Adaptively Secure (Aggregatable) PVSS and Application to Distributed Randomness Beacons

Renas Bacho
CISPA Helmholtz Center for Information Security,
Universität des Saarlandes
Saarbrücken, Germany
renas.bacho@cispa.de

Julian Loss
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
lossjulian@gmail.com

ABSTRACT

Publicly Verifiable Secret Sharing (PVSS) is a fundamental primitive that allows to share a secret S among n parties via a publicly verifiable transcript T. Existing (efficient) PVSS are only proven secure against *static adversaries* who must choose who to corrupt ahead of a protocol execution. As a result, any protocol (e.g., a distributed randomness beacon) that builds on top of such a PVSS scheme inherits this limitation. To overcome this barrier, we revisit the security of PVSS under *adaptive corruptions* and show that, surprisingly, many protocols from the literature already achieve it in a meaningful way:

- We propose a new security definition for *aggregatable PVSS*, i.e., schemes that allow to homomorphically combine multiple transcripts into one compact aggregate transcript AT that shares the sum of their individual secrets. Our notion captures that if the secret shared by AT contains at least one contribution from an honestly generated transcript, it should not be predictable. We then prove that several existing schemes satisfy this notion against adaptive corruptions in the algebraic group model.
- To motivate our new notion, we show that it implies the adaptive security of two recent random beacon protocols, SPURT (S&P '22) and OptRand (NDSS '23), who build on top of aggregatable PVSS schemes satisfying our notion of unpredictability. For a security parameter λ, our result improves the communication complexity of the best known adaptively secure random beacon protocols to $O(\lambda n^2)$ for synchronous networks with $t < n/2$ corruptions and partially synchronous networks with $t < n/3$ corruptions.

KEYWORDS

Adaptive Security, Randomness Beacon, Aggregatable PVSS, Pairing-Based Cryptography

1 INTRODUCTION

In *publicly verifiable secret sharing* (PVSS) [74], a dealer D shares a secret S among n parties $P_1, \ldots, P_n$ by broadcasting a transcript T consisting of encrypted shares $\vec{E} = (E_1, \ldots, E_n)$ along with a proof π. Any subset of $t + 1$ parties can pool their (decrypted) shares to reconstruct S, whereas t or fewer shares give no information about S. Using π, anyone can efficiently determine whether the shares in $\vec{E}$ can be decrypted by the appropriate parties and indeed yield a sharing of S. This sets PVSS apart from the more common notion of *verifiable secret sharing* (VSS) [31], which typically requires expensive communication among parties to ensure that the sharing is correct. As such, PVSS is an important building block in high-performance distributed protocols that aim to minimize communication. Recent examples of such protocols include distributed randomness beacons [15, 16, 34, 70] and distributed key generation (DKG) [3, 53]. In these types of protocols, one typically assumes a malicious adversary who can corrupt some $t < n$ of the parties and make them behave arbitrarily. Most of the literature considers a *static adversary* who must commit to its corruptions before the protocol execution begins. However, a recent trend in this area has been toward considering a stronger *adaptive adversary* who can corrupt parties dynamically over the course of the protocol execution [2, 7, 16, 33].

Unfortunately, protocol designers currently face the following limitation: existing (efficient) PVSS schemes are only proven secure with respect to static corruptions. Hence, adaptively secure protocols must often resort to less efficient (but adaptively secure) alternatives such as VSS. To ameliorate this unsatisfactory state of affairs, we ask the following question: *Are there efficient and adaptively secure PVSS protocols?*

1.1 Our Contribution

In this work, we provide a nuanced answer to the above question. Our contributions are summarized in the following.

New Security Notions for Aggregatable PVSS. One particularly useful feature supported by some PVSS schemes is the ability to homomorphically *aggregate* sharings. In more detail, suppose that we are given $t + 1$ PVSS transcripts $T_1, \ldots T_{t+1}$ sharing respective secrets $S_1, \ldots, S_n$. Then aggregation allows to efficiently combine them into a compact transcript T sharing $S = \sum_i S_i$.

Aggregate PVSS has served as an indispensible building block in many higher-order constructions, most notably leader-based randomness beacons [15, 16, 34]. In such constructions, a designated leader L aggregates PVSS transcripts of different parties and commits them to consensus. To ensure that a malicious leader cannot propose a self-chosen value, T should prove that at least one honest party has contributed to the combined secret S. This, intuitively, ensures that S remains unpredictable. We observe that while several constructions from the literature already have this property, it is usually proven as part of a security proof for a broader system (see, e.g., the recent work of Bhat et al. [15]). Given the importance of aggregated PVSS as a modular building block, we believe that it is useful to capture the above unpredictability property in a new standalone security notion that we call *aggregated unpredictability.* While aggregatable unpredictability does not ensure full secrecy in the sense of previous indistinguishability-based notions [56], we show that it is sufficient to prove the security of recent distributed randomness beacons (see below).

We prove that several existing aggregatable PVSS protocols achieve our notion of unpredictability against *adaptive corruptions* in the algebraic group model (AGM). Here, we rely on techniques from the recent work of Bacho and Loss [7], who gave the first adaptive security analysis of the threshold BLS signature [17, 19]. Our proof faces many additional challenges compared to theirs that we elaborate on in more detail in our technical overview. In particular, our proofs are complicated by the fact that the adversary obtains partial information about the secret S from the encrypted shares $\vec{E}$. Therefore, it must be argued that it cannot use this information to cancel out honest parties' contribution to the aggregated secret S and render it predictable.

Applications to Randomness Beacons. We conclude by showing that our newly introduced notion of unpredictability for PVSS suffices to prove the security of two recent distributed randomness beacon protocols, SPURT [34] and OptRand [15]. Recall that the objective of a distributed random beacon protocol is for n parties $P_1, \dots, P_n$ to agree on an a sequence of (computationally) uniformly random values $\sigma_1, \sigma_2, \dots$. The crucial property of a randomness beacon is that an adversary controlling some minority of $t < n$ parties can neither *predict* these values too early before they are output nor *bias* them. While both SPURT and OptRand achieve these properties (under different network conditions and corruption regimes), both of them are proven secure only with respect to *static adversaries.* We observe, however, that this limitation is directly inherited from the respective (statically secure) PVSS schemes that they are built on. Hence, it is plausible that their security can be improved to the same number of adaptive corruptions if the underlying PVSS provides such security guarantees. We confirm this intuition by introducing a weak unpredictability notion for randomness beacons and showing that both SPURT and OptRand achieve it against adaptive adversaries. In our new notion, a beacon produces values that remain unpredictable, yet possibly not uniformly distributed from the perspective of the adversary, up to a certain point before being output. However, since most beacon protocols assume the random oracle model [14] (ROM) anyway, it is trivial to transform an unpredictable beacon into one fully-fledged one. To do so, each party simply hashes each value that it outputs from the weak (i.e., unpredictable) beacon to obtain its final output. In this manner, one immediately obtains the first adaptively secure randomness beacons achieving $O(\lambda n^2)$ communication complexity per computed value (λ denotes a security parameter) in the synchronous regime with $t < n/2$ corruptions and in the partially synchronous regime with $t < n/3$ corruptions. Previously, adaptively secure randomness beacons in these settings relied on (more expensive) VSS [16], thus incurring at least $O(\lambda n^3)$ communication per output.

Our proofs also give a modular template which allows to infer unpredictability of leader-based beacon protocols in a black-box fashion from the unpredictability of the underlying PVSS scheme. Thus, we believe that our new security notions will be of use to the design of randomness beacons in the future.

1.2 Technical Overview

We proceed with a brief overview of our techniques. We remark that the discussion below is informal and as such does not depend on particular components of the PVSS we consider in this work. For example, non-interactive zero-knowledge proofs (NIZKs) can be implemented using Fiat-Shamir type proofs of discrete logarithm equality, pairing-based proofs, or code-based proofs, but we omit these distinctions here as they are not relevant for this high-level overview.

A Short Recap of PVSS. To begin, we describe the common high-level idea behind many efficient PVSS schemes in the literature. Let again g and h be known generators of some cyclic group $\mathbb{G}$ of prime order p. In the sharing phase, the dealer D picks a value $\alpha \in \mathbb{Z}_p$ and computes a (t, n)-sharing of a group element $S := h^\alpha$ by interpolating a random polynomial P over $\mathbb{Z}_p$ of degree t through points $\alpha_i =: P(i)$ and computing the shares h^{α_i} for all $i = 0, \dots, n$. In addition, D also computes the commitments $g^{\alpha_0}, \dots, g^{\alpha_n}$. It then computes ciphertexts $E_i := \mathsf{Enc}(pk_i, h^{\alpha_i})$ for all i and shares the vector $\vec{E}$ of encrypted shares together with the proof π consisting of the values $g^{\alpha_0}, \dots, g^{\alpha_n}$ and NIZKs proving that $\vec{E}$ is an encryption of values h^{α_i}. This can be achieved by using the values $g^{\alpha_0}, \dots, g^{\alpha_n}$. Using the NIZKs and these values, anyone is convinced that $\vec{E}$ provides a correct sharing of S. To reconstruct, party i decrypts its share via $h^{\alpha_i} = \mathsf{Dec}(sk_i, \vec{E}_i)$ and sends this value to all parties. Upon receiving $t + 1$ shares $h^{\alpha_{k_1}}, \dots, h^{\alpha_{k_{t+1}}}$, the secret can be recovered via Lagrange interpolation in the exponent of h.

