Revisiting the Security of Sparkle

Ojaswi Acharya[1,⋆], Georg Fuchsbauer[3], Adam O'Neill[2], and Marek Sefranek[3]

[1] Stevens Institute of Technology, USA
oacharya@stevens.edu
[2] Manning CICS, UMass Amherst, USA
adamoneill@umass.edu
[3] TU Wien, Austria
firstname.lastname@tuwien.ac.at

Abstract. We revisit the three-round threshold Schnorr signature scheme Sparkle of Crites, Komlo, and Maller (CRYPTO 2023), as well as its variant Sparkle+. While Sparkle+ was accompanied by a claim of full adaptive security, subsequent work identified a gap in the analysis. The original—and simpler and more efficient—Sparkle scheme has so far lacked even a proof of *static* security.

We resolve this by proving static security for Sparkle; our main result is then a *tight* proof of its *full adaptive security* in the *pure* random oracle model, *i.e.*, without relying on the algebraic group model. The core obstacle is that, in the fully adaptive setting, rewinding arguments fundamentally break down. To address this, we base our proof on the *Vandermonde circular discrete-logarithm* (VCDL) assumption, an interactive strengthening of the circular discrete-logarithm assumption of Cho *et al.* (CRYPTO 2025), originally introduced to prove tight security of basic Schnorr signatures. Circular-style assumptions eliminate the need for rewinding. Beyond tightness, our analysis highlights circular-style assumptions as a new approach to achieving security in other settings where rewinding is problematic.

We justify VCDL by reducing it to the *low-dimensional vector representation* (LDVR) problem of Crites *et al.* (CRYPTO 2025) in the elliptic-curve generic group model. Finally, we generalize VCDL (and similarly LDVR) and identify a different assumption within this framework that yields a tight proof of adaptive multi-user security for the basic Schnorr signature scheme, a result of independent interest.

Keywords: threshold signatures · Schnorr signatures · adaptive security · tight security

⋆ Majority of the work was done while the author was a PhD student at UMass Amherst.

Table of Contents

1 Introduction . . . 3
1.1 Background and Goals . . . 3
1.2 Our Contributions . . . 4
1.3 Related Work . . . 8
1.4 Paper Organization . . . 10
2 Preliminaries . . . 10
2.1 Signature Schemes . . . 11
2.2 Threshold Signature Schemes . . . 12
3 VCDL: Our New Assumption . . . 14
4 Full Adaptive Security of Sparkle . . . 15
4.1 The Sparkle Scheme . . . 15
4.2 Full Adaptive Security Under VCDL . . . 15
5 Relationship Between VCDL and LDVR . . . 28
5.1 VCDL Implies LDVR . . . 28
5.2 LDVR Implies VCDL in the EC-GGM . . . 29
6 Adaptive Multi-User Security of Schnorr Signatures . . . 46
6.1 Adaptive Multi-User Unforgeability . . . 47
6.2 Standard-Basis CDL . . . 48
6.3 Main Result . . . 49
6.4 The ICDL and gLDVR Frameworks . . . 53
6.5 Relationship Between ICDL and gLDVR, and Open Questions . . . 56
Acknowledgements . . . 57
References . . . 57
A Reduction from LDVR to VCDL . . . 62
B Error in the Original Full Adaptive Sparkle+ Proof . . . 64

1 Introduction

1.1 Background and Goals

Threshold Schnorr signatures. Threshold signatures distribute signing power across multiple parties while still producing a single short signature. In an t-out-of-n threshold signature (TS) scheme, there are n parties, any t of which can jointly generate a valid signature under a common public key. Among such schemes, *threshold Schnorr signatures* are especially attractive, since the resulting signature is a standard Schnorr signature. This enables drop-in compatibility with existing verification infrastructure (*e.g.*, BIP-340 for Bitcoin), without requiring protocol or software changes. Such compatibility is particularly important in blockchain and distributed-systems settings, where minimizing overhead and preserving interoperability are critical.

Our focus: Sparkle. The three-round threshold Schnorr signature scheme Sparkle (the first round being message-independent), introduced by Crites, Komlo, and Maller (CKM) [CKM23b] is compelling from both foundational and practical standpoints. From a foundational perspective the scheme is remarkably natural given CKM's explicit goal of adaptive security. From a practical perspective, it achieves the lowest communication and computational cost per signer among recent low-round adaptively secure threshold Schnorr signatures schemes (see Table 2 for a concrete comparison). With the ongoing NIST threshold cryptography standardization process [BD24, BP26], it is crucial to know what level of security the scheme achieves and under what assumptions. Indeed, Sparkle essentially follows one of the blueprints laid out in the NIST call [BD24, Section 4.2.2].

In terms of existing analyses, Sparkle originally came with a claim of *full adaptive security*, in which the adversary may corrupt parties on the fly (up to one fewer than the signing threshold), even after observing protocol transcripts, and learns their entire internal state (without assuming secure erasures). This notion is challenging to achieve and well motivated in recent work. Moreover, as above, full adaptive security is explicitly identified as a desideratum in the NIST call [BD24, BP26].

CKM argued their claim of full adaptive security for Sparkle in the random oracle model [BR93] combined with the algebraic group model [FKL18], under the algebraic one-more discrete-logarithm (AOMDL) assumption. AOMDL, which will be relevant to us later, is an interactive strengthening of the discrete-logarithm assumption in which the adversary is given access to a discrete-log oracle that answers queries on linear combinations of the challenges.

Gaps in existing analyses. Unfortunately, subsequent work to CKM identified two fundamental gaps in the original security analysis of Sparkle and its variants:

- The first concerns *static security*. Bacho *et al.* [BLT+24] observed that the original proof does not correctly handle executions in which honest parties hold inconsistent local views, a situation that can arise even under static

corruptions.[1] As a result, the original simulation strategy is invalid. To address this issue, Crites, Komlo, and Maller proposed a modified scheme, Sparkle+, in the full version of their work [CKM23a], in which parties sign their local views using an auxiliary signature scheme. This introduces significant overhead, making the scheme less appealing in practice. While the modification suffices to recover a (loose) proof of static security in the random oracle model, the original Sparkle protocol has remained without such a proof.
- The second gap concerns *full adaptive security*. Crites and Stewart [CS25b] and subsequently Crites *et al.* [CKK+25] showed that any threshold Schnorr signature scheme such as Sparkle whose public key shares are Shamir shares of the secret key in the exponent necessarily requires the hardness of algebraic problems over the underlying field to achieve full adaptive security. The original analysis of CKM does not seem to rely on such an assumption (and, in fact, their reduction is flawed, even in the case of Sparkle+ as we show in Appendix B), a gap they later acknowledged [CKM23a].

While neither gap yields an explicit attack on the scheme itself, they leave Sparkle and Sparkle+ without any proof-based support for full adaptive security (and, in the case of Sparkle, even for static security). We therefore raise a fundamental question:

> *What assumptions and idealized models are sufficient to establish static and more importantly full adaptive security for* Sparkle*, if any?*

Our goal: Tight full adaptive security in the pure ROM. In answering this question, our goal will be to obtain *tight* reductions establishing *full adaptive security* in the *pure random oracle model*, without appealing to the algebraic group model (AGM) as done in CKM's original (flawed) analysis. Firstly, tightness matters: loose reductions mean increased concrete parameters, an issue already present for basic Schnorr signatures that can be significantly worse in the threshold setting. Moreover, while the AGM+ROM can enable both tight bounds and full adaptive security (*e.g.*, for FROST [CKK+25]), it imposes strong algebraic restrictions on the adversary that obscure insight into the scheme's true security. Indeed, in comparison to the ROM, the AGM is a comparatively recent idealization that is not yet fully understood; several works have highlighted subtleties and limitations in its use [Zha22, ZZK22]. It is therefore prudent to avoid the AGM when possible and stick to the pure ROM.

1.2 Our Contributions

Static security of Sparkle. Along the way to full adaptive security, we first revisit static security of the original Sparkle scheme. By adapting the delayed-programming and equivalence-class techniques of Bacho *et al.* [BLT+24], which they originally developed for their scheme Twinkle, we obtain a proof of static security in the random oracle model. While the techniques are primarily due

[1] See "A Gap in the Proof of Sparkle" in [BLT+24, Section 1.2] for a detailed account.

to [BLT+24], there is no prior result adapting them to Sparkle as far as we know. The reduction is either loose under the discrete-logarithm assumption, or *tight* under the circular discrete-logarithm (CDL) assumption of Cho *et al.* [CFOS25]. In the body, we frame this as a corollary to our proof of full adaptive security.

Why full adaptive security is different. Full adaptive security poses a fundamentally different challenge, and extending the above proof to this setting requires entirely new techniques. In the full adaptive setting, the adversary may corrupt up to t parties throughout multiple signing sessions, where $t+1$ is the signing threshold. In threshold Schnorr schemes where public keys are Shamir shares of the secret key in the exponent, each corruption reveals an evaluation of a degree-t polynomial whose constant term is the signing key. After sufficiently many corruptions, *no hidden secret remains for a reduction to leverage*. As a result, classical rewinding [PS96] arguments break down, as the adversary can corrupt different users in each branch of the forked execution.

To overcome this issue, our main idea is to use *circular* assumptions, in the spirit of the circular discrete-logarithm (CDL) assumption. CDL is related to the semi-logarithm problem introduced by Brown [Bro05] and a special case of its generalization by Fersch, Kiltz, and Poettering [FKP17]. Cho *et al.* [CFOS25] use CDL to obtain tight security for basic Schnorr signatures. In this case, a successful forgery under key $X = g^x$ does not lead to extraction of the signing key x. Instead, it yields a solution $(R, z) \in \mathbb{G} \times \mathbb{Z}_p$ to a circular relation of the form $g^z = R \cdot X^{f(R)}$ for a fixed conversion function $f\colon \mathbb{G} \to \mathbb{Z}_p$ (*e.g.*, the ECDSA conversion function).[2] Therefore, the reduction never needs to extract the signing key, rewinding is avoided entirely, and the reduction is tight.

In other words, we leverage circular-style assumptions as a new approach to achieve full adaptive security for threshold Schnorr schemes with Shamir shares in the exponent, *even disregarding tightness*. Our approach may be useful in future work that considers other settings where rewinding is inherently problematic.

The VCDL assumption. Following this approach, we isolate an interactive extension of the CDL assumption suitable for full adaptive security of Sparkle. (Note that interactive assumptions are somewhat inherent here [CKM25], see Section 1.3.) We call the resulting assumption the *Vandermonde circular discrete-logarithm* (VCDL) assumption. In our assumption $\mathsf{VCDL}[\mathbb{G}, f, n, t]$, the adversary receives $(X_0, X_1, \ldots, X_t) \in \mathbb{G}^{t+1}$ with $X_i = g^{x_i}$. It may query a discrete-log oracle on indices $k \in [n]$, obtaining

$$x_0 + \sum_{i=1}^{t} k^i x_i,$$

that is, evaluations of a degree-t polynomial. To win, the adversary must output (R, z) satisfying

$$g^z = R \cdot X_0^{f(R)} \quad \text{with } f(R) \neq 0.$$

[2] Note that f can be *any* efficient function, not necessarily the hash function used to instantiate Schnorr; indeed, f appears only in the proof, not the scheme.

From VCDL to tight full adaptive security. As our main result, we prove that full adaptive security of Sparkle reduces *tightly* to VCDL in the pure ROM. Adaptive corruption queries in Sparkle reveal Shamir shares of the signing key, which match the Vandermonde evaluations provided by the discrete-log oracle in VCDL. Our reduction therefore answers corruption queries using its discrete-log oracle. Signing queries are simulated using the static-corruption techniques and additional bookkeeping, without invoking the VCDL oracle. Since a successful forgery yields a VCDL solution, the reduction does not need to extract the signing key or rewind.

Our proof only invokes the VCDL oracle for corruption queries, and the parameters of the VCDL assumption are independent of the number of signing queries made by the adversary. On the other hand, the proofs of FROST [BTZ22, CKK+25] under the AOMDL assumption need to invoke the DL oracle for every signing query in addition to corruption queries. While signing queries can exceed the number of parties by orders of magnitude, our assumption is parameterized by only the signing threshold, scaling with the static scheme parameters rather than the adversary's dynamic resources.

Discussion of VCDL. For non-circular assumptions such as AOMDL, allowing the adversary to query arbitrary linear combinations of the underlying secrets still yields a plausible assumption. For circular assumptions, however, such freedom immediately renders the assumption false. If arbitrary linear queries were permitted, then querying the vector $(f(X_1), 1, 0, \ldots, 0)$ would reveal $z := f(X_1)x_0 + x_1$, which yields a valid circular solution together with $R := X_1$.

VCDL is useful for Sparkle but not for FROST [KG20], in which the signers' public keys are also Shamir shares of the signing key in the exponent. For FROST, answering signing queries in the reduction requires discrete-log oracle queries for arbitrary linear combinations of the underlying secrets. But, as we have pointed out, in the setting of circular assumptions (which is needed to avoid rewinding), such access corresponds to a circular analogue of AOMDL—which is false. This helps to understand why the proof of full adaptive security for FROST [CKK+25] requires the algebraic group model whereas our Sparkle proof does not.

Justifying VCDL via LDVR. It is customary to justify a new group-theoretic assumption by showing it is hard when idealizing the group. In our case, we obtain a conditional result based on the *low-dimensional vector representation* (LDVR) problem of Crites *et al.* [CKK+25], which is known to be necessary for full adaptive security of Sparkle and related schemes. Namely, we show that VCDL tightly reduces to LDVR in the elliptic-curve generic group model of Groth and Shoup [GS22] (and conversely that VCDL tightly implies LDVR in the standard model). For this, we combine techniques from the proof of CDL in the EC-GGM due to Cho *et al.* [CFOS25] with the proof of one-more discrete log (OMDL) [BNPS03] in Shoup's GGM [Sho97] due to Bauer, Fuchsbauer, and Plouviez [BFP21] in a non-trivial manner, as well as new arguments tailored to LDVR. Interestingly, in addition to (approximate) regularity of the conversion function f as required for hardness of CDL in the EC-GGM as shown by [CFOS25], we need that f is *preimage-sampleable*. This is satisfied by the ECDSA conversion

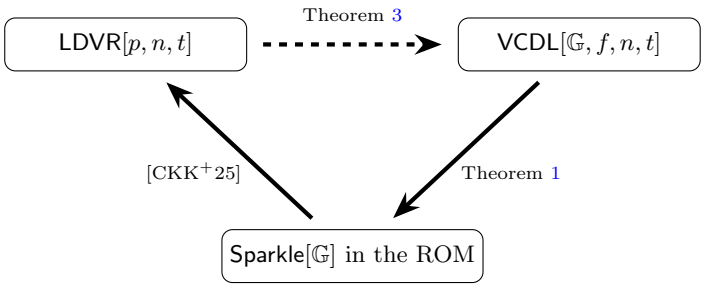

Fig. 1. Implications shown in this work. $\mathsf{VCDL}[\mathbb{G}, f, n, t]$ is for regular, preimage-sampleable f. "$\mathsf{Sparkle}[\mathbb{G}]$ in the ROM" refers to full adaptive security. A dashed arrow indicates the implication is shown in the EC-GGM. All reductions are tight.

function originally suggested by [CFOS25] as a candidate choice for f. Thus, up to idealization of the group, VCDL and LDVR are *equivalent*, positioning VCDL as the natural group-theoretic analogue of LDVR (cf. Figure 1).

Discussion of LDVR. While it is a new assumption, LDVR and its generalization *Problem P* [CS25b] are connected to existing linear-algebra problems: Crites and Stewart demonstrate the relationship between Problem P and bounded distance decoding as well as proximity gaps of Reed–Solomon codes [CS25b, CS25a]. Proximity gaps, in turn, are essential in hash-based succinct non-interactive arguments of knowledge (SNARKs).

Furthermore, Crites et al. [CKK+25] discuss various parameter regimes for hardness of LDVR. For n total signers, corruption threshold t_c, and signing threshold $t+1$, there are $\binom{n}{t_c}$ possibilities for the corrupt set. For t_c close to 0 or n, the problem is unconditionally hard because a solution only exists with negligible probability. Additionally, when $t_c < t/2$, the adversary's winning probability is bounded to a trivial $q/2^\kappa$ (where q is the number of queries and κ is the security parameter).

Multi-user Schnorr and a general assumption framework. As a result of independent interest, we show a *tight* proof of adaptive multi-user unforgeability for the basic Schnorr signature scheme in the ROM. Despite the widespread use of the scheme, to the best of our knowledge, no such proof was previously known. (A trivial reduction to single-user unforgeability incurs a factor-n loss that is prohibitive for large-scale deployments.) To this end, we abstract away the Vandermonde vectors in VCDL (and similarly LDVR) and replace them with an arbitrary family of vectors satisfying a simple non-degeneracy condition. We identify another special case—the *standard-basis circular discrete-logarithm* (SB-CDL) assumption—under which our reduction goes through. We justify SB-CDL by proving it is unconditionally hard in the EC-GGM. From this perspective, VCDL is not merely an isolated, scheme-specific assumption, but part of a broader framework that we expect to be useful in future work.

Table 1. Security guarantees for recent low-round fully adaptively secure threshold Schnorr signature schemes. Among these, Sparkle is the only scheme achieving a tight reduction in the *pure* random oracle model. ↯ denotes a claimed result later shown to be invalid.

Scheme	Source	Tight Reduction	Model	Assumptions
Sparkle	**This work**	**Yes**	**ROM**	**VCDL**
Rackle	[NRT26]	No	ROM	DL
FROST	[CKK+25]	Yes	ROM+AGM	AOMDL+LDVR
ms-FROST	[BCL+26]	Yes	ROM+AGM	AOMDL
FaFROST	[BGC+25]	Yes	ROM+AGM	AOMDL
Sparkle+↯	[CKM23a]	Yes	ROM+AGM	AOMDL
Gargos	[BDLR25a]	No	ROM	DDH

Perspective. We do not claim that Sparkle is the optimal threshold Schnorr signature scheme, nor that VCDL is a standard assumption. Rather, we want to understand which assumptions and idealized models suffice to prove full adaptive security of threshold Schnorr signature schemes, particularly in which each signer's public key is simply a Shamir share of the secret key in the exponent, as is the case for Sparkle and FROST. For such schemes, rewinding introduces a substantial technical challenge, which can be overcome by resorting to the AGM. Nevertheless, our results show that for Sparkle we can avoid the AGM and give a proof in the *pure* ROM, even with a *tight* reduction.

1.3 Related Work

Impossibility results. Recently, Crites, Komlo, and Maller [CKM25] presented impossibility results that clarify the space of assumptions necessary to prove full adaptive security of both *key-unique* threshold signatures, as well as giving stronger results for the special case of threshold Schnorr signatures like Sparkle. At a high level, they show that: (1) the former cannot achieve full adaptive security, via a black-box reduction, under any non-interactive search assumption; and (2) the latter cannot achieve full adaptive security, via certain *rewinding* reductions, even under certain interactive assumptions. Our results do not contradict theirs: we avoid (1) because VCDL is interactive and (2) because we do not rewind.

Recent threshold Schnorr signatures. Threshold Schnorr signatures have long been studied, see *e.g.* Gennaro *et al.* [GJKR99]. Since Komlo and Goldberg introduced FROST [KG20], there has been a surge of work on highly efficient schemes [KG20, BCK+22, RRJ+22, Mak22, CKM23a, CKM23b, CGRS23, Lin24, KRT24, BDLR25b, Che25b, GCRS26]. These works explore various tradeoffs in round complexity, security models, assumptions, communication cost, robustness, and network synchrony (cf. Tables 1 and 2). In particular, several refinements and

Table 2. Concrete efficiency per signer across all rounds for recent low-round fully adaptively secure threshold Schnorr signature schemes. $\mathbb{G}$ denotes a group element, $\mathbb{Z}_p$ a field element, and Exp a group exponentiation.

Scheme	Rounds	Comm./signer	Comp./signer
Sparkle	**3**	$\mathbf{1}\mathbb{G} + \mathbf{2}\mathbb{Z}_p$	**1 Exp**
FROST	2	$2\mathbb{G} + 1\mathbb{Z}_p$	3 Exp
ms-FROST	2	$2\mathbb{G} + 1\mathbb{Z}_p$	3 Exp + masking
FaFROST	2	$2\mathbb{G} + 1\mathbb{Z}_p$	3 Exp + masking
Rackle	3	$1\mathbb{G} + 2\mathbb{Z}_p +$ NIZK	1 Exp + NIZK.P + t NIZK.V
Sparkle+	3	$1\mathbb{G} + 2\mathbb{Z}_p +$ DS	1 Exp + DS.S + t DS.V
Gargos	3	$2\mathbb{G} + 7\mathbb{Z}_p$	8 Exp + NIZK.P + t NIZK.V

variants of FROST have been proposed; see, for example, Bellare *et al.* [BCK+22]. When discussing the costs of FROST, we use the optimized variant introduced in that work (often referred to as FROST2).

Tight threshold signatures. A closely related line of work explores threshold signatures with *tight* security from standard, non-interactive assumptions. Recent constructions include Twinkle-T [BW26], Dazzle-T [Che25a], and Earpicks-TS [BC26], the first two of which achieve adaptive security. All of these schemes are proven secure under the decisional Diffie–Hellman (DDH) assumption in the random oracle model. However, unlike Sparkle, they are not compatible with standard Schnorr signatures, limiting their drop-in compatibility with existing verification infrastructure.

Lattice-based schemes. A parallel line of work has explored analogous *lattice-based* threshold signatures, beginning with TRaccoon [DKM+24], which follows the blueprint of Sparkle while introducing masking and noise-management techniques needed in the lattice setting. This further underscores the relevance of analysis of the core Sparkle design. Subsequent work [TZ23, EKT24, KRT24, CATZ24, BKL+25, PN25, PKN+25, GHK+25] proposes alternatives. While practitioners appear to favor threshold Schnorr signatures for near-term deployment, lattice-based threshold signatures provide potential post-quantum successors.

Multi-user security of Schnorr signatures. The *static* multi-user setting for signature schemes goes back at least to Galbraith, Malone-Lee, and Smart [GMS02], who noted a trivial reduction to single-user security incurring a factor-n loss. Bernstein [Ber15] later pointed out an error in the Schnorr-specific tight reduction in [GMS02] and advocated for "key-prefixing," proving a tight implication for the key-prefixed variant of basic Schnorr signatures. This was subsequently addressed by Kiltz, Masny, and Pan [KMP16], who gave tight multi-user security proofs for Schnorr signatures (without key-prefixing) within a broader framework for Fiat–Shamir signatures.

Tight *adaptive* multi-user security for signature schemes was introduced by Bader *et al.* [BHJ+15] as a tool for constructing tightly secure authenticated key exchange, along with efficient constructions satisfying the new notion. The subject continues to be an active area of study; *e.g.*, recent work of Hashimoto *et al.* [HOS25] achieves tight adaptive multi-user security from non-interactive search assumptions; however, the proposed schemes are not Schnorr-based.

Concurrent work. Several concurrent works study full adaptive security for low-round threshold Schnorr signatures:

- Bacho *et al.* [BCL+26], and independently Baecker *et al.* [BGC+25], introduce masked variants of FROST called msFROST and FaFROST, respectively. These schemes achieve full adaptive security in the ROM+AGM under the AOMDL assumption, avoiding LDVR. Their approach handles adaptive corruptions by modifying the scheme to blind final-round signature shares using pairwise one-time masks.
- Niot *et al.* [NRT26] propose Rackle, a three-round threshold Schnorr scheme achieving full adaptive security in the pure ROM under the discrete-log assumption. While their construction avoids the AGM, they do not achieve tight security and rely on an auxiliary simulation-extractable proof system.

Our analysis of Sparkle takes a complementary approach to the above works. Rather than modifying the scheme or introducing auxiliary components, we show that full adaptive security can be established for the original Sparkle scheme *as-is*, via a tight reduction in the pure random oracle model under VCDL.

1.4 Paper Organization

Section 2 recalls preliminaries, including signature schemes, threshold signatures, and their security notions. Section 3 introduces our new *Vandermonde circular discrete-logarithm* (VCDL) assumption. Section 4 presents the Sparkle scheme from [CKM23b] and our main result: a tight proof of full adaptive security of Sparkle under VCDL in the pure ROM. Section 5 establishes a tight equivalence between VCDL and the low-dimensional vector representation (LDVR) assumption in the elliptic-curve generic group model. Section 6 introduces our new *standard-basis circular discrete-logarithm* (SB-CDL) assumption and uses it to give a tight proof of adaptive multi-user unforgeability for basic Schnorr signatures in the pure ROM. It further develops the *interactive circular discrete-logarithm* (ICDL) and *generalized low-dimensional vector representation* (gLDVR) frameworks, which generalize VCDL/SB-CDL and LDVR, respectively. Appendix A contains the full details of the reduction from LDVR to VCDL, and Appendix B pinpoints the error in the original full adaptive security proof of Sparkle+.

2 Preliminaries

Notation. For a vector $\boldsymbol{v}$, we write $|\boldsymbol{v}|$ for its dimension (*i.e.*, the number of coordinates), and $\boldsymbol{v}[i]$ for its i-th coordinate. For vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, we denote

their concatenation by $\boldsymbol{v}\|\boldsymbol{w}$. If Q is a list of tuples, then $Q[x, y, \ldots]$ denotes the sublist of all tuples in Q whose prefix is $(x, y, \ldots)$; we use '$\cdot$' as a wildcard entry.

For a finite set S, we write $|S|$ for its cardinality and use $x \leftarrow\!\!\$\ S$ to denote sampling an element uniformly at random from S and assigning it to x. We write $[n] := \{1, \ldots, n\} \subset \mathbb{N}$ and $[0, n] := \{0, 1, \ldots, n\} \subset \mathbb{N}$. For a set $S \subseteq [n]$, the i-th Lagrange coefficient with respect to S is defined as $\lambda_i^S := \prod_{j \in S \setminus \{i\}} \frac{j}{j-i}$.

For a function $f\colon X \to Y$, we use $\mathrm{dom}(f) := X$ and $\mathrm{ran}(f) := \{f(x) \mid x \in X\}$ for its domain and range, respectively. We say that f is *regular* if, for every $y \in \mathrm{ran}(f)$, the preimage $f^{-1}(y)$ has the same size; equivalently, f is d-to-1 where $d := |\mathrm{dom}(f)|/|\mathrm{ran}(f)|$.

For an algorithm A, we let $y \leftarrow \mathrm{A}^{\mathrm{O}_1,\cdots}(x_1, \ldots; \omega)$ denote running A on inputs $x_1, \ldots$ and coins ω, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. By $y \leftarrow\!\!\$\ \mathrm{A}^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ we denote picking ω uniformly at random and letting $y \leftarrow \mathrm{A}^{\mathrm{O}_1,\cdots}(x_1, \ldots; \omega)$.

Games. We use the code-based game-playing framework of Bellare and Rogaway [BR06]. By $\Pr[\mathbf{G} \Rightarrow y]$ we denote the probability that the execution of game $\mathbf{G}$ results in the output being y.

2.1 Signature Schemes

A *signature scheme* with message space MS is defined by a tuple of algorithms $\mathsf{DS} = (\mathsf{DS.K}, \mathsf{DS.S}, \mathsf{DS.V})$. The key-generation algorithm $\mathsf{DS.K}$ outputs a key pair (vk, sk). The signing algorithm $\mathsf{DS.S}$ takes as input a signing key sk and a message m, and outputs a signature σ. The verification algorithm $\mathsf{DS.V}$ takes as input a verification key vk, a message m, and a signature σ, and outputs a bit.

Correctness and security. Informally, a signature scheme is correct if a message-signature pair generated from a secret key passes verification under the corresponding public key. Similarly, security of a signature scheme requires that an adversary, even after observing many valid message-signature pairs, cannot generate a valid signature on a message not previously signed.

Schnorr signatures. Let $\mathbb{G}$ be a cyclic group of prime order $p = |\mathbb{G}|$, generated by g. Let $H\colon \mathbb{G} \times \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_p$ be a hash function. (Note that we hash the verification key as the first input, which is common in the multi-user setting we consider.) The *Schnorr signature scheme* [FS87, Sch90] $\mathsf{Sch}[\mathbb{G}, H] = (\mathsf{Sch.K}, \mathsf{Sch.S}, \mathsf{Sch.V})$ with message space $\{0,1\}^*$ works as follows. Algorithm $\mathsf{Sch.K}$ chooses $x \leftarrow\!\!\$\ \mathbb{Z}_p$, sets $X \leftarrow g^x$, and returns $(vk = X, sk = x)$. Algorithm $\mathsf{Sch.S}$ on input x, m chooses $r \leftarrow\!\!\$\ \mathbb{Z}_p$, sets $R \leftarrow g^r$ and $c \leftarrow H(X, m, R)$, then returns $\big(R, (r + cx) \bmod p\big)$. Algorithm $\mathsf{Sch.V}$ on input $X, m, (R, z)$ returns 1 iff $g^z = R \cdot X^c$ where $c \leftarrow H(X, m, R)$. Correctness is straightforward to check. In the ROM, we denote the scheme by $\mathsf{Sch}[\mathbb{G}]$.

