computational cost					
$\mathbb{F}\leftarrow\$$	2	2	2	1	2
$\mathbb{F}+$	$t+1$	$t+1$	$t+1$	t	$t+1$
$\mathbb{F}\times$	3	3	3	2	3
$\mathbb{G}\times$	$4t-4$	$4t-4$	$4t+2$	t	$5t-2$
$\mathbb{G}^\wedge$	$2t+4$	$2t+4$	4	3	2
H	$2t+1$	3	3	$2t+4$	$2t+3$
Lag	1	1	1	1	1
δ	1	1	1	3	2
Bandwidth					
$\mathbb{F}$	2	2	2	3	4
$\mathbb{G}$	3	3	5	3	3

Table 2. Comparison of all considered protocols, with the notation of Section 1.

Security. FROST-like schemes achieve static security under AOMDL, a relatively young assumption. The static security of MIRAGE and SPARKLE+ is based on better understood assumptions, with the latter also achieving adaptive security up to $t-1$ corruptions under AOMDL.

Computational Complexity. Group exponentiation is often one of the costliest operations; if $\mathcal{E}$ is the Edwards Curve25519 and $\mathbb{G}$ its point-group, this is computing $[a]\mathcal{P}$. Another is evaluating hash functions.

Bandwidth requirement. Remark that we analysed the bandwidth from the point of view of a single party $\mathcal{P}_i$. The total communication cost per protocol execution involves all t signers, thus the per-party bandwidth values or Table 2 must be multiplied by t. This is e.g. the total traffic on a signing device executing Π_{TSign}. If $\mathbb{G}$ is the point-group of Curve25519, its points are 32 bytes, and elements in $\mathbb{F}_{2^{255}-19}$ also take around 32 bytes. For FROST, a bandwidth cost of $3\beta_{\mathbb{F}} + 2\beta_{\mathbb{G}}$ per party results in $t \cdot (2 \cdot 32 + 3 \cdot 32) = t \cdot 160$ bytes. See Table 3.

(t, n)	FROST	FROST2	FROST3	MIRAGE	SPARKLE+
(2, 3)	320	320	448	384	448
(5, 10)	800	800	1120	960	1120

Table 3. Total bandwidth (in bytes) using Curve25519 parameters.

5 Industrial Applications

We already discussed how the properties of threshold signature schemes make them well-suited for real-world decentralised infrastructures. Recent research brought Schnorr variants up to the requirements for practical deployment and studied additional properties (e.g. adversarial model, identifiable aborts).
In this final section, we explore how different Schnorr variants can meet the varying needs of applied decentralised domains: blockchain, custodial services, IoT, and electronic voting. We will also formulate educated adoption proposals.

5.1 Blockchain and Cryptocurrencies

Challenges. Threshold signatures underpin the protocols for multisig wallets, validator coordination, and Layer-2 infrastructures [28]. Bitcoin Improvement Proposal 340 introduced Schnorr support to facilitate threshold signing, and actors such as Cardano, Coinbase, ING, and ZenGo maintain open-source implementations. However, they still face adoption barriers. Secure DKG remains complex [18], and interactive signing can suffer from liveness issues in asynchronous or adversarial settings.

On-chain verification is hindered by the absence of native threshold support, requiring expensive or custom logic [21]. Finally, some schemes lack signer traceability, complicating slashing or audit mechanisms.

Performance and scalability. FROST2 and FROST3 excel in bandwidth-sensitive, high-throughput settings (e.g. Layer-2 roll-ups or smart contract platforms) thanks to their low interaction and minimal computations. On the other hand, MIRAGE and SPARKLE+ provide stronger guarantees but may introduce higher latency in the system.

Governance and key rotation. Dynamic systems such as staking protocols, DAOs, or side-chains benefit from support for evolving participant sets. FROST3 offers lightweight key material and efficient signing, making it ideal for mobile validators or rotating committees. SPARKLE+ enables greater auditability and adaptive security, more suited to regulated environments or slashing-based consensus.

Requirement	**Scheme**	**Details**
Low-latency signing	FROST2 FROST3	Minimal number of heavy computations during Sign.
Dynamic membership	MIRAGE	Extensible decentralised Keygen.
Low Bandwidth	FROST2 FROST3	Less broadcasts of lightweight messages.
High auditability	SPARKLE+	Improved signer traceability.
Adaptive security	SPARKLE+	Proven adaptively secure.

Table 4. Blockchain requirements and suitable threshold schemes.

5.2 Custodianship

Challenges. Custodians (e.g. banks, asset managers, enterprise key vaults) operate in centralized but regulated environments and face three main issues [15]. First, key compromising, natively solved upon the adoption of a threshold scheme. Second, the regulatory frameworks (e.g., GDPR [16], SOC2 [1]) could mandate multiparty control, auditability, or restricted insider access. Third, recovery and emergency mechanisms require robust key management.