A Common Proof Strategy. The main difficulty in the context of adaptive corruptions that our simulator Sim has to overcome is to balance two seemingly mutually exclusive tasks. First, Sim has to simulate the security experiment without knowing the values sk_i and the secret shares of all of the honest parties. If, instead, it knew all these values, the adversary's final output would be useless to Sim with regards to breaking the hardness assumption underlying the security of the PVSS scheme. Also, it is clear that guessing the subset of eventually corrupted parties is out of the question, as it would lead to a security loss exponential in t and n. On the other hand, Sim must be able to provide the values sk_i along with the share of party i, upon i becoming adaptively corrupted during simulation. We stress that this issue does not occur when corruptions are static, as Sim knows all the corrupted parties upfront. This allows Sim to interpolate a properly distributed polynomial through a secret S as well as the corrupted parties' shares, without actually knowing them. This simulation strategy is well-known from the the literature on distributed key generation protocols [24, 47, 48, 57]. Unfortunately, it is not applicable to a setting with adaptive corruptions, which, instead, requires different arguments. As such, schemes commonly resort to heavy machinery such as non-committing encryption [57] to attain adaptive security.

Our techniques for addressing this problem are inspired by the recent work of Bacho and Loss [7] who proved adaptive security of the (symmetric) threshold BLS signature scheme in the AGM under the OMDL assumption. Loosely speaking, the OMDL assumption of degree k asserts that it is difficult to return the discrete logarithms $z_1, \dots, z_k$ of k discrete logarithm challenges $g^{z_1}, \dots, g^{z_k}$ when given $(k - 1)$-time access to a (perfect) discrete logarithm oracle DL_g. In more detail, on input a group element $\xi \in \mathbb{G}$, DL_g returns its discrete logarithm $z \in \mathbb{Z}_p$ to base g (where p is the prime order of $\mathbb{G}$). The key insight of their work is the construction of a simulator Sim which reduces from this assumption and hence can leverage the oracle DL_g to simulate adaptive corruptions. Followup works have leveraged similar techniques to obtain adaptively secure asynchronous DKG [2] and threshold Schnorr signatures [33].

Challenges in the Context of PVSS. As explained above, aggregatable PVSS are typically composed of three main components: an encryption scheme, a commitment scheme, and a NIZK. This combination opens up many different vectors of attack that add unique challenges to our security proofs when compared to structurally simpler primitives such as signatures and VSS. Intuitively, there are three ways in which an attacker can learn the secret S, each corresponding to one of the three aforementioned components: (1) it can break security of the encryption scheme Enc to learn a $(t+1)st$ share h^{α_i}, (2) it can find the discrete logarithm a of h to base g and compute h^{α_0} from the commitment (g^{α_0}) via $(g^{\alpha_0})^a$, (3) it can pick up to t transcripts $T_1, \ldots, T_t$ dependent on a single honest transcript T^* in such a way that their aggregate becomes entirely independent of T^*. In particular, it could choose them in such a way that T^*'s contribution is cancelled out entirely, in which case the secret shared by the aggregate transcript AT is no longer unpredictable. Intuitively, this lane of attack should be prevented by the NIZK component of the scheme, as it forces the attacker to know the discrete logarithms of the secrets.

Following this high-level template, our proof broadly distinguishes multiple cases by providing appropriate simulations of the unpredictability experiment to the adversary in each of which the OMDL instance is embedded into different components of the scheme. The main difficulty of our proof is to balance these simulation strategies without the adversary being able to tell them apart.

Additional Issues with Asymmetric Groups. While we could prove all of our claims for symmetric variants of the pairing-based PVSS schemes we consider directly under the OMDL assumption, we insist on proving these schemes directly in their original and more performant versions over asymmetric pairing groups. Because of this, the OMDL assumption unfortunately turns out to be insufficient for our purposes. To see the issue, note that PVSS schemes over asymmetric pairing groups typically share the secret in *both source groups* $\mathbb{G}_1$ *and* $\mathbb{G}_2$. As a consequence, our reduction would have to supply the discrete logarithm challenges in both groups as $g_1^{x_1}, \ldots, g_1^{x_k}$, $g_2^{x_1}, \ldots, g_2^{x_k}$, where g_1 and g_2 are the groups' respective generators. To remedy this issue, we introduce a natural extension of OMDL to asymmetric pairing groups, in which the adversary obtains all of these generators and can query the oracle DL_{g_1} for elements in $\mathbb{G}_1$. We refer to this assumption as *Co-OMDL* and provide a rigorous proof of its hardness in the generic group model (GGM). Our proof follows along the lines of Bauer et al. [8], but requires a new mathematical lemma due to the higher degree of polynomials in the exponents of target group elements.

We believe that, similar to established asymmetric hardness assumptions such as SXDH [5] or Co-CDH [19], Co-OMDL has many applications to schemes based on asymmetric pairing groups and, as such, is of independent interest. As an example, we refer again to the work of Bacho and Loss who prove adaptive security for the (symmetric) threshold BLS from OMDL. As we explain in more detail in Appendix D, their proof faces similar issues in the asymmetric setting that would also require Co-OMDL.

1.3 Related Work and Discussion

We give an overview of the literature on PVSS and how our work fits in. We also briefly discuss some limitations of our work.

Publicly Verifiable Secret Sharing. The idea of publicly verifiable secret sharing (PVSS) was first formally stated in the seminal work of Stadler [74], although Stadler notes that it already was implicitly conceived in the work of Chor et al. [32]. Over the years, many improved schemes have been proposed. The common idea behind many of these schemes is the following. The dealer samples a polynomial $f \in \mathbb{Z}_p[X]$ of degree t uniformly at random and commits to it via Feldman commitments [40] (i.e. it commits to the $t+1$ coefficients of f). The dealer also provides encryptions of shares to an (t, n)-Shamir secret sharing of f. The Feldman commitments are used by non-dealer parties to compute commitments to the shares $f(i)$ that are proven via zero-knowledge proofs to correspond to the encrypted shares. Stadler realized these proofs via the Fiat-Shamir heuristics in the random oracle model. Security of the scheme is reduced from the Decisional Diffie-Hellman (DDH) assumption. Schoenmakers [71] gives a more efficient variant of Stadler's construction, in which the security of the scheme is reduced from the computational Diffie-Hellman (CDH) assumption. Ruiz and Villar [68] and Jhanwar et al. [59] gave standard model constructions which replace random oracle model proofs through checks based on Paillier encryption [66]. The security of both these schemes is reduced from the Decisional Composite Residuosity (DCR) assumption. Heidarvand and Villar [56] and Jhanwar [58] proposed alternative pairing-based PVSS constructions in the plain model with security under the Decisional Bilinear Square (DBS) assumption and the multi-sequence of exponents Diffie-Hellman (MSE-DDH) assumption. A significant drawback of these schemes is that parties must each compute $O(nt)$ exponentiations to verify the validity of the encrypted shares. Since one is mostly interested in the case $t \in O(n)$, this results in high computation cost of $O(n^2)$.

This barrier in quadratic computation cost was first overcome by SCRAPE, an elegant scheme proposed by Cascudo and David [25]. The idea of their scheme is the following. Instead of committing to the coefficients of f, the dealer directly commits to the polynomial evaluations $f(i)$ by publishing $g^{f(i)}$. With the help of linear error correcting codes (and their dual codes), parties can verify with high probability that the commitments published by the dealer actually correspond to a polynomial of degree t. In this manner, the total computation cost reduces to $O(n)$ exponentiations. The authors provide two construction based on the underlying model. In the random oracle model, the proofs are realized via NIZKs and security of the scheme is reduced from the DDH assumption. In the standard model, the authors use pairings to realize these proofs and security of the scheme is reduced from the Decisional Bilinear Square (DBS) assumption. This construction has inspired several followups, which we elaborate on below.

In the context of randomness beacons, Das et al. [34] propose SPURT, which gives a variant of SCRAPE that relies on the standard Decisional Bilinear Diffie-Hellman (DBDH) assumption and achieves similar performance to SCRAPE. The work of Gurkan et al. [53] also gives a variant of the pairing-based SCRAPE and uses it as a building block to design a DKG protocol. To support efficient aggregation of PVSS transcripts, their construction relies on signatures of knowledge and is proven secure under the Symmetric External Diffie-Hellman (SXDH) assumption for Type 3 pairings.

Security of Publicly Verifiable Secret Sharing. The literature on PVSS has considered two main security notions, both of which capture the notion of indistinguishability of secrets. These notions were first formally defined in [56]. In the weaker notion of *IND1-secrecy*, the adversary cannot distinguish between the sharings of two secrets S_1, S_2 chosen uniformly at random by the challenger. In the stronger notion of *IND2-secrecy*, the adversary has the additional power to choose the two secrets S_1, S_2 by itself. As already pointed out by Heidarvand and Villar, there is a generic transformation from an IND1-secure PVSS scheme to an IND2-secure PVSS scheme. Omitting some details, the transform uses an IND1-secret PVSS to share a uniform key K which in turn is used to encrypt a secret S.

In the following, we compare these notions to our notion of unpredictability. Intuitively, IND-secrecy says that an adversary cannot learn any information about the secret S shared in the distribution protocol. Therefore, proofs achieving this security notion have to provide simulator Sim that on input a uniformly random S simulates protocol execution in which the secret S is shared. As discussed above, it is unknown how to instantiate Sim efficiently for adaptive corruption without modifying the scheme. Our notion of (aggregated) unpredictability obviates the need for this type of simulation. This is because unpredictability allows the adversary to obtain partial information about the secret S, with the only condition that it cannot *fully recover* the secret.