Note that, unlike in [CFOS25], we define the hash function in Schnorr signatures above to be *public-key prefixed*; that is, a hash query by user i is prepended with its public key X_i. This will be important for our proof of tight adaptive multi-user unforgeability of Schnorr signatures in Section 6.

2.2 Threshold Signature Schemes

Our formalizations here largely follow Crites, Komlo, and Maller [CKM23a], with some adaptations to be consistent with the rest of our work. In particular, following their work, we consider 3-round threshold signature schemes where the first round is *message independent*.

3-ROUND THRESHOLD SIGNATURES. A *three-round threshold signature scheme* with message space MS is a tuple of algorithms

$$\mathsf{TS} = (\mathsf{TS.KeyGen}, (\mathsf{TS.Sign}_i)_{i\in\{1,2,3\}}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$$

that work as follows:

- $\mathsf{TS.KeyGen}(n, t+1)$: The key-generation algorithm, on input the number of signers $n \in \mathbb{N}$ and signing threshold $t+1 \leq n$, outputs a verification key vk, verification key shares $(vk_i)_{i\in[n]}$, and private key shares $(sk_i)_{i\in[n]}$.
- $\mathsf{TS.Sign}_1(i, sk_i)$: The round-one signing algorithm, on input a signer index $i \in [n]$ and signing key share sk_i, outputs a round-one protocol message $pm_{i,1}$ and signer state $st_{i,1}$.
- $\mathsf{TS.Sign}_2(i, \mathcal{S}, m, (pm_{j,1})_{j\in\mathcal{S}}, st_{i,1})$: The round-two signing algorithm, on input a signer index $i \in [n]$, set of signers $\mathcal{S} \subseteq [n]$, message $m \in \mathsf{MS}$, round-one protocol messages $(pm_{j,1})_{j\in\mathcal{S}}$, and signer state $st_{i,1}$, outputs a round-two protocol message $pm_{i,2}$ and signer state $st_{i,2}$.
- $\mathsf{TS.Sign}_3(i, (pm_{j,2})_{j\in\mathcal{S}}, st_{i,2})$: The round-three signing algorithm, on input a signer index $i \in [n]$, round-two protocol messages $(pm_{j,2})_{j\in\mathcal{S}}$, and signer state $st_{i,2}$, outputs a round-three protocol message $pm_{i,3}$.
- $\mathsf{TS.Combine}(\mathcal{S}, m, (pm_{j,2})_{j\in\mathcal{S}}, (pm_{j,3})_{j\in\mathcal{S}})$: The aggregation algorithm, on input the set of signers $\mathcal{S} \subseteq [n]$, message $m \in \mathsf{MS}$, round-two protocol messages $(pm_{j,2})_{j\in\mathcal{S}}$, and round-three protocol messages $(pm_{j,3})_{j\in\mathcal{S}}$, outputs a signature σ.
- $\mathsf{TS.Verify}(vk, m, \sigma)$: The verification algorithm, on input a verification key vk, message $m \in \mathsf{MS}$, and signature σ, outputs a bit.

For correctness, we require that for every $n, t \in \mathbb{N}$ with $n > t$, $\mathcal{S} \subseteq [n]$ such that $|\mathcal{S}| \geq t+1$, and $m \in \mathsf{MS}$, $\Pr\left[\mathbf{G}^{\mathrm{cor}}_{\mathsf{TS},n,t,\mathcal{S},m} \Rightarrow 1\right] = 1$ where the game is in Figure 2.

Game $\mathbf{G}^{\mathrm{cor}}_{\mathsf{TS},n,t,\mathcal{S},m}$

1 $(vk, (vk_i)_{i\in[n]}, (sk_i)_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{TS.KeyGen}(n, t+1)$
2 For $i \in \mathcal{S}$: $(pm_{i,1}, st_{i,1}) \leftarrow\!\!\$\ \mathsf{TS.Sign}_1(i, sk_i)$
3 For $i \in \mathcal{S}$: $(pm_{i,2}, st_{i,2}) \leftarrow\!\!\$\ \mathsf{TS.Sign}_2(i, \mathcal{S}, m, (pm_{j,1})_{j\in\mathcal{S}}, st_{i,1})$
4 For $i \in \mathcal{S}$: $pm_{i,3} \leftarrow\!\!\$\ \mathsf{TS.Sign}_3(i, (pm_{j,2})_{j\in\mathcal{S}}, st_{i,2})$
5 $\sigma \leftarrow \mathsf{TS.Combine}(\mathcal{S}, m, (pm_{j,2})_{j\in\mathcal{S}}, (pm_{j,3})_{j\in\mathcal{S}})$
6 Return $\mathsf{TS.Verify}(vk, m, \sigma)$

Fig. 2. Game defining correctness of TS.

Game $\mathbf{G}^{\text{adp-ts-uf}}_{\mathsf{TS},n,t}$

INITIALIZE:
1 $\mathsf{Q}_{st}, \mathsf{Q}_m \leftarrow \emptyset$; $eid \leftarrow 0$
2 $\mathsf{HS} \leftarrow [n]$; $\mathsf{CS} \leftarrow \emptyset$
3 $(vk, (vk_i)_{i\in[n]}, (sk_i)_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{TS.KeyGen}(n, t+1)$
4 Return $(vk, (vk_i)_{i\in[n]})$

SIGNO$_1(k)$: // $k \in \mathsf{HS}$
5 $eid \leftarrow eid + 1$
6 $(pm_{k,eid,1}, st_{k,eid,1}) \leftarrow\!\!\$\ \mathsf{TS.Sign}_1(k, sk_k)$
7 $\mathsf{Q}_{st}[k, eid, 1] \leftarrow st_{k,eid,1}$
8 Return $(eid, pm_{k,eid,1})$

SIGNO$_2(k, eid, \mathcal{S}, m, (pm_{i,eid_i,1})_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 1] \neq \bot, \mathsf{Q}_{st}[k, eid, 2] = \bot$
9 $\mathsf{Q}_m \leftarrow \mathsf{Q}_m \cup \{m\}$
10 $st_{k,eid,1} \leftarrow \mathsf{Q}_{st}[k, eid, 1]$
11 $(pm_{k,eid,2}, st_{k,eid,2}) \leftarrow\!\!\$\ \mathsf{TS.Sign}_2(k, \mathcal{S}, m, (pm_{i,eid_i,1})_{i\in\mathcal{S}}, st_{k,eid,1})$
12 $\mathsf{Q}_{st}[k, eid, 2] \leftarrow st_{k,eid,2}$
13 Return $pm_{k,eid,2}$

SIGNO$_3(k, eid, (pm_{i,eid_i,2})_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 2] \neq \bot, \mathsf{Q}_{st}[k, eid, 3] = \bot$
14 $st_{k,eid,2} \leftarrow \mathsf{Q}_{st}[k, eid, 2]$
15 $pm_{k,eid,3} \leftarrow\!\!\$\ \mathsf{TS.Sign}_3(k, (pm_{2,eid_i,1})_{i\in\mathcal{S}}, st_{k,eid,2})$
16 $\mathsf{Q}_{st}[k, eid, 3] \leftarrow pm_{k,eid,3}$
17 Return $pm_{k,eid,3}$

CORRUPTO(k): // $k \in \mathsf{HS}, |\mathsf{CS}| < t$
18 $\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$; $\mathsf{HS} \leftarrow \mathsf{HS} \setminus \{k\}$
19 $st_k \leftarrow \mathsf{Q}_{st}[k, \cdot, \cdot]$
20 Return (sk_k, st_k)

FINALIZE(m, σ):
21 If $m \in \mathsf{Q}_m$: Return 0
22 Return $\mathsf{TS.Verify}(vk, m, \sigma)$

Fig. 3. Game defining adaptive security of TS.

FULL ADAPTIVE SECURITY. We define adaptive existential unforgeability under chosen-message attack for TS schemes (ADP-TS-UF). Our definition of security is adapted from [CKM23a, Definition 8]. Let

$$\mathsf{TS} = (\mathsf{TS.KeyGen}, (\mathsf{TS.Sign}_i)_{i\in\{1,2,3\}}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$$

with message space MS be a three-round TS scheme. For an adversary A, we let its ADP-TS-UF advantage against TS be

$$\mathbf{Adv}^{\text{adp-ts-uf}}_{\mathsf{TS},n,t}(\mathrm{A}) = \Pr\left[\mathbf{G}^{\text{adp-ts-uf}}_{\mathsf{TS},n,t}(\mathrm{A}) \Rightarrow 1\right],$$

where the game is in Figure 3.

Intuitively, this notion of security allows the adversary to initiate concurrent signing sessions while being able to adaptively corrupt signers. To simplify handling of each signer's state during concurrent sessions, the security game increments a global counter *eid* (execution identifier) whenever the adversary

initiates a new session with an honest signer. The adversary wins if it forges on a message for which it never initiated an execution of the signing protocol. Moreover, the adversary fully controls the network; in particular, we do *not* assume broadcast or authenticated channels.

3 VCDL: Our New Assumption

We introduce the *Vandermonde circular discrete-logarithm* (VCDL) assumption, which combines the circular structure of CDL [CFOS25] with a restricted discrete-log oracle. Namely, the oracle returns evaluations $x_0 + \sum_{i=1}^{t} k^i x_i$ of a degree-t polynomial at queried indices $k \in [n]$. The Vandermonde structure is not merely convenient for our analysis of Sparkle; it is *necessary for soundness*: arbitrary linear queries as in the algebraic one-more discrete-logarithm (AOMDL) assumption would falsify the assumption, since querying $(f(X_1), 1, 0, \ldots, 0)$ yields a valid circular solution with $R := X_1$. We later justify VCDL via the low-dimensional vector representation (LDVR) assumption [CKK+25], as detailed in Section 5.

THE DEFINITION. Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, generated by g. Let $f\colon \mathbb{G} \to \mathbb{Z}_p$ be an efficient function. Let $n, t \in \mathbb{N}$ be parameters with $t < n$. For an adversary A we let its VCDL-advantage against $\mathbb{G}, g, f, n, t$ be

$$\mathbf{Adv}^{\mathrm{vcdl}}_{\mathbb{G},g,f,n,t}(\mathrm{A}) = \Pr\left[\mathbf{G}^{\mathrm{vcdl}}_{\mathbb{G},g,f,n,t}(\mathrm{A}) \Rightarrow 1\right],$$

where the game is in Figure 4 (right). We include the CDL game for comparison in Figure 4 (left).

Game $\mathbf{G}^{\mathrm{cdl}}_{\mathbb{G},g,f}$

INITIALIZE:
1 $x \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ X \leftarrow g^x$
2 Return X

FINALIZE(R, z):
3 Return $\left(f(R) \neq 0 \wedge g^z = R \cdot X^{f(R)}\right)$

Game $\mathbf{G}^{\mathrm{vcdl}}_{\mathbb{G},g,f,n,t}$

INITIALIZE:
1 $Q \leftarrow \emptyset$
2 For $i \in [0, t]$:
3 $\quad x_i \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ X_i \leftarrow g^{x_i}$
4 Return $(X_0, X_1, \ldots, X_t)$

DLOG(k): // $k \in [n], |Q| < t$
5 $Q \leftarrow Q \cup \{k\}$
6 $y_k \leftarrow \left(x_0 + \sum_{i \in [t]} k^i \cdot x_i\right) \bmod p$
7 Return y_k

FINALIZE(R, z):
8 Return $\left(f(R) \neq 0 \wedge g^z = R \cdot X_0^{f(R)}\right)$

Fig. 4. Games defining the CDL and VCDL problems.

RELATION TO CDL. While it is immediate that VCDL implies CDL, the converse direction also holds for a *restricted* variant of VCDL in which the queries are

fixed *non-adaptively*. Recall that for a query k, the oracle returns evaluations of the polynomial

$$y_k = x_0 + \sum_{i=1}^{t} k^i x_i.$$

For any $m \leq t$ distinct nonzero queries $k_1, \ldots, k_m$, the linear mapping from $(x_1, \ldots, x_t)$ to $(y_{k_1}, \ldots, y_{k_m})$ is defined by a full-rank Vandermonde matrix. Since $x_1, \ldots, x_t$ are uniform and independent in $\mathbb{Z}_p$, the resulting vector $(y_{k_1}, \ldots, y_{k_m})$ is uniformly distributed in $\mathbb{Z}_p^m$, completely masking x_0. Therefore, a CDL reduction can simulate the VCDL oracle by embedding its CDL challenge as $X_0 = g^{x_0}$, sampling the values y_{k_j} uniformly at random, and computing the other VCDL challenge values $X_1, \ldots, X_t$ using y_{k_j}. Since these oracle responses reveal no information about x_0, the adversary's view is identical to that in the restricted VCDL game, and any circular solution directly yields a valid CDL solution. Ultimately, VCDL's separation from CDL crucially hinges on the ability of the adversary to make *adaptive* queries.

4 Full Adaptive Security of Sparkle

We recall the Sparkle threshold signature scheme and then give a proof of its full adaptive security under VCDL in the random oracle model (ROM).

4.1 The Sparkle Scheme

Let $\mathbb{G}$ be a cyclic group of prime order p, generated by g. Let $H_{cm} \colon \mathbb{N} \times \mathbb{G} \to \mathbb{Z}_p$ and $H_{sig} \colon \mathbb{G} \times \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_p$ be hash functions. The Sparkle three-round threshold signature scheme [CKM23a]

$$\mathsf{Sparkle}[\mathbb{G}, H_{cm}, H_{sig}] = \big(\mathsf{KeyGen}, (\mathsf{Sign}_i)_{i\in\{1,2,3\}}, \mathsf{Combine}, \mathsf{Verify}\big)$$

with message space $\{0,1\}^*$ is given in Figure 5. In the ROM, we denote the scheme by $\mathsf{Sparkle}[\mathbb{G}]$.

4.2 Full Adaptive Security Under VCDL

We prove the following main result.

Theorem 1 (Full Adaptive Security of Sparkle). *Let $\mathbb{G}$ be a group of prime order p. Let* A *be an adversary against full adaptive security of* $\mathsf{Sparkle}[\mathbb{G}]$ *in the ROM for parameters $n, t \in \mathbb{N}$ where $n > t$ and assume* A *makes at most q queries to its oracles. Let $f \colon \mathbb{G} \to \mathbb{Z}_p$ be arbitrary and efficient. Then there exists an adversary* B *against VCDL for parameters f, n, t running in roughly the same time as* A *plus simulation overhead proportional to $q \cdot T_f$, where T_f is the time to compute f, such that*

$$\mathbf{Adv}^{\text{adp-ts-uf}}_{\mathsf{Sparkle}[\mathbb{G}],n,t}(\mathrm{A}) \leq \mathbf{Adv}^{\text{vcdl}}_{\mathbb{G},g,f,n,t}(\mathrm{B}) + \frac{4q^2 - 2q + |f^{-1}(0)|}{p}\,.$$

$\underline{\mathsf{KeyGen}(n, t+1)}$:

$x \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ vk \leftarrow g^x$
$\{(i, y_i)\}_{i\in[n]} \leftarrow\!\!\$\ \mathsf{Share}(n, t+1, x)$ // $(t+1)$-out-of-n Shamir secret sharing
$(vk_i)_{i\in[n]} \leftarrow (g^{y_i})_{i\in[n]}\ ;\ (sk_i)_{i\in[n]} \leftarrow (y_i, vk, (vk_j)_{j\in[n]})_{i\in[n]}$
Return $(vk, (vk_i)_{i\in[n]}, (sk_i)_{i\in[n]})$

$\underline{\mathsf{Sign}_1(k, sk_k)}$:

$r_k \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ R_k \leftarrow g^{r_k}$
$cm_k \leftarrow H_{cm}(k, R_k)$
$st_{k,1} \leftarrow (sk_k, r_k, R_k, cm_k)$
Return $(cm_k, st_{k,1})$

$\underline{\mathsf{Sign}_2(k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}, st_{k,1})}$:

$(sk_k, r_k, R_k, cm'_k) \leftarrow st_{k,1}$
If $\mathcal{S} \not\subseteq [n] \vee |\mathcal{S}| \leq t \vee k \notin \mathcal{S} \vee cm_k \neq cm'_k$: Return $\perp$
$st_{k,2} \leftarrow (sk_k, r_k, R_k, cm_k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$
Return $(R_k, st_{k,2})$

$\underline{\mathsf{Sign}_3(k, (R_i)_{i\in\mathcal{S}}, st_{k,2})}$:

$(sk_k, r_k, R'_k, cm_k, \mathcal{S}', m, (cm_i)_{i\in\mathcal{S}'}) \leftarrow st_{k,2}$
If $(R_k, \mathcal{S}) \neq (R'_k, \mathcal{S}')$ or $\exists i \in \mathcal{S} \setminus \{k\}$ s.t. $cm_i \neq H_{cm}(i, R_i)$: Return $\perp$
$R \leftarrow \prod_{i\in\mathcal{S}} R_i\ ;\ c \leftarrow H_{sig}(vk, m, R)$
$z_k \leftarrow (r_k + c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$ // $\lambda_k^{\mathcal{S}}$ is the k-th Lagrange coefficient for $\mathcal{S}$
Return z_k

$\underline{\mathsf{Combine}(\mathcal{S}, m, (R_i)_{i\in\mathcal{S}}, (z_i)_{i\in\mathcal{S}})}$:

$R \leftarrow \prod_{i\in\mathcal{S}} R_i\ ;\ z \leftarrow \left(\sum_{i\in\mathcal{S}} z_i\right) \bmod p$
Return (R, z) // Same form as basic Schnorr

$\underline{\mathsf{Verify}(vk, m, (R, z))}$:

$c \leftarrow H_{sig}(vk, m, R)$
Return $(g^z = R \cdot vk^c)$

Fig. 5. The Sparkle three-round threshold signature scheme.

Proof intuition. The most important part of the reduction is how to simulate signing on behalf of party k without knowing the corresponding secret key share y_k; in particular, how partial signatures $z_k = (r_k + c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$ returned by Sign_3 can be simulated, where r_k is the discrete logarithm of nonce R_k revealed by Sign_2. The basic idea is similar to plain Schnorr signatures [PS96, CKM23a]: first sample $z_k \leftarrow\!\!\$\ \mathbb{Z}_p$, and then set the nonce revealed in a Sign_2 query to

$$R_k \leftarrow g^{z_k} \cdot vk_k^{-c\cdot\lambda_k^{\mathcal{S}}},$$

such that the term involving y_k cancels out.

Crucially, this simulation technique requires to fix the value of c during a Sign_2 query, before learning the combined nonce $R = \prod R_i$ and being able to program $H_{sig}(vk, m, R) \leftarrow c$. This is where our simulation strategy diverges from the original security proofs given by Crites et al. [CKM23a]. Building on ideas by Bacho *et al.* in their proof of full adaptive security of Twinkle [BLT+24], we

avoid the gap in the original Sparkle proof by delaying the programming of H_{sig} on c as much as possible.

To achieve this, we define an equivalence relation of signing sessions—similar, in spirit, to [BLT+24]: informally, two signing sessions $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ and $(\mathcal{S}', m, (cm'_i)_{i\in\mathcal{S}'})$, each fixed as the input to a Sign_2 query, should fall in the same equivalence class if and only if their resulting combined nonces turn out to be equal, once all the nonces $(R_i)_{i\in\mathcal{S}}$ and $(R'_i)_{i\in\mathcal{S}'}$ are revealed. Consequently, they should use the same challenge c when programming $H_{sig}(vk, m, \prod_{i\in\mathcal{S}} R_i)$ for the correct simulation of partial signatures.

We show that our equivalence relation is preserved over time, except for negligible probability that two sessions that are initially not equivalent, later end up having the same combined nonce. Moreover, the reduction proceeds in a sequence of hybrid games that account for the following bad events, each occurring with negligible probability:

1. Coll_{cm}: If any commitments returned by H_{cm} collide, we cannot uniquely extract a preimage (i, R_i) from a commitment cm_i provided by the adversary.
2. $\mathsf{RepeatSign}_1$: If the randomness used by Sign_1 repeats, we cannot resample the nonce in Sign_2, which is needed for simulating z_k.
3. $\mathsf{FindPreimage}_{cm}$: Similarly, if the adversary manages to recover the preimage (i, R) of a nonce commitment cm returned by Sign_1, we abort.
4. $\mathsf{InvalidSign}_2$: If the adversary queries Sign_2 with some commitment cm_j that is not yet fixed in the function table of H_{cm}, we also abort.
5. $\mathsf{CantProgH}_{cm}$: When we program H_{cm} in Sign_2, we need to abort if the sampled value $R \leftarrow\!\!\$\ \mathbb{G}$ is inconsistent with the current function table of H_{cm}.
6. $\mathsf{CantProgH}_{sig}$: The programming of H_{sig} leads to a similar abort.

This leads to a final game that can be simulated by a VCDL adversary extending techniques similar to the proof of static security of Sparkle+ under CDL in [CFOS25]. In particular, a corruption query for Sparkle mimics the Vandermonde structure of the VCDL oracle queries, and a Sparkle forgery can be converted into a VCDL solution. Altogether, this establishes a *tight* proof of *full adaptive security* of Sparkle in the ROM under the hardness of VCDL.

Corollary 1 (Static Security of Sparkle). *Let $\mathbb{G}$ be a group of prime order p. Let* A *be an adversary against static security of* $\mathsf{Sparkle}[\mathbb{G}]$ *in the ROM for parameters $n, t \in \mathbb{N}$ where $n > t$ and assume* A *makes at most q queries to its oracles. Let $f\colon \mathbb{G} \to \mathbb{Z}_p$ be arbitrary and efficient. Then there exists an adversary* B *against CDL for f running in roughly the same time as* A *plus simulation overhead proportional to $q \cdot T_f$, where T_f is the time to compute f, such that*

$$\mathbf{Adv}^{\text{ts-uf}}_{\mathsf{Sparkle}[\mathbb{G}]}(\mathrm{A}) \leq \mathbf{Adv}^{\text{cdl}}_{\mathbb{G},g,f}(\mathrm{B}) + \frac{4q^2 - 2q + |f^{-1}(0)|}{p} \ .$$

In the static security definition (TS-UF) of a threshold signature scheme, the adversary must declare its corruption queries in advance before receiving any keys. As a corollary, the static security of Sparkle follows directly by combining the simulation strategy of Theorem 1 with the final reduction used in the static

security proof of Sparkle+ in [CFOS25]. In this reduction, a CDL adversary simulates the final game by embedding its challenge value X as the Schnorr verification key.

Proof (of Theorem 1). Let $\mathbf{G}_0 := \mathbf{G}^{\text{adp-ts-uf}}_{\mathsf{Sparkle}[\mathbb{G}],n,t}$ be the full adaptive security game for Sparkle in the ROM, given in Figure 6. Let A be the adversary from Theorem 1 interacting with $\mathbf{G}_0$ and making at most q oracle queries. By a *successful* query to $\textsc{SignO}_1$, $\textsc{SignO}_2$, or $\textsc{SignO}_3$, we mean a query that passes all the checks, and thus does not result in the output $\perp$. Without loss of generality, we assume that the adversary A only makes successful queries (as it can efficiently check whether a query would return $\perp$ before making it). Moreover, we assume that all queries to the two random oracles $\textsc{HashO}_{cm}$ and $\textsc{HashO}_{sig}$ are well-formed, i.e., inputs to $\textsc{HashO}_{cm}$ are of the form $(i, R) \in [n] \times \mathbb{G}$ and inputs to $\textsc{HashO}_{sig}$ are of the form $(vk, m, R) \in \mathbb{G} \times \{0,1\}^* \times \mathbb{G}$, where vk is the verification key given to A.

$\underline{\mathbf{G}_0 \to \mathbf{G}_1.}$ In game $\mathbf{G}_1$, given in Figure 6, we introduce an abort in line 26 of $\textsc{HashO}_{cm}$ which is triggered if any of the cm values stored in Q_{cm} collide (Coll_{cm}). Moreover, we add an abort in line 7 of $\textsc{SignO}_1$ ($\mathsf{RepeatSign}_1$). This change ensures that all the cm_k values returned by $\textsc{SignO}_1$ are distinct by aborting if the entry $(k, R_k, \cdot)$ already exists in Q_{cm}, i.e., when the r_k value sampled in line 6 repeats, or when A previously queried $\textsc{HashO}_{cm}$ on (k, R_k).

$\underline{\mathsf{Coll}_{cm}.}$ Since at most q values $cm_1, \ldots, cm_q$ get sampled from $\mathbb{Z}_p$, by the birthday bound, the probability that there is a collision is at most

$$\Pr[\,\mathsf{Coll}_{cm}\,] \leq \frac{q^2 - q}{2p}.$$

$\underline{\mathsf{RepeatSign}_1.}$ For the i-th query, the probability that $(k, R_k, \cdot) \in \mathsf{Q}_{cm}$ in line 7 of $\textsc{SignO}_1$ is at most $(i-1)/p$ since R_k is a uniformly random group element and there are at most $i-1$ possible entries $(k, \cdot, \cdot)$ defined in Q_{cm} it could hit. By a union bound over at most q queries, we get:

$$\Pr[\,\mathsf{RepeatSign}_1\,] \leq \frac{q^2 - q}{2p}.$$

Since $\mathbf{G}_0$ and $\mathbf{G}_1$ are equivalent if neither of these two aborts occurs, we have:

$$\Pr[\,\mathbf{G}_0(\mathrm{A}) \Rightarrow 1\,] \leq \Pr[\,\mathbf{G}_1(\mathrm{A}) \Rightarrow 1\,] + \frac{q^2 - q}{p}. \tag{1}$$

$\underline{\mathbf{G}_1 \to \mathbf{G}_2.}$ In $\mathbf{G}_2$, given in Figure 6, we introduce an abort in line 23 of $\textsc{HashO}_{cm}$ ($\mathsf{FindPreimage}_{cm}$). This is triggered if the adversary manages to query $\textsc{HashO}_{cm}$ on (k, R_k) where $(k, R_k, cm_k) \in \mathsf{Q}_{cm}$ for some cm_k output by $\textsc{SignO}_1(k)$, *before* obtaining R_k via a query to $\textsc{SignO}_2$ or $\textsc{CorruptO}$. The other changes are for bookkeeping, using the list $\mathsf{Q}'_{cm} \subseteq \mathsf{Q}_{cm}$ to keep track of the preimages (k, R_k) of commitments cm_k returned by $\textsc{SignO}_1$ which are unknown to A.