Robust access control and compliance. Its adaptive security and auditability make SPARKLE+ particularly attractive. Its non-malleability and signer traceability allow precise attribution and enforceable policies. MIRAGE offers similar benefits with lower memory overhead but at the cost of additional rounds.

Hardware-constrained environments. Custodial systems often rely on HSMs or isolated enclaves with tight limits on communication and computation. In such setups, FROST3 and FROST2 are advantageous due to low memory footprint and efficient signing. In particular, FROST3 minimises key material and interaction, enabling lightweight deployment without compromising performance.

Requirement	Scheme	Details
High auditability	SPARKLE+	Improved signer traceability.
Low memory footprint	FROST3	Suitable for constrained environments.
Robust, simple design	FROST3	Balances security assumptions and performance in enterprise settings.
Dynamic membership	MIRAGE	Extensible decentralised Keygen.

Table 5. Custodianship requirements and suitable threshold schemes.

5.3 Internet of Things

Challenges. IoT systems are a network of heterogeneous and resource-constrained devices (e.g. sensors) that must authenticate and cooperate securely. Industrial IoT (IIoT) demands particularly high availability and traceability. Most challenges are of physical nature, e.g. limited computing power or connectivity [20]. Centralized key management, for instance, can be fragile or undesirable, as many use cases usually require distributed trust and fault tolerance [31].

Lightweight and secure orchestration. FROST3 is ideal for low-power, low-memory devices. However, the decentralized, dealerless key generation of MIRAGE could be desirable in some IoT networks. In the case of IIoT, SPARKLE+ offers stronger security and traceability, improving compliance and enabling operational audits.

Requirement	Scheme	Details
Low-power signing	FROST3	Minimal cryptographic cost and key size.
Dealerless Keygen	MIRAGE	Enables trusted setup.
High auditability	SPARKLE+	Improved signer traceability.

Table 6. IoT and IIoT requirements and recommended schemes.

5.4 Electronic voting

Challenges. Secure electronic voting systems must ensure anonymity, public verifiability, coercion-resistance, and fault-tolerance while remaining transparent and verifiable [6]. The election infrastructure must withstand failures and malicious behaviour while minimizing trust assumptions. In some designs, tallying authorities are distributed, requiring joint signing of election results or ballots without revealing individual inputs.

Verifiable and private tallying. Threshold schemes can be used to jointly sign the final tally, ensuring that no single party can alter the result [9]. Due to its strong auditability and adaptive security, SPARKLE+ is particularly well-suited to building trust and ensuring that only authorized coalitions can produce valid results. MIRAGE, computationally-wise comparable to it, could instead be valuable if distributed key generation is required.

Scalable deployment. While not offering the strongest security, FROST3 reduces bandwidth and computations with minimal and efficient interactions, making it suitable for large-scale elections with constrained devices or limited network availability, offering the best trade-off for heterogeneous infrastructures.

Requirement	**Scheme**	**Details**
High auditability	SPARKLE+	Improved signer traceability.
Dealerless Keygen	MIRAGE	Enables trusted setup.
Scalable deployment	FROST3	Efficient and resilient signing protocols.

Table 7. Electronic voting requirements and suitable threshold schemes.

6 Conclusion

Threshold signature schemes are foundational to secure multi-party applications like blockchain, e-voting, and IoT. This work presented a comparative analysis of recent threshold Schnorr variant, with particular focus on their concrete efficiency, auditability, and suitability in practical scenarios. While prior works such as [28] offer broad overviews of the field, our study complements them by providing a detailed and quantitative comparison of specific Schnorr-based threshold protocols.

Future work. Future research may extend this theoretical comparison to threshold variants of other classical signature schemes, such as ECDSA and RSA.
Another promising direction is to broaden the scope of the current study by including emerging Schnorr-based constructions like GLACIUS [2], which pushes the boundaries of adaptive security and efficiency.

Acknowledgements and conflict of interest. The authors acknowledge support from Ripple's University Blockchain Research Initiative. The second and third authors conducted the majority of this research at the University of Trento (Trento, IT). The second author is currently a Ph.D. student at Polytechnic of Turin (Turin, IT). The third author is currently affiliated with the University of Bari "Aldo Moro" (Bari, IT). The second and third authors are members of the INdAM Research Group GNSAGA.