1.4 Organization of this Article

In Chapter 2, we define preliminaries and our model. In Chapter 3, we formalize the notion of an aggregatable PVSS scheme and introduce a new security notion for it. Following this, we show that the PVSS schemes used in OptRand and SPURT are secure under this notion. In Chapter 4, we infer the adaptive security of the randomness beacons OptRand and SPURT. In Appendix A, we provide a warm-up for our main body, in which we introduce the syntax and a new security notion for standard PVSS schemes, prove that Schoenmakers' PVSS is secure under it, and infer the adaptive security of the randomness beacon GRandPiper [16]. In Appendix B, we provide a detailed discussion on the literature of randomness beacons. Due to space constraints, some formal definitions and proofs are deferred to Appendices C and E. Finally, we provide in Appendix D a proof for the hardness of our newly introduced Co-OMDL assumption in the generic group model.

2 PRELIMINARIES AND MODEL

Throughout the paper, we consider a complete network $\mathcal{P}$ of n parties connected by pairwise authenticated channels, i.e. the receiver of a message is aware of the sender's identity. We assume known party identifiers, w.l.o.g from P_1 to P_n. An unknown subset of these parties is faulty and controlled by an adversary.

General Notation. We denote the set of integers by $\mathbb{Z}$, the group of integers modulo p by $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ and its multiplicative unit group by $\mathbb{Z}_p^*$. We denote the set of integers from a to b by $[a, b]$; if $a = 1$, we write $[b]$, and if $a = 0$, we write $[\![b]\!]$. For an element x in a set S, we write $x \leftarrow S$ to indicate that x was sampled from S uniformly at random. We consider the standard notion of probabilistic polynomial time algorithms. As such, all of our algorithms are randomized (unless stated otherwise) and are written in uppercase serif-free letters. We write $x \leftarrow \mathsf{A}(x_1, \ldots, x_n)$ to denote that algorithm A is run on inputs $(x_1, \ldots, x_n)$ and produces output x. We write $x \in \mathsf{A}(x_1, \ldots, x_n)$ to denote that x is a possible output of a (randomized) algorithm A on input $(x_1, \ldots, x_n)$. If A has oracle access to some algorithm B during its execution, we write A^{B}. Furthermore, we write $\mathbf{G}^{\mathsf{A}}$ to denote the output of the experiment $\mathbf{G}$ involving algorithm A. We define $\mathcal{LC}$ to be the Reed-Solomon code over $\mathbb{Z}_p$ of length n and dimension $t + 1$ of the following form

$$\mathcal{LC} := \{(f(1), \ldots, f(n)) \mid f(X) \in \mathbb{Z}_p[X], \deg(f) \leq t\},$$

where $f(X)$ ranges over all polynomials in $\mathbb{Z}_p[X]$ of degree at most t. Its dual code $\mathcal{LC}^\perp$ is defined as

$$\mathcal{LC}^\perp := \{(\vartheta_1 r(1), \ldots, \vartheta_n r(n)) \mid r(X) \in \mathbb{Z}_p[X], \deg(r) \leq n - t\}$$

with the coefficients $\vartheta_i := \prod_{j \in [n] \setminus \{i\}} 1/(i - j)$. Equivalently, $\mathcal{LC}^\perp$ is the vector space consisting of all $c^\perp \in \mathbb{Z}_p^n$ that are orthogonal to all of $\mathcal{LC}$, i.e. $\langle c^\perp, c \rangle = 0$ for all $c \in \mathcal{LC}$ where $\langle \cdot, \cdot \rangle$ is the standard inner product operation on $\mathbb{Z}_p^n$.

Setup Assumptions and Adversary Model. We assume that parties have established a public key infrastructure (PKI) via a public bulletin board. This means that every party P_i is associated with a public-secret key pair (pk_i, sk_i) of a public key encryption scheme, where pk_i is known to all parties.[1]We assume an adversary who can take full control of up to $t < n$ parties and may cause them to deviate from the protocol arbitrarily. We refer to the correct parties as *honest* and to the faulty parties as *corrupt*. The adversary is adaptive, i.e. it chooses the faulty parties at any time during the execution of the protocol. We do not assume that the keys are computed in a trusted manner. Instead, we assume only that each party generates its keys locally (faulty parties may choose their keys arbitrarily) and then makes its public keys known to all other parties by using the public bulletin board. However, the adversary is assumed to be *rushing* and can corrupt some subset $C \subset [n]$ of parties so as to replace their keys with keys of its own choice, before they get posted to the bulletin board.

Random Oracle Model. We assume the random oracle model (ROM) [14]. In this model, a hash function H is treated as an idealized random function. Concretely, H is modeled as an oracle with the following properties. The oracle internally keeps a list H for bookkeeping purposes. At the beginning, all entries of H are set to $\perp$. On input m from the domain of H, the oracle first checks whether $H[m] \neq \perp$. If so, it returns $H[m]$. Otherwise, it sets $H[m]$ to a uniformly random value in the codomain of H and then returns $H[m]$. We write q_h to denote the maximum number of allowed hash queries, i.e. the number of times the adversary may query the oracle H.

Cryptographic Groups. Let λ be the security parameter. Throughout, we assume that global system parameters par are fixed and known to all parties. Depending on the setting, we either assume

[1]In some specific cases, we will also require that parties share private and public keys for a digital signature scheme.

that $par = (\mathbb{G}, g, h, p)$ defines a cyclic group $\mathbb{G}$ of prime order p with generators g, h or that $par = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \hat{g}, h, e)$ defines a triple of groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p such that $g, \hat{g} \in \mathbb{G}_1, h \in \mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear asymmetric pairing of Type 3. That is, there is no efficiently computable isomomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ and vice versa. For concrete choices, we will assume $\lambda = 128$ and that $\mathbb{G}_1, \mathbb{G}_2$ are instantiated with 256-bit elliptic curves.

Algebraic Group Model. In the algebraic group model (AGM) [43], all algorithms are treated as algebraic. Intuitively, whenever an algorithm outputs a group element, it must also output a representation of that element relative to all of the inputs the algorithm has received up to that point. This captures the intuition that any reasonable algorithm should know how it computes its outputs from its inputs. In terms of assumptions, the algebraic group model lies in between the generic group model [73] and the plain model.

Definition 2.1 (Algebraic Algorithm). An algorithm A is called *algebraic* (over group $\mathbb{G}$) if for all group elements $\zeta \in \mathbb{G}$ that A outputs, it additionally outputs a vector $\vec{z} = (z_0, \ldots, z_m)$ of integers such that $\zeta = \prod_i g_i^{z_i}$, where $(g_0, \ldots, g_m)$ is the list of group elements A has received so far.

3 ADAPTIVELY SECURE APVSS SCHEMES

In this section, we provide a formal definition for *aggregatable PVSS (APVSS)*. Additionally, we propose our new security notion and prove several schemes from the literature secure with respect to it.

An APVSS scheme allows a dealer to share a secret S via ADist via a *transcript* $T = (\vec{E}, \pi)$. $\vec{E}$ contains a vector of encrypted shares of S, each to a different public key pk_i, such that any $t+1$ decryptions uniquely reconstruct S. T is publicly verifiable using algorithm Ver. The *aggregation* routine Agg allows to homomorphically combine the secrets corresponding to transcripts $T_1, \ldots, T_k$ into an *aggregate transcript* AT. To be useful as a building block, we endow an aggregatable PVSS scheme with an additional verification routine AVer for aggregated transcripts. Intuitively, AVer can be used to detect whether an aggretated transcript AT has at least one contribution from an honest party. In order for this to be well-defined, we also define the notion of *ownership* of a transcript T. This is captured via the auxiliary algorithm OwnId that can efficiently find the creator of T. In concrete schemes, this is usually implemented by parties holding signing keys in addition to their encryption keys and digitally signing their transcripts. Rather than unnecessarily convoluting our syntax, we simply require that the distribution algorithm ADist take in a party's secret key as part of its input. (This would, for example, allow a party holding a signing key as part of its overall secret key to sign its transcript upon distributing it.) We remark that in our definitions, we syntactically distinguish between transcripts and aggregated transcripts.

Definition 3.1 (Aggregatable PVSS Scheme). Let $\hat{\mathbb{G}}$ be a cyclic group of prime order p specified by par. A (t, n)*-threshold aggregatable PVSS (APVSS) scheme over* $\hat{\mathbb{G}}$ is a tuple of algorithms APVSS = (Keys, Enc, Dec, ADist, OwnId, Ver, AVer, Rec, Agg) with the following properties:

- Keys: The randomized *key generation algorithm* takes as input system parameters par and an identity index $i \in [n]$. It outputs a public key pk_i and a secret key sk_i.
- Enc: The randomized *encryption algorithm* takes as input a public key pk_i and a message m. It outputs a ciphertext c.
- Dec: The deterministic *decryption algorithm* takes as input a secret key sk_i and a ciphertext c. It outputs a message m (optionally with a proof of correct decryption). We require that for all messages m,
$$\Pr[\mathsf{Dec}_{sk_i}(\mathsf{Enc}_{pk_i}(m)) = m] = 1.$$
- ADist: The randomized *aggregatable secret sharing algorithm* takes as input a secret key sk_i and public keys $pk_1, \ldots, pk_n$. It outputs a vector of encrypted shares $\vec{E} = (\mathsf{Enc}_{pk_1}(S_1), \ldots, \mathsf{Enc}_{pk_n}(S_n))$ and a proof π, where $S_1, \ldots, S_n$ are shares of a secret $S \in \hat{\mathbb{G}}$. We refer to $T := (\vec{E}, \pi)$ as a *PVSS transcript*.
- Ver: The deterministic *verification algorithm* takes as input public keys $pk_1, \ldots, pk_n$, and a PVSS transcript $T = (\vec{E}, \pi)$. It outputs 1 (accept) or 0 (reject). In the first case we call the transcript T *valid* (relative to $pk_1, \ldots, pk_n$); otherwise we call it *invalid*.
- OwnId: The deterministic *owner identifier algorithm* takes as input a PVSS transcript $T = (\vec{E}, \pi)$ and a public key pk_i. It outputs 1 (accept) or 0 (reject). In the first case, we refer to P_i as the *owner of* T.[2]
- Agg: The deterministic *aggregation algorithm* takes as input $t+1$ PVSS transcripts $(\vec{E}_1, \pi_1), \ldots, (\vec{E}_{t+1}, \pi_{t+1})$ with pairwise distinct owners. It outputs an *aggregated PVSS transcript* $AT := (\vec{E}, \pi)$.
- AVer: The deterministic *aggregation verification algorithm* takes as input public keys $pk_1, \ldots, pk_n$, and an aggregated PVSS transcript $AT = (\vec{E}, \pi)$. It outputs 1 (accept) or 0 (reject). In the first case we call the aggregated transcript AT *valid*; otherwise we call it *invalid*.
- Rec: The deterministic *reconstruction algorithm* takes as input $t+1$ shares $S_1, \ldots, S_{t+1}$. It outputs a reconstructed secret $S \in \hat{\mathbb{G}}$. *In case* Rec *gets more than* $t+1$ *shares as input, it takes the first lexicographical* $t+1$.