$\underline{\mathsf{FindPreimage}_{cm}.}$ We claim that for a single query (to either $\textsc{HashO}_{cm}$ directly, or to $\textsc{SignO}_3$ which calls $\textsc{HashO}_{cm}$ in line 16), the probability of $\mathsf{FindPreimage}_{cm}$

Game $\mathbf{G}_0(\mathbb{G}, g)$ / $\boxed{\mathbf{G}_1(\mathbb{G}, g)}$ / $\boxed{\boxed{\mathbf{G}_2(\mathbb{G}, g)}}$

INITIALIZE:
1 $\mathsf{Q}_{st}, \mathsf{Q}_m, \mathsf{Q}_{cm}, \boxed{\mathsf{Q}'_{cm},} \mathsf{Q}_{sig} \leftarrow \emptyset$; $eid \leftarrow 0$
2 $\mathsf{HS} \leftarrow [n]$; $\mathsf{CS} \leftarrow \emptyset$; $x \leftarrow\!\!\$\ \mathbb{Z}_p$; $vk \leftarrow g^x$
3 $\{(i, y_i)\}_{i\in[n]} \leftarrow\!\!\$\ \mathsf{Share}(n, t+1, x)$ // Shamir secret sharing
4 $(vk_i)_{i\in[n]} \leftarrow (g^{y_i})_{i\in[n]}$; $(sk_i)_{i\in[n]} \leftarrow (y_i, vk, (vk_j)_{j\in[n]})_{i\in[n]}$
5 Return $(vk, (vk_i)_{i\in[n]})$

SIGNO$_1(k)$: // $k \in \mathsf{HS}$
6 $eid \leftarrow eid + 1$; $r_k \leftarrow\!\!\$\ \mathbb{Z}_p$; $R_k \leftarrow g^{r_k}$
7 $\boxed{\text{If } (k, R_k, \cdot) \in \mathsf{Q}_{cm}\text{: Abort} \quad // \ \mathsf{RepeatSign}_1}$
8 $cm_k \leftarrow \text{HASHO}_{cm}(k, R_k)$
9 $\boxed{\boxed{\mathsf{Q}'_{cm} \leftarrow \mathsf{Q}'_{cm} \cup \{(k, R_k, cm_k)\}\ ;}}\ \mathsf{Q}_{st}[k, eid, 1] \leftarrow (r_k, R_k, cm_k)$
10 Return (eid, cm_k)

SIGNO$_2(k, eid, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 1] \neq \bot, \mathsf{Q}_{st}[k, eid, 2] = \bot$
11 $\mathsf{Q}_m \leftarrow \mathsf{Q}_m \cup \{m\}$; $(r_k, R_k, cm'_k) \leftarrow \mathsf{Q}_{st}[k, eid, 1]$
12 If $\mathcal{S} \not\subseteq [n] \vee |\mathcal{S}| \leq t \vee k \notin \mathcal{S} \vee cm_k \neq cm'_k$: Return $\bot$
13 $\boxed{\boxed{\mathsf{Q}'_{cm} \leftarrow \mathsf{Q}'_{cm} \cup \{(k, R_k, cm_k)\}\ ;}}\ \mathsf{Q}_{st}[k, eid, 2] \leftarrow (r_k, R_k, cm_k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$
14 Return R_k

SIGNO$_3(k, eid, (R_i)_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 2] \neq \bot, \mathsf{Q}_{st}[k, eid, 3] = \bot$
15 $(r_k, R'_k, cm_k, \mathcal{S}', m, (cm_i)_{i\in\mathcal{S}'}) \leftarrow \mathsf{Q}_{st}[k, eid, 2]$
16 If $(R_k, \mathcal{S}) \neq (R'_k, \mathcal{S}')$ or $\exists i \in \mathcal{S} \setminus \{k\}$ s.t. $cm_i \neq \text{HASHO}_{cm}(i, R_i)$: Return $\bot$
17 $R \leftarrow \prod_{i\in\mathcal{S}} R_i$; $c \leftarrow \text{HASHO}_{sig}(vk, m, R)$; $z_k \leftarrow (r_k + c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$
18 $\mathsf{Q}_{st}[k, eid, 3] \leftarrow z_k$
19 Return z_k

CORRUPTO(k): // $k \in \mathsf{HS}, |\mathsf{CS}| < t$
20 $\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$; $\mathsf{HS} \leftarrow \mathsf{HS} \setminus \{k\}$
21 $\boxed{\boxed{\mathsf{Q}'_{cm} \leftarrow \{(i, \cdot, \cdot) \in \mathsf{Q}'_{cm} \mid i \neq k\}\ ;}}\ st_k \leftarrow \mathsf{Q}_{st}[k, \cdot, \cdot]$
22 Return (sk_k, st_k)

HASHO$_{cm}(i, R)$:
23 $\boxed{\boxed{\text{If } (i, R, \cdot) \in \mathsf{Q}'_{cm}\text{: Abort} \quad // \ \mathsf{FindPreimage}_{cm}}}$
24 If $(i, R, \cdot) \in \mathsf{Q}_{cm}$: Return $\mathsf{Q}_{cm}[i, R]$
25 $cm \leftarrow\!\!\$\ \mathbb{Z}_p$
26 $\boxed{\text{If } (\cdot, \cdot, cm) \in \mathsf{Q}_{cm}\text{: Abort} \quad // \ \mathsf{Coll}_{cm}}$
27 $\mathsf{Q}_{cm} \leftarrow \mathsf{Q}_{cm} \cup \{(i, R, cm)\}$
28 Return cm

HASHO$_{sig}(vk, m, R)$:
29 If $(vk, m, R, \cdot) \in \mathsf{Q}_{sig}$:
30 Return $\mathsf{Q}_{sig}[vk, m, R]$
31 $c \leftarrow\!\!\$\ \mathbb{Z}_p$
32 $\mathsf{Q}_{sig} \leftarrow \mathsf{Q}_{sig} \cup \{(vk, m, R, c)\}$
33 Return c

FINALIZE$(m, (R, z))$:
34 If $m \in \mathsf{Q}_m$: Return 0
35 $c \leftarrow \text{HASHO}_{sig}(vk, m, R)$
36 Return $(g^z = R \cdot vk^c)$

Fig. 6. Games $\mathbf{G}_0$, $\mathbf{G}_1$, and $\mathbf{G}_2$ for the proof of Theorem 1. Changes from $\mathbf{G}_0$ to $\mathbf{G}_1$ are boxed, while changes from $\mathbf{G}_1$ to $\mathbf{G}_2$ are double-boxed.

is at most q/p. For a HASHO_{cm} query on input (i^*, R^*), the probability that $(i^*, R^*, \cdot) \in \mathsf{Q}'_{cm}$ is at most $a/(p-b)$, where $a = |\mathsf{Q}'_{cm}|$ is the current number of entries (i, R, cm) in Q'_{cm} (with each R value being a uniformly random group element unknown to A), and b is the number of entries that were added to Q'_{cm} but then later removed by a query to either SIGNO_2 or CORRUPTO (i.e., A knows their R values and could exclude them when choosing R^*). The same bound also holds for a query to $\text{SIGNO}_3(k, eid, (R_i)_{i\in\mathcal{S}})$ considering the analogously defined quantities a_i, b_i for each party $i \in \mathcal{S} \setminus \{k\}$. The probability that $\mathsf{FindPreimage}_{cm}$ occurs for party i in line 16 is then bounded by $a_i/(p-b_i)$. Using $b_i \leq b$ and $\sum a_i \leq a$, we still obtain the same overall bound:

$$\sum_{i\in\mathcal{S}\setminus\{k\}} \frac{a_i}{p-b_i} \leq \frac{\sum_{i\in\mathcal{S}\setminus\{k\}} a_i}{p-b} \leq \frac{a}{p-b}.$$

Since $a + b \leq q \leq p$,[3] we can further rewrite the bound as

$$\frac{a}{p-b} \leq \frac{a+b}{p} \leq \frac{q}{p}.$$

By a union bound over at most q queries, we get:

$$\Pr[\,\mathbf{G}_1(\text{A}) \Rightarrow 1\,] \leq \Pr[\,\mathbf{G}_2(\text{A}) \Rightarrow 1\,] + \frac{q^2}{p}. \tag{2}$$

$\underline{\mathbf{G}_2 \to \mathbf{G}_3.}$ Next, we modify game $\mathbf{G}_2$ to obtain $\mathbf{G}_3$, given in Figure 7. We change SIGNO_2, adding the helper functions GETCHAL and EQUIV, and introduce an abort in line 24 of SIGNO_3 ($\mathsf{InvalidSign}_2$).

We say a query to SIGNO_2 on input $(k, eid, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ is *valid* if each commitment cm_i provided by the adversary satisfies $(i, R_i, cm_i) \in \mathsf{Q}_{cm}$ for some element R_i, i.e., $\text{HASHO}_{cm}(i, R_i) = cm_i$. Note that an invalid SIGNO_2 query must contain at least one commitment cm_{i^*} which violates this condition, and thus will result in the output $\perp$ in a subsequent SIGNO_3 query, except with negligible probability that A manages to add (i^*, R_{i^*}, cm_{i^*}) to Q_{cm} for some R_{i^*}. In this case, $\mathbf{G}_3$ aborts due to $\mathsf{InvalidSign}_2$ in line 24 of SIGNO_3.

Moreover, on a valid SIGNO_2 query in $\mathbf{G}_3$, the preimage R_k of the commitment cm_k from SIGNO_1 is resampled. Since the abort due to $\mathsf{FindPreimage}_{cm}$ in line 23 of HASHO_{cm} ensures that A never learned the previous value of R_k, this does not change anything from the view of the adversary. We will argue that the resulting output distribution of SIGNO_2 is the same as in $\mathbf{G}_2$, i.e., that $R_k \in \mathbb{G}$ is distributed independently and uniformly and satisfies $\text{HASHO}_{cm}(k, R_k) = cm_k$ in $\mathbf{G}_3$, except for the abort caused by $\mathsf{CantProgH}_{cm}$. In line 8, we set $R_k \leftarrow g^{z_k - c\cdot y_k \cdot \lambda_k^{\mathcal{S}}}$ for $z_k \leftarrow\!\!\$\ \mathbb{Z}_p$ and $c \leftarrow \text{GETCHAL}(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$, which is equivalent to choosing $R_k \leftarrow\!\!\$\ \mathbb{G}$ since z_k is independent of $c \cdot y_k \cdot \lambda_k^{\mathcal{S}}$. Moreover, line 10 programs $\text{HASHO}_{cm}(k, R_k) \leftarrow cm_k$.

[3] Note that if $q > p$, the bound holds trivially.

Game $\mathbf{G}_3(\mathbb{G}, g, n, t)$ / $\boxed{\mathbf{G}_4(\mathbb{G}, g, n, t)}$

INITIALIZE:

$\mathsf{Q}_{st}, \mathsf{Q}_m, \mathsf{Q}_{cm}, \mathsf{Q}'_{cm}, \mathsf{Q}_{sig}, \mathsf{Q}_{ses}, \mathsf{Q}_z \leftarrow \emptyset$; $eid \leftarrow 0$

... // the same as in $\mathbf{G}_2$

$\text{SIGNO}_2(k, eid, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 1] \neq \bot, \mathsf{Q}_{st}[k, eid, 2] = \bot$

... // the same as in $\mathbf{G}_2$

$\mathsf{Q}'_{cm} \leftarrow \mathsf{Q}'_{cm} \setminus \{(k, R_k, cm_k)\}$

If $\left(\bigwedge_{i\in\mathcal{S}}(i, \cdot, cm_i) \in \mathsf{Q}_{cm}\right)$: // valid signing session

  $c \leftarrow \text{GETCHAL}(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$

  $\mathsf{Q}_{cm} \leftarrow \mathsf{Q}_{cm} \setminus \{(k, R_k, cm_k)\}$

  $z_k \leftarrow\!\!\$\ \mathbb{Z}_p$; $r_k \leftarrow (z_k - c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$; $R_k \leftarrow g^{r_k}$

  If $(k, R_k, \cdot) \in \mathsf{Q}_{cm}$: Abort // $\mathsf{CantProgH}_{cm}$

  $\mathsf{Q}_{cm} \leftarrow \mathsf{Q}_{cm} \cup \{(k, R_k, cm_k)\}$

  $\mathsf{Q}_{st}[k, eid, 1] \leftarrow (r_k, R_k, cm_k)$; $\mathsf{Q}_z[k, eid] \leftarrow (z_k, c)$

$\mathsf{Q}_{st}[k, eid, 2] \leftarrow (r_k, R_k, cm_k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$

Return R_k

$\text{GETCHAL}(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$: // not accessible to the adversary

For $(\mathcal{S}', m, (cm'_i)_{i\in\mathcal{S}'}, c) \in \mathsf{Q}_{ses}$:

  If $\text{EQUIV}\big((\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}), (\mathcal{S}', m, (cm'_i)_{i\in\mathcal{S}'})\big) = 1$: Return c

$c \leftarrow\!\!\$\ \mathbb{Z}_p$; $\mathsf{Q}_{ses} \leftarrow \mathsf{Q}_{ses} \cup \{(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}, c)\}$

Return c

$\text{EQUIV}\big((\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}), (\mathcal{S}', m', (cm'_i)_{i\in\mathcal{S}'})\big)$: // not accessible to the adversary

$I \leftarrow \{i \in \mathcal{S} \mid (i, \cdot, cm_i) \in \mathsf{Q}'_{cm}\}$; $I' \leftarrow \{i \in \mathcal{S}' \mid (i, \cdot, cm'_i) \in \mathsf{Q}'_{cm}\}$

$R \leftarrow \prod_{i\in\mathcal{S}\setminus I} \mathsf{Q}_{cm}^{-1}[i, cm_i]$ // $\mathsf{Q}_{cm}^{-1}[i, cm]$ returns unique R s.t. $\mathsf{Q}_{cm}[i, R] = cm$

$R' \leftarrow \prod_{i\in\mathcal{S}'\setminus I'} \mathsf{Q}_{cm}^{-1}[i, cm'_i]$

Return $\left(m = m' \wedge I = I' \wedge (cm_i)_{i\in I} = (cm'_i)_{i\in I'} \wedge R = R'\right)$

$\text{SIGNO}_3(k, eid, (R_i)_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 2] \neq \bot, \mathsf{Q}_{st}[k, eid, 3] = \bot$

$(r_k, R'_k, cm_k, \mathcal{S}', m, (cm_i)_{i\in\mathcal{S}'}) \leftarrow \mathsf{Q}_{st}[k, eid, 2]$

If $(R_k, \mathcal{S}) \neq (R'_k, \mathcal{S}')$ or $\exists i \in \mathcal{S} \setminus \{k\}$ s.t. $cm_i \neq \text{HASHO}_{cm}(i, R_i)$: Return $\bot$

If $\mathsf{Q}_z[k, eid] = \bot$: Abort // $\mathsf{InvalidSign}_2$

$R \leftarrow \prod_{i\in\mathcal{S}} R_i$; $c \leftarrow \text{HASHO}_{sig}(vk, m, R)$; $z_k \leftarrow (r_k + c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$

$\boxed{(z_k, c) \leftarrow \mathsf{Q}_z[k, eid]\ ;}$ $\mathsf{Q}_{st}[k, eid, 3] \leftarrow z_k$

Return z_k

$\text{HASHO}_{sig}(vk, m, R)$:

For $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}, c) \in \mathsf{Q}_{ses}$:

  If $\left(\bigwedge_{i\in\mathcal{S}}(i, \cdot, cm_i) \notin \mathsf{Q}'_{cm}\right)$: // fully determined signing session

    If $\prod_{i\in\mathcal{S}} \mathsf{Q}_{cm}^{-1}[i, cm_i] = R$:

      If $(vk, m, R, c') \in \mathsf{Q}_{sig}$ with $c' \neq c$: Abort // $\mathsf{CantProgH}_{sig}$

      $\mathsf{Q}_{sig} \leftarrow \mathsf{Q}_{sig} \cup \{(vk, m, R, c)\}$

      $\mathsf{Q}_{ses} \leftarrow \mathsf{Q}_{ses} \setminus \{(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}, c)\}$

... // the same as in $\mathbf{G}_2$

Fig. 7. Games $\mathbf{G}_3$ and $\mathbf{G}_4$ for the proof of Theorem 1. Changes from $\mathbf{G}_2$ are highlighted in blue, while changes from $\mathbf{G}_3$ to $\mathbf{G}_4$ are additionally boxed (SIGNO_1, CORRUPTO, HASHO_{cm}, and FINALIZE remain the same).

When simulating the partial signature z_k of party k involved in a (valid) signing session $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$, we need the value of c used to compute $R_k \leftarrow g^{z_k - c\cdot y_k\cdot\lambda_k^{\mathcal{S}}}$ in line 8 of SIGNO_2 to correspond to the value of $\text{HASHO}_{sig}(vk, m, \prod_{i\in\mathcal{S}} R_i)$, where the $(R_i)_{i\in\mathcal{S}}$ are the *unique*[4] preimages of the commitments $(cm_i)_{i\in\mathcal{S}}$, i.e., $\text{HASHO}_{cm}(i, R_i) = cm_i$ for each $i \in \mathcal{S}$. To ensure this, we make use of an equivalence relation similar to the one used by Bacho et al. to prove adaptive security of their threshold signature scheme Twinkle [BLT+24]. We say two valid signing sessions $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ and $(\mathcal{S}', m', (cm'_i)_{i\in\mathcal{S}'})$ are equivalent if and only if the following holds:

(a) $m = m'$.
(b) Let $I := \{i \in \mathcal{S} \mid (i, \cdot, cm_i) \in \mathsf{Q}'_{cm}\}$ and $I' := \{i \in \mathcal{S}' \mid (i, \cdot, cm'_i) \in \mathsf{Q}'_{cm}\}$ be the indices of the commitments whose preimages are unknown to A, respectively. Then $I' = I$ and $(cm'_i)_{i\in I'} = (cm_i)_{i\in I}$.
(c) Let $(R_i)_{i\in\mathcal{S}\setminus I}$ and $(R'_i)_{i\in\mathcal{S}'\setminus I'}$ be the preimages of $(cm_i)_{i\in\mathcal{S}\setminus I}$ and $(cm'_i)_{i\in\mathcal{S}'\setminus I'}$, respectively. Then the partial combined nonces are equal:

$$\prod_{i\in\mathcal{S}\setminus I} R_i = \prod_{i\in\mathcal{S}'\setminus I'} R'_i. \tag{3}$$

This check is performed by EQUIV in GETCHAL to decide whether to sample a fresh challenge $c \leftarrow\!\!\$\ \mathbb{Z}_p$ or reuse an existing one when simulating the partial signature z_k in line 8 of SIGNO_2 for a given valid signing session. We use the notation $\mathsf{Q}_{cm}^{-1}[i, cm]$ to retrieve the unique preimage R such that $(i, R, cm) \in \mathsf{Q}_{cm}$.

Next, we make the following claim:

Claim 1. *The equivalence relation defined above is preserved over time. In particular, two valid signing sessions that are equivalent at any point in time will stay equivalent throughout the execution and two sessions that are not equivalent will not become equivalent later in time, except with probability at most* $1/p$.

Proof (of Claim 1). Note that the message in a signing session is fixed when the session is first introduced in a SIGNO_2 query, so we only need to consider sessions with the same message. Further, the index sets I and I' are defined with respect to the set Q'_{cm}, and so any change can only occur when some entry is added to or removed from Q'_{cm}.

Adding an entry to Q'_{cm} (via a SIGNO_1 query), does not affect the equivalence relation since, due to the abort in line 7, SIGNO_1 only adds "fresh" entries (k, R_k, cm_k) distinct from all Q'_{cm} entries tied to existing valid signing sessions.

On the other hand, when an entry is removed from Q'_{cm}, indicating that the corresponding R_k is revealed to the adversary, we consider the following two cases for two arbitrary (valid) signing sessions $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ and $(\mathcal{S}', m, (cm'_i)_{i\in\mathcal{S}'})$:

1. If the two signing sessions are equivalent, they stay equivalent for the remainder of the execution of the game. Suppose an entry is removed from Q'_{cm}. Since I and I' are equal before, this affects I and I' identically: either the index of that entry was in both sets or in neither. If it was in both sets, then the product in Eq. (3) is multiplied by the same value, and if it was in neither set, then nothing changes. Thus, the two sessions are still equivalent.

[4] Uniqueness is guaranteed by the abort event Coll_{cm} introduced in $\mathbf{G}_1$.

2. If the two signing sessions are not equivalent, they can only become equivalent later with negligible probability. Suppose an entry (i^*, R_{i^*}, cm_{i^*}) is removed from Q'_{cm} that makes the two signing sessions equivalent. Assuming without loss of generality that $|I| \geq |I'|$, the only way this can happen is when $I \setminus I' = \{i^*\}$ prior to this change, so that condition (b) can be satisfied. Moreover, for Eq. (3) in condition (c) to hold, we must have

$$R_{i^*} = \prod_{i \in \mathcal{S}' \setminus I'} R'_i \cdot \prod_{i \in \mathcal{S} \setminus I} R_i^{-1}.$$

Since the element R_{i^*} is chosen independently and uniformly at random (either in line 6 of SIGNO_1 or in line 8 of SIGNO_2), the probability is $1/p$. Importantly, for any two fixed signing sessions, such a transition scenario can occur at most once *throughout* the game (when the last honestly contributed nonce not shared across both sessions is revealed), at which point, the sessions either become equivalent with probability $1/p$ or remain non-equivalent for the rest of the execution.

This finishes the proof of Claim 1. □

$\underline{\mathsf{CantProgH}_{cm}}$. For the i-th query, the probability that $(k, R_k, \cdot) \in \mathsf{Q}_{cm}$ in line 9 of SIGNO_2 is at most $(i-1)/p$ since $R_k = g^{z_k - c \cdot y_k \cdot \lambda_k^{\mathcal{S}}}$ for $z_k \leftarrow\!\!\$\ \mathbb{Z}_p$ is a uniformly random group element and there are at most $i-1$ possible entries $(k, \cdot, \cdot)$ defined in Q_{cm} it could hit. By a union bound over at most q queries, we get:

$$\Pr[\,\mathsf{CantProgH}_{cm}\,] \leq \frac{q^2 - q}{2p}. \tag{4}$$

$\underline{\mathsf{InvalidSign}_2}$. Note that the abort in line 24 of SIGNO_3 can only occur if the adversary previously successfully queried SIGNO_1 and SIGNO_2 for the same party k. Moreover, $\mathsf{Q}_z[k, eid] = \bot$ means that the query to SIGNO_2 was not valid (i.e., the checks in line 5 did not pass). In particular, this means there is a party $i^* \in \mathcal{S} \setminus \{k\}$ for which $(i^*, \cdot, cm_{i^*}) \notin \mathsf{Q}_{cm}$ at the time of the query to SIGNO_2. From this point on, the only way for the game to reach the abort in line 24 of SIGNO_3 is if the check $cm_{i^*} = \text{HASHO}_{cm}(i^*, R_i^*)$ in line 23 passes. This means HASHO_{cm} must have added (i^*, R_{i^*}, cm_{i^*}) to Q_{cm} at some point after A made the SIGNO_2 query fixing cm_{i^*}.

So to bound $\mathsf{InvalidSign}_2$, we can instead bound the probability that HASHO_{cm} adds $(j, \cdot, cm)$ to Q_{cm} where cm was previously used as the "alleged" commitment for party j in a SIGNO_2 query. For the i-th query (either direct or implicit via a SIGNO_1 query) to HASHO_{cm} on input (j, R), the probability that $cm \leftarrow\!\!\$\ \mathbb{Z}_p$ hits one of at most $i-1$ possible commitment values used in previous SIGNO_2 queries for party j is at most $(i-1)/p$. Applying a union bound over at most q queries gives:

$$\Pr[\,\mathsf{InvalidSign}_2\,] \leq \frac{q^2 - q}{2p}. \tag{5}$$

Combining Equations (4) and (5), we obtain:

$$\Pr[\,\mathbf{G}_2(\mathrm{A}) \Rightarrow 1\,] \leq \Pr[\,\mathbf{G}_3(\mathrm{A}) \Rightarrow 1\,] + \frac{q^2 - q}{p}. \tag{6}$$

$\underline{\mathbf{G}_3 \to \mathbf{G}_4.}$ We also provide game $\mathbf{G}_4$ in Figure 7 and argue that—from the view of the adversary—it is equivalent to $\mathbf{G}_3$, except when the newly introduced abort due to $\mathsf{CantProgH}_{sig}$ in line 31 of $\textsc{HashO}_{sig}$ occurs:

1. $\textsc{Initialize}$, $\textsc{SignO}_1$, $\textsc{SignO}_2$, $\textsc{CorruptO}$, $\textsc{HashO}_{cm}$ and $\textsc{Finalize}$ are all the same.
2. $\textsc{HashO}_{sig}$ is the same, except for the abort caused by $\mathsf{CantProgH}_{sig}$. The programming in line 32 is consistent with a random oracle since the used value of c stored in Q_{ses} was chosen uniformly at random in line 16.
3. $\textsc{SignO}_3$ has the same output distribution, except for the abort caused by $\mathsf{CantProgH}_{sig}$. Observe that when $\mathsf{InvalidSign}_2$ does not occur, z_k is sampled in line 8 of $\textsc{SignO}_2$ before being added to Q_z. Let $(k, eid, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ be the input to the corresponding $\textsc{SignO}_2$ query. Since the checks in line 23 of $\textsc{SignO}_3$ pass, let $(R_i)_{i\in\mathcal{S}}$ be the preimages of the commitments $(cm_i)_{i\in\mathcal{S}}$ and set $R := \prod_{i\in\mathcal{S}} R_i$. In $\mathbf{G}_3$, r_k is sampled in $\textsc{SignO}_1$ and z_k is then computed in $\textsc{SignO}_3$ as

   $$r_k \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ c \leftarrow \textsc{HashO}_{sig}(vk, m, R)\ ;\ z_k \leftarrow (r_k + c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$$

   On the other hand, in $\mathbf{G}_4$, z_k and r_k are both chosen in $\textsc{SignO}_2$ as

   $$z_k \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ c \leftarrow \textsc{GetChal}(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})\ ;\ r_k \leftarrow (z_k - c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p.$$

   Clearly, these two distributions are equivalent if the value of c used to compute z_k in $\mathbf{G}_4$ satisfies $c = \textsc{HashO}_{sig}(vk, m, R)$. This is ensured by the code added to $\textsc{HashO}_{sig}$ in line 28 onward, which is called by line 25 at the latest, before $\textsc{SignO}_3$ returns z_k. Indeed, since the checks in line 23 of $\textsc{SignO}_3$ pass and thus $\mathsf{FindPreimage}_{cm}$ in line 23 of $\textsc{HashO}_{cm}$ does not occur, we have $(i, R_i, cm_i) \notin \mathsf{Q}'_{cm}$ for all $i \in \mathcal{S}$. Consequently, if $\mathsf{CantProgH}_{sig}$ also does not occur, the loop in line 28 of $\textsc{HashO}_{sig}$ programs $\textsc{HashO}_{sig}(vk, m, R) \leftarrow c$.

$\underline{\mathsf{CantProgH}_{sig}.}$ We say a (valid) signing session $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}, \cdot) \in \mathsf{Q}_{ses}$ is *fully determined* if $(i, \cdot, cm_i) \notin \mathsf{Q}'_{cm}$ for all $i \in \mathcal{S}$, which implies that the adversary received all the preimages $(R_i)_{i\in\mathcal{S}}$ (either by querying $\textsc{SignO}_2$ or $\textsc{CorruptO}$). Moreover, for each fully determined signing session $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$, define its combined nonce R as the product $\prod_{i\in\mathcal{S}} R_i$. Note that the loop in line 28 of $\textsc{HashO}_{sig}$ only considers fully determined signing sessions.

Recall the equivalence relation from Claim 1. We claim that any two distinct, fully determined signing sessions $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}) \neq (\mathcal{S}', m, (cm'_i)_{i\in\mathcal{S}'})$ whose resulting combined nonces collide, i.e., $R = R'$, will belong to the same equivalence class at the end of the execution. This is because fully determined signing sessions with the same m trivially satisfy conditions (a) and (b), while condition (c) is equivalent to $R = R'$.