References

1. AICPA, SOC 2 Compliance: A Guide for Financial Institutions, (2022). https://www.aicpa.org/resources/article/soc-2-compliance.
2. Bacho, R., Das, S., Loss, J., Ren, L.: Glacius: Threshold Schnorr Signatures from DDH with Full Adaptive Security. In: Fehr, S., Fouque, P.-A. (eds.) EUROCRYPT 2025, Part II. LNCS, pp. 304–334. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-91124-8_11
3. Battagliola, M., Galli, A., Longo, R., Meneghetti, A., *et al.*: A Provably-Unforgeable Threshold Schnorr Signature With an Offline Recovery Party. In: DLT:ITASEC, CEUR Workshop Proceedings, pp. 60–76 (2022). https://api.semanticscholar.org/CorpusID:251026965
4. Battagliola, M., Longo, R., Meneghetti, A.: Extensible Decentralized Secret Sharing and Application to Schnorr Signatures, Cryptology ePrint Archive, Report 2022/1551 (2022). https://eprint.iacr.org/2022/1551.
5. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93, pp. 62–73. ACM Press (1993). https://doi.org/10.1145/168588.168596
6. Benaloh, J.: Simple verifiable elections. EVT/WOTE (2006)
7. Boneh, D.: The decision Diffie-Hellman problem. In: Third Algorithmic Number Theory Symposium (ANTS). LNCS. Springer (1998)
8. Boneh, D., Drijvers, M., Neven, G.: Compact Multi-signatures for Smaller Blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, pp. 435–464. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_15
9. Burton, C., Culnane, C., Schneider, S.: vVote: Verifiable Electronic Voting in Practice. IEEE Security & Privacy **14**(4), 64–73 (2016). https://doi.org/10.1109/MSP.2016.69
10. Chu, H., Gerhart, P., Ruffing, T., Schröder, D.: Practical Schnorr Threshold Signatures Without the Algebraic Group Model. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, pp. 743–773. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38557-5_24
11. Crites, E., Komlo, C., Maller, M.: 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.
12. Crites, E.C., Komlo, C., Maller, M.: Fully Adaptive Schnorr Threshold Signatures. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, pp. 678–709. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38557-5_22

13. Damgård, I., Koprowski, M.: Practical Threshold RSA Signatures without a Trusted Dealer. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, pp. 152–165. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_10
14. Desmedt, Y.: Society and Group Oriented Cryptography: A New Concept. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, pp. 120–127. Springer, Berlin, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_8
15. Di Nicola, V., Longo, R., Mazzone, F., Russo, G.: Resilient Custody of Crypto-Assets, and Threshold Multisignatures. Mathematics **8**(10), 1773 (2020). https://doi.org/10.3390/math8101773
16. European Union, Understanding the GDPR and its Implications on Key Management, (2021). https://gdpr.eu.
17. Fuchsbauer, G., Kiltz, E., Loss, J.: The Algebraic Group Model and its Applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
18. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. Journal of Cryptology **20**(1), 51–83 (2007). https://doi.org/10.1007/s00145-006-0347-3
19. Komlo, C., Goldberg, I.: FROST: Flexible Round-Optimized Schnorr Threshold Signatures. In: Dunkelman, O., Jacobson, M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, pp. 34–65. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-81652-0_2
20. Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W.: A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. IEEE Internet of Things Journal **4**(5), 1125–1142 (2017). https://doi.org/10.1109/jiot.2017.2683200
21. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple Schnorr multi-signatures with applications to Bitcoin. DCC **87**(9), 2139–2164 (2019). https://doi.org/10.1007/s10623-019-00608-x
22. McCurley, K.S.: The discrete logarithm problem. In: Proc. of Symp. in Applied Math, pp. 49–74 (1990)
23. Micali, S., Rivest, R.L.: Transitive Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, pp. 236–243. Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45760-7_16
24. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple Two-Round Schnorr Multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, pp. 189–221. Springer, Cham, Virtual Event (2021). https://doi.org/10.1007/978-3-030-84242-0_8
25. Nicolosi, A., Krohn, M.N., Dodis, Y., Mazières, D.: Proactive Two-Party Signatures for User Authentication. In: NDSS 2003. The Internet Society (2003)
26. Ruffing, T., Ronge, V., Jin, E., Schneider-Bensch, J., Schröder, D.: ROAST: Robust Asynchronous Schnorr Threshold Signatures. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 2551–2564. ACM Press (2022). https://doi.org/10.1145/3548606.3560583
27. Schnorr, C.-P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22

28. Sedghighadikolaei, K., Yavuz, A.A.: A Comprehensive Survey of Threshold Signatures: NIST Standards, Post-Quantum Cryptography, Exotic Techniques, and Real-World Applications, (2023). https://doi.org/10.48550/ARXIV.2311.05514.
29. Shoup, V.: Practical Threshold Signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, pp. 207–220. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_15
30. Shoup, V.: The many faces of Schnorr, Cryptology ePrint Archive, Report 2023/1019 (2023). https://eprint.iacr.org/2023/1019.
31. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in Internet of Things: The road ahead. Computer networks **76**, 146–164 (2015)