For an aggregatable PVSS scheme APVSS = (Keys, Enc, Dec, ADist, OwnId, Ver, AVer, Rec, Agg) as defined above, we define public verifiability of transcripts and aggregated transcripts as well as correctness as follows:

- *Correctness (of Aggregatable PVSS).* We say that APVSS is *correct* if for all $(pk_1, sk_1), \ldots, (pk_n, sk_n) \in \mathsf{Keys}(par)$ and all $i \in [n]$,
$$\Pr[\mathsf{Ver}(\{pk_j\}_{j\in[n]}, T) = 1 \land \mathsf{OwnId}(pk_i, T) = 1] = 1,$$
where the probability is taken over all $T \leftarrow \mathsf{ADist}(sk_i, \{pk_j\}_{j\in[n]})$.
- *Public Verifiability (of Transcripts).* We say that APVSS is *publicly verifiable* if for all $(pk_1, sk_1), \ldots, (pk_n, sk_n) \in \mathsf{Keys}(par)$ and all $(\vec{E}, \pi)$ s.t. $\mathsf{Ver}(\{pk_j\}_{j\in[n]}, (\vec{E}, \pi)) = 1$, there exists a unique $S \in \hat{\mathbb{G}}$ s.t.
$$\mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}_i)\}_{i\in\mathcal{I}}) = S \quad \forall \mathcal{I} \subset [n], |\mathcal{I}| = t+1.$$
- *Public Verifiability (of Aggregated Transcripts).* We say that APVSS is *publicly verifiable* if for all $(pk_1, sk_1), \ldots, (pk_n, sk_n)$ $\in \mathsf{Keys}(par)$ and all aggregated transcripts $AT = (\vec{E}, \pi)$ s.t.

[2]We remark that OwnId could return 1 on an invalid transcript.

$\mathsf{AVer}(\{pk_j\}_{j\in[n]}, (\vec{E}, \pi)) = 1$, there exists a unique $S \in \hat{\mathbb{G}}$ s.t.

$$\mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}_i)\}_{i\in\mathcal{I}}) = S \quad \forall \mathcal{I} \subset [n], |\mathcal{I}| = t + 1.$$

We say that an APVSS scheme is publicly verifiable if both its transcripts and aggregated transcripts are publicly verifiable. We would also like to guarantee that the secret reconstructed from an aggregated transcript $AT = \mathsf{Agg}(T_1, \dots, T_{t+1})$ corresponds to the sum of the secrets S_i that can be reconstructed from T_i. This is captured in the following definition.

Definition 3.2 (Correctness of Aggregation). We say that an aggregatable and publicly verifiable (t, n)-threshold APVSS scheme APVSS = (Keys, Enc, Dec, ADist, OwnId, Ver, AVer, Rec, Agg) over $\hat{\mathbb{G}}$ is *correctly aggregatable* if for all keys $(pk_1, sk_1), \dots, (pk_n, sk_n) \in \mathsf{Keys}(par)$ and all PVSS transcripts $T_1 = (\vec{E}_1, \pi_1), \dots, T_{t+1} = (\vec{E}_{t+1}, \pi_{t+1})$ with pairwise distinct owners, the following is true. If for all $i \in [t+1]$, $\mathsf{Ver}(\{pk_j\}_{j\in[n]}, T_i) = 1$, then for all $\mathcal{I} \subset [n]$, $|\mathcal{I}| = t+1$, the aggregated transcript $AT = (\vec{E}', \pi') := \mathsf{Agg}(T_1, \dots, T_{t+1})$ satisfies

$$\mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}'_i)\}_{i\in\mathcal{I}}) = \prod_{j\in[t+1]} \mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}_{j,i})\}_{i\in\mathcal{I}}),$$

where we write $\vec{E}_j = (\vec{E}_{j,1}, \dots, \vec{E}_{j,n})$.

3.1 New Security Notions for APVSS

We introduce a new security notion for APVSS schemes called *aggregated unpredictability*. This is a kind of non-malleability kind property specifically for aggregatable PVSS schemes. It prohibits an adversary controlling t parties from learning the secret of an aggregated transcript with at least one honest contribution, even if the adversary is allowed to contribute itself to the aggregate. This models an active adversary who can contribute to the final secret itself. In the following, we define this notion formally.

Definition 3.3 (Aggregated Unpredictability of Aggregatable PVSS Scheme). Let APVSS = (Keys, Enc, Dec, ADist, OwnId, Ver, AVer, Rec, Agg) be a publicly verifiable aggregatable (t, n)-PVSS scheme over $\hat{\mathbb{G}}$. For an algorithm A, define the *aggregated unpredictability* experiment $\mathbf{AggPred}^{\mathsf{A}}_{\mathsf{APVSS},t}$ as follows:

- *Offline Phase.* For all $i \in [n]$, run Keys on input (par, i) to generate keys $(pk_i, sk_i) \leftarrow \mathsf{Keys}(par, i)$. On input par and $\{pk_i\}_{i\in[n]}$, A returns an index set $C \subset [n]$ of initially corrupted parties along with updated public keys $\{\hat{pk}_j\}_{j\in C}$. Set $pk_j := \hat{pk}_j$ for all $j \in C$.
- *Corruption Queries.* At any point of the experiment, A may submit an index $i \in [n] \setminus C$. In this case, return the secret key sk_i and update $C := C \cup \{i\}$. *If A is static, it submits an index set $C' \subset [n] \setminus C$ at the beginning of the experiment. Return the secret keys $\{sk_i\}_{i\in C'}$ and update $C := C' \cup C$.*
- *Random Oracle Queries.* At any point of the experiment, A gets access to an oracle that answers queries of the following type: When A submits a query m, check if $H[m] = \perp$. If so, set $H[m] \leftarrow \mathbb{Z}_p^*$ and return $H[m]$. Otherwise, return $H[m]$.
- *Transcript Queries.* At any point of the experiment, A gets access to an oracle that answers queries of the following type: When A submits a request (givePVSS, i) for an $i \in [n] \setminus C$, do the following. On behalf of dealer P_i, run ADist on input sk_i and $pk_1, \dots, pk_n$. Return the output transcript $T = (\vec{E}, \pi)$.
- *Output Determination.* When A outputs an aggregated transcript $AT' = (\vec{E}', \pi')$ and an element $S^* \in \hat{\mathbb{G}}$, do:
  - Return 1 if $|C| \le t$, $\mathsf{AVer}(\{pk_i\}_{i\in[n]}, (\vec{E}', \pi')) = 1$, and $S^* = \mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}'_i)\}_{i\in[t+1]})$.
  - Return 0 otherwise.

We say that APVSS is $(\varepsilon, T, t, q_k, q_h)$-*aggregated unpredictable* if for all algorithms A that run in time at most T, make at most q_k transcript queries, and make at most q_h random oracle queries, $\Pr[\mathbf{AggPred}^{\mathsf{A}}_{\mathsf{APVSS},t} = 1] \le \varepsilon$. Conversely, we say that A $(\varepsilon, T, t, q_k, q_h)$-*breaks aggregated unpredictability* of APVSS if it runs in time at most T, makes at most q_k transcript queries, makes at most q_h random oracle queries, and $\Pr[\mathbf{AggPred}^{\mathsf{A}}_{\mathsf{APVSS},t} = 1] > \varepsilon$.

3.2 Security Analysis of APVSS in the AGM, OptRand's & SPURT's Scheme

We analyze the (adaptive) security of two recent APVSS schemes from the literature that are designed upon Type 3 asymmetric pairings, OptRand's and SPURT's APVSS. As already explained in the introduction, the standard OMDL assumption is not sufficient anymore for this setting. The reason is that the secret is shared in both source groups, which makes it impossible for the simulation to work when relying on OMDL. We elaborate on this in more detail in Appendix D. We observe that this issue can be resolved by relying on an extended version of OMDL, which we call the *co one-more discrete logarithm (COMDL) assumption.* In the following, let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p with respective generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. We denote by DL_g an oracle that on input $\xi := g^z \in \mathbb{G}_1$ returns the discrete logarithm z of ξ to base g.