We first bound the probability that the equivalence relation is not preserved throughout the game, i.e., two initially non-equivalent signing sessions $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ and $(\mathcal{S}', m, (cm'_i)_{i\in\mathcal{S}'})$ later become equivalent. We let Coll_R denote this event. By Claim 1, for any fixed pair of sessions, this happens with probability at most $1/p$. By a union bound over all distinct pairs among at most q signing sessions, we get

$$\Pr[\,\mathsf{Coll}_R\,] \leq \frac{q^2-q}{2p}.$$

Next, conditioned on no such collision occurring, we will bound the probability of $\mathsf{CantProgH}_{sig}$. Note that in this case, each call to $\textsc{HashO}_{sig}$ runs the check "$(vk, m, R, c') \in \mathsf{Q}_{sig}$ with $c' \neq c$" in line 31 for at most one equivalence class of fully determined signing sessions. For the i-th such query on input (vk, m, R), let $(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$ be the corresponding fully determined signing session with $R = \prod_{i\in\mathcal{S}} R_i$. Consider the point when A first learned the value of R by obtaining the last preimage R_{i^*} of the commitments $(cm_i)_{i\in\mathcal{S}}$ for some honest party $i^* \in \mathcal{S}$. At this point, there are at most $i-1$ possible entries $(vk, m, \cdot, \cdot)$ defined in Q_{sig}, and since R is a uniformly random group element perfectly blinded by R_{i^*}, the probability that $(vk, m, R, \cdot) \in \mathsf{Q}_{sig}$ is at most $(i-1)/p$. By a union bound over at most $q+1$ total queries (including the call to $\textsc{HashO}_{sig}$ in Finalize), we get:

$$\Pr[\,\mathsf{CantProgH}_{sig} \mid \neg\mathsf{Coll}_R\,] \leq \frac{q^2+q}{2p}.$$

In total, this means

$$\Pr[\,\mathsf{CantProgH}_{sig}\,] \leq \Pr[\,\mathsf{Coll}_R\,] + \Pr[\,\mathsf{CantProgH}_{sig} \mid \neg\mathsf{Coll}_R\,] \leq \frac{q^2}{p},$$

and thus

$$\Pr[\,\mathbf{G}_3(\mathrm{A}) \Rightarrow 1\,] \leq \Pr[\,\mathbf{G}_4(\mathrm{A}) \Rightarrow 1\,] + \frac{q^2}{p}. \tag{7}$$

$\underline{\mathbf{G}_4 \to \mathbf{G}_5.}$ We claim that $\mathbf{G}_5$ in Figure 8 is equivalent to $\mathbf{G}_4$, i.e.,

$$\Pr[\,\mathbf{G}_4(\mathrm{A}) \Rightarrow 1\,] = \Pr[\,\mathbf{G}_5(\mathrm{A}) \Rightarrow 1\,]. \tag{8}$$

First of all, the changes in Initialize perform a simulated Shamir secret sharing of $\log_g(vk)$. Clearly, the resulting distribution of $(vk, (vk_i)_{i\in[n]})$ is equivalent to $\mathbf{G}_4$. As a side effect, the secret key shares $(y_i)_{i\in[n]}$ are now all set to $\perp$. To perform the simulation of z_k in line 11 of $\textsc{SignO}_2$, instead of sampling $z_k \leftarrow\!\!\$\ \mathbb{Z}_p$ and then computing $R_k \leftarrow g^{r_k}$ for $r_k \leftarrow (z_k - c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$, we simply set

$$R_k \leftarrow g^{z_k} \cdot vk_k^{-c\cdot\lambda_k^{\mathcal{S}}}.$$

The only other modifications happen in CorruptO, where we now need to restore the secret key share sk_k by setting $y_k \leftarrow \log_g(vk_k)$ (note that this step will be performed efficiently by the final VCDL reduction B in Figure 9 by calling

Game $\mathbf{G}_5(\mathbb{G}, g, n, t)$ / $\boxed{\mathbf{G}_6(\mathbb{G}, g, f, n, t)}$

INITIALIZE:

$\mathsf{Q}_{st}, \mathsf{Q}_m, \mathsf{Q}_{cm}, \mathsf{Q}'_{cm}, \mathsf{Q}_{sig}, \mathsf{Q}_{ses}, \mathsf{Q}_z \boxed{, \mathsf{Q}_{ab}} \leftarrow \emptyset$; $eid \leftarrow 0$
$\mathsf{HS} \leftarrow [n]$; $\mathsf{CS} \leftarrow \emptyset$
$X_0, \ldots, X_t \leftarrow\!\!\$\ \mathbb{G}$; $vk \leftarrow X_0$
For $i \in [n]$: $y_i \leftarrow \bot$; $vk_i \leftarrow \prod_{j\in[0,t]} (X_j)^{i^j}$
$(sk_i)_{i\in[n]} \leftarrow (y_i, vk, (vk_j)_{j\in[n]})_{i\in[n]}$
Return $(vk, (vk_i)_{i\in[n]})$

SIGNO$_2(k, eid, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$: // $k \in \mathsf{HS}, \mathsf{Q}_{st}[k, eid, 1] \neq \bot, \mathsf{Q}_{st}[k, eid, 2] = \bot$

... // the same as in $\mathbf{G}_4$
If $(\bigwedge_{i\in\mathcal{S}}(i, \cdot, cm_i) \in \mathsf{Q}_{cm})$: // valid signing session
  $c \leftarrow$ GETCHAL$(\mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$
  $\mathsf{Q}_{cm} \leftarrow \mathsf{Q}_{cm} \setminus \{(k, R_k, cm_k)\}$
  $z_k \leftarrow\!\!\$\ \mathbb{Z}_p$; $r_k \leftarrow \bot$; $R_k \leftarrow g^{z_k} \cdot vk_k^{-c\cdot\lambda_k^{\mathcal{S}}}$
  If $(k, R_k, \cdot) \in \mathsf{Q}_{cm}$: Abort // $\mathsf{CantProgH}_{cm}$
  $\mathsf{Q}_{cm} \leftarrow \mathsf{Q}_{cm} \cup \{(k, R_k, cm_k)\}$
  $\mathsf{Q}_{st}[k, eid, 1] \leftarrow (r_k, R_k, cm_k)$; $\mathsf{Q}_z[k, eid] \leftarrow (z_k, c)$
$\mathsf{Q}_{st}[k, eid, 2] \leftarrow (r_k, R_k, cm_k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$
Return R_k

CORRUPTO(k): // $k \in \mathsf{HS}, |\mathsf{CS}| < t$

$y_k \leftarrow \log_g(vk_k)$; $sk_k \leftarrow (y_k, vk, (vk_i)_{i\in[n]})$
$\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$; $\mathsf{HS} \leftarrow \mathsf{HS} \setminus \{k\}$
$\mathsf{Q}'_{cm} \leftarrow \{(i, \cdot, \cdot) \in \mathsf{Q}'_{cm} \mid i \neq k\}$
For $(k, eid_k, z_k, c) \in \mathsf{Q}_z$:
  $(\bot, R_k, cm_k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}}) \leftarrow \mathsf{Q}_{st}[k, eid_k, 2]$
  $r_k \leftarrow (z_k - c \cdot y_k \cdot \lambda_k^{\mathcal{S}}) \bmod p$ // update $r_k \leftarrow \log_g(R_k)$
  $\mathsf{Q}_{st}[k, eid_k, 2] \leftarrow (r_k, R_k, cm_k, \mathcal{S}, m, (cm_i)_{i\in\mathcal{S}})$
  $\mathsf{Q}_{st}[k, eid_k, 1] \leftarrow (r_k, R_k, cm_k)$
$st_k \leftarrow \mathsf{Q}_{st}[k, \cdot, \cdot]$
Return (sk_k, st_k)

HASHO$_{sig}(vk, m, R)$:

... // the same as in $\mathbf{G}_4$
$c \leftarrow\!\!\$\ \mathbb{Z}_p$
$\boxed{a, b \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ R' \leftarrow R \cdot vk^a \cdot g^b}$
$\boxed{c \leftarrow (f(R') + a) \bmod p}$
$\boxed{\mathsf{Q}_{ab} \leftarrow \mathsf{Q}_{ab} \cup \{(vk, m, R, a, b)\}}$
$\mathsf{Q}_{sig} \leftarrow \mathsf{Q}_{sig} \cup \{(vk, m, R, c)\}$
Return c

FINALIZE$(m, (R, z))$:

If $m \in \mathsf{Q}_m$: Return 0
$c \leftarrow$ HASHO$_{sig}(vk, m, R)$
$\boxed{(a, b) \leftarrow \mathsf{Q}_{ab}[vk, m, R]}$
$\boxed{R' \leftarrow R \cdot vk^a \cdot g^b}$
$\boxed{\text{If } f(R') = 0\text{: Abort} \quad // \ \mathsf{Zero}_f}$
Return $(g^z = R \cdot vk^c)$

Fig. 8. Games $\mathbf{G}_5$ and $\mathbf{G}_6$ for the proof of Theorem 1. Changes from $\mathbf{G}_4$ are highlighted in blue, while changes from $\mathbf{G}_5$ to $\mathbf{G}_6$ are additionally boxed (SIGNO$_1$, SIGNO$_3$, and HASHO$_{cm}$ remain the same).

its DLOG oracle). Moreover, for each valid SIGNO_2 query, we also need to update the value r_k stored in Q_{st} (see line 20 onward).

$\underline{\mathbf{G}_5 \to \mathbf{G}_6.}$ In $\mathbf{G}_6$, given in Figure 8, we change how HASHO_{sig} queries are answered. We claim that the resulting value c has the same uniform distribution as in $\mathbf{G}_5$. This is similar to an argument in the proof of [CFOS25, Theorem 1]. Namely, since $b \leftarrow\!\!\$\ \mathbb{Z}_p$ is uniform, the value of $R' = R \cdot vk^a \cdot g^b$ is independent of a. Thus, $f(R')$ is also independent of a, making $c = (f(R') + a) \bmod p$ uniform in $\mathbb{Z}_p$.

Furthermore, $\mathbf{G}_6$ introduces an abort in line 38. To bound the probability of the corresponding event Zero_f, note that (a, b) in line 36 are always well defined, because $m \notin \mathsf{Q}_m$ implies that no entry of the form $(\cdot, m, \cdot, \cdot)$ was ever added to Q_{ses}, and so $\mathsf{Q}_{sig}[vk, m, R]$ was not programmed in line 32 of HASHO_{sig}. Since the value of R' is uniform and independent of (vk, m, R) and c, the probability that $f(R') = 0$ is exactly $|f^{-1}(0)|/p$. Overall, we get:

$$\Pr[\,\mathbf{G}_5(\mathrm{A}) \Rightarrow 1\,] \le \Pr[\,\mathbf{G}_6(\mathrm{A}) \Rightarrow 1\,] + \frac{|f^{-1}(0)|}{p}. \tag{9}$$

Adversary $\mathrm{B}^{\text{DLOG}}_{\mathbb{G},g,f,n,t}(X_0, X_1, \ldots, X_t)$

1 $\mathsf{Q}_{st}, \mathsf{Q}_m, \mathsf{Q}_{cm}, \mathsf{Q}'_{cm}, \mathsf{Q}_{sig}, \mathsf{Q}_{ses}, \mathsf{Q}_z, \mathsf{Q}_{ab} \leftarrow \emptyset$; $eid \leftarrow 0$
2 $\mathsf{HS} \leftarrow [n]$; $\mathsf{CS} \leftarrow \emptyset$; $vk \leftarrow X_0$
3 For $i \in [n]$: $y_i \leftarrow \perp$; $vk_i \leftarrow \prod_{j\in[0,t]} (X_j)^{i^j}$
4 $(sk_i)_{i\in[n]} \leftarrow (y_i, vk, (vk_j)_{j\in[n]})_{i\in[n]}$
5 $(m^*, (R^*, z^*)) \leftarrow\!\!\$\ \mathrm{A}^{\text{SIGNO}_{1,2,3},\text{CORRUPTO},\text{HASHO}_{cm},\text{HASHO}_{sig}}(vk, (vk_i)_{i\in[n]})$
6 If $m^* \in \mathsf{Q}_m$: Abort
7 $c^* \leftarrow \text{HASHO}_{sig}(vk, m^*, R^*)$; $(a^*, b^*) \leftarrow \mathsf{Q}_{ab}[vk, m^*, R^*]$
8 $R \leftarrow R^* \cdot vk^{a^*} \cdot g^{b^*}$
9 If $f(R) = 0$: Abort
10 $z \leftarrow (z^* + b^*) \bmod p$
11 Return (R, z)

CORRUPTO(k):
12 $y_k \leftarrow \text{DLOG}(k)$; $sk_k \leftarrow (y_k, vk, (vk_i)_{i\in[n]})$
13 ... // the same as in $\mathbf{G}_6$

Fig. 9. The VCDL reduction B that simulates game $\mathbf{G}_6$ for adversary A, where SIGNO_1, SIGNO_2, SIGNO_3, HASHO_{cm}, and HASHO_{sig} are the same as in $\mathbf{G}_6$.

$\underline{\mathbf{G}_6.}$ Finally, consider the adversary B defined in Figure 9 against VCDL (cf. Figure 4, right side) that perfectly simulates game $\mathbf{G}_6$ for the adversary A. On input a VCDL instance $(X_0, X_1, \ldots, X_t)$, B performs a simulated Shamir secret sharing of $\log_g(X_0)$. Then, for a corruption query on signer k, B makes a query

with index k to its DLOG oracle in line 12 to receive y_k, which is precisely the discrete logarithm of the corresponding verification key share vk_k.

Suppose A wins $\mathbf{G}_6$ and outputs a forgery (m^*, σ^*) such that

(i) $m^* \notin \mathsf{Q}_m$, and
(ii) $\mathsf{Sparkle.Verify}(vk, m^*, \sigma^*) = 1$.

From (i), it follows that A never queried SIGNO_2 on m^*, while (ii) means that the signature $\sigma^* = (R^*, z^*)$ satisfies $g^{z^*} = R^* \cdot vk^{c^*}$ for $c^* \leftarrow \text{HASHO}_{sig}(vk, m^*, R^*)$. Thus, B can use the values (a^*, b^*) used to program HASHO_{sig} on (vk, m^*, R^*) to output

$$(R, z) := (R^* \cdot vk^{a^*} \cdot g^{b^*}, (z^* + b^*) \bmod p).$$

Note that B makes at most t DLOG queries since that is the corruption threshold in $\mathbf{G}_6$. Similarly, $f(R) \neq 0$ as required to win the game since B would have aborted otherwise. Finally, since $c^* = (f(R) + a^*) \bmod p$ and $vk = X_0$, we have:

$$g^z = g^{z^*} \cdot g^{b^*} = R^* \cdot vk^{f(R)+a^*} \cdot g^{b^*} = (R^* \cdot vk^{a^*} \cdot g^{b^*}) \cdot vk^{f(R)} = R \cdot X_0^{f(R)}.$$

Therefore, (R, z) is a valid VCDL solution and we get:

$$\Pr[\,\mathbf{G}_6(\text{A}) \Rightarrow 1\,] \leq \mathbf{Adv}^{\text{vcdl}}_{\mathbb{G},g,f,n,t}(\text{B}). \tag{10}$$

Combining Equations (1), (2) and (6) to (10) yields the bound in Theorem 1, finishing the proof. □

5 Relationship Between VCDL and LDVR

We study the relationship between the VCDL assumption introduced in this paper and the low-dimensional vector representation (LDVR) assumption of [CKK+25]. We prove that the two assumptions are equivalent in the elliptic-curve generic group model (EC-GGM) [GS22]: VCDL implies LDVR in the standard model (Theorem 2), and LDVR implies VCDL in the EC-GGM (Theorem 3). Together, these results situate VCDL precisely within the landscape of assumptions underlying threshold Schnorr security, and justify treating VCDL as a natural "group-based" reformulation of LDVR.

We recall the LDVR problem in Figure 10. The advantage of an adversary A solving the (p, n, t, t_c)-LDVR problem is:

$$\mathbf{Adv}^{\text{ldvr}}_{p,n,t,t_c}(\text{A}) = \Pr[\mathbf{G}^{\text{ldvr}}_{p,n,t,t_c}(\text{A}) \Rightarrow 1].$$

5.1 VCDL Implies LDVR

Inspired by the relationship between the LDVR assumption and adaptive security of a threshold Schnorr signature scheme, we now analyze the relationship between the LDVR assumption and our VCDL assumption. First, VCDL implies LDVR in the standard model. To that end, we prove the following theorem:

Game $\mathbf{G}^{\mathrm{ldvr}}_{p,n,t,t_c}$

INITIALIZE:
1 $\mathsf{ctr} \leftarrow 0$
2 For $j \in [0, n]$:
3 $\quad \boldsymbol{v}_j \leftarrow (1, j, \ldots, j^t) \in \mathbb{Z}_p^{t+1}$

$\mathcal{O}(\boldsymbol{\alpha})$: // $\boldsymbol{\alpha} \in \mathbb{Z}_p^{n+1}$
4 $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
5 $\boldsymbol{\alpha}_{\mathsf{ctr}} \leftarrow \boldsymbol{\alpha}$
6 $c_{\mathsf{ctr}} \leftarrow\!\!\$\ \mathbb{Z}_p$
7 Return c_{ctr}

FINALIZE(CS, i): // $\mathsf{CS} \subseteq [n], |\mathsf{CS}| \leq t_c, i \in [\mathsf{ctr}]$
8 $\boldsymbol{w} \leftarrow (\boldsymbol{\alpha}_i[0] + c_i) \cdot \boldsymbol{v}_0 + \sum_{j \in [n]} \boldsymbol{\alpha}_i[j] \cdot \boldsymbol{v}_j$
9 Return $(\boldsymbol{w} \in \mathrm{span}(\{\boldsymbol{v}_j\}_{j \in \mathsf{CS}}))$

Fig. 10. The LDVR problem introduced by [CKK+25].

Theorem 2 (VCDL ⇒ LDVR). *Let $\mathbb{G}$ be a group of prime order p. Let* A *be an adversary against LDVR for the parameters $n, t \in \mathbb{N}$ where $n > t$. Let $f\colon \mathbb{G} \to \mathbb{Z}_p$ be arbitrary and efficient. Then there exists an adversary* B *against VCDL for parameters f, n, t running in roughly the same time as* A *plus simulation overhead proportional to $q \cdot T_f$, where T_f is the time to compute f, such that*

$$\mathbf{Adv}^{\mathrm{ldvr}}_{p,n,t,t}(\mathrm{A}) \leq \mathbf{Adv}^{\mathrm{vcdl}}_{\mathbb{G},g,f,n,t}(\mathrm{B}) + \frac{|f^{-1}(0)|}{p} .$$

PROOF OVERVIEW. A detailed proof is given in Appendix A. Here we provide an overview. We construct a VCDL adversary B that simulates the LDVR game for an adversary A. The VCDL adversary B gets $t+1$ group elements $\boldsymbol{X} = (X_0, X_1, \ldots, X_t)$ and has access to a discrete-log oracle. When A makes its ℓ-th oracle query for vector $\boldsymbol{\alpha}_\ell$, B returns $c_\ell := f(R_\ell) + a_\ell$ where R_ℓ is a group element derived from $\boldsymbol{\alpha}_\ell$, $\boldsymbol{X}$, and fresh random scalars a_ℓ and b_ℓ. Since a_ℓ is uniform and independent of $f(R_\ell)$, the response c_ℓ is uniformly distributed over $\mathbb{Z}_p$, perfectly simulating the LDVR oracle. When A outputs a winning tuple (CS, ℓ^*), we obtain coefficients $\boldsymbol{\beta}$ with $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{k \in \mathsf{CS}} \beta[k]\, y_k$. Defining $z^* := b_{\ell^*} + \langle \boldsymbol{w}, \boldsymbol{x} \rangle$, substituting $c_{\ell^*} = f(R_{\ell^*}) + a_{\ell^*}$ and expanding gives:

$$z^* = b_{\ell^*} + (c_{\ell^*} + \boldsymbol{\alpha}_{\ell^*}[0]) \cdot x_0 + \sum_{j \in [n]} \boldsymbol{\alpha}_{\ell^*}[j] \cdot \langle \boldsymbol{v}_j, \boldsymbol{x} \rangle,$$

which satisfies the VCDL forgery condition $g^{z^*} = R_{\ell^*} \cdot X_0^{f(R_{\ell^*})}$. The only failure is when $f(R_{\ell^*}) = 0$, which occurs with probability $|f^{-1}(0)|/p$ since R_{ℓ^*} is uniformly distributed in $\mathbb{G}$.

5.2 LDVR Implies VCDL in the EC-GGM

In the other direction, since LDVR is not a group-based assumption, a standard-model reduction solely from the VCDL to the LDVR assumption seems unlikely. Interestingly, we show this implication in the EC-GGM [GS22], where VCDL can be expressed in terms of polynomials. Specifically, we prove that LDVR

implies VCDL in the EC-GGM for regular and preimage-sampleable conversion functions.

Theorem 3 (LDVR ⇒ VCDL in EC-GGM). *Let E be an elliptic curve of prime order p. Let $f\colon E \to \mathbb{Z}_p$ be a regular function that is efficiently computable and preimage-sampleable. Let* A *be an adversary against VCDL for the parameters $n, t \in \mathbb{N}$ where $n > t$ in the EC-GGM with E that makes at most q queries to its oracles. Then, there exists an adversary* B *against LDVR with parameters (p, n, t, t) such that*

$$\mathbf{Adv}^{\text{ec-ggm-vcdl}}_{E,f,n,t}(\mathrm{A}) \leq \mathbf{Adv}^{\text{ldvr}}_{p,n,t,t}(\mathrm{B}) + \frac{1}{p - 4q - 2t - 5} + \frac{q^2}{p - q^2} + \frac{q}{p}.$$

Note that, as in CFOS [CFOS25], more conventional idealized models such as the algebraic group model (AGM) [FKL18] are unsuitable here because the conversion function maps points on the curve to bits.

Remark 1. The requirement that f be regular can be relaxed and our proof still goes through. Concretely, it suffices if f is *approximately* regular in the sense that for every $c \in \mathrm{dom}(f)$,

$$\Pr\big[y \leftarrow\!\!\$\ \mathrm{ran}(f); x \leftarrow\!\!\$\ f^{-1}(y) : x = c\big] \approx 1/|\,\mathrm{dom}(f)|.$$

For simplicity of exposition, we omit this generalization in the proof.

Remark 2. As in CFOS, the ECDSA conversion function $(x, y) \mapsto x \bmod p$ is a suitable choice for f above. Approximate regularity of the ECDSA conversion function holds since almost all outputs have two preimages (a small amount can have four), and preimage sampleability follows by the Tonelli–Shanks algorithm.

Proof intuition. Recall that in Shoup's *generic group model* (GGM) [Sho97], group elements are represented by random labels, and the adversary can perform the group operation by querying an oracle with two labels to receive the label of the corresponding result. In the *elliptic-curve GGM* (EC-GGM) [GS22], we fix a curve E and labels are random elements of E respecting basic properties such as the identity and inverses. Since E is fixed, the adversary can also query the group operation oracle with labels it never received from a group operation query. Moreover, the adversary is given a map oracle that takes $x \in \mathbb{Z}_p$ and returns the label of G^x where G generates E.

To reduce VCDL to LDVR in the EC-GGM, we begin with the strategy from [BFP21]: group elements are (gradually) internally represented as formal polynomials in $\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_t$, with the adversary's DL queries revealing polynomial evaluations. We then incorporate the core insight from the CFOS [CFOS25] analysis of CDL in the EC-GGM: in a valid CDL solution (R, z), the element R must correspond to a *non-constant* polynomial. In CFOS's univariate setting, such a solution is unconditionally hard to produce. In our multivariate setting, the CFOS argument no longer applies because the DL oracle reveals an equivalent form of the secret-sharing polynomial, and any two polynomials whose difference

is in the span of the adversary's knowledge are represented by the *same* group element. Our contribution is to show that in this case a CDL solution lets us solve LDVR.

More concretely, these multivariate polynomials can be interpreted as coefficient vectors in the underlying vector space of LDVR. The main technical trick is to connect outputs of the LDVR oracle on such vectors to outputs of f, using preimage sampleability of f. Specifically, on a map query, the reduction extracts the coefficient vector of the relevant polynomial, converts it from the standard basis to the Vandermonde basis, and forwards this vector to its LDVR oracle. The oracle's response c is then interpreted as an output of f, and by preimage sampleability of f, the reduction samples $R \in f^{-1}(c)$ uniformly and returns it. Finally, when the VCDL adversary outputs a CDL solution (R, z), the reduction identifies the LDVR query index that produced the embedding of R. Together with the set of discrete-log queries asked during the execution, this forms a valid LDVR solution, since correctness of the CDL equation implies that the coefficient vector of R lies in the span of the Vandermonde vectors defined by the DL queries. Thus, if the VCDL adversary is successful then our LDVR adversary is successful as well.

Proof. We provide a series of games starting from VCDL in the EC-GGM, denoted by $\mathbf{G}_0$ in Figure 11, to the final game $\mathbf{G}_6$ in Figure 14, with intermediate games $\mathbf{G}_1$–$\mathbf{G}_5$ in Figures 12 and 13. We assume that the adversary makes exactly t queries to the DLOG oracle. This is without loss of generality, since if the number of queries $|Q| < t$ during FINALIZE, then the game can make the remaining $t - |Q|$ queries with arbitrary inputs $k \in [n] \setminus Q$.

First, we include a helpful lemma which will be used throughout the proof. The lemma is from Bauer *et al.* [BFP21] with some notation changed to be consistent with our proof. We consider polynomials of degree 1 in $\mathbb{Z}_p[\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_t]$ for a prime p. A polynomial D can be written as:

$$D(\mathbf{X}) = \sum_{i \in [0,t]} d_i \cdot \mathsf{X}_i + d_{t+1} = \langle \boldsymbol{d}, \mathbf{X} \rangle + d_{t+1}$$

where $\mathbf{X} = (\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_t)$ and the vector $\boldsymbol{d} = (d_0, d_1, \ldots, d_t) \in \mathbb{Z}_p^{t+1}$ excludes the constant scalar term d_{t+1}. For a polynomial D, let $\mathcal{D}$ be the set of roots of D in $\mathbb{Z}_p^{t+1}$: $\mathcal{D} = \{\boldsymbol{x} \in \mathbb{Z}_p^{t+1} : D(\boldsymbol{x}) = 0\}$. For a polynomial Q, $\mathcal{Q}$ is defined similarly. Let $\mathcal{C}$ be defined as follows:

$$\mathcal{C} := \Big(\bigcap_{j \in [\ell]} \mathcal{Q}_j \Big) \setminus \Big(\bigcup_{i \in [m]} \mathcal{D}_i \Big). \tag{11}$$

Lemma 1 ([BFP21]). *Let $D_1, \ldots, D_m, Q_1, \ldots, Q_{\ell+1} \in \mathbb{Z}_p[\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_t]$ be polynomials of degree 1. If we assume (1) $\mathcal{Q}_{\ell+1} \cap \mathcal{C} \neq \emptyset$, (2) $\boldsymbol{q}_{\ell+1}$ is linearly independent of $\{\boldsymbol{q}_j\}_{j \in [\ell]}$, and (3) $\boldsymbol{x} \leftarrow\!\!\$\, \mathcal{C}$ where $\mathcal{C}$ is defined in Eq. (11), then*

$$\frac{p-m}{p^2} \;\leq\; \Pr\left[Q_{\ell+1}(\boldsymbol{x}) = 0\right] \;\leq\; \frac{1}{p-m}.$$

Game $\mathbf{G}^{\text{ec-ggm-vcdl}}_{E,f,n,t}$

INITIALIZE:
$Q \leftarrow \emptyset$
$\tau \leftarrow \{(0, 1_E)\}$
$G \leftarrow \text{MAP}(1)$
For $i \in [0, t]$:
  $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $X_i \leftarrow \text{MAP}(x_i)$
Return $(G, X_0, X_1, \ldots X_t)$

MAP(a):
If $a \in \text{dom}(\tau)$: Return $\tau(a)$
$A \leftarrow\!\!\$\ E \setminus \text{ran}(\tau)$
$\tau \leftarrow \tau \cup \{(a, A), (-a, A^{-1})\}$
Return $\tau(a)$

ADD(c_1, A_1, c_2, A_2):
For $b \in \{1, 2\}$:
  If $A_b \notin \text{ran}(\tau)$:
    $a \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \text{dom}(\tau)$
    $\tau \leftarrow \tau \cup \{(a, A_b), (-a, A_b^{-1})\}$
$a \leftarrow c_1 \cdot \tau^{-1}(A_1) + c_2 \cdot \tau^{-1}(A_2)$
Return MAP(a)

DLOG(k): // $k \in [n],\ |Q| < t$
$Q \leftarrow Q \cup \{k\}$
$y_k \leftarrow x_0 + \sum_{i \in [t]} k^i \cdot x_i$
Return y_k

FINALIZE(R, z):
Return $\big(f(R) \neq 0 \ \wedge\ \text{MAP}(z) = \text{ADD}(1, R, f(R), X_0)\big)$

Fig. 11. The EC-GGM game for the Vandermonde circular discrete-logarithm (VCDL) problem ($\mathbf{G}_0$).

$\mathbf{G}_0 \to \mathbf{G}_1$. The first hybrid game $\mathbf{G}_1$ is given in Figure 12. Here, we use a helper function $\text{MAP}_{\text{HELPER}}$ that keeps track of polynomials throughout the execution of the game. Additionally, we include a check in the FINALIZE procedure.

In order to analyze this game, we give two claims about the $\text{MAP}_{\text{HELPER}}$ function and the polynomials used in the proof.

Claim 2. *In the games that use a* $\text{MAP}_{\text{HELPER}}$ *function and where the domain of* τ *consists of a tuple of a scalar and a polynomial, any* (a, P) *in the domain of* τ *is such that* $P(\boldsymbol{x}) = a$ *where* $\boldsymbol{x} = (x_0, x_1, \ldots, x_t)$ *is sampled during* INITIALIZE.