Definition 3.4 (Co One-More Discrete Logarithm Problem). For an algorithm A and $n \in \mathbb{N}$, define the co one-more discrete logarithm experiment n-$\mathbf{COMDL}^{\mathsf{A}}$ for $\mathbb{G}_1$ and $\mathbb{G}_2$ as follows:

- *Offline Phase.* Sample $(z_1, \dots, z_n) \leftarrow \mathbb{Z}_p^n$ uniformly at random and set $\xi_i := (g^{z_i}, h^{z_i}) \in \mathbb{G}_1 \times \mathbb{G}_2$ for all $i \in [n]$.
- *Online Phase.* Run A on input $(par, \xi_1, \dots, \xi_n)$. In this phase, A gets access to the oracle DL_g in $\mathbb{G}_1$.
- *Output Determination.* When A outputs $(z'_1, \dots, z'_n)$, return 1 if (i) $z'_i = z_i$ for all $i \in [n]$, and (ii) DL_g was queried at most $n - 1$ times during the online phase. Otherwise, return 0.

We say that the co one-more discrete logarithm problem of degree n is (ε, T)-hard if for all algorithms A that run in time at most T, $\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}} = 1] \le \varepsilon$. Conversely, we say that an algorithm A (ε, T)-solves the co one-more discrete logarithm problem of degree n if it runs in time at most T, and $\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}} = 1] > \varepsilon$.

In Appendix D, we provide a proof of hardness of COMDL in the generic group model (GGM) when the groups are equipped with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. This structure gives the adversary additional power and makes our proof even more valuable. Our proof follows along the lines of Bauer et al.'s [8] proof of hardness of OMDL in the GGM. At the heart of their proof is a

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric pairing and independent generators $g, \hat{g} \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Let (pk_i, sk_i) be the key pair of party P_i with $pk_i = h^{sk_i}$. The dealer P_L with key pair (pk_L, sk_L) wants to share secret $e(\hat{g}, h^\alpha)$ for an $\alpha \leftarrow \mathbb{Z}_p^*$. The ADist algorithm takes as input sk_L and public keys $pk_1, \ldots, pk_n$. It outputs the transcript $T_L := \{C_i, Y_i, \pi\}_{i \in [n]}$ defined as follows. In the following, $\langle m \rangle_i := (m, \sigma)$ denotes the pair consisting of message m and a signature σ on m from party P_i.

(1) Choose a polynomial $f(X) = \alpha + \alpha_1 X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ of degree t uniformly at random.
(2) Publish commitments $C_i = g^{f(i)} \in \mathbb{G}_1$ for $i \in [n]$. Also publish encrypted shares $Y_i = pk_i^{f(i)} \in \mathbb{G}_2$ for $i \in [n]$.
(3) Compute $\zeta = g^\alpha$ and a NIZK proof $\theta = (c, r)$ of knowledge of α where the challenge is $c = \mathsf{H}(g^r \zeta^{-c}, \zeta)$. Publish $\pi := \langle \zeta, \theta \rangle_L$.

The *transcript verification algorithm* Ver takes as input the public keys $pk_1, \ldots, pk_n$ (including pk_L) and transcript T_L. It outputs 1 (accept) or 0 (reject). Let $\mathcal{LC}$ be the linear code as defined in General Notation 2 and let $\mathcal{LC}^\perp$ be its dual code.

(4) Check that $e(g, Y_i) = e(C_i, pk_i)$ for all $i \in [n]$. Sample a random codeword $(v_1, \ldots, v_n) \in \mathcal{LC}^\perp$ and check that $C_1^{v_1} \cdot \ldots \cdot C_n^{v_n} = 1$.
(5) Check that $\zeta = g^{f(0)}$ via Lagrange interpolation in the exponent from the C_i.
(6) Check that the NIZK proof $\theta = (c, r)$ verifies using ζ and H. Check that the signature on $\langle \zeta, \theta \rangle_L$ verifies using pk_L.
(7) If one of the above checks fails, output 0 (invalid transcript). Otherwise, output 1 (valid transcript).

Figure 1: Aggregatable distribution protocol ADist **and transcript verification algorithm** Ver **of OptRand's APVSS.**

On input the encrypted shares $Y_1, \ldots, Y_n$, the decryption Dec and reconstruction Rec algorithms work as follows.

(1) Using sk_i, compute the secret share $S_i = h^{f(i)}$ from Y_i via extracting the root $S_i = Y_i^{1/sk_i}$. Publish the decryption S_i.
(2) Upon receiving a secret share S_ℓ from party P_ℓ, check that $e(C_\ell, h) = e(g, S_\ell)$. Otherwise, the secret share is invalid.
(3) Upon receiving $t + 1$ valid secret shares $S_j = h^{f(j)}$ from different parties, compute $S = h^{f(0)}$ via Lagrange interpolation in the exponent. Finally, the secret is computed as $e(\hat{g}, S) \in \mathbb{G}_T$ and output.

Figure 2: Decryption Dec **and reconstruction** Rec **algorithms of OptRand's APVSS.**

We demonstrate aggregation for the first $t + 1$ parties $P_1, \ldots, P_{t+1}$. The algorithm Agg takes as input the individual parties' transcripts $\{C_{i,j}, Y_{i,j}, \pi_j\}_{i \in [n]}$ for all party indices $j \in [t + 1]$ and outputs an aggregated transcript $AT := \{C_i, Y_i, \pi\}_{i \in [n]}$. In the following, let $\mu_1, \ldots, \mu_{t+1}$ denote the Lagrange coefficients for the set $[t + 1]$ at the point $x = 0$, i.e. $\mu_i := \prod_{j \in [t+1] \setminus \{i\}} j/(j - i)$ for $i \in [t + 1]$.

(1) For $i \in [n]$, compute $C_i := C_{i,1} \cdot \ldots \cdot C_{i,t+1}$ and $Y_i := Y_{i,1} \cdot \ldots \cdot Y_{i,t+1}$. Let $\pi := (\pi_1, \ldots, \pi_{t+1})$ where as above $\pi_j = \langle \zeta_j, \theta_j \rangle_j$ for all $j \in [t + 1]$. Publish the aggregated transcript $AT := \{C_i, Y_i, \pi\}_{i \in [n]}$.

The *aggregation transcript verification algorithm* AVer takes as input public keys $pk_1, \ldots, pk_n$ and an aggregated transcript $AT := \{C_i, Y_i, \pi\}_{i \in [n]}$ as above. It outputs 1 (valid aggregated transcript) or 0 (invalid aggregated transcript).

(2) Check as usual that $\{C_i, Y_i\}_{i \in [n]}$ and that $\langle \zeta_i, \theta_i \rangle_i$ for $i \in [t + 1]$ verify. Also check that $\zeta_1 \cdot \ldots \cdot \zeta_{t+1} = C_1^{\mu_1} \cdot \ldots \cdot C_{t+1}^{\mu_{t+1}}$. If one of these checks fails, output 0 (invalid aggregated transcript). Otherwise, output 1 (valid aggregated transcript).

Figure 3: Aggregation algorithm Agg **and aggregation transcript verification algorithm** AVer **of OptRand's APVSS.**

technical lemma on vector spaces generated by the vanishing set of linear (multivariate) polynomials. Their proof relies on techniques from linear algebra, especially the theory of linear vector spaces. In our case, however, this lemma does not suffice anymore, since we obtain polynomials of degree 2 from the pairing operation. Nevertheless, using techniques from algebraic geometry and the theory (of rational points) on projective varieties, we are able to extend their lemma to our setting and thus get a proof of hardness of COMDL in the generic group model. We note that the proof

Some notes on OptRand's APVSS.. In their scheme, the authors use an unspecified digital signature scheme to sign the commitment $\zeta = g^\alpha$ along with the NIZK proof of knowledge θ. In our description of their scheme, we assume for convenience that the generated pairs (pk_i, sk_i) also (implicitly) include the verification-signing key pair (vk_i, dk_i) of the underlying signature scheme for party P_i so that we do not have to keep track of these pairs in our description of the scheme. We will use a signature scheme DS = (SKey, Sign, Ver) as defined in Definition C.1 to implement their (and SPURT's) underlying APVSS scheme. For this, we will write APVSS_{DS} to denote that APVSS is implemented with DS. In particular, $(vk_i, dk_i) \leftarrow \text{SKey}(par, i)$ is used for the owner identifier algorithm. In Definition C.2, we define the security of a signature scheme by means of the unforgeability under chosen message game.

Subsequently, we give a tight security reduction from the hardness of n-COMDL to the aggregated unpredictability of OptRand's APVSS scheme. In the following, we provide an intuition for our proof. Our analysis starts with the observation that the adversary controlling t parties essentially has four options to successfully predict the secret S of the aggregate. Firstly, it learns an additional $(t + 1)$-th decryption key controlled by an honest party, in which case it can derive S from enough decryptions of secret shares. Secondly, it breaks the underlying encryption scheme directly and thus obtains an additional secret share. Thirdly, it finds the discrete logarithm ℓ of the second generator $\hat{g} \in \mathbb{G}_1$ to base g, in which case it can compute the secret $S = e(\hat{g}, h^{\alpha})$ from the element g^{α} (which is derived via Lagrange interpolation in the exponent from the public commitments) by the identity $e(\hat{g}, h^{\alpha}) = e(g^{\alpha}, h)^{\ell}$. Lastly, it forms its contributions to the aggregate such that honest parties' contributions erase (malleability attack). The key idea of our reduction therefore is to embed the n-COMDL challenge ξ in the public keys $pk_1, \dots, pk_n$ of parties, the polynomial $f \in \mathbb{Z}_p[X]$ chosen by the challenger to answer a transcript query, or the second generator $\hat{g} \in \mathbb{G}_1$, a choice that remains hidden from the adversary. In the first case, we simulate by using the discrete logarithm oracle DL_g to answer corruption queries. In the second case, we simulate by using an honest-verifier zero knowledge simulation in the random oracle to generate the NIZK proofs for the transcripts of honest parties. In the third case, we execute the protocol honestly. Additionally, our reduction is able to leverage the algebraic equations that result from the random oracle queries by the adversary to generate its NIZK proofs of knowledge to handle the malleability attack. Overall, our reduction is tight and loses only a factor of 1/6. The running time of the reduction has only a quadratic overhead.

Theorem 3.5. *If n-**COMDL** is (ε, T)-hard in the AGM and* DS *is $(\varepsilon_s, T_s, q_s)$-secure, then OptRand's* $\mathsf{APVSS}_{\mathsf{DS}}$ *is $(\varepsilon', T', t, q_k, q_h)$-aggregated unpredictable in the AGM & ROM, where*

$$\varepsilon \geq \frac{\varepsilon' - \varepsilon_s}{6} - \frac{q_h}{6p}, \quad T \leq T' + T_s + O(n^2).$$

Proof. Let A be an algebraic adversary that $(\varepsilon', T', t, q_k, q_h)$-breaks aggregated unpredictability of APVSS. In our proof, we assume that all parties are honest prior to the execution of APVSS. It is easy to adjust the proof to the case where the adversary has already corrupted some parties before the execution of the protocol. Additionally, we assume that the aggregated transcript output by the adversary at the end of the game has contribution from exactly one corrupt party. At the end of the proof we explain how to adjust the proof (at one place) to obtain the general case. In the following, let $\mathcal{C} \subset \mathcal{P} = \{P_1, \dots, P_n\}$ be the dynamically changing set of corrupt parties and $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$ the set of honest parties. In particular, we assume that $\mathcal{C} = \{\}$ prior to the execution of the protocol. We consider the following game between a challenger and the adversary.

Game $\mathbf{G}$: This is the real game. The challenger generates the system parameters $(\mathbb{G}_1, \mathbb{G}_2, p, g, \hat{g}, h)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an asymmetric pairing of cyclic groups of prime order p with independent generators $g, \hat{g} \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Furthermore, the challenger generates the public-secret key pairs $(pk_i, sk_i) = (h^{x_i}, x_i)$ of the honest parties. Whenever A decides to corrupt a party P_i, the challenger returns the internal state of that party, which consists of P_i's secret key $x_i = sk_i$, and sets $\mathcal{C} = \mathcal{C} \cup \{P_i\}$, $\mathcal{H} = \mathcal{H} \setminus \{P_i\}$. In addition, A gets full control over party P_i. Random oracle queries m_i are answered by sampling $r_i \leftarrow \mathbb{Z}_p^*$ uniformly at random and returning $H[m_i] = r_i$. Transcript oracle queries are answered by sampling a polynomial $f_k \leftarrow \mathbb{Z}_p[X]$ of degree t uniformly at random, running the ADist algorithm and returning the transcript T_k with reconstructed secret $e(\hat{g}, h^{\alpha_{0,k}})$ where $\alpha_{0,k} = f_k(0)$. The transcript also includes $\zeta_k = g^{\alpha_{0,k}}$ and a Chaum-Pedersen non-interactive zero-knowledge (NIZK) proof of knowledge $\pi_k = (c_k, r_k)$ of $\alpha_{0,k}$. The challenge c_k for the proof is computed as the hash $\mathsf{H}(-)$ of $g^{s_k} \| \zeta_k$, where $s_k = r_k - c_k \alpha_{0,k}$ and $\|$ denotes the concatenation of elements in $\mathbb{G}_1$. From now on, we write $P := f_1 \in \mathbb{Z}_p[X]$ for f_1. At the end of the game, A outputs an aggregated transcript AT with contribution from $t + 1$ different parties along with a secret $\sigma^* \in \mathbb{G}_T$.

Game $\mathbf{G}_1$: This game is identical to the game before, except that the game aborts and the adversary loses when it forges a signature of an honest party. Clearly, the statistical distance between game $\mathbf{G}$ and $\mathbf{G}_1$ is bounded by the advantage ε_s of A in the UF-CMA game of the underlying signature scheme DS. This observation is necessary, otherwise A could forge signatures on the NIZK and aggregate $t+1$ transcripts it sampled itself. Note that in the APVSS scheme the signature is used as proof of ownership of a transcript.

The strategy of our reduction will be to embed the COMDL instance into the generator $\hat{g}$, the public keys $pk_1, \dots, pk_n$ of parties, or the polynomials $f_k \in \mathbb{Z}_p[X]$ of transcripts T_k. In the following, we make the simplification by embedding the instance in only one particular polynomial, w.l.o.g. the first one f_1, and that the adversary picks the corresponding transcript T_1 for his aggregated transcript. At the end of the proof, we will eliminate these simplifications. Having said that, our reduction now samples all but the first queried transcript honestly. As A is an algebraic adversary, it returns the secret σ^* together with a representation

$$\left(a, b, \{c_i\}_{i=1}^n, \{d_i\}_{i=1}^n, \{e_i\}_{i=1}^n, \{f_i\}_{i=1}^n, \{u_i\}_{i=1}^n, \{v_{i,j}\}_{i,j=1}^n, \{w_{i,j}\}_{i,j=1}^n\right)$$

of elements in $\mathbb{Z}_p$ such that

$$\sigma^* = e(g,h)^a \cdot e(\hat{g},h)^b \cdot \prod_{i=1}^n e(C_i,h)^{c_i} \cdot \prod_{i=1}^n e(g,pk_i)^{d_i} \cdot \prod_{i=1}^n e(g,Y_i)^{e_i} \cdot \prod_{i=1}^n e(\hat{g},pk_i)^{f_i} \cdot \prod_{i=1}^n e(\hat{g},Y_i)^{u_i} \cdot \prod_{i,j=1}^n e(C_i,pk_j)^{v_{i,j}} \cdot \prod_{i,j=1}^n e(C_i,Y_j)^{w_{i,j}}. \tag{1}$$

Here, the representation is split (from left to right) into powers of pairing evaluations on combinations of the generators $g, \hat{g} \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, the public keys $pk_1, \dots, pk_n \in \mathbb{G}_2$, the polynomial commitments $C_1, \dots, C_n \in \mathbb{G}_1$ of f_1, and the encrypted shares $Y_1, \dots, Y_n \in \mathbb{G}_2$. As already clarified, we do not explicitly present the elements from the outputs $\{T_k\}_{k \geq 2}$ in the equation because these can directly be put into the other terms in on the right-hand side of the equation (since the T_k for $k > 1$ are honestly generated). We also do not include ζ into the equation because it can be computed via Lagrange interpolation in the exponent from the commitments $C_1, \dots, C_n$. In the following, let R_i for $i \in [q_h]$ denote the random oracle queries made by the adversary. Let $R_\dagger$ and ζ' be the elements corresponding to the contribution of the corrupt party. Since A is an algebraic adversary, it returns the elements $R_\dagger \overset{!}{=} g^{r'} \zeta'^{-c'} \in \mathbb{G}_1$ and $\zeta' \in \mathbb{G}_1$ together with an algebraic representation. Note that we assume w.l.o.g. that the adversary queries the random oracle on $R_\dagger \| \zeta$ to obtain a challenge for the NIZK π' corresponding to its contribution. For $R_\dagger$, let $(a', b', c_1', \dots, c_n')$ be elements in $\mathbb{Z}_p$ such that

$$R_\dagger = g^{a'} \cdot \hat{g}^{b'} \cdot C_1^{c_1'} \cdot \ldots \cdot C_n^{c_n'}. \tag{1'}$$

And for ζ', let $(a^\dagger, b^\dagger, c_1^\dagger, \dots, c_n^\dagger)$ be elements in $\mathbb{Z}_p$ such that

$$\zeta = g^{a^\dagger} \cdot \hat{g}^{b^\dagger} \cdot C_1^{c_1^\dagger} \cdot \ldots \cdot C_n^{c_n^\dagger}. \tag{1''}$$

In the following, let $\ell \in \mathbb{Z}_p$ denote the discrete logarithm of $\hat{g}$ to base g (i.e. $g^\ell = \hat{g}$). And let $\alpha_{0,i} = f_i(0)$ for $i \in [2, q_k]$ denote the secret field elements chosen by the reduction to answer the i-th transcript oracle query T_i. Assuming the adversary wins the game $\mathbf{G}$ by outputting the secret of the aggregated transcript AT (w.l.o.g. it has contributions $(\alpha', \alpha, \alpha_{0,2}, \dots, \alpha_{0,t})$, where α' comes from the adversary), the above equation (1) to base $e(g, h)$ yields

$$\ell(\alpha + \alpha' + \sum_{i=2}^{t} \alpha_{0,i}) = a + \ell b + \sum_{i=1}^{n} P(i)c_i + \sum_{i=1}^{n} x_i d_i + \sum_{i=1}^{n} x_i P(i) e_i + \ell \sum_{i=1}^{n} x_i f_i + \ell \sum_{i=1}^{n} x_i P(i) u_i + \sum_{i,j=1}^{n} P(i) x_j v_{i,j} + \sum_{i,j=1}^{n} P(i)P(j) x_j w_{i,j}.$$