Proof. We prove this claim by induction. The first entries of τ are $((0, 0), 1_E)$ and $((1, 1), G)$. In both entries, the polynomial is constant and thus evaluates to that constant value for any $\boldsymbol{x}$, which is not sampled at this point. Similarly, any entry coming from MAP is redirected as a tuple of the scalar and a constant

Game $\mathbf{G}_1(E, f, n, t)$ / $\boxed{\mathbf{G}_2(E, f, n, t)}$

INITIALIZE:
$Q \leftarrow \emptyset$; $\boxed{L \leftarrow \emptyset}$
$\tau \leftarrow \{((0,0), 1_E)\}$
$G \leftarrow \text{MAP}(1)$
For $i \in [0, t]$:
  $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$
  $X_i \leftarrow \text{MAP}_{\text{HELPER}}(x_i, \mathsf{X}_i)$
Return $(G, X_0, X_1, \ldots, X_t)$

$\text{MAP}(a)$:
Return $\text{MAP}_{\text{HELPER}}(a, a)$

$\text{MAP}_{\text{HELPER}}(a, P)$: // not accessible to the adversary
If $(a, P) \in \text{dom}(\tau)$: Return $\tau(a, P)$
$\boxed{\text{If } (\exists (a, P') \in \text{dom}(\tau) : P - P' \notin \text{span}(L)) \text{: Abort}}$
If $\exists P' \neq P : (a, P') \in \text{dom}(\tau)$: // pick the first satisfying P'
  $A \leftarrow \tau(a, P')$
Else:
  $A \leftarrow\!\!\$\ E \setminus \text{ran}(\tau)$
$\tau \leftarrow \tau \cup \{((a, P), A), ((-a, -P), A^{-1})\}$
Return $\tau(a, P)$

$\text{ADD}(c_1, A_1, c_2, A_2)$:
For $b \in \{1, 2\}$:
  If $A_b \notin \text{ran}(\tau)$:
    $a \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{a' \mid (a', \cdot) \in \text{dom}(\tau)\}$
    $\tau \leftarrow \tau \cup \{((a, a), A_b), ((-a, -a), A_b^{-1})\}$
Let (a_1, P_1) be the first entry in τ such that $\tau(a_1, P_1) = A_1$
Let (a_2, P_2) be the first entry in τ such that $\tau(a_2, P_2) = A_2$
$P \leftarrow c_1 \cdot P_1 + c_2 \cdot P_2$
$a \leftarrow c_1 \cdot a_1 + c_2 \cdot a_2$
Return $\text{MAP}_{\text{HELPER}}(a, P)$

$\text{DLOG}(k)$: // $k \in [n]$, $|Q| < t$
$Q \leftarrow Q \cup \{k\}$
$P_k \leftarrow \mathsf{X}_0 + \sum_{i \in [t]} k^i \cdot \mathsf{X}_i$
$y_k \leftarrow P_k(\boldsymbol{x})$
$\boxed{L \leftarrow L \cup \{P_k - y_k\}}$
Return y_k

$\text{FINALIZE}(R, z)$:
If $R \notin \text{ran}(\tau)$: Return 0
Return $\big(f(R) \neq 0 \ \wedge\ \text{MAP}(z) = \text{ADD}(1, R, f(R), X_0)\big)$

Fig. 12. Games $\mathbf{G}_1$ and $\mathbf{G}_2$ for the proof of Theorem 3. Changes from game $\mathbf{G}_0$ are highlighted in blue, while changes from game $\mathbf{G}_1$ to $\mathbf{G}_2$ are additionally boxed.

polynomial equal to the same scalar. When (a, a) is added to the domain of τ, then $P(\boldsymbol{x}) = a$ holds for any $\boldsymbol{x}$.

When $\text{MAP}_{\text{HELPER}}((x_i, \mathsf{X}_i))$ is called, the polynomial consists of exactly one variable X_i and its corresponding scalar is the value x_i, so the statement is true here. Finally, when $\text{MAP}_{\text{HELPER}}$ is called from ADD, we can see that P and a are computed using the same coefficients. In other words, if $P = c_1 \cdot P_1 + c_2 \cdot P_2$, then $P(\boldsymbol{x}) = c_1 \cdot P_1(\boldsymbol{x}) + c_2 \cdot P_2(\boldsymbol{x})$. Thus, since it holds that $P_1(\boldsymbol{x}) = a_1$ and $P_2(\boldsymbol{x}) = a_2$, then $P(\boldsymbol{x}) = a$.

We cover all instances of additions to the domain of τ, so the statement holds. □

Claim 3. *At any point of execution, if a polynomial D is in the span of L, we have $D(\boldsymbol{x}) = 0$ where $\boldsymbol{x} = (x_0, x_1, \ldots, x_t)$ is sampled during* INITIALIZE.

Proof. By construction, for each k in the challenge index set Q, the list L contains a degree-1 polynomial of the form

$$D_k = P_k - y_k, \text{ where } P_k = \mathsf{X}_0 + \sum_{i \in [t]} k^{\,i}\, \mathsf{X}_i \text{ and } y_k = P_k(\boldsymbol{x}).$$

When evaluating $D_k(\boldsymbol{x})$, we get $(P_k - y_k)(\boldsymbol{x}) = P_k(\boldsymbol{x}) - P_k(\boldsymbol{x}) = 0$. So, every polynomial in L vanishes at $\boldsymbol{x}$. Then, for a polynomial $D' \in \text{span}(L)$, it can be written as a linear combination of polynomials in L as $D = \sum_{D_k \in L} \beta_k \cdot D_k$ for some $\beta_k \in \mathbb{Z}_p$ where all $k \in Q$. Evaluated under $\boldsymbol{x}$, we have $D(\boldsymbol{x}) = \sum_{D_j \in L} \beta_j \cdot D_k(\boldsymbol{x}) = \sum_{D_j \in L} \beta_j \cdot 0 = 0$. Thus, every polynomial in the span of L evaluates to 0 under $\boldsymbol{x}$.

As a corollary, this means that if $P - P' \in \text{span}(L)$ for two polynomials P and P', then $P(\boldsymbol{x}) = P'(\boldsymbol{x})$ under $\boldsymbol{x}$. □

We make the following claim about MAP and $\text{MAP}_{\text{HELPER}}$ in games $\mathbf{G}_0$ and $\mathbf{G}_1$, showing that this restructure does not affect the game.

Claim 4. *The execution of* MAP *in game $\mathbf{G}_0$ is identical to the execution of* MAP *and* $\text{MAP}_{\text{HELPER}}$ *combined in game $\mathbf{G}_1$. In particular, the second component P in the domain of τ does not affect the execution.*

Proof. In game $\mathbf{G}_1$, $\text{MAP}(a)$ acts as a wrapper that calls $\text{MAP}_{\text{HELPER}}(a, a)$. Thus, it suffices to compare MAP in game $\mathbf{G}_0$ with the execution of $\text{MAP}_{\text{HELPER}}$ in this game.

In game $\mathbf{G}_0$, if a is already in the domain of τ, then the game returns the associated group element A. Otherwise, if $a \notin \text{dom}(\tau)$, then the game samples a new group element A, adds (a, A) to τ and returns A. On the other hand, in game $\mathbf{G}_1$, if a is already in the domain of τ, it is either associated with the input polynomial P or some other polynomial P'. In both cases, the game returns the corresponding group element A. Similarly, if a is not in the domain of τ at all, the game reaches the else statement in line 13 and samples a new group element A and returns it. In this game, every unique tuple is added to τ which can create multiple entries of the form $((a, \cdot), A)$.

Thus each a is associated with a unique A, and all calls to $\text{MAP}_{\text{HELPER}}(a, P)$ yield this A. Hence, the two executions are identical. □

Claim 5. *The probability that an adversary* A *wins game* $\mathbf{G}_0$ *such that* $R \notin \text{ran}(\tau)$ *is given by*

$$\Pr[\mathbf{G}_0(\text{A}) \Rightarrow 1 \wedge R \notin \text{ran}(\tau)] \leq \frac{1}{p - 4q - 2t - 5},$$

where q *is the total number of oracle queries made by the adversary.*

Proof. When an adversary wins game $\mathbf{G}_0$ with (R, z), we have $f(R) \neq 0$ and $\text{MAP}(z) = \text{ADD}(1, R, f(R), X_0)$. If R was not already in τ, then the call to ADD samples a fresh $r \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \text{dom}(\tau)$ and assigns it to R. This value must equal $z - f(R) \cdot x_0$, which is fixed in advance, so this happens with probability $1/|\mathbb{Z}_p \setminus \text{dom}(\tau)|$.

Next, we compute the size of the set $\mathbb{Z}_p \setminus \text{dom}(\tau)$. The set τ consists of the entry $(0, 1_E)$, the $2(t+2)$ entries from the $t+2$ calls to MAP in INITIALIZE, and at most $4q$ entries from q calls to oracles by the adversary. One call to ADD adds a maximum of 4 entries to τ when both A_1 and A_2 are not in $\text{ran}(\tau)$. Thus, the total number of entries in τ is at most $1 + 2(t+2) + 4q = 4q + 2t + 5$. Hence, the probability is at most $1/(p - 4q - 2t - 5)$. □

Finally, note that the check inside FINALIZE is the only difference that affects the execution of games $\mathbf{G}_0$ and $\mathbf{G}_1$. Hence, we bound this difference as follows:

$$\begin{aligned}\Pr[\mathbf{G}_0(\text{A}) \Rightarrow 1] &= \Pr[\mathbf{G}_0(\text{A}) \Rightarrow 1 \wedge R \in \text{ran}(\tau)] + \Pr[\mathbf{G}_0(\text{A}) \Rightarrow 1 \wedge R \notin \text{ran}(\tau)] \\ &\leq \Pr[\mathbf{G}_1(\text{A}) \Rightarrow 1] + \frac{1}{p - 4q - 2t - 5}\ .\end{aligned}$$

$\underline{\mathbf{G}_1 \to \mathbf{G}_2.}$ In game $\mathbf{G}_2$ (Figure 12), we add an abort condition. In order to create the abort condition, we introduce a set L of polynomials that is populated during DLOG queries, which tracks the knowledge of the adversary. In particular, when a query k is made to the oracle, the oracle computes $y_k = P_k(\boldsymbol{x})$ where P_k is defined in line 27 and adds the polynomial $P_k - y_k$ to L. This represents the information that the adversary has about the secret vector $\boldsymbol{x}$. This extra computation does not affect the execution of the game, which is only affected by the abort condition elaborated next.

The game aborts on input (a, P) to $\text{MAP}_{\text{HELPER}}$ if there already exists an entry (a, P') in the domain of τ but $P - P'$ is not in the span(L). Intuitively, this means that the existing knowledge the adversary has is insufficient to reveal that P and P' evaluate to the same value a under the secret $\boldsymbol{x}$, but returning the same group element would reveal that information. Let F denote this abort event for a fixed (a, P). Since the number of queries made by the adversary is bounded by q, the difference in advantage between the two games can be computed as:

$$\Pr[\mathbf{G}_1(\text{A}) \Rightarrow 1] \leq \Pr[\mathbf{G}_2(\text{A}) \Rightarrow 1] + q \cdot \Pr[F]\ . \tag{12}$$

To bound $\Pr[F]$ for (a, P), consider another fixed (a', P') already in $\mathrm{dom}(\tau)$ where $a' = a$. Let $P^* := P - P'$. We bound the probability that $P^*(\boldsymbol{x}) = P(\boldsymbol{x}) - P'(\boldsymbol{x}) = 0$ while $P^* \notin \mathrm{span}(L)$.

Using the language of Lemma 1, let $L = \{Q_1, \ldots, Q_\ell\}$ and $Q_{\ell+1} := P^*$ where ℓ is the number of DLOG queries already made when (a, P) is first input to $\text{MAP}_{\text{HELPER}}$. For each $Q_j \in L$, the adversary knows that $Q_j(\boldsymbol{x}) = (P_{k_j} - y_{k_j})(\boldsymbol{x}) = 0$. Similarly, for any pair of group elements $A_1 \neq A_2$ in the range of τ, the adversary knows that the corresponding (a_1, P_1) and (a_2, P_2) satisfy $a_1 \neq a_2$ and $P_1 \neq P_2$ from Claim 2. Again using the language of the lemma, let $D_i = P_1 - P_2$ for all pairs $A_1 \neq A_2$ and let m be the total number of polynomials D_i. Since $P_1(\boldsymbol{x}) \neq P_2(\boldsymbol{x})$, the adversary knows that $D_i(\boldsymbol{x}) \neq 0$. Hence, the challenge vector $\boldsymbol{x}$ lies in the set $\mathcal{C}$ defined as:

$$\mathcal{C} := \Big(\bigcap_{j\in[\ell]} \mathcal{Q}_j\Big) \setminus \Big(\bigcup_{i\in[m]} \mathcal{D}_i\Big).$$

Intuitively, $\mathcal{C}$ contains vectors that vanish at polynomials in L but do not vanish at polynomials D_i.

With the setup defined as in the lemma, we verify the three requirements: (1) $\boldsymbol{x}$ is uniform in $\mathcal{C}$, (2) $\mathcal{Q}_{\ell+1} \cap \mathcal{C} \neq \emptyset$, and (3) $\boldsymbol{q}_{\ell+1}$ is independent of $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$.

$\boldsymbol{x}$ IS UNIFORM IN $\mathcal{C}$. We first fix the randomness in the game, which includes all the coins of the challenger and the adversary. Now, consider the transcript of what the adversary receives from the game:

$$G, X_0, \ldots, X_t, A_1, \ldots, A_{q'}, y_{k_1}, \ldots, y_{k_\ell}.$$

Here, $G, X_0, \ldots, X_t$ are output by INITIALIZE, $A_1, \ldots, A_{q'}$ are output by queries to MAP and ADD for some $q' \leq q$, and $y_{k_1}, \ldots, y_{k_\ell}$ are output by DLOG for queries $k_1, \ldots, k_\ell$. These values are generated assuming a fixed secret $\boldsymbol{x}$. We want to show that this transcript is distributed identically for an arbitrary $\boldsymbol{v} \in \mathcal{C}$, meaning these values only depend on membership in $\mathcal{C}$ and not on the specific choice of $\boldsymbol{x}$.

First, during INITIALIZE when MAP or $\text{MAP}_{\text{HELPER}}$ is called, there are no elements in L, so the game does not abort. Then, for both $\boldsymbol{x}$ and $\boldsymbol{v}$, the same group elements are chosen to be $X_0, \ldots, X_t$.

Next, we show by induction that each A_γ and each y_{k_j} is the same for both challenge vectors $\boldsymbol{x}$ and $\boldsymbol{v}$.

- During a call to DLOG, the adversary makes the same queries since everything in the transcript and randomness are the same. For a query k_j, first the polynomial P_{k_j} is computed as $\mathsf{X}_0 + \sum_{i\in[t]} {k_j}^i \cdot \mathsf{X}_i$. Then, y_{k_j} is computed as $P_{k_j}(\boldsymbol{x})$ for $\boldsymbol{x}$ and let y'_{k_j} be $P_{k_j}(\boldsymbol{v})$ for $\boldsymbol{v}$. Note that $Q_j = P_{k_j} - y_{k_j}$ and $Q'_j = P_{k_j} - y'_{k_j}$ are the corresponding polynomials added to L. We know that $\boldsymbol{v} \in \mathcal{C}$ which means that $\boldsymbol{v} \in \mathcal{Q}_j$ as well. Thus, $Q_j(\boldsymbol{v}) = P_{k_j}(\boldsymbol{v}) - y_{k_j} = 0$. This means that $P_{k_j}(\boldsymbol{v}) = y_{k_j} = y'_{k_j}$ and $Q_j = Q'_j$.

– Then, during a call to MAP or ADD, the adversary again makes the same queries in both cases. For a call to $\text{MAP}_{\text{HELPER}}$ coming from MAP or ADD, let (a_γ, P_γ) be the input and A_γ be the output.
  - If for all $(\hat{a}, \hat{P}) \in \text{dom}(\tau)$, $\hat{a} \neq a_\gamma$ meaning $\hat{P}(\boldsymbol{x}) \neq P_\gamma(\boldsymbol{x})$, then the game with challenge $\boldsymbol{x}$ outputs a new group element A_γ. For the game with $\boldsymbol{v}$, since $\boldsymbol{v} \in \mathcal{C}$, it means that $\boldsymbol{v} \notin \mathcal{D}_i$ for any $D_i = P_\gamma - \hat{P}$ for all $(\hat{a}, \hat{P}) \in \text{dom}(\tau)$. Thus, $P_\gamma(\boldsymbol{v}) \neq \hat{P}(\boldsymbol{v})$ and $a_\gamma \neq \hat{a}$, so this game also outputs the same new group element A_γ.
  - If there exists $(a, \hat{P}) \in \text{dom}(\tau)$, then the game with $\boldsymbol{x}$ assigns $\tau(a, \hat{P})$ to A_γ. Since the game did not abort yet, it means that for the same $\hat{P}$, $P_\gamma - \hat{P} \in \text{span}(L)$. From $\boldsymbol{v} \in \cap_{j \in [\ell]} \mathcal{Q}_j$, we get $P_\gamma(\boldsymbol{v}) = \hat{P}(\boldsymbol{v})$ and $a_\gamma = \hat{a}$. Thus, this game also assigns $\tau(a, \hat{P})$ to A_γ.

  In both cases, the group element A_γ is the same.

Thus, the adversary always gets the same transcript for any $\boldsymbol{v} \in \mathcal{C}$, so $\boldsymbol{x}$ is distributed uniformly in the set $\mathcal{C}$ from the point of view of the adversary.

$\mathcal{Q}_{\ell+1} \cap \mathcal{C} \neq \emptyset$. If $\mathcal{Q}_{\ell+1} \cap \mathcal{C} = \emptyset$, then $P^*(\boldsymbol{v}) = 0$ for all $\boldsymbol{v} \in \mathcal{C}$. Since we are interested in bounding the probability that $P^*(\boldsymbol{v}) = 0$, we can assume that this set is not empty.

$\boldsymbol{q}_{\ell+1}$ IS INDEPENDENT OF $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$. For contradiction, suppose $\boldsymbol{q}_{\ell+1}$ is linearly dependent on $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$. Recall that $\boldsymbol{q}$ is the vector of coefficients of the polynomial Q excluding the scalar value. Thus, there exists $Q \in \text{span}(L)$ such that we can write $P^* = Q_{\ell+1}$ as $Q + \alpha$ where $\alpha \in \mathbb{Z}_p$. By the abort event F we know that $P^* \notin \text{span}(L)$, so it follows that $\alpha \neq 0$.

Moreover, by construction, we have $\mathcal{C} \subset \mathcal{Q}_j$ for every $j \in [\ell]$, so we have $Q_j(\boldsymbol{v}) = 0$ for all $\boldsymbol{v} \in \mathcal{C}$. Since $Q \in \text{span}(L)$, we get $Q(\boldsymbol{v}) = 0$. Substituting into P^*, we can write $P^*(\boldsymbol{v}) = Q(\boldsymbol{v}) + \alpha = \alpha$. Thus, $P^*(\boldsymbol{v}) \neq 0$ for all $\boldsymbol{v} \in \mathcal{C}$ and $\mathcal{Q}_{\ell+1} \cap \mathcal{C} = \emptyset$. This contradicts the previous condition, so it cannot be that $\boldsymbol{q}_{\ell+1}$ is linearly dependent on $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell$.

Since each of the conditions above is satisfied, the lemma can be applied as follows:

$$\Pr_{\boldsymbol{x} \leftarrow \$ \mathcal{C}}[P^*(\boldsymbol{x}) = 0] \leq \frac{1}{p - m} \leq \frac{1}{p - q^2},$$

where $m \leq q^2$ since the adversary A knows at most q^2 many pairs of group elements. For each fixed input (a, P) to $\text{MAP}_{\text{HELPER}}$, there are at most q entries (a', P') already in the domain of τ, so $\Pr[F] \leq \frac{q}{p - q^2}$ and finally

$$\Pr[\mathbf{G}_1(\text{A}) \Rightarrow 1] \leq \Pr[\mathbf{G}_2(\text{A}) \Rightarrow 1] + \frac{q^2}{p - q^2}\ .$$

$\mathbf{G}_2 \rightarrow \mathbf{G}_3$. In game $\mathbf{G}_3$ (Figure 13), we refactor the mapping set τ. From game $\mathbf{G}_2$ to game $\mathbf{G}_3$, the difference is in the oracles $\text{MAP}_{\text{HELPER}}$ and MAP. Next, we show that the combined execution of MAP and $\text{MAP}_{\text{HELPER}}$ in game $\mathbf{G}_2$ is identical to the execution of MAP in game $\mathbf{G}_3$.

Game $\mathbf{G}_3(E,f,n,t)$ / $\boxed{\mathbf{G}_4(E,f,n,t)}$ / $\boxed{\boxed{\mathbf{G}_5(E,f,n,t)}}$

INITIALIZE:
$Q \leftarrow \emptyset$; $L \leftarrow \emptyset$
$\tau \leftarrow \{(0, 1_E)\}$
$G \leftarrow \text{MAP}(1)$
For $i \in [0,t]$:
  $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$
  $X_i \leftarrow \text{MAP}(\mathsf{X}_i)$
Return $(G, X_0, X_1, \ldots, X_t)$

MAP(P): // adversary query can only be a constant
If $(\exists P' \in \text{dom}(\tau) : P(\boldsymbol{x}) = P'(\boldsymbol{x}) \wedge P - P' \notin \text{span}(L))$: Abort
If $\exists P' \in \text{dom}(\tau) : P - P' \in \text{span}(L)$: // pick the first satisfying P'
  $A \leftarrow \tau(P')$
Else:
  $A \leftarrow\!\!\$\ E \setminus \text{ran}(\tau)$
$\tau \leftarrow \tau \cup \{(P, A), (-P, A^{-1})\}$
Return $\tau(P)$

ADD(c_1, A_1, c_2, A_2):
For $b \in \{1,2\}$:
  If $A_b \notin \text{ran}(\tau)$:
    $a \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \text{dom}(\tau)$
    $\tau \leftarrow \tau \cup \{(a, A_b), (-a, A_b^{-1})\}$
Let P_1 be the first entry in τ such that $\tau(P_1) = A_1$
Let P_2 be the first entry in τ such that $\tau(P_2) = A_2$
$P \leftarrow c_1 \cdot P_1 + c_2 \cdot P_2$
Return MAP(P)

DLOG(k): // for adversary query: $k \in [n]$, $|Q| < t$
$Q \leftarrow Q \cup \{k\}$
$P_k \leftarrow \mathsf{X}_0 + \sum_{i \in [t]} k^i \cdot \mathsf{X}_i$
$y_k \leftarrow P_k(\boldsymbol{x})$
$\boxed{y_k \leftarrow\!\!\$\ \mathbb{Z}_p}$
$L \leftarrow L \cup \{P_k - y_k\}$
$\boxed{\text{If } \exists P_1, P_2 \in \text{dom}(\tau) : P_1 - P_2 \in \text{span}(L) \wedge \tau(P_1) \neq \tau(P_2)\text{: Abort}}$
Return y_k

FINALIZE(R, z):
$\boxed{\text{DLOG}(0)}$
If $R \notin \text{ran}(\tau)$: Return 0
Return $\big(f(R) \neq 0 \ \wedge\ \text{MAP}(z) = \text{ADD}(1, R, f(R), X_0)\big)$

Fig. 13. Games $\mathbf{G}_3$ (which includes the dashed box), $\mathbf{G}_4$ and $\mathbf{G}_5$ (which do not include the dashed box) for the proof of Theorem 3. Changes from game $\mathbf{G}_2$ are highlighted in blue, while changes from game $\mathbf{G}_3$ to $\mathbf{G}_4$ use solid boxes and changes from game $\mathbf{G}_4$ to $\mathbf{G}_5$ use double boxes.

Fix an input pair (a, P) in game $\mathbf{G}_2$ and input P in game $\mathbf{G}_3$ where $a = P(\boldsymbol{x})$. We distinguish two cases depending on whether a polynomial P' such that $P'(\boldsymbol{x}) = a$ is already in the domain of τ.

- **Case 1:** There exists such a polynomial P'. In game $\mathbf{G}_2$, the domain of τ contains (a, P'), while in game $\mathbf{G}_3$ it contains P'. We analyze two subcases.
  - Subcase 1a: $P - P' \notin \text{span}(L)$. In this case, both games abort. By Claim 2, whenever (a, P) is in the domain of τ, we have $P(\boldsymbol{x}) = a$. Thus, checking $P(\boldsymbol{x}) = P'(\boldsymbol{x})$ is equivalent to requiring the same scalar a.
  - Subcase 1b: $P - P' \in \text{span}(L)$. Here, by the construction of τ, we have $P(\boldsymbol{x}) = P'(\boldsymbol{x}) = a$. In game $\mathbf{G}_2$, if $(a, P') \in \text{dom}(\tau)$ and $P - P' \in \text{span}(L)$, then the game does not abort and the encoding $\tau(a, P')$ is reused for (a, P). In game $\mathbf{G}_3$, the check $P - P' \in \text{span}(L)$ is made directly, and $\tau(P)$ is set equal to $\tau(P')$.
- **Case 2:** There does not exist a polynomial P' with $P'(\boldsymbol{x}) = a$. In game $\mathbf{G}_2$, the game directly checks this and samples a new group element to assign to $\tau((a, P))$. In game $\mathbf{G}_3$, since no P' with $P(\boldsymbol{x}) = P'(\boldsymbol{x})$ exists, neither of the earlier conditions apply, and a fresh group element is sampled for $\tau(P)$.

Note that if P itself is already in the domain τ, then $P - P' = 0 \in \text{span}(L)$ at any point in execution including when $L = \emptyset$. This is why we do not need to separately check if P is in the domain of τ and it can be folded into subcase 1b. Thus, in all possible cases, the games behave identically on inputs (a, P) and P respectively.

Hence, the two games behave identically:

$$\Pr[\mathbf{G}_2(\mathrm{A}) \Rightarrow 1] = \Pr[\mathbf{G}_3(\mathrm{A}) \Rightarrow 1].$$

$\underline{\mathbf{G}_3 \to \mathbf{G}_4.}$ We present game $\mathbf{G}_4$ in Figure 13. The only change from game $\mathbf{G}_3$ to game $\mathbf{G}_4$ is that the abort condition checking inconsistency is moved from Map to Dlog. Intuitively, this means that the check is performed when L is updated instead of on each call to Map. Formally, we show that game $\mathbf{G}_3$ aborts if and only if game $\mathbf{G}_4$ aborts by proving both directions.

Before this proof, we make the following claim.

Claim 6. *After the query with $k = 0$ to* Dlog *in* Finalize*, the set L contains $t + 1$ polynomials and, for each $i \in [0, t]$, $\mathsf{X}_i - x_i$ is in the span of L.*

Proof. Just before Finalize, L contains t polynomials of the form $P - P(\boldsymbol{x}) = \mathsf{X}_0 + \sum_{i \in [t]} k^i \cdot \mathsf{X}_i - P(\boldsymbol{x})$, each corresponding to a distinct nonzero value $k \in \mathbb{Z}_p$. We assume without loss of generality that A makes exactly t queries to Dlog. At Finalize, the game makes an additional call to Dlog with input $k = 0$, which inserts the polynomial $\mathsf{X}_0 - x_0$ to the set L. Now, the set L contains $t + 1$ polynomials. Note that the restriction on the size of the set Q only applies to adversary queries.

Consider the coefficients of the polynomials in L including the polynomial $\mathsf{X}_0 - x_0$. In particular, if we rearrange and take the coefficients with respect to $(\mathsf{X}_0 - x_0, \mathsf{X}_1 - x_1, \ldots, \mathsf{X}_t - x_t)$, we get the Vandermonde vectors $(1, k^1, k^2, \ldots, k^t)$

for $t{+}1$ distinct values $k_0 = 0, k_1, \ldots, k_t$. Since the k_i are distinct, the $(t{+}1){\times}(t{+}1)$ Vandermonde matrix is invertible, so these rows are linearly independent.

Moreover, invertibility of this matrix implies the standard basis vectors are in their span; in particular, for each $i \in [0, t]$, the vector with a 1 in position i and zeros elsewhere is a linear combination of the row vectors. Translating back to polynomials, this means $\mathsf{X}_i - x_i \in \mathrm{span}(L)$ for all $i \in [0, t]$, as claimed. □

Now, we show that game $\mathbf{G}_3$ aborts if and only if game $\mathbf{G}_4$ aborts.