Since the $\alpha_{0,i}$ for $i > 1$ are known and the sum with coefficients e_i also appears in the sum with coefficients $v_{i,i}$, this equation reduces to (using the same symbols for the coefficients)

$$\ell(\alpha + \alpha') = \ell\Big(b + \sum_{i=1}^{n} x_i f_i + \sum_{i=1}^{n} x_i P(i) u_i\Big) + a + \sum_{i=1}^{n} P(i)c_i + \sum_{i=1}^{n} x_i d_i + \sum_{i,j=1}^{n} P(i) x_j v_{i,j} + \sum_{i,j=1}^{n} P(i)P(j) x_j w_{i,j}, \tag{2}$$

which we write as $\ell(\alpha + \alpha') = \alpha A + B$ for appropriate variables A and B. On the other hand, equation (1') together with the condition that $R_\dagger \overset{!}{=} g^{r'} \zeta^{-c'}$ yields

$$\alpha' c' = r' - a' - \ell b' - \sum_{i=1}^{n} P(i) c_i'. \tag{2'}$$

We make the crucial observation that the adversary necessarily queries the random oracle on input $R_\dagger \| \zeta$ before obtaining the challenge c', thus fixing the value α' before c' was chosen by the reduction. Therefore, all appearing variables including α' are independent from c' and the above equation (2') is equivalent to $\alpha' = \tilde{a} + \ell \tilde{b} + \sum_{i \le n} P(i)\tilde{c}_i / c'$ (2"), where the elements $\tilde{a}, \tilde{b}, \tilde{c}_i \in \mathbb{Z}_p$ are appropriately defined. Plugging this equation into the above one (2) with the same notation for A and B yields

$$\ell^2 \tilde{b} + \ell(\alpha + \tilde{a} + \sum_{i=1}^{n} P(i)\tilde{c}_i / c' - A) - B = 0. \tag{$\spadesuit$}$$

Not to forget, equation (1") yields

$$\alpha' = a^\dagger + \ell b^\dagger + \sum_{i=1}^{n} P(i) c_i^\dagger, \tag{$\dagger$}$$

where the coefficients on the right-hand side are again independent from c'. In the following, we denote by V the Vandermonde matrix corresponding to the polynomial $\omega(X) = 1 + X + \ldots + X^t \in \mathbb{Z}_p[X]$ of degree t at the points $\{1, 2, \dots, n\}$. Since V is Vandermonde, its rank is $t+1$ and thus its kernel $\ker(V)$ is of dimension $n - (t+1) = t$.

We define the following four events:

- E_1 defined by $\tilde{b} = 0 \ \wedge \ \alpha + \tilde{a} + \sum_{i=1}^{n} P(i)\tilde{c}_i / c' - A = 0$.
- E_2 defined by: $(\tilde{c}_1, \dots, \tilde{c}_n) \in \mathbb{Z}_p^n$ is in the kernel of V.
- E_3 defined by $1 = x_1 u_1 + \ldots + x_n u_n$.
- E_4 defined by: There is no index $i \in \mathcal{H}$ s.t. $u_i \neq 0$.[3]

We have the following technical lemma.

Lemma 3.6. *Let* $\mathbf{G}_1$ *and* E_i *for* $i \in [4]$ *be defined as above. Then there exist (algebraic) algorithms* A_j *for* $j \in [5]$ *playing in game* n-**COMDL** *that run in time at most* T *such that:*

$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_1} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge \neg E_1],$$
$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_2} = 1] = \left(1 - \frac{1}{p}\right) \cdot \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge \neg E_2],$$
$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_3} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge \neg E_3],$$
$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_4} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge E_3 \wedge \neg E_4],$$
$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_5} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge E_3 \wedge E_4].$$

Moreover, $T \le T' + O(n^2)$.

Proof. Let $\xi = (\xi_1, \dots, \xi_n) \in (\mathbb{G}_1 \times \mathbb{G}_2)^n$ with $\xi_i = (g^{z_i}, h^{z_i})$ for $i \in [n]$ be the COMDL instance of degree n. Algorithms A_i for $i \in [5]$ have access to a (perfect) discrete logarithm oracle DL_g in $\mathbb{G}_1$ (to base g) which they can query at most $n-1$ times. When we say algorithm A_i queries the discrete logarithm oracle on ξ_j, we mean that it queries DL_g on the first component of ξ_j which is a group element in $\mathbb{G}_1$. The algorithms A_i, $i \in [5]$, simulate game $\mathbf{G}_1$ as described in the following.

Algorithm $\mathsf{A}_1(\xi, par)$: Algorithm A_1 works as follows. On input ξ, A_1 queries the discrete logarithm oracle DL_g on $\xi_2, \dots, \xi_n$ and gets $(z_2, \dots, z_n)$. It publishes the generator $\hat{g}$ by setting $\hat{g} = \xi_{1,1}$. In particular, it is $\ell = \mathsf{DL}_g(\hat{g}) = z_1$. Furthermore, A_1 generates the public-secret key pairs of parties and the polynomial $P(X) \in \mathbb{Z}_p[X]$ honestly (by sampling $sk_i, \alpha_j \leftarrow \mathbb{Z}_p$ uniformly at random). Random oracle queries m_i are answered honestly by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning $H[m_i] = r_i$. Transcript oracle queries are answered honestly by sampling a polynomial $f_k \leftarrow \mathbb{Z}_p[X]$ of degree t uniformly at random and running the distribution phase on it. Corruption queries are answered by returning the secret key of the corresponding party. It is not hard to see that A_1's simulation of $\mathbf{G}_1$ is perfect. Suppose that A_1 wins $\mathbf{G}_1$ and that event $\neg E_1$ happens. Equation ($\spadesuit$) is then a non-trivial equation of degree one or two in ℓ over the field $\mathbb{Z}_p$ (either the coefficient of ℓ^2 or the one from ℓ is non-zero). By standard techniques, A_1 can efficiently compute $\ell = z_1$ and thus solve the COMDL instance. Overall, we obtain

$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_2} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge \neg E_1].$$

The bound on the running time of A_1 is obvious.

Algorithm $\mathsf{A}_2(\xi, par)$: Algorithm A_2 works on input $\xi_i = (g^{z_i}, h^{z_i})$, $i \in [n]$, as follows. It samples $\ell \leftarrow \mathbb{Z}_p$ uniformly at random and publishes $\hat{g} = g^\ell$. It generates the public-secret key pairs of parties honestly. It chooses the polynomial $P(X) = \alpha_0 + \alpha_1 X + \ldots + \alpha_t X^t$ s.t. $g^{\alpha_i} = \xi_{i+1,1}$ for all $i \in [\![t]\!]$. In particular, it is $\alpha_i = z_{i+1}$ for $i \in [\![t]\!]$. Commitments $C_i = g^{P(i)}$ and encryptions $Y_i = pk_i^{P(i)} = (h^{P(i)})^{x_i}$ are computed via Lagrange interpolation in the exponent from the elements $\xi_1, \dots, \xi_{t+1}$ and returned to the adversary. The NIZK proof π is generated via an HVZK simulation and returned. Random oracle queries m_i are answered by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning

[3]At this stage, $\mathcal{H} \subset \mathcal{P}$ is the set of parties that remain honest at the end of the game.

$H[m_i] = r_i$. Transcript oracle queries for $k > 1$ are answered by sampling a polynomial $f_k \leftarrow \mathbb{Z}_p[X]$ of degree t uniformly at random and running the distribution phase on it. Corruption queries are answered by returning the secret key of the corresponding party. It is not hard to see that A_2's simulation of $\mathbf{G}_1$ is perfect.

Suppose that A_2 wins $\mathbf{G}_1$ and that event $E_1 \wedge \neg E_2$ happens. In particular, the vector $(\tilde{c}_1, \ldots, \tilde{c}_n) \in \mathbb{Z}_p^n$ is not in the kernel of V. Comparison of the equations (2") and (†) coming from the random oracle query on $R_\dagger \| \zeta'$ gives

$$\tilde{a} + \ell\tilde{b} + \sum_{i=1}^{n} P(i)\tilde{c}_i/c' = a^\dagger + \ell b^\dagger + \sum_{i=1}^{n} P(i)c_i^\dagger$$
$$\iff \quad \sum_{i=1}^{n} P(i)\delta_i = a^\dagger - \tilde{a} + \ell(b^\dagger - \tilde{b}),$$

where $\delta_i := (\tilde{c}_i/c' - c_i^\dagger)$ for all $i \in [n]$. With the previously defined notation $P(X) = \alpha_0 + \alpha_1 X + \ldots + \alpha_t X^t$, the last equation is equivalent

$$\sum_{i=0}^{t} \alpha_i(\delta_1 + 2^i\delta_2 + 3^i\delta_3 + \ldots + n^i\delta_n) = a^\dagger - \tilde{a} + \ell(b^\dagger - \tilde{b})$$
$$\iff \quad \sum_{i=0}^{t} \alpha_i F(i) = a^\dagger - \tilde{a} + \ell(b^\dagger - \tilde{b}), \qquad (♥)$$

where $F(X) := \delta_1 + 2^X\delta_2 + 3^X\delta_3 + \ldots + n^X\delta_n$. Assuming $F(i) = 0$ for all $i \in [\![t]\!]$, we get the following system of linear equations in the variables $\delta_1, \ldots, \delta_n$ written in matrix form:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & n \\ \vdots & \vdots & & \vdots \\ 1 & 2^t & \cdots & n^t \end{pmatrix} \cdot \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \iff V \cdot \begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ \vdots \\ \tilde{c}_n \end{pmatrix} / c' = V \cdot \begin{pmatrix} c_1^\dagger \\ c_2^\dagger \\ \vdots \\ c_n^\dagger \end{pmatrix}.$$

By assumption that event $\neg E_2$ happens, the left-hand side is an n-dimensional non-zero vector with scaling factor $1/c'$, whereas the right-hand side is an n-dimensional vector independent from c' (since the coefficients $c_i^\dagger$ and $\tilde{c}_i$ were fixed by the adversary before seeing c'). As a result, both sides are equal with probability at most $1/p$. Therefore, there is an $\tilde{i} \in [\![t]\!]$ such that $F(\tilde{i}) \neq 0$ with probability $1 - 1/p$ and algorithm A_2 proceeds as follows. It queries the discrete logarithm oracle DL_g on ξ_{i+1} for all $i \in [\![n-1]\!] \setminus \{\tilde{i}\}$ and obtains z_i for all $i \in [n] \setminus \{\tilde{i}+1\}$. In particular, A_3 has knowledge of the polynomial coefficients α_i for all $i \neq \tilde{i}$ and computes the remaining value $\alpha_{\tilde{i}}$ from the above equation (♥) using $F(\tilde{i}) \neq 0$. As a result, it solves the COMDL instance with $n-1$ oracle queries. Overall, we obtain

$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_2} = 1] = \left(1 - \frac{1}{p}\right) \cdot \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge \neg E_2].$$

The bound on the running time of A_2 is clear.