- If game $\mathbf{G}_3$ aborts, then so does $\mathbf{G}_4$. Suppose game $\mathbf{G}_3$ aborts. Then, during some call to MAP with input P, there exists $P' \neq P$ such that $P(\boldsymbol{x}) = P'(\boldsymbol{x})$ but $P - P' \notin \mathrm{span}(L)$. In game $\mathbf{G}_4$, this check is not performed inside MAP, so MAP would sample a fresh group element. It would not reuse an existing value from $\mathrm{ran}(\tau)$, since reassignment requires $P - P' \in \mathrm{span}(L)$.
  However, once L contains $t + 1$ linearly independent polynomials, Claim 6 implies that any two polynomials agreeing at $\boldsymbol{x}$ must differ by an element in $\mathrm{span}(L)$. Thus, the abort condition $P_1 - P_2 \in \mathrm{span}(L)$ but $\tau(P_1) \neq \tau(P_2)$ is necessarily triggered during one of the DLOG queries. Therefore, whenever game $\mathbf{G}_3$ aborts, then so does $\mathbf{G}_4$.
- If game $\mathbf{G}_4$ aborts, then $\mathbf{G}_3$ aborts. In this case, during some DLOG query, there exist polynomials $P_1, P_2 \in \mathrm{dom}(\tau)$ such that $P_1 - P_2 \in \mathrm{span}(L)$ but $\tau(P_1) \neq \tau(P_2)$. Since $P_1 - P_2 \in \mathrm{span}(L)$, we know that $P_1(\boldsymbol{x}) = P_2(\boldsymbol{x})$. Without loss of generality, assume that P_1 was added to τ before P_2. Then when P_2 was first given to MAP, $P_1 - P_2$ must not have been in the $\mathrm{span}(L)$, otherwise $\tau(P_1)$ would have been reused for P_2. This exactly matches the abort condition of game $\mathbf{G}_3$. Hence, if game $\mathbf{G}_4$ aborts, then game $\mathbf{G}_3$ would already have aborted when P_2 was processed.

Putting both directions together, we have shown that game $\mathbf{G}_3$ aborts if and only if game $\mathbf{G}_4$ aborts. Therefore, moving the inconsistency check from MAP to DLOG does not change the execution of the game and we can conclude:

$$\Pr[\mathbf{G}_3(\mathrm{A}) \Rightarrow 1] = \Pr[\mathbf{G}_4(\mathrm{A}) \Rightarrow 1].$$

$\underline{\mathbf{G}_4 \to \mathbf{G}_5.}$ In game $\mathbf{G}_5$ (Figure 13), we remove all uses of $\boldsymbol{x}$ from the game.

The change from $\mathbf{G}_4$ to $\mathbf{G}_5$ concerns how $(y_{k_1}, \ldots, y_{k_t}, x_0)$ is defined. Note that we assume that an adversary makes exactly t queries to DLOG, which makes the size of the set L equal to $t + 1$ after the last DLOG query in FINALIZE. We want to bound the probability that for each query k to the DLOG, when $\boldsymbol{x} \leftarrow\!\!\$\ \mathcal{C}$, $P_k(\boldsymbol{x}) = y_k$ in game $\mathbf{G}_4$ for some $\mathcal{C}$ using Lemma 1.

As in the hop $\mathbf{G}_1 \to \mathbf{G}_2$, we condition on the knowledge already available to A. However, unlike game $\mathbf{G}_2$, because MAP does not abort on equalities under $\boldsymbol{x}$, the adversary does *not* learn implications of the form $\tau(P) \neq \tau(P') \Rightarrow P(\boldsymbol{x}) \neq P'(\boldsymbol{x})$. Note that $\mathrm{dom}(\tau)$ consists only of polynomials in this game. Thus, at any point in the execution, the set that describes the knowledge of A simplifies to $\mathcal{C} = \bigcap_{j \in [\ell]} \mathcal{Q}_j$ where ℓ is the number of DLOG queries made so far and $\mathcal{C}$ does not include any $\mathcal{D}_i$ terms. For a fixed query k^*, we define $Q_{\ell+1} := P_{k^*} - y_{k^*}$ and use the lemma to compute the probability that $Q_{\ell+1}(\boldsymbol{x}) = 0$.

The lemma has three requirements: (1) $\boldsymbol{x}$ is uniform in $\mathcal{C}$, (2) $\boldsymbol{q}_{\ell+1}$ is independent of $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$, and (3) $\mathcal{Q}_{\ell+1} \cap \mathcal{C} \neq \emptyset$.

$\boldsymbol{x}$ IS UNIFORM IN $\mathcal{C}$. We fix the randomness in the game, which includes all the coins of the challenger and the adversary. Again, consider the transcript of what the adversary receives from the game:

$$G, X_0, \ldots, X_t, A_1, \ldots, A_{q'}, y_{k_1}, \ldots, y_{k_\ell}.$$

Here, $G, X_0, \ldots, X_t$ are output by INITIALIZE, $A_1, \ldots, A_{q'}$ are output by queries to MAP and ADD for some $q' \leq q$, and $y_{k_1}, \ldots, y_{k_\ell}$ are output by DLOG. These values are generated assuming a fixed secret $\boldsymbol{x}$. We want to show that this transcript is distributed identically for an arbitrary $\boldsymbol{v} \in \mathcal{C}$, meaning these values only depend on membership in $\mathcal{C}$ and not on the specific choice of $\boldsymbol{x}$.

First, during INITIALIZE when MAP is called, there are no elements in L, so for both $\boldsymbol{x}$ and $\boldsymbol{v}$, the same group elements are chosen to be $X_0, \ldots, X_t$.

Then, we show by induction that each A_γ and each y_{k_j} is the same for both challenge vectors $\boldsymbol{x}$ and $\boldsymbol{v}$.

- During a call to DLOG, the adversary makes the same queries since everything in the transcript and randomness are the same. Using the same arguments as the first application of the lemma, we deduce $P_{k_j}(\boldsymbol{v}) = y_{k_j} = y'_{k_j}$ and $Q_{k_j} = Q'_{k_j}$.
- Then, during a call to MAP or ADD, the adversary again makes the same queries in both cases. For a call to MAP, either direct or coming from ADD, let P_γ be the input and A_γ be the output.
  - If for all $\hat{P} \in \mathrm{dom}(\tau)$: $P_\gamma - \hat{P} \notin \mathrm{span}(L)$, then the game samples a new A_γ in both cases, which is independent of $\boldsymbol{x}$ or $\boldsymbol{v}$.
  - If there exists $\hat{P} \in \mathrm{dom}(\tau)$: $P_\gamma - \hat{P} \in \mathrm{span}(L)$, then the game with $\boldsymbol{x}$ assigns A_γ to be $\tau(\hat{P})$. For the challenge vector $\boldsymbol{v}$, by the induction hypothesis, all polynomials in $\mathrm{dom}(\tau)$ are the same, so $P_\gamma - \hat{P} \in \mathrm{span}(L)$ still holds. Thus, this game also assigns A_γ to be $\tau(\hat{P})$.

  In both cases, the group element A_γ is the same.

Thus, the adversary always gets the same transcript for any $\boldsymbol{v} \in \mathcal{C}$, so $\boldsymbol{x}$ appears uniform in the set $\mathcal{C}$ from the point of view of the adversary.

$\boldsymbol{q}_{\ell+1}$ IS INDEPENDENT OF $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$. By construction, when query k^* is made, the corresponding polynomial P_{k^*} is linearly independent of the set L at that point. This means that P_{k^*} introduces a new linear relation that is not implied by previous relations in L. Additionally, any nonzero constant polynomial $P_1 = a_1 \neq 0$ is also not contained in $\mathrm{span}(L)$. Therefore, the coefficient vector $\boldsymbol{q}_{\ell+1}$, obtained from P_{k^*} by removing its constant term, is linearly independent of the set $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$.

$\mathcal{Q}_{\ell+1} \cap \mathcal{C} \neq \emptyset$. Since $\boldsymbol{q}_{\ell+1}$ is independent of $(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell)$, the polynomial $Q_{\ell+1}$ cannot be expressed as a linear combination of $\{Q_1, \ldots, Q_\ell\}$. We know that $(P_{k^*} - y_{k^*})(\boldsymbol{x}) = 0$ and $(P_{k_j} - y_{k_j})(\boldsymbol{x}) = 0$ for all $j \in [\ell]$. Thus, the challenge vector $\boldsymbol{x}$ lies in the intersection $\mathcal{Q}_{\ell+1} \cap \mathcal{C}$, and hence it is not empty.

Game $\mathbf{G}_6(E, f, n, t)$

MAP(P): // adversary query can only be a constant
1 If $\exists P' \in \mathrm{dom}(\tau) : P - P' \in \mathrm{span}(L)$: // Pick the first satisfying P'
2 $A \leftarrow \tau(P')$
3 Else:
4 $A \leftarrow$ SAMP()
5 $\tau \leftarrow \tau \cup \{(P, A), (-P, A^{-1})\}$
6 Return $\tau(P)$

SAMP(): // not accessible to the adversary
7 $i \leftarrow 0$
8 While true:
9 $c \leftarrow\!\!\$\ \mathbb{Z}_p$
10 If $c \in \mathrm{ran}(f)$:
11 $A \leftarrow\!\!\$\ f^{-1}(c)$
12 If $A \notin \mathrm{ran}(\tau)$: Return A
13 $i \leftarrow i + 1$
14 If $i = \left\lceil \frac{pd}{p-4q-2t-5} \cdot \ln p \right\rceil$: Abort

Fig. 14. Final game $\mathbf{G}_6$ for the proof of Theorem 3. Changes from $\mathbf{G}_5$ are highlighted in blue (INITIALIZE, ADD, DLOG, and FINALIZE remain the same).

Since each of the conditions above is satisfied, the lemma can be applied as follows:

$$\Pr_{\boldsymbol{x} \leftarrow\!\!\$ \mathcal{C}}[Q_{\ell+1}(\boldsymbol{x}) = 0] \leq \frac{1}{p - m}$$

where $m = 0$ since the adversary A does not get any information about which polynomials do not evaluate to the same value.

There are $t + 1$ such polynomials as the adversary makes t queries, plus one more query made in FINALIZE. The same probability holds for each query made to DLOG, so for each P_{k_j} for $j \in [t]$ we have:

$$\Pr_{\boldsymbol{x} \leftarrow\!\!\$ \mathcal{C}}[P_{k_j}(\boldsymbol{x}) = y_{k_j}] \leq \frac{1}{p}.$$

This means that each y_{k_j} is uniform in $\mathbb{Z}_p$ and can be replaced with any uniform value in $\mathbb{Z}_p$. This is what we do in game $\mathbf{G}_5$. Thus:

$$\Pr[\mathbf{G}_4(\mathrm{A}) \Rightarrow 1] = \Pr[\mathbf{G}_5(\mathrm{A}) \Rightarrow 1] \ .$$

$\mathbf{G}_5 \rightarrow \mathbf{G}_6$. First, we will show that the two games are equivalent when ignoring the abort in line 14 of $\mathbf{G}_6$. In this case, the while loop in SAMP runs until the sampled value $c \in \mathbb{Z}_p$ lies in the range of f and the randomly chosen preimage $A \in f^{-1}(c)$ is not in the range of τ. To argue that this distribution is the same as sampling $A \leftarrow\!\!\$\ E \setminus \mathrm{ran}(\tau)$, we will use the property that the conversion function f is regular. Note that repeatedly sampling a value $c \leftarrow\!\!\$\ \mathbb{Z}_p$ until it lies in the

range of f is the same as uniformly sampling $c \leftarrow\!\!\$\ \mathrm{ran}(f)$. Moreover, if f is regular, then each group element $A \in E$ has the same probability to be sampled in line 11 of Samp (over the random choice of $c \leftarrow\!\!\$\ \mathrm{ran}(f)$). Since we only return if $A \notin \mathrm{ran}(\tau)$, this is the same as sampling $A \leftarrow\!\!\$\ E \setminus \mathrm{ran}(\tau)$.

Next, let us bound the probability that $\mathbf{G}_6$ aborts in line 14 of Samp. Let f be a d-to-1 function, which means that $|\mathrm{ran}(f)| = p/d$. Let $r := 4q+2t+5 \geq |\mathrm{ran}(\tau)|$. Define $\overline{\mathsf{R}}$ as the event that Samp does not return after a single iteration of the while loop. We get:

$$\begin{aligned}
\Pr[\overline{\mathsf{R}}] &= \Pr[\overline{\mathsf{R}} \wedge c \notin \mathrm{ran}(f)] + \Pr[\overline{\mathsf{R}} \wedge c \in \mathrm{ran}(f)] \\
&= \Pr[\overline{\mathsf{R}} \mid c \notin \mathrm{ran}(f)] \cdot \Pr[c \notin \mathrm{ran}(f)] + \Pr[\overline{\mathsf{R}} \mid c \in \mathrm{ran}(f)] \cdot \Pr[c \in \mathrm{ran}(f)] \\
&= 1 \cdot \Pr[c \notin \mathrm{ran}(f)] + \Pr[\overline{\mathsf{R}} \mid c \in \mathrm{ran}(f)] \cdot \Pr[c \in \mathrm{ran}(f)] \\
&= \left(1 - \frac{1}{d}\right) + \Pr[\overline{\mathsf{R}} \mid c \in \mathrm{ran}(f)] \cdot \frac{1}{d} \\
&\leq 1 - \frac{1}{d} + \frac{r}{p} \cdot \frac{1}{d} = 1 - \frac{p-r}{pd}.
\end{aligned}$$

Note that $\Pr[\overline{\mathsf{R}} \mid c \notin \mathrm{ran}(f)] = 1$ since the loop does not return in this case, and $\Pr[\overline{\mathsf{R}} \mid c \in \mathrm{ran}(f)] \leq r/p$ since in this case the group element A is chosen uniformly at random from E due to f being regular.

Let Abort denote the event that Samp aborts in line 14, i.e., that the loop has not returned after $\lambda := \left\lceil \frac{pd}{p-4q-2t-5} \cdot \ln p \right\rceil = \left\lceil \frac{pd}{p-r} \cdot \ln p \right\rceil$ iterations. Using $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$, we get:

$$\begin{aligned}
\Pr[\mathsf{Abort}] = \Pr[\overline{\mathsf{R}}]^\lambda &\leq \left(1 - \frac{p-r}{pd}\right)^\lambda \\
&\leq \exp\left(-\frac{p-r}{pd} \cdot \lambda\right) \\
&\leq \exp\left(-\frac{p-r}{pd} \cdot \frac{pd}{p-r} \cdot \ln p\right) = e^{-\ln p} = \frac{1}{p}.
\end{aligned}$$

Since the adversary can make at most q queries to Samp (by querying Map), the difference between the two games can be bounded as follows:

$$\Pr[\mathbf{G}_5(\mathrm{A}) \Rightarrow 1] \leq \Pr[\mathbf{G}_6(\mathrm{A}) \Rightarrow 1] + \frac{q}{p}\,.$$

LDVR reduction. Finally, we give an adversary B against LDVR that simulates the game $\mathbf{G}_6$ for an adversary A in Figure 15. Specifically, our adversary solves the case of $t_c = t$ as this aligns with full adaptive security where the signing threshold is $t + 1$.

Adversary $\mathrm{B}^{\mathcal{O}}_{p,n,t,t}$

$Q \leftarrow \emptyset$; $L \leftarrow \emptyset$; $\mathsf{ctr} \leftarrow 0$
$\tau \leftarrow \{(0, 1_E)\}$
$G \leftarrow \textsc{Map}(1)$
For $i \in [0, t]$:
  $X_i \leftarrow \textsc{Map}(\mathsf{X}_i)$
$(R, z) \leftarrow \mathrm{A}^{\textsc{Map},\textsc{Add},\textsc{Dlog}}(G, X_0, X_1, \ldots, X_t)$
Let $\mathsf{ctr}^* \in [\mathsf{ctr}]$ be such that $R = A_{\mathsf{ctr}^*}$. If such ctr^* does not exist: Abort
Return (Q, ctr^*)

$\textsc{Map}(P)$:
If $\exists P' \in \mathrm{dom}(\tau) : P - P' \in \mathrm{span}(L)$: // Pick the first satisfying P'
  $A \leftarrow \tau(P')$
Else:
  Parse P as $r + \sum_{i=0}^{t} r_i \mathsf{X}_i$
  Compute $\boldsymbol{\alpha}' \in \mathbb{Z}_p^{t+1}$ such that $r_i = \sum_{j=0}^{t} \boldsymbol{\alpha}'[j] \cdot j^i$ for all $i \in [0, t]$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}' \| \mathbf{0}^{n-t}$
  $A \leftarrow \textsc{Samp}(\boldsymbol{\alpha})$
$\tau \leftarrow \tau \cup \{(P, A), (-P, A^{-1})\}$
Return $\tau(P)$

$\textsc{Samp}(\boldsymbol{\alpha})$:
$i \leftarrow 0$
While true:
  $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
  $c \leftarrow \mathcal{O}(\boldsymbol{\alpha})$ // the same as $c \leftarrow\!\!\$\ \mathbb{Z}_p$
  If $c \in \mathrm{ran}(f)$:
    $A_{\mathsf{ctr}} \leftarrow\!\!\$\ f^{-1}(c)$
    If $A_{\mathsf{ctr}} \notin \mathrm{ran}(\tau)$: Return A_{ctr}
  $i \leftarrow i + 1$
  If $i = \left\lceil \frac{pd}{p-4q-2t-5} \cdot \ln p \right\rceil$: Abort

$\textsc{Add}(c_1, A_1, c_2, A_2)$:
... // the same as in $\mathbf{G}_6$

$\textsc{Dlog}(k)$:
... // the same as in $\mathbf{G}_6$

Fig. 15. The LDVR reduction B that simulates game $\mathbf{G}_6$ for adversary A.

Lemma 2. *Let* A *be an adversary against game* $\mathbf{G}_6$ *for the parameters* (E, f, n, t). *There exists an adversary* B *against LDVR with parameters* (p, n, t, t) *such that*

$$\Pr[\mathbf{G}_6(\mathrm{A}) \Rightarrow 1] \leq \mathbf{Adv}^{\mathrm{ldvr}}_{p,n,t,t}(\mathrm{B}).$$

Proof. Here, adversary B perfectly simulates game $\mathbf{G}_6$ for adversary A. Adversary B simulates everything in game $\mathbf{G}_6$ honestly except the Map oracle. In the LDVR game, the oracle query output is a uniform random value for any input, which is the same distribution as in game $\mathbf{G}_6$ when sampling c inside Map.

In Map, we parse the input polynomial P as $r + \sum_{i=0}^{t} r_i \mathsf{X}_i$. The coefficient values of this polynomial $r_0, \ldots, r_t$ can be interpreted as a vector. Since the vector is in the space $\mathbb{Z}_p^{t+1}$, we can perform a change-of-basis to get a representation in any other basis that spans the same space. In particular, we consider the Vandermonde vectors $\{\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_t\}$ which span $\mathbb{Z}_p^{t+1}$. The new representation is given by $\boldsymbol{V}^{-1} \cdot \boldsymbol{r}$, where $\boldsymbol{V} \in \mathbb{Z}_p^{t+1} \times \mathbb{Z}_p^{t+1}$ is the matrix containing the basis vectors as its columns. Since the vectors are unique Vandermonde vectors, the resulting matrix is invertible, and the new representation vector is uniquely determined and efficiently computable.

Suppose adversary A wins game $\mathbf{G}_6$. Then, we have:

(1) $R \in \mathrm{ran}(\tau)$,
(2) $f(R) \neq 0$,
(3) $\textsc{Map}(z) = \textsc{Add}(1, R, f(R), X_0)$.

Let $P(\mathsf{X}_0, \ldots, \mathsf{X}_t) = r + \sum_{i=0}^{t} r_i \mathsf{X}_i$ denote the first entry in τ such that $\tau(P) = R$, which exists by condition (1).

Note that condition (3) corresponds to $\tau(z) = \tau(P(\mathsf{X}_0, \ldots, \mathsf{X}_t) + f(R)\mathsf{X}_0)$. Moreover, $\tau(P_1) = \tau(P_2)$ for two polynomials P_1, P_2 only if $P_1 - P_2 \in \mathrm{span}(L)$ (even in the case $P_1 = P_2$, since the zero polynomial is always contained in the span of L). We conclude that

$$P(\mathsf{X}_0, \ldots, \mathsf{X}_t) + f(R)\mathsf{X}_0 - z \in \mathrm{span}(L).$$

When excluding the constant term from the polynomials, this means that

$$\begin{pmatrix} r_0 + f(R) \\ r_1 \\ \vdots \\ r_t \end{pmatrix} \in \mathrm{span}(\{\boldsymbol{v}_i\}_{i \in Q}). \tag{13}$$

Since Vandermonde vectors $\boldsymbol{v}_i$ are linearly independent and $0 \notin Q$, we cannot have a multiple of $\boldsymbol{v}_0$ in the span of $\mathrm{span}(\{\boldsymbol{v}_i\}_{i \in Q})$. If P were a constant polynomial, then the vector on the left-hand side of Eq. (13) would be $f(R) \cdot \boldsymbol{v}_0$. Together with $f(R) \neq 0$ from condition (2), this implies that P cannot be constant.

Consider the point that R is added to the range of τ. Any "fresh" input R to Add will be associated to a constant polynomial. Since P is non-constant, R must have been added to τ during a call to Map. Thus, R was sampled using Samp and there exists a $\mathsf{ctr}^* \in [\mathsf{ctr}]$ such that $R = A_{\mathsf{ctr}^*}$.

Let $\boldsymbol{\alpha}^* \in \mathbb{Z}_p^{n+1}$ denote the input to the LDVR oracle on query ctr^* and let c^* denote the corresponding output, i.e., $c^* := \mathcal{O}(\boldsymbol{\alpha}^*)$. We have $c^* = f(R)$ because $R \in f^{-1}(c^*)$ by construction of the Samp procedure. Moreover, $\boldsymbol{\alpha}^*$ satisfies $r_i = \sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \cdot j^i$ for all $i \in [0, t]$.

Now, the output (Q, ctr^*) of B is an LDVR solution if and only if

$$\boldsymbol{w} := c^* \boldsymbol{v}_0 + \sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \cdot \boldsymbol{v}_j \in \mathrm{span}(\{\boldsymbol{v}_i\}_{i \in Q}).$$

Expanding the sum,

$$\sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \cdot \boldsymbol{v}_j = \sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \cdot \begin{pmatrix} 1 \\ j \\ \vdots \\ j^t \end{pmatrix} = \begin{pmatrix} \sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \\ \sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \cdot j \\ \vdots \\ \sum_{j=0}^{n} \boldsymbol{\alpha}^*[j] \cdot j^t \end{pmatrix},$$

together with Eq. (13) shows that

$$\boldsymbol{w} = \begin{pmatrix} r_0 + f(R) \\ r_1 \\ \vdots \\ r_t \end{pmatrix} \in \mathrm{span}(\{\boldsymbol{v}_i\}_{i \in Q}),$$

which is exactly the LDVR winning condition for output (Q, ctr^*). This concludes the proof. □

FINAL BOUND. We are now ready to conclude the proof of Theorem 3. Accumulating the differences from $\mathbf{G}_0$ through $\mathbf{G}_6$ and applying Lemma 2, we obtain:

$$\begin{aligned} \mathbf{Adv}^{\text{ec-ggm-vcdl}}_{E,f,n,t}(\mathrm{A}) &\leq \frac{1}{p - 4q - 2t - 5} + \frac{q^2}{p - q^2} + 0 + 0 + 0 + \frac{q}{p} + \mathbf{Adv}^{\text{ldvr}}_{p,n,t,t}(\mathrm{B}) \\ &\leq \mathbf{Adv}^{\text{ldvr}}_{p,n,t,t}(\mathrm{B}) + \frac{1}{p - 4q - 2t - 5} + \frac{q^2}{p - q^2} + \frac{q}{p}. \end{aligned}$$

This is the bound in the theorem statement and concludes the proof. □

6 Adaptive Multi-User Security of Schnorr Signatures

In this section, we establish *tight adaptive multi-user unforgeability* for *basic* Schnorr signatures in the ROM, under a new *standard-basis circular discrete-logarithm* (SB-CDL) assumption.

To obtain this result, we develop a broader framework that captures VCDL, and similarly LDVR, as special cases. Specifically, we introduce the *interactive circular discrete-logarithm* (ICDL) assumption and, correspondingly, the *generalized low-dimensional vector representation* (gLDVR) problem. These frameworks abstract away the Vandermonde vectors, replacing them with an arbitrary family of vectors satisfying a simple non-degeneracy condition.

We identify another simple and natural special case of ICDL, which we call the *standard-basis circular discrete-logarithm* (SB-CDL) assumption. This instantiation enables a tight reduction for adaptive multi-user Schnorr signatures. Beyond this application, we believe that viewing VCDL and LDVR through the ICDL and gLDVR frameworks will aid their future study.

6.1 Adaptive Multi-User Unforgeability

MOTIVATION. Separately from threshold settings, deployments of basic Schnorr signatures are inherently multi-user, *e.g.*, in TLS, Signal, or Bitcoin, where verifiers process signatures under a large and evolving set of public keys over long periods of time. In such settings, key compromise events are unavoidable and may occur adaptively as the system evolves. This motivates targeting *adaptive multi-user unforgeability*.

THE DEFINITION. We define (strong) adaptive multi-user existential unforgeability under chosen-message attack (AMU-EUF-CMA). We define it specifically for Schnorr signatures, which is the only scheme we consider here, and because the properties of Schnorr signatures allow us to substantially simplify the definition. Let $\mathsf{Sch}[\mathbb{G}, H]$ be the Schnorr signature scheme. For an adversary A and $n \in \mathbb{N}$, we define its *AMU-EUF-CMA advantage* against $\mathsf{Sch}[\mathbb{G}, H]$ as

$$\mathbf{Adv}^{\text{amu-euf-cma}}_{\mathsf{Sch}[\mathbb{G},H],n}(\mathrm{A}) \;=\; \Pr\big[\mathbf{G}^{\text{amu-euf-cma}}_{\mathsf{Sch}[\mathbb{G},H],n}(\mathrm{A}) \Rightarrow 1\big],$$

where the game is given in Figure 16.

Game $\mathbf{G}^{\text{amu-euf-cma}}_{\mathsf{Sch}[\mathbb{G},H],n}$

INITIALIZE:
1 $\mathsf{CS} \leftarrow \emptyset$
2 For $i \in [n]$:
3 $\quad (X_i, x_i) \leftarrow\!\!\$\ \mathsf{Sch.K}$; $S_i \leftarrow \emptyset$
4 Return $(X_i)_{i\in[n]}$

SIGNO(k, m): // $k \in [n] \setminus \mathsf{CS}$
5 $(R, z) \leftarrow\!\!\$\ \mathsf{Sch.S}(x_k, m)$
6 $S_k \leftarrow S_k \cup \{(m, (R, z))\}$
7 Return (R, z)

CORRUPT(k): // $k \in [n] \setminus \mathsf{CS}$
8 $\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$
9 Return x_k

FINALIZE$(i, m, (R, z))$:
10 If $i \in \mathsf{CS} \ \vee\ (m, (R, z)) \in S_i$: Return 0
11 Return $\mathsf{Sch.V}(X_i, m, (R, z))$

Fig. 16. Game defining AMU-EUF-CMA for Schnorr signatures.

Remark 3. The following relationship between adaptive multi-user EUF-CMA security and standard single-user EUF-CMA security is immediate. For *any* signature scheme DS, given an AMU adversary A, we can construct an EUF-CMA adversary B that guesses an index $i \leftarrow\!\!\$\ [n]$, embeds its challenge verification

key as vk_i, and generates the remaining key pairs honestly. The adversary B simulates the AMU game for A, aborting if party i gets corrupted, and outputs any forgery produced for user i. Thus,

$$\mathbf{Adv}^{\text{amu-euf-cma}}_{\mathsf{DS},n}(\mathrm{A}) \le n \cdot \mathbf{Adv}^{\text{euf-cma}}_{\mathsf{DS}}(\mathrm{B}).$$

The running times of A and B are essentially the same. However, this factor-n loss is prohibitive in large-scale applications such as those mentioned above, motivating dedicated AMU-EUF-CMA analyses of specific schemes such as Schnorr signatures.

6.2 Standard-Basis CDL

We introduce a variant of VCDL that we call the *standard-basis circular discrete-logarithm* (SB-CDL) assumption. Later, we show that both VCDL and SB-CDL can be viewed as special cases of a unified assumption framework.