Algorithm $\mathsf{A}_3(\xi, par)$: Algorithm A_3 works on input $\xi_i = (g^{z_i}, h^{z_i})$, $i \in [n]$, as follows. It queries the discrete logarithm oracle DL_g on $\xi_2, \ldots, \xi_n$ and gets $(z_2, \ldots, z_n)$. It samples $\ell \leftarrow \mathbb{Z}_p$ uniformly at random and publishes $\hat{g} = g^\ell$. It generates the public-secret key pairs of parties honestly. It chooses the polynomial $q(X) = \alpha_1 X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ uniformly at random and lets $P(X) = \alpha + q(X)$ such that $g^\alpha = \xi_{1,1}$. In particular, it is $\alpha = z_1$ and A_3 knows the coefficients α_i for $i \in [t]$ (since it chose them uniformly at random). Commitments $C_i = g^{P(i)}$ are computed as $C_i = \xi_{1,1} g^{q(i)}$ and returned. Encrypted shares $Y_i = pk_i^{P(i)}$ are computed as $Y_i = (h^{P(i)})^{x_i}$ where $h^{P(i)} = \xi_{1,2} h^{q(i)}$ and returned. The NIZK proof π is generated via an HVZK simulation and returned. Random oracle queries m_i are answered honestly by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning $H[m_i] = r_i$. Transcript oracle queries for $k > 1$ are answered honestly by sampling a polynomial $f_k \leftarrow \mathbb{Z}_p[X]$ of degree t uniformly at random and running the distribution phase on it. Corruption queries are answered by returning the secret key of the corresponding party. It is not hard to see that A_3's simulation of $\mathbf{G}_1$ is perfect.

Suppose that A_3 wins $\mathbf{G}_1$ and that event $E_1 \wedge E_2 \wedge \neg E_3$ happens. The equation defining event E_1 is given by

$$\alpha + \tilde{a} + \sum_{i=1}^{n} P(i)\tilde{c}_i/c' = b + \sum_{i=1}^{n} x_i f_i + \sum_{i=1}^{n} x_i P(i) u_i.$$

The knowledge that $(\tilde{c}_1, \ldots, \tilde{c}_n) \in \ker(V)$ given by event E_2 reduces this equation to

$$\alpha + \tilde{a} = b + \sum_{i=1}^{n} x_i f_i + \sum_{i=1}^{n} x_i P(i) u_i.$$

With the same notation $P(X) = \alpha + q(X)$ as above, this yields

$$\alpha + \tilde{a} = b + \sum_{i=1}^{n} x_i f_i + \sum_{i=1}^{n} x_i q(i) u_i + \alpha \sum_{i=1}^{n} x_i u_i. \qquad (♦)$$

With the condition that event $\neg E_3$ happens, equation (♦) is a non-trivial linear equation in α and thus yields

$$\alpha = \left(b - \tilde{a} + \sum_{i=1}^{n} x_i f_i + \sum_{i=1}^{n} x_i q(i) u_i\right)\left(1 - \sum_{i=1}^{n} x_i u_i\right)^{-1},$$

since the second factor is non-zero. As a result, A_3 can efficiently compute $\alpha = z_1$ and thus solve the COMDL instance with $n-1$ oracle queries. Overall, we obtain

$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_3} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge \neg E_3].$$

The bound on the running time of algorithm A_3 is obvious.

Algorithm $\mathsf{A}_4(\xi, par)$: Algorithm A_4 works on input $\xi_i = (g^{z_i}, h^{z_i})$, $i \in [n]$, as follows. It samples $\ell \leftarrow \mathbb{Z}_p$ uniformly at random and publishes $\hat{g} = g^\ell$. It generates the polynomial $P(X) \in \mathbb{Z}_p[X]$ honestly by sampling $\alpha_i \leftarrow \mathbb{Z}_p$ for all $i \in [\![t]\!]$ uniformly at random. It chooses party P_j's public key pk_j as $pk_j = \xi_{j,2}$ for all $j \in [n]$. In particular, it is $x_j = sk_j = z_j$ for all $j \in [n]$. Commitments C_i, encrypted shares Y_i, and NIZK proof π are computed honestly and returned (which is possible, since the polynomial $P(X)$ is completely known to A_4). Random oracle queries m_i are answered honestly by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning $H[m_i] = r_i$. Transcript oracle queries for $k > 1$ are answered honestly by sampling a polynomial $f_k \leftarrow \mathbb{Z}_p[X]$ of degree t uniformly at random and running the distribution phase on it. Corruption queries are answered with the help of the discrete logarithm oracle DL_g. A corruption query on party P_j is answered by computing $\mathsf{DL}_g(pk_j)$ and returning the secret key sk_j. It is easy to see that A_4's simulation of $\mathbf{G}_1$ is perfect.

Suppose that A_4 wins $\mathbf{G}_1$ and that event $E_1 \wedge E_2 \wedge E_3 \wedge \neg E_4$ happens. In particular, event $\neg E_4$ implies that there is an index $\tilde{i} \in \mathcal{H}$ such that $u_{\tilde{i}} \neq 0$. Given the equation $1 = x_1 u_1 + \ldots + x_n u_n$ (♣)

defined by E_3, algorithm A_4 proceeds as follows. It queries the discrete logarithm oracle DL_g on ξ_i for all $i \in \mathcal{H} \setminus \{\tilde{i}\}$ and obtains x_i for all $i \in \mathcal{H} \setminus \{\tilde{i}\}$. Therefore, A_4 has knowledge of x_i for all $i \in \mathcal{H} \setminus \{\tilde{i}\} \cup C = \mathcal{P} \setminus \{\tilde{i}\}$ and computes the remaining value $x_{\tilde{i}}$ by the above equation (♣). As a result, it solves the COMDL instance with $n-1$ oracle queries. Overall, we obtain

$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_4} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge E_3 \wedge \neg E_4].$$

The bound on the running time of algorithm A_4 is clear.

Algorithm $\mathsf{A}_5(\xi, par)$: Algorithm A_5 works on input $\xi_i = (g^{z_i}, h^{z_i})$, $i \in [n]$, as follows. The simulation of the game $\mathbf{G}_1$ is identical to that of A_2; in particular A_5 chooses the polynomial $P(X) = \alpha_0 + \alpha_1 X + \ldots + \alpha_t X^t$ such that $g^{\alpha_i} = \xi_{i+1,1}$ for all $i \in [\![t]\!]$, and thus it is $\alpha_i = z_{i+1}$ for all $i \in [\![t]\!]$. Again, A_5's simulation of $\mathbf{G}_1$ is perfect. Suppose that A_5 wins $\mathbf{G}$ and that event $E_1 \wedge E_2 \wedge E_3 \wedge E_4$ happens. With the same notation $P(X) = \alpha + q(X)$ as before, events E_1 to E_4 yield the identities (note that $u_i = 0$ for all $i \in \mathcal{H}$ by E_4)

$$\tilde{a} = b + \sum_{i=1}^{n} x_i f_i + \sum_{i \in C} x_i q(i) u_i, \quad 1 = \sum_{i \in C} x_i u_i. \qquad (\star)$$

The latter implies that there is an index $\tilde{j} \in C$ such that $x_{\tilde{j}} u_{\tilde{j}} \neq 0$. The first equation in the above identity ($\star$) is then equivalent to

$$q(\tilde{j}) = -\frac{1}{x_{\tilde{j}} u_{\tilde{j}}} \cdot \left(b - \tilde{a} + \sum_{i=1}^{n} x_i f_i + \sum_{i \in C \setminus \{\tilde{j}\}} x_i q(i) u_i \right).$$

Algorithm A_5 proceeds as follows. It queries the discrete logarithm DL_g on $\xi_1, \xi_{t+2}, \ldots, \xi_n$ and $g^{q(i)}$ for all $i \in C \setminus \{\tilde{j}\}$. Note that since $g^{P(X)} = g^{\alpha_0} \cdot g^{q(X)} = \xi_1 \cdot g^{q(X)}$, algorithm A_5 can compute (and hence query) $g^{q(i)}$ for any $i \in \mathbb{Z}_p$. W.l.o.g. we may assume that $|C| = t$ (otherwise, A_5 simply simulates, for itself, $t - |C|$ corruption queries for random parties from $\mathcal{H}$). As a result, A_5 can compute $q(\tilde{j})$ from the above identity and has finally knowledge of $t+1$ points in the range of $[n]$ on the polynomial q of degree t. In particular, A_5 knows the coefficients of q, i.e. $\alpha_1 = z_2, \ldots, \alpha_t = z_{t+1}$. From previous oracle queries it knows $z_1, z_{t+2}, \ldots, z_n$ and thus solves the COMDL instance with $1 + (n-t-1) + (t-1) = n-1$ queries to DL_g. Overall, we obtain

$$\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_5} = 1] = \