To formally define SB-CDL, let $\mathbb{G}$ be a cyclic group of prime order $p = |\mathbb{G}|$ generated by g. Let $f\colon \mathbb{G} \to \mathbb{Z}_p$ be an efficiently computable function. For $n \in \mathbb{N}$, the *standard-basis circular discrete-logarithm* (SB-CDL) problem is defined via the game in Figure 17. For an adversary A, we let its *SB-CDL advantage* be

$$\mathbf{Adv}^{\text{sb-cdl}}_{\mathbb{G},g,f,n}(\mathrm{A}) \;=\; \Pr\big[\mathbf{G}^{\text{sb-cdl}}_{\mathbb{G},g,f,n}(\mathrm{A}) \Rightarrow 1\big].$$

Game $\mathbf{G}^{\text{sb-cdl}}_{\mathbb{G},g,f,n}$

INITIALIZE:
1 $Q \leftarrow \emptyset$
2 For $i \in [n]$:
3 $\quad x_i \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ X_i \leftarrow g^{x_i}$
4 Return $(X_1, \ldots, X_n)$

DLOG(k): // $k \in [n]$
5 $Q \leftarrow Q \cup \{k\}$
6 Return x_k

FINALIZE($i, (R, z)$): // $i \in [n] \setminus Q$
7 Return $\big(f(R) \neq 0 \ \wedge\ g^z = R \cdot X_i^{f(R)}\big)$

Fig. 17. Game defining the standard-basis circular discrete-log (SB-CDL) problem.

HARDNESS OF SB-CDL IN THE EC-GGM. We sketch a proof of hardness of SB-CDL *unconditionally* (*i.e.*, without assuming LDVR) in the EC-GGM for any efficiently computable, approximately regular conversion function f. For this, we adapt the proof of Theorem 3; it turns out that, in the standard-basis case, the analysis is substantially easier than for Vandermonde vectors. As before, we

represent each group element internally by a degree-1 polynomial P over formal variables $\mathsf{X}_1, \ldots, \mathsf{X}_n$, and we maintain a list L of linear constraints capturing the adversary's knowledge: for each $i \in Q$ we add $\mathsf{X}_i - x_i$ to L. Two polynomials P, P' are represented by the same group element iff $P - P' \in \mathrm{span}(L)$.

Now, fix an SB-CDL adversary A in the EC-GGM. Consider the final symbolic game in which every fresh encoding corresponds to a fresh uniform point in E subject to the consistency rule above. Suppose A outputs $(k, (R, z))$ and wins, *i.e.*,

$$f(R) \neq 0 \quad \text{and} \quad \textsc{Map}(z) = \textsc{Add}(1, R, f(R), X_k)$$

with $k \notin Q$. Let P be the (first) polynomial in $\mathrm{dom}(\tau)$ with $\tau(P) = R$. Writing

$$P = r_0 + \sum_{i=1}^{n} r_i \mathsf{X}_i,$$

the verification equation implies

$$P + f(R)\mathsf{X}_k - z \in \mathrm{span}(L).$$

Isolating the coefficient of X_k and using that $k \notin Q$ yields $r_k + f(R) = 0 \pmod p$. Additionally, $r_k \neq 0$, since otherwise $f(R) = 0$. Thus, the winning condition requires $f(R) = -r_k$, and for uniformly random $R \in E$, this event occurs with probability at most

$$\max_{y \in \mathbb{Z}_p^*} \frac{|f^{-1}(y)|}{p}.$$

A union bound over all Map queries completes the proof sketch.

6.3 Main Result

Now, we give a tight security reduction from adaptive multi-user security of basic Schnorr signatures to the SB-CDL assumption.

Theorem 4 (Adaptive Multi-User Security of Schnorr). *Let $\mathbb{G}$ be a group of prime order p with generator g, and $n \in \mathbb{N}$. Let* A *be an adversary against n-user AMU-EUF-CMA security of Schnorr signatures $\mathsf{Sch}[\mathbb{G}]$ in the ROM that makes at most q_s signing queries, q_h hash queries, and q_c corruption queries. Let $f\colon \mathbb{G} \to \mathbb{Z}_p$ be any efficient function. Then there exists an adversary* B *against SB-CDL for parameters $\mathbb{G}, g, f, n$ with running time comparable to* A *that makes at most q_c queries to its discrete-log oracle such that*

$$\mathbf{Adv}^{\text{amu-euf-cma}}_{\mathsf{Sch}[\mathbb{G}],n}(\mathrm{A}) \leq \mathbf{Adv}^{\text{sb-cdl}}_{\mathbb{G},g,f,n}(\mathrm{B}) + \frac{q_s(q_s + q_h) + |f^{-1}(0)|}{p}\,.$$

Proof. This proof follows the techniques of CFOS [CFOS25], carefully lifting their analysis of EUF-CMA of Schnorr signatures in the ROM under CDL to AMU-EUF-CMA in the ROM under SB-CDL. Consider the sequence of games given in Figure 18. To simplify the reduction, we make the following w.l.o.g. assumption

$\mathbf{G}_0(\mathbb{G}, g, n)$ / $\boxed{\mathbf{G}_1(\mathbb{G}, g, n)}$	$\mathbf{G}_2(\mathbb{G}, g, n)$	$\mathbf{G}_3(\mathbb{G}, g, f, n)$ / $\boxed{\mathbf{G}_4(\mathbb{G}, g, f, n)}$
INITIALIZE:	INITIALIZE:	INITIALIZE:
$\mathsf{T} \leftarrow ()$; $\mathsf{CS} \leftarrow \emptyset$	$\mathsf{T} \leftarrow ()$; $\mathsf{CS} \leftarrow \emptyset$	$\mathsf{T} \leftarrow ()$; $\mathsf{T}' \leftarrow ()$; $\mathsf{CS} \leftarrow \emptyset$
For $i \in [n]$:	For $i \in [n]$:	For $i \in [n]$:
$\quad x_i \leftarrow\!\$\ \mathbb{Z}_p$; $X_i \leftarrow g^{x_i}$	$\quad x_i \leftarrow\!\$\ \mathbb{Z}_p$; $X_i \leftarrow g^{x_i}$	$\quad x_i \leftarrow\!\$\ \mathbb{Z}_p$; $X_i \leftarrow g^{x_i}$
$\quad S_i \leftarrow \emptyset$	$\quad S_i \leftarrow \emptyset$	$\quad S_i \leftarrow \emptyset$
Return $(X_i)_{i \in [n]}$	Return $(X_i)_{i \in [n]}$	Return $(X_i)_{i \in [n]}$
SIGNO(k, m): // $k \notin \mathsf{CS}$	SIGNO(k, m): // $k \notin \mathsf{CS}$	SIGNO(k, m): // $k \notin \mathsf{CS}$
$r \leftarrow\!\$\ \mathbb{Z}_p$; $R \leftarrow g^r$	$z \leftarrow\!\$\ \mathbb{Z}_p$; $c \leftarrow\!\$\ \mathbb{Z}_p$	$z \leftarrow\!\$\ \mathbb{Z}_p$; $c \leftarrow\!\$\ \mathbb{Z}_p$
$\boxed{\text{If } \mathsf{T}[X_k, m, R] \neq \bot:}$	$R \leftarrow g^z / X_k^c$	$R \leftarrow g^z / X_k^c$
$\boxed{\quad \text{Abort}}$	If $\mathsf{T}[X_k, m, R] \neq \bot$:	If $\mathsf{T}[X_k, m, R] \neq \bot$:
$c \leftarrow$ HASHO(X_k, m, R)	$\quad$ Abort	$\quad$ Abort
$z \leftarrow (r + cx_k) \bmod p$	$\mathsf{T}[X_k, m, R] \leftarrow c$	$\mathsf{T}[X_k, m, R] \leftarrow c$
$S_k \leftarrow S_k \cup \{(m, (R, z))\}$	$S_k \leftarrow S_k \cup \{(m, (R, z))\}$	$S_k \leftarrow S_k \cup \{(m, (R, z))\}$
Return (R, z)	Return (R, z)	Return (R, z)
CORRUPT(k): // $k \notin \mathsf{CS}$	CORRUPT(k): // $k \notin \mathsf{CS}$	CORRUPT(k): // $k \notin \mathsf{CS}$
$\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$	$\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$	$\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$
Return x_k	Return x_k	Return x_k
HASHO(X_k, m, R):	HASHO(X_k, m, R):	HASHO(X_k, m, R):
If $\mathsf{T}[X_k, m, R] = \bot$:	If $\mathsf{T}[X_k, m, R] = \bot$:	If $\mathsf{T}[X_k, m, R] = \bot$:
$\quad c \leftarrow\!\$\ \mathbb{Z}_p$	$\quad c \leftarrow\!\$\ \mathbb{Z}_p$	$\quad a, b \leftarrow\!\$\ \mathbb{Z}_p$
$\quad \mathsf{T}[X_k, m, R] \leftarrow c$	$\quad \mathsf{T}[X_k, m, R] \leftarrow c$	$\quad R' \leftarrow R \cdot X_k^a \cdot g^b$
Return $\mathsf{T}[X_k, m, R]$	Return $\mathsf{T}[X_k, m, R]$	$\quad c \leftarrow (f(R') + a) \bmod p$
		$\quad \mathsf{T}'[X_k, m, R] \leftarrow (a, b)$
		$\quad \mathsf{T}[X_k, m, R] \leftarrow c$
		Return $\mathsf{T}[X_k, m, R]$
FINALIZE$(i, m, (R, z))$:	FINALIZE$(i, m, (R, z))$:	FINALIZE$(i, m, (R, z))$:
If $i \in \mathsf{CS}$: Return 0	If $i \in \mathsf{CS}$: Return 0	If $i \in \mathsf{CS}$: Return 0
If $(m, (R, z)) \in S_i$:	If $(m, (R, z)) \in S_i$:	If $(m, (R, z)) \in S_i$:
$\quad$ Return 0	$\quad$ Return 0	$\quad$ Return 0
$c \leftarrow$ HASHO(X_i, m, R)	$c \leftarrow$ HASHO(X_i, m, R)	$c \leftarrow$ HASHO(X_i, m, R)
Return $(g^z = R \cdot X_i^c)$	Return $(g^z = R \cdot X_i^c)$	$\boxed{\text{If } \mathsf{T}'[X_i, m, R] \neq \bot:}$
		$\boxed{\quad (a, b) \leftarrow \mathsf{T}'[X_i, m, R]}$
		$\boxed{\quad R' \leftarrow R \cdot X_i^a \cdot g^b}$
		$\boxed{\quad \text{If } f(R') = 0\text{: Abort}}$
		Return $(g^z = R \cdot X_i^c)$

Fig. 18. Games for the proof of Theorem 4. Changes are highlighted in blue. Boxed games additionally include the boxed code.

on the adversary A: whenever INITIALIZE returns the same verification keys $X_i = X_j$ for two different users $i \neq j$, then A queries CORRUPT on one of the users and then returns an arbitrary honestly computed signature for the other user as its forgery.

$\underline{\mathbf{G}_0 \to \mathbf{G}_1.}$ Game $\mathbf{G}_1$ aborts in SIGNO(k, m) if $\mathsf{T}[X_k, m, R] \neq \perp$. For any fixed (k, m), the value $R = g^r$ is uniform in $\mathbb{G}$, while the table contains at most $q_s + q_h$ defined entries of the form $(X_k, m, \cdot)$. Thus, the abort probability for this signing query is at most $(q_s + q_h)/p$. A union bound over at most q_s signing queries yields

$$\Pr[\,\mathbf{G}_0(\mathrm{A}) \Rightarrow 1\,] \leq \Pr[\,\mathbf{G}_1(\mathrm{A}) \Rightarrow 1\,] + \frac{q_s(q_s + q_h)}{p}.$$

$\underline{\mathbf{G}_1 \to \mathbf{G}_2.}$ In $\mathbf{G}_2$ we modify SIGNO to avoid using x_k by sampling $z, c \leftarrow\!\!\$\ \mathbb{Z}_p$ and setting $R \leftarrow g^z / X_k^c$, and we define $\mathsf{T}[X_k, m, R] \leftarrow c$. Conditioned on not aborting, the pair (R, z) is distributed exactly as in $\mathbf{G}_1$, since in both games c is uniform and R is uniform subject to $g^z = R \cdot X_k^c$. Hence,

$$\Pr[\,\mathbf{G}_1(\mathrm{A}) \Rightarrow 1\,] = \Pr[\,\mathbf{G}_2(\mathrm{A}) \Rightarrow 1\,].$$

$\underline{\mathbf{G}_2 \to \mathbf{G}_3.}$ In $\mathbf{G}_3$ we change how a fresh hash value $\mathsf{T}[X_k, m, R]$ is defined: instead of sampling $c \leftarrow\!\!\$\ \mathbb{Z}_p$, we sample $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$, set $R' \leftarrow R \cdot X_k^a \cdot g^b$, and return $c := (f(R') + a) \bmod p$. Since a and b are uniform in $\mathbb{Z}_p$, and R' is independent of a (over the choice of b), the value $(f(R') + a) \bmod p$ is uniform in $\mathbb{Z}_p$ as well. Therefore,

$$\Pr[\,\mathbf{G}_2(\mathrm{A}) \Rightarrow 1\,] = \Pr[\,\mathbf{G}_3(\mathrm{A}) \Rightarrow 1\,].$$

$\underline{\mathbf{G}_3 \to \mathbf{G}_4.}$ Game $\mathbf{G}_4$ additionally aborts when the input $(i, m, (R, z))$ to FINALIZE is such that $\mathsf{T}'[X_i, m, R] \neq \perp$ and $f(R') = 0$, where $R' \leftarrow R \cdot X_i^a \cdot g^b$ and $(a, b) \leftarrow \mathsf{T}'[X_i, m, R]$. We first argue that any execution in which A outputs a valid forgery $(i, m, (R, z))$ satisfying

(i) $(m, (R, z)) \notin S_i$, and
(ii) $g^z = R \cdot X_i^c$ with $c \leftarrow \mathsf{T}[X_i, m, R]$,

necessarily has $\mathsf{T}'[X_i, m, R] \neq \perp$. Suppose instead that $\mathsf{T}[X_i, m, R]$ was set by SIGNO rather than HASHO. When SIGNO(i, m) programs $\mathsf{T}[X_i, m, R] \leftarrow c$, it also records $(m, (R, z')) \in S_i$, where z' is the unique scalar satisfying $g^{z'} = R \cdot X_i^c$. Since by (ii), $z' = z$, this however contradicts (i). Therefore any valid forgery $(i, m, (R, z))$ for which FINALIZE reaches that check must have $\mathsf{T}[X_i, m, R]$ defined by HASHO, and in particular $\mathsf{T}'[X_i, m, R] \neq \perp$. Since $R' = R \cdot X_i^a \cdot g^b$ with $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$, the element R' is uniform in $\mathbb{G}$ and, in particular, independent of (X_i, m, R) and $c = (f(R') + a) \bmod p$. Thus, the probability that $f(R') = 0$ is exactly $|f^{-1}(0)|/p$, and we get

$$\Pr[\,\mathbf{G}_3(\mathrm{A}) \Rightarrow 1\,] \leq \Pr[\,\mathbf{G}_4(\mathrm{A}) \Rightarrow 1\,] + \frac{|f^{-1}(0)|}{p}.$$

Adversary $\mathrm{B}^{\mathrm{DLOG}}_{\mathbb{G},g,f,n}(X_1,\ldots,X_n)$

$\mathsf{T} \leftarrow ()$; $\mathsf{T}' \leftarrow ()$; $\mathsf{CS} \leftarrow \emptyset$
For $i \in [n]$: $S_i \leftarrow \emptyset$
$(i, m, (R, z)) \leftarrow\!\!\$\ \mathrm{A}^{\textsc{SignO},\textsc{Corrupt},\textsc{HashO}}(X_1,\ldots,X_n)$
If $i \in \mathsf{CS} \vee (m,(R,z)) \in S_i$: Abort
$c \leftarrow \textsc{HashO}(X_i, m, R)$
$(a, b) \leftarrow \mathsf{T}'[X_i, m, R]$
$R' \leftarrow R \cdot X_i^a \cdot g^b$
If $f(R') = 0$: Abort
Return $\big(i, (R', (z + b) \bmod p)\big)$

$\textsc{SignO}(k, m)$:
... // the same as in $\mathbf{G}_4$

$\textsc{Corrupt}(k)$:
$\mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$
Return $\textsc{Dlog}(k)$

$\textsc{HashO}(X_k, m, R)$:
... // the same as in $\mathbf{G}_4$

Fig. 19. The SB-CDL reduction B used in the proof of Theorem 4.

Reduction to SB-CDL. Finally, let B be the SB-CDL adversary in Figure 19, which perfectly simulates $\mathbf{G}_4$ for A, answering each corruption query by forwarding it to $\textsc{Dlog}(\cdot)$. We show that if $\mathbf{G}_4$ outputs 1 then B wins the SB-CDL game. Let $(i, m, (R, z))$ be the valid forgery output by A in $\mathbf{G}_4$. As shown in the analysis of $\mathbf{G}_3 \to \mathbf{G}_4$ above, $\mathsf{T}[X_i, m, R]$ must have been defined on a call to HashO. Letting $a, b \in \mathbb{Z}_p$ be the values sampled by that execution of HashO and stored in $\mathsf{T}'[X_i, m, R]$, we have

$$R' = R \cdot X_i^a \cdot g^b, \qquad c = (f(R') + a) \bmod p,$$

and $f(R') \neq 0$. Since the forgery is valid, we have $i \notin \mathsf{CS}$, i.e., B did not query Dlog on index i, and $g^z = R \cdot X_i^c$. Substituting terms and multiplying both sides by g^b we get

$$g^{z+b} = R' \cdot X_i^{f(R')},$$

which means $(i, (R', z + b))$ as returned by B is an SB-CDL solution. Hence,

$$\Pr[\,\mathbf{G}_4(\mathrm{A}) \Rightarrow 1\,] \leq \mathbf{Adv}^{\text{sb-cdl}}_{\mathbb{G},g,f,n}(\mathrm{B}).$$

Combining this with the above game hops gives the theorem. □

Remark 4. Using the form of Schnorr signatures that prepends the public key X_i to hash queries appears essential for our tight reduction. To see why, suppose we drop key-prefixing, so the hash table is keyed by (m, R) rather than (X_i, m, R). Consider the following scenario in the reduction B: a signing query $\textsc{SignO}(k, m)$

produces a nonce R and programs $\mathsf{T}[m, R] \leftarrow c$. Later, the adversary A outputs a valid forgery $(i, m, (R, z'))$ for a different user $i \neq k$, reusing the same pair (m, R). Since $\mathsf{T}[m, R]$ was set by SignO rather than HashO, the auxiliary values (a, b) needed by B for extracting an SB-CDL solution were never defined. Whether tight AMU-EUF-CMA security for Schnorr signatures without key-prefixing is achievable under a natural assumption remains an interesting open question.

6.4 The ICDL and gLDVR Frameworks

We introduce a general framework or family of assumptions capturing both VCDL and SB-CDL as special cases, which we call the *interactive circular discrete-logarithm* (ICDL) assumption. In this way, we move beyond viewing VCDL and SB-CDL as construction-specific and isolate the core algebraic structure they both rely on. Accordingly, ICDL abstracts away the Vandermonde and standard basis vectors in these assumptions and replaces them with an arbitrary family of vectors satisfying a simple non-degeneracy condition. Moreover, ICDL is parameterized by a *class of adversaries* to capture fixed vs. flexible targets.

ICDL. Let $\mathbb{G}$ be a cyclic group of prime order $p = |\mathbb{G}|$ generated by g, and let $f\colon \mathbb{G} \to \mathbb{Z}_p$ be an efficiently computable conversion function. Let $n, t \in \mathbb{N}$ where $t < n$, and let

$$V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\} \qquad \text{with} \qquad \boldsymbol{v}_k \in \mathbb{Z}_p^t.$$

We require that V satisfies the following *non-degeneracy condition*: for every index $k \in [n]$ and every subset $S \subseteq [n] \setminus \{k\}$ with $|S| = t - 1$,

$$\boldsymbol{v}_k \notin \operatorname{span}\big(\{\boldsymbol{v}_j\}_{j \in S}\big).$$

Equivalently, every subset of t vectors from V is linearly independent. As mentioned above, ICDL is parameterized by an *adversary class* $\mathcal{A}$, which specifies which target indices may be selected by the adversary. For $\mathrm{A} \in \mathcal{A}$, we let its *ICDL advantage* be

$$\mathbf{Adv}^{\mathrm{icdl},\mathcal{A}}_{\mathbb{G},g,f,V,n,t}(\mathrm{A}) \;=\; \Pr\big[\mathbf{G}^{\mathrm{icdl}}_{\mathbb{G},g,f,V,n,t}(\mathrm{A}) \Rightarrow 1\big],$$

where the ICDL game is defined in Figure 20.

Why non-degeneracy is needed. The non-degeneracy condition ensures that the oracle response associated with any target index k cannot be computed from oracle responses to $t-1$ other indices. If this condition is violated, then the ICDL assumption becomes false. Indeed, suppose there exist a target index $k \in [n]$, a subset $S \subseteq [n] \setminus \{k\}$ with $|S| = t - 1$, and coefficients $\{\alpha_j\}_{j \in S} \subset \mathbb{Z}_p$ such that

$$\boldsymbol{v}_k = \sum_{j \in S} \alpha_j \boldsymbol{v}_j.$$

An adversary may then query $\textsc{Dlog}(j)$ for each $j \in S$, obtaining

$$y_j = \langle \boldsymbol{v}_j, (x_1, \ldots, x_t) \rangle.$$

Game $\mathbf{G}^{\mathrm{icdl}}_{\mathbb{G},g,f,V,n,t}$

INITIALIZE:
1 $Q \leftarrow \emptyset$
2 For $i \in [t]$:
3 $\quad x_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $X_i \leftarrow g^{x_i}$
4 Return $(X_1, \ldots, X_t)$

DLOG(k): // $k \in [n]$, $|Q| < t-1$
5 $Q \leftarrow Q \cup \{k\}$
6 $y_k \leftarrow \sum_{i\in[t]} \boldsymbol{v}_k[i] \cdot x_i$
7 Return y_k

FINALIZE($k, (R, z)$): // $k \in [n] \setminus Q$
8 $Y_k \leftarrow \prod_{i\in[t]} X_i^{\boldsymbol{v}_k[i]}$
9 Return $\big(f(R) \neq 0 \ \wedge\ g^z = R \cdot Y_k^{f(R)}\big)$

Fig. 20. Game defining the interactive circular discrete-logarithm (ICDL) problem.

By linearity, it can compute

$$\langle \boldsymbol{v}_k, (x_1, \ldots, x_t)\rangle = \sum_{j\in S} \alpha_j y_j$$

without ever querying DLOG(k). Let

$$Y_k := g^{\langle \boldsymbol{v}_k, (x_1,\ldots,x_t)\rangle}.$$

The adversary may now choose $r \leftarrow\!\!\$\ \mathbb{Z}_p$, set $R \leftarrow g^r$ (so $f(R) \neq 0$ with high probability), and output (k, z, R) where

$$z := r + f(R) \cdot \langle \boldsymbol{v}_k, (x_1, \ldots, x_t)\rangle \bmod p.$$

This satisfies

$$g^z = g^r \cdot \big(g^{\langle \boldsymbol{v}_k, (x_1,\ldots,x_t)\rangle}\big)^{f(R)} = R \cdot Y_k^{f(R)},$$

and hence constitutes a valid solution for the ICDL game.

SPECIAL CASE: VCDL. The VCDL assumption with parameters $n, t \in \mathbb{N}$ with $t < n$ is recovered as a special case of fixed-target ICDL instantiated with $t{+}1$ in place of t, by taking

$$\boldsymbol{v}_j \ =\ (1,\, j-1,\, (j-1)^2,\, \ldots,\, (j-1)^t) \ \in\ \mathbb{Z}_p^{t+1} \quad \text{for all } j \in [n+1],$$

with fixed target $k = 1$. The $t+1$ ICDL scalars $(x_1, \ldots, x_{t+1})$ correspond to the VCDL scalars $(x_0, \ldots, x_t)$; the queryable indices $[n+1] \setminus \{1\}$ correspond to VCDL's query set $[n]$; and $Y_1 = \prod_{i\in[t+1]} X_i^{\boldsymbol{v}_1[i]} = X_1$ corresponds to VCDL's X_0. The fixed-target winning condition $g^z = R \cdot Y_1^{f(R)}$ is then exactly the VCDL

winning condition. The non-degeneracy condition holds since any distinct $t+1$ Vandermonde vectors are linearly independent.

Special case: SB-CDL. We observe that SB-CDL with parameter $n \in \mathbb{N}$ is exactly the ICDL game with parameters n and $t = n$ when taking V with the *standard-basis* vector family. Namely, let

$$V_{\mathsf{sb}} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\},$$

where $\boldsymbol{e}_j \in \mathbb{Z}_p^n$ is the j-th standard basis vector, i.e., $\boldsymbol{e}_j[i] = 1$ if $i = j$ and $\boldsymbol{e}_j[i] = 0$ otherwise. For this choice of V, a discrete-log query $\textsc{Dlog}(k)$ returns

$$\sum_{i \in [t]} \boldsymbol{e}_k[i] \cdot x_i \;=\; x_k,$$

so the oracle reveals individual coordinates. Moreover, for any target k we have

$$Y_k \;=\; \prod_{i \in [t]} X_i^{\boldsymbol{e}_k[i]} \;=\; X_k,$$

and the ICDL winning condition is then:

$$f(R) \neq 0 \;\wedge\; g^z = R \cdot X_k^{f(R)},$$

which is exactly the SB-CDL winning condition. Then, defining $\mathcal{A}_{\mathsf{flex}}$ to be the class of adversaries that may output an *arbitrary* target index $k \in [n]$, we can view SB-CDL advantage as:

$$\mathbf{Adv}^{\text{sb-cdl}}_{\mathbb{G},g,f,n}(\mathrm{A}) \;:=\; \mathbf{Adv}^{\text{icdl},\,\mathcal{A}_{\mathsf{flex}}}_{\mathbb{G},g,f,V_{\mathsf{sb}},n,n-1}(\mathrm{A}).$$

The non-degeneracy condition holds trivially for V_{sb}, since no standard basis vector lies in the span of any others. Note that relative to $\mathcal{A}_{\mathsf{fixed}}$, the assumption collapses to the (non-interactive) CDL assumption.

Generalized LDVR. Toward studying hardness of ICDL and special cases, we generalize LDVR along the same lines. Fix a prime p and integers $n, t, t_c \in \mathbb{N}$ with $t < n$ and $t_c \leq t$. Let

$$V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\} \qquad \text{where} \qquad \boldsymbol{v}_j \in \mathbb{Z}_p^t.$$

We require that V satisfies the same *non-degeneracy* condition as in ICDL. For $\mathrm{A} \in \mathcal{A}$, we let its *gLDVR advantage* be

$$\mathsf{Adv}^{\text{gldvr},\,\mathcal{A}}_{p,n,t,t_c,V}(\mathrm{A}) \;=\; \Pr\big[\mathbf{G}^{\text{gldvr}}_{p,n,t,t_c,V}(\mathrm{A}) \Rightarrow 1\big],$$

where the gLDVR game is defined in Figure 21.

Special case: LDVR. The LDVR assumption can be recovered as a special case of fixed-target gLDVR instantiated with $t+1$ in place of t, by taking

$$\boldsymbol{v}_j \;:=\; (1,\, j-1,\, (j-1)^2,\, \ldots,\, (j-1)^t) \;\in\; \mathbb{Z}_p^{t+1} \quad \text{for all } j \in [n+1],$$

with fixed target $k = 1$. We have already seen that this Vandermonde family satisfies the required non-degeneracy condition.

Game $\mathbf{G}^{\text{gldvr}}_{p,n,t,t_c,V}$

INITIALIZE:

1 $\mathsf{ctr} \leftarrow 0$

$\mathcal{O}(\boldsymbol{\alpha})$: // $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$

2 $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

3 $\boldsymbol{\alpha}_{\mathsf{ctr}} \leftarrow \boldsymbol{\alpha}$

4 $c_{\mathsf{ctr}} \leftarrow\!\!\$\ \mathbb{Z}_p$

5 Return c_{ctr}

FINALIZE(k, CS, i): // $\mathsf{CS} \subseteq [n] \setminus \{k\},\ |\mathsf{CS}| \leq t_c,\ i \in [1, \mathsf{ctr}]$

6 $\beta_i[k] \leftarrow \boldsymbol{\alpha}_i[k] + c_i\ ;\ \beta_i[j] \leftarrow \boldsymbol{\alpha}_i[j]$ for $j \in [n] \setminus \{k\}$

7 $\boldsymbol{w} \leftarrow \sum_{j\in[n]} \beta_i[j] \cdot \boldsymbol{v}_j$

8 Return $\big(\boldsymbol{w} \in \mathrm{span}(\{\boldsymbol{v}_j\}_{j\in\mathsf{CS}})\big)$

Fig. 21. Game defining the generalized LDVR (gLDVR) problem.

6.5 Relationship Between ICDL and gLDVR, and Open Questions

We study the relationship between ICDL and gLDVR, and its consequences for the hardness of both assumptions. We show that for any fixed non-degenerate vector family $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\} \subseteq \mathbb{Z}_p^t$, the *fixed-target* variants of ICDL and gLDVR are tightly equivalent: fixed-target ICDL implies fixed-target gLDVR in the standard model, and conversely fixed-target gLDVR implies fixed-target ICDL in the EC-GGM. This generalizes the equivalence between VCDL and LDVR established in Theorems 2 and 3, showing it is not a special property of Vandermonde vectors but a consequence of non-degeneracy alone. We give proof sketches below; throughout, we fix a target index k.

ICDL $\Rightarrow$ gLDVR. We directly generalize the proof of Theorem 2, replacing the Vandermonde vectors throughout by V. The target group element X_0 in that proof is replaced by

$$Y_k \ := \ \prod_{i\in[t]} X_i^{\boldsymbol{v}_k[i]}\,.$$

In the simulation of the gLDVR oracle $\mathcal{O}$, the reduction programs the group element using Y_k, setting

$$R_\ell \ \leftarrow \ R'_\ell \cdot Y_k^{a_\ell} \cdot g^{b_\ell}$$

in place of $R'_\ell \cdot X_0^{a_\ell} \cdot g^{b_\ell}$ (cf. line 16). The vector $\boldsymbol{w}$ is then defined as (cf. line 6)

$$\boldsymbol{w} \ \leftarrow \ c_{\ell^*}\, \boldsymbol{v}_k \ + \ \sum_{j\in[n]} \boldsymbol{\alpha}_{\ell^*}[j] \cdot \boldsymbol{v}_j.$$

The remainder of the argument is identical to Theorem 2, with V in place of the Vandermonde family.

gLDVR $\Rightarrow$ ICDL (in the EC-GGM). We directly generalize the proof of Theorem 3, again replacing the Vandermonde vectors throughout by V. By non-degeneracy, every subset of t vectors in V is linearly independent; in particular, we may select a subset $S \subseteq V$ of size t containing $\boldsymbol{v}_k$, which then forms a basis of $\mathbb{Z}_p^t$. For a Map query corresponding to a coefficient vector $\boldsymbol{r} = (r_1, \ldots, r_t) \in \mathbb{Z}_p^t$, the reduction computes the unique vector $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$ with nonzero entries only at indices j with $\boldsymbol{v}_j \in S$, satisfying

$$\sum_{j \in [n]} \alpha[j]\, \boldsymbol{v}_j \;=\; \boldsymbol{r}\,;$$

existence and uniqueness follow from S being a basis. Since $\boldsymbol{\alpha}$ has at most t nonzero entries, each Map query is simulated by a single gLDVR oracle query. The hybrid argument and final step then proceed exactly as in Theorem 3, yielding an ICDL solution from any successful gLDVR adversary.

Open questions. We mention two open questions for future work, both concerning the flexible-target setting.

The first is whether the equivalence above extends to flexible-target adversaries. The above proofs extend trivially to this setting via a guessing argument, but with a factor-n loss. Whether this loss can be avoided remains open.

The second is whether non-degeneracy of V alone suffices for hardness of ICDL and gLDVR. If hardness holds for all non-degenerate V, it would follow that VCDL and LDVR are not special but instances of a phenomenon determined entirely by the non-degeneracy condition. Establishing or refuting this would sharpen our understanding of what makes these assumptions hard.

Acknowledgments

We thank Balthazar Bauer for help with the proof of Theorem 3. We thank Stefano Tessaro for pointing out similarities between our simulation technique for Sparkle and their technique for Twinkle. We also thank Benedikt Wagner for explaining details of their technique for Twinkle, leading to the discovery of an oversight and subsequent fix of how we simulate Sparkle. Finally, we thank the anonymous reviewers for their feedback.

The work of G.F. and M.S. was funded by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18002] and [10.47379/ICT25-075], and by the Austrian Science Fund (FWF) [10.55776/F8515-N].

References

BC26. Renas Bacho and Yanbo Chen. Earpicks: Tightly secure two-round multi and threshold signatures. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part I*, volume 16541 of *LNCS*, pages 154–183. Springer, Cham, May 2026. doi:10.1007/978-3-032-25291-3_6. 9

BCK+22. Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022. doi:10.1007/978-3-031-15985-5_18. 8, 9

BCL+26. Renas Bacho, Yanbo Chen, Julian Loss, Stefano Tessaro, and Chenzhi Zhu. Adaptively secure partially non-interactive threshold schnorr signatures in the AGM. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part I*, volume 16541 of *LNCS*, pages 33–62. Springer, Cham, May 2026. doi:10.1007/978-3-032-25291-3_2. 8, 10

BD24. Luís T. A. N. Brandão and Michael Davidson. Notes on threshold EdDSA/Schnorr signatures. https://csrc.nist.gov/pubs/ir/8214/b/ipd, 2024. [Online; accessed 4-June-2024]. 3

BDLR25a. Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. Adaptively secure three-round threshold Schnorr signatures from DDH. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 390–422. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_13. 8

BDLR25b. Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. Glacius: Threshold schnorr signatures from DDH with full adaptive security. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part II*, volume 15602 of *LNCS*, pages 304–334. Springer, Cham, May 2025. doi:10.1007/978-3-031-91124-8_11. 8

Ber15. Daniel J. Bernstein. Multi-user Schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996, 2015. URL: https://eprint.iacr.org/2015/996. 9

BFP21. Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 587–617. Springer, Cham, December 2021. doi:10.1007/978-3-030-92068-5_20. 6, 30, 31

BGC+25. Ruben Baecker, Paul Gerhart, Davide Li Calsi, Luigi Russo, Dominique Schröder, and Arkady Yerukhimovich. Fully adaptive FROST in the algebraic group model from falsifiable assumptions. Cryptology ePrint Archive, Report 2025/1950, 2025. URL: https://eprint.iacr.org/2025/1950. 8, 10

BHJ+15. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Berlin, Heidelberg, March 2015. doi:10.1007/978-3-662-46494-6_26. 10

BKL+25. Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. Ringtail: Practical two-round threshold signatures from learning with errors. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 149–164. IEEE Computer Society Press, May 2025. doi:10.1109/SP61157.2025.00070. 9

BLT+24. Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from DDH with full adaptive security. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*,

volume 14651 of *LNCS*, pages 429–459. Springer, Cham, May 2024. doi:10.1007/978-3-031-58716-0_15. 3, 4, 5, 16, 17, 22

BNPS03. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. doi:10.1007/s00145-002-0120-1. 6

BP26. Luís T. A. N. Brandão and Rene Peralta. NIST first call for multi-party threshold schemes. https://csrc.nist.gov/pubs/ir/8214/c/final, 2026. [Online; accessed 21-May-2026]. 3

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596. 3

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_25. 11

Bro05. Daniel R. L. Brown. On the provable security of ECDSA. In Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, editors, *Advances in Elliptic Curve Cryptography*, London Mathematical Society Lecture Note Series, pages 21–40. Cambridge University Press, 2005. doi:10.1017/CBO9780511546570.004. 5

BTZ22. Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger security for non-interactive threshold signatures: BLS and FROST. Cryptology ePrint Archive, Report 2022/833, 2022. URL: https://eprint.iacr.org/2022/833. 6

BW26. Renas Bacho and Benedikt Wagner. Tightly secure threshold signatures over pairing-free groups. *CiC*, 2(4):8, 2026. doi:10.62056/ay73z10k. 9

CATZ24. Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 268–302. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0894-2_9. 9

CFOS25. Gavin Cho, Georg Fuchsbauer, Adam O'Neill, and Marek Sefranek. Schnorr signatures are tightly secure in the ROM under a non-interactive assumption. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 223–255. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_8. 5, 6, 7, 11, 14, 17, 18, 27, 30, 49

CGRS23. Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical Schnorr threshold signatures without the algebraic group model. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 743–773. Springer, Cham, August 2023. doi:10.1007/978-3-031-38557-5_24. 8

Che25a. Yanbo Chen. Dazzle: Improved adaptive threshold signatures from DDH. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 233–261. Springer, Cham, May 2025. doi:10.1007/978-3-031-91826-1_8. 9

Che25b. Yanbo Chen. Round-efficient adaptively secure threshold signatures with rewinding. *CiC*, 2(2):16, 2025. doi:10.62056/ah893z10k. 8

CKK+25. Elizabeth C. Crites, Jonathan Katz, Chelsea Komlo, Stefano Tessaro, and Chenzhi Zhu. On the adaptive security of FROST. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 480–511. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_16. 4, 6, 7, 8, 14, 28, 29, 64, 65

CKM23a. Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. Cryptology ePrint Archive, Report 2023/445, 2023. URL: https://eprint.iacr.org/2023/445. 4, 8, 12, 13, 15, 16

CKM23b. Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023. doi:10.1007/978-3-031-38557-5_22. 3, 8, 10, 64, 65

CKM25. Elizabeth Crites, Chelsea Komlo, and Mary Maller. On the adaptive security of key-unique threshold signatures. Cryptology ePrint Archive, Report 2025/943, 2025. URL: https://eprint.iacr.org/2025/943. 5, 8

CS25a. Elizabeth Crites and Alistair Stewart. On reed–solomon proximity gaps conjectures. Cryptology ePrint Archive, Report 2025/2046, 2025. URL: https://eprint.iacr.org/2025/2046. 7

CS25b. Elizabeth C. Crites and Alistair Stewart. A plausible attack on the adaptive security of threshold Schnorr signatures. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 457–479. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_15. 4, 7

DKM+24. Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024. doi:10.1007/978-3-031-58723-8_8. 9

EKT24. Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024. doi:10.1007/978-3-031-68394-7_13. 9

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. doi:10.1007/978-3-319-96881-0_2. 3, 30, 65

FKP17. Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the one-per-message unforgeability of (EC)DSA and its variants. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 519–534. Springer, Cham, November 2017. doi:10.1007/978-3-319-70503-3_17. 5

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12. 11

GCRS26. Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. Fully-adaptive two-round threshold schnorr signatures from DDH. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part II*,

volume 16542 of *LNCS*, pages 251–283. Springer, Cham, May 2026. doi:10.1007/978-3-032-25317-0_9. 8

GHK+25. Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. Olingo: Threshold lattice signatures with DKG and identifiable abort. Cryptology ePrint Archive, Report 2025/1789, 2025. URL: https://eprint.iacr.org/2025/1789. 9

GJKR99. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer, Berlin, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_21. 8

GMS02. Steven Galbraith, John Malone-Lee, and Nigel P. Smart. Public key signatures in the multi-user setting. *Information Processing Letters*, 83(5):263–266, 2002. doi:10.1016/S0020-0190(01)00338-6. 9

GS22. Jens Groth and Victor Shoup. On the security of ECDSA with additive key derivation and presignatures. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 365–396. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-06944-4_13. 6, 28, 29, 30

HOS25. Keitaro Hashimoto, Wakaha Ogata, and Yusuke Sakai. Signatures with tight adaptive corruptions from search assumptions. Cryptology ePrint Archive, Report 2025/115, 2025. URL: https://eprint.iacr.org/2025/115. 10

KG20. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020. doi:10.1007/978-3-030-81652-0_2. 6, 8

KMP16. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Berlin, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_2. 9

KRT24. Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024. doi:10.1007/978-3-031-68394-7_15. 8, 9

Lin24. Yehuda Lindell. Simple three-round multiparty Schnorr signing with full simulatability. *CiC*, 1(1):25, 2024. doi:10.62056/a36c0l5vt. 8

Mak22. Nikolaos Makriyannis. On the classic protocol for MPC Schnorr signatures. Cryptology ePrint Archive, Report 2022/1332, 2022. URL: https://eprint.iacr.org/2022/1332. 8

NRT26. Guilhem Niot, Michael Reichle, and Kaoru Takemure. Adaptively-secure three-round threshold schnorr from DL. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part I*, volume 16541 of *LNCS*, pages 3–32. Springer, Cham, May 2026. doi:10.1007/978-3-032-25291-3_1. 8, 10

PKN+25. Rafaël Del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking TRaccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume

16005 of *LNCS*, pages 423–456. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_14. 9

PN25. Rafaël Del Pino and Guilhem Niot. Finally! A compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 169–199. Springer, Cham, May 2025. doi:10.1007/978-3-031-91826-1_6. 9

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Berlin, Heidelberg, May 1996. doi:10.1007/3-540-68339-9_33. 5, 16

RRJ+22. Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. ROAST: Robust asynchronous Schnorr threshold signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2551–2564. ACM Press, November 2022. doi:10.1145/3548606.3560583. 8

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990. doi:10.1007/0-387-34805-0_22. 11

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18. 6, 30

TZ23. Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_22. 9

Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_3. 4

ZZK22. Cong Zhang, Hong-Sheng Zhou, and Jonathan Katz. An analysis of the algebraic group model. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 310–322. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_11. 4

A Reduction from LDVR to VCDL

We prove Theorem 2 here, namely that the VCDL assumption implies the LDVR assumption.

Proof (of Theorem 2). We can directly construct a reduction adversary B against VCDL that runs the assumed adversary A against LDVR. This adversary is given in Figure 22.

First, the adversary B perfectly simulates the oracle responses. In the LDVR game, the oracle simply returns a uniform random scalar value for each query $\boldsymbol{\alpha}$. In the oracle simulation by B, the response to an oracle query is $c_\ell = f(R_\ell) + a_\ell$. Regardless of the value of $f(R_\ell)$, since a_ℓ is a freshly sampled value in $\mathbb{Z}_p$, the response c_ℓ is also uniform in $\mathbb{Z}_p$.

Adversary $\mathrm{B}^{\mathrm{DLOG}}_{\mathbb{G},g,f,n,t}(X_0, X_1, \ldots, X_t)$

$\ell \leftarrow 0$
For $j \in [0,n]$: $\boldsymbol{v}_j \leftarrow (1, j^1, \ldots, j^t) \in \mathbb{Z}_p^{t+1}$
For $j \in [n]$: $Y_j \leftarrow \prod_{i\in[0,t]} X_i^{\boldsymbol{v}_j[i]}$
$(\mathsf{CS}, \ell^*) \leftarrow\!\$\ \mathrm{A}^{\mathcal{O}}$ // $\mathsf{CS} \subseteq [n], |\mathsf{CS}| \le t, \ell^* \in [\ell]$
If $f(R_{\ell^*}) = 0$: Abort
$\boldsymbol{w}_{\ell^*} \leftarrow c_{\ell^*} \cdot \boldsymbol{v}_0 + \sum_{j\in[0,n]} \boldsymbol{\alpha}_{\ell^*}[j] \cdot \boldsymbol{v}_j$
If $\boldsymbol{w}_{\ell^*} \notin \mathrm{span}(\{\boldsymbol{v}_k\}_{k\in\mathsf{CS}})$: Abort
Compute $\boldsymbol{\beta}$ such that $\boldsymbol{w}_{\ell^*} = \sum_{k\in\mathsf{CS}} \boldsymbol{\beta}[k] \cdot \boldsymbol{v}_k$
For $k \in \mathsf{CS}$: $y_k \leftarrow \textsc{Dlog}(k)$
$z \leftarrow \sum_{k\in\mathsf{CS}} \boldsymbol{\beta}[k] \cdot y_k$
$z^* \leftarrow b_{\ell^*} + z$
Return (R_{ℓ^*}, z^*)

$\mathcal{O}(\boldsymbol{\alpha})$:
$\ell \leftarrow \ell + 1$; $\boldsymbol{\alpha}_\ell \leftarrow \boldsymbol{\alpha}$
$R'_\ell \leftarrow \prod_{j\in[0,n]} Y_j^{\boldsymbol{\alpha}_\ell[j]}$
$a_\ell, b_\ell \leftarrow\!\$\ \mathbb{Z}_p$
$R_\ell \leftarrow R'_\ell \cdot X_0^{a_\ell} \cdot g^{b_\ell}$
$c_\ell \leftarrow f(R_\ell) + a_\ell$
Return c_ℓ

Fig. 22. The VCDL reduction B from an LDVR adversary A.

Next, adversary B receives $t+1$ challenge values and makes the same number of queries to its oracle as the size of the set CS returned by adversary A. If A is a valid adversary, then the size of $\mathsf{CS} \le t$. Since B makes at most t oracle queries, it is a valid VCDL adversary.

Finally, we analyze the case where the adversary A wins the game. If A wins the game with the returned values (CS, ℓ^*), then there exists a coefficient vector $\boldsymbol{\beta}$ such that:

$$\boldsymbol{w} := c_{\ell^*} \cdot \boldsymbol{v}_0 + \sum_{j\in[0,n]} \boldsymbol{\alpha}_{\ell^*}[j] \cdot \boldsymbol{v}_j = \sum_{k\in\mathsf{CS}} \boldsymbol{\beta}[k] \cdot \boldsymbol{v}_k.$$

Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_t)$ be the discrete logs of $(X_0, X_1, \ldots, X_t)$ and recall that $y_j = \langle \boldsymbol{v}_j, \boldsymbol{x} \rangle$ for $j \in [n]$. Then, consider z^* returned by the adversary B:

$$z^* = b_{\ell^*} + \sum_{k\in\mathsf{CS}} \boldsymbol{\beta}[k] \cdot y_k = b_{\ell^*} + \langle \boldsymbol{w}, \boldsymbol{x} \rangle.$$

Expanding $\boldsymbol{w}$ gives:

$$z^* = b_{\ell^*} + (c_{\ell^*} + \boldsymbol{\alpha}_{\ell^*}[0]) \cdot x_0 + \sum_{j\in[n]} \boldsymbol{\alpha}_{\ell^*}[j] \cdot \langle \boldsymbol{v}_j, \boldsymbol{x} \rangle.$$

Then, substituting $c_{\ell^*} = f(R_{\ell^*}) + a_{\ell^*}$:

$$z^* = \left(b_{\ell^*} + (a_{\ell^*} + \boldsymbol{\alpha}_{\ell^*}[0]) \cdot x_0 + \sum_{j \in [n]} \boldsymbol{\alpha}_{\ell^*}[j] \cdot y_j \right) + f(R_{\ell^*}) \cdot x_0.$$

On the other hand,

$$R_{\ell^*} = R'_{\ell^*} \cdot X_0^{a_{\ell^*}} \cdot g^{b_{\ell^*}} = g^{b_{\ell^*}} \cdot X_0^{(a_{\ell^*} + \boldsymbol{\alpha}_{\ell^*}[0])} \cdot \prod_{j \in [n]} Y_j^{\boldsymbol{\alpha}_{\ell^*}[j]}.$$

Therefore, (R_{ℓ^*}, z^*) satisfies the VCDL condition $g^{z^*} = R^* \cdot X_0^{f(R^*)}$.

The only case where the LDVR adversary A returns a winning solution but the VCDL adversary does not is when it aborts in line 7. For any query index $\ell^* \in \ell$, R_ℓ is uniform in the group $\mathbb{G}$ since it is randomized by b_ℓ which is freshly sampled from $\mathbb{Z}_p$. Thus, the probability that the game aborts is the probability that $f(R_{\ell^*}) = 0$ for a uniform random R_{ℓ^*}. This probability is $\frac{|f^{-1}(0)|}{p}$.

We can conclude that the adversary B has the same advantage of winning as the adversary A except for when the game aborts due to $f(R_{\ell^*}) = 0$. Thus, we have the inequality of the theorem statement:

$$\mathbf{Adv}^{\mathrm{ldvr}}_{p,n,t,t}(\mathrm{A}) \leq \mathbf{Adv}^{\mathrm{vcdl}}_{\mathbb{G},g,f,n,t}(\mathrm{B}) + \frac{|f^{-1}(0)|}{p} \ .$$

□

B Error in the Original Full Adaptive Sparkle+ Proof

For completeness, we pinpoint the error in the original proof of the full adaptive security of Sparkle+ as given by Crites *et al.* [CKM23b]; to our knowledge, it has not appeared explicitly in the literature.

Recall that Crites *et al.* [CKK+25] later provide a proof showing that the LDVR assumption is necessary for adaptive security of a class of threshold Schnorr signatures, including Sparkle+. This implies that the original Sparkle+ proof must contain an error, since it claims full adaptive security of Sparkle+ under only the AOMDL assumption (which does not seem to imply the LDVR assumption) in the AGM+ROM. We show that this proof is indeed invalid: We construct a forger A* who, given the ability to solve (a restricted form of) LDVR, produces a valid forgery against Sparkle+ for which the reduction to AOMDL B presented in [CKM23b] fails to extract an AOMDL solution. Since A*'s success is contingent on solving LDVR, this confirms that the original proof implicitly relies on LDVR hardness—precisely the conclusion of [CKK+25]. Our analysis draws on ideas from the discussion of the hardness and cryptanalysis of LDVR in [CKK+25].

At a high level, B's extraction of $x = \log_g(vk)$ from a valid forgery relies on a formula whose denominator involves the hash output c^* and quantities determined by the forger's output. We construct A* so that this denominator is

zero, causing extraction to fail, while the forgery itself is valid. Specifically, A^* finds a corruption set CS forcing the denominator to vanish.

NOTATION. We largely follow the same notation as [CKM23b]. Let $p, n, t \in \mathbb{N}$ be the parameters of the scheme where p is the field size, $t+1$ is the signing threshold, t is the corruption threshold, and $n > t$ is the total number of signers. We denote by $L_k^T(Z)$ the Lagrange basis polynomial with respect to a set T where $k \in T$ and Z is the indeterminate, and by $L_{k,i}^T$ the coefficient of the i-th term Z^i in $L_k^T(Z)$. Throughout, CS denotes the set of corrupted indices and $\mathcal{S} := \mathsf{CS} \cup \{0\}$. For $k \in [n]$, let

$$\mu_k^* = 1 + \sum_{i \in [t]} L_{0,i}^{\mathcal{S}}\, k^i \qquad \text{and} \qquad \nu_k^* = \sum_{i \in [t]} \Big(\sum_{j \in \mathsf{CS}} x_j\, L_{j,i}^{\mathcal{S}} \Big) k^i .$$

STRUCTURE OF B. The reduction B of [CKM23b] runs an ADP-TS-UF adversary A against $\mathsf{Sparkle}+[\mathbb{G}]$ in the algebraic group model (AGM) [FKL18] and attempts to use a forgery to solve AOMDL. Specifically, B receives an AOMDL challenge $(Y_0, Y_1, \ldots, Y_t)$ and sets up public key shares for A as $X_k = \prod_{i=0}^{t} Y_i^{k^i}$, so that $x_k := \log_g(X_k)$ is a degree-t polynomial evaluated at k, with $vk = Y_0$ as the public key. On a corruption query $j \in [n]$, B queries its DL oracle on X_j with the AGM representation $(1, j, j^2, \ldots, j^t)$ with respect to $(Y_0, \ldots, Y_t)$ to obtain x_j. Since $\mathcal{S} = \mathsf{CS} \cup \{0\}$ has size $t+1$, Lagrange interpolation gives the identity $X_k = vk^{\mu_k^*} \cdot g^{\nu_k^*}$, where $\mu_k^* = L_0^{\mathcal{S}}(k)$ depends only on $\mathcal{S}$, and ν_k^* involves only the known values $(x_j)_{j \in \mathsf{CS}}$ (see [CKM23b, Appendix C]). Given a valid forgery from A, B attempts to extract $x = \log_g(vk)$ via the following *extraction formula*:

$$x = \frac{\eta^* - \sum_{k=1}^{n} \nu_k^*\, \zeta_k^*}{c^* + \xi^* + \sum_{k=1}^{n} \mu_k^*\, \zeta_k^*} ,$$

where $\eta^* = z^* - \gamma^* - \sum_{i,k} z_{i,k} \rho_{i,k}^*$ and $\gamma^*, \xi^*, \zeta_k^*, \rho_{i,k}^*$ are given by A's AGM representation of the forgery nonce R^*. Note that this formula is undefined whenever its denominator vanishes. We now exhibit a specific adversary A^* for which this denominator is always zero.

FORGER A^*. Now, we give the details of our specific $\mathsf{Sparkle}+$ adversary A^*:

- During initialization, A^* receives $(vk, Y_1, \ldots, Y_n)$.
- A^* makes one query to $\textsc{Hash}\mathrm{O}_{sig}$ on (vk, m^*, Y_1) for an arbitrary message m^* and receives $c^* = H_{sig}(vk, m^*, Y_1)$.
- A^* finds a set $\mathsf{CS} \subseteq [2, n]$ with $|\mathsf{CS}| = t$ satisfying $\prod_{j \in \mathsf{CS}} (1 - j^{-1}) = -c^* \pmod p$.
- A^* corrupts all indices in CS to obtain $(x_k)_{k \in \mathsf{CS}}$. Let $\mathcal{S} = \mathsf{CS} \cup \{0\}$.
- A^* returns $(m^*, (Y_1, z^*))$ where $z^* = \nu_1^* = \sum_{i \in [t]} \sum_{j \in \mathsf{CS}} x_j\, L_{j,i}^{\mathcal{S}}$, with AGM representation of $R^* = Y_1 = X_1$ given by $\gamma^* = 0$, $\xi^* = 0$, $\zeta_1^* = 1$, and $\zeta_k^* = 0$ for $k \neq 1$.

Note that finding the required CS is precisely solving a restricted form of LDVR with parameters (p, n, t): by [CKK+25, Section 4.5], the condition $\prod_{j \in \mathsf{CS}} (1 - j^{-1}) = -c^* \bmod p$ is equivalent to $c^* \boldsymbol{v}_0 + \boldsymbol{v}_1 \in \mathrm{span}(\{\boldsymbol{v}_j\}_{j \in \mathsf{CS}})$, with

solution sets restricted to $[2, n]$. Thus, if this restricted form of LDVR is easy, A^* succeeds and the reduction B fails to extract an AOMDL solution. Specifically, we show: **(1)** A^*'s output is a valid forgery and **(2)** B cannot extract an AOMDL solution from it. Both statements rely on the following claim, which shows that the choice of CS forces the denominator of B's extraction formula to be zero.

Claim 7. *If* $c^*\boldsymbol{v}_0 + \boldsymbol{v}_1 \in \mathrm{span}(\{\boldsymbol{v}_k\}_{k\in\mathsf{CS}})$ *then* $\mu_1^* = 1 + \sum_{i\in[t]} L_{0,i}^{\mathcal{S}} = -c^*$.

Proof. Since $|\mathcal{S}| = t+1$ and $0 \notin \mathsf{CS}$, the set $\{\boldsymbol{v}_k\}_{k\in\mathcal{S}}$ is a basis of $\mathbb{Z}_p^{t+1}$, so we may write $\boldsymbol{v}_1 = \sum_{k\in\mathcal{S}} \alpha_k \boldsymbol{v}_k$ for unique $\alpha_k \in \mathbb{Z}_p$. Substituting into $c^*\boldsymbol{v}_0 + \boldsymbol{v}_1 \in \mathrm{span}(\{\boldsymbol{v}_j\}_{j\in\mathsf{CS}})$ gives $(c^* + \alpha_0)\boldsymbol{v}_0 \in \mathrm{span}(\{\boldsymbol{v}_j\}_{j\in\mathsf{CS}})$. Since $\boldsymbol{v}_0 \notin \mathrm{span}(\{\boldsymbol{v}_j\}_{j\in\mathsf{CS}})$, we get $\alpha_0 = -c^*$. By the definition of the Lagrange basis polynomial, $\alpha_0 = L_0^{\mathcal{S}}(1) = \mu_1^*$, so $\mu_1^* = -c^*$. □

Proof (of statements (1) and (2)).

(1) Validity. The message m^* is fresh since A^* makes no signing queries. We have $c^* = H_{sig}(vk, m^*, Y_1)$ from A^*'s hash query. Using $X_1 = vk^{\mu_1^*} \cdot g^{\nu_1^*}$ and the claim above,

$$Y_1 \cdot vk^{c^*} = vk^{\mu_1^*} \cdot g^{\nu_1^*} \cdot vk^{c^*} = vk^{\mu_1^* + c^*} \cdot g^{z^*} = vk^0 \cdot g^{z^*} = g^{z^*}\,.$$

Thus the signature (Y_1, z^*) is valid.

(2) Reduction failure. Since A^* makes no signing queries, $\rho_{i,k}^*$ vanish. Its AGM representation gives $\gamma^* = 0$, $\xi^* = 0$, $\zeta_1^* = 1$, and $\zeta_k^* = 0$ for $k \neq 1$, so $\eta^* = z^*$ and the extraction formula reduces to

$$x = \frac{z^* - \nu_1^*}{c^* + \mu_1^*}\,.$$

The numerator is zero since $z^* = \nu_1^*$ by definition, and the denominator is zero by the claim since $\mu_1^* = -c^*$. Hence B cannot extract $x = \log_g(vk)$, completing the argument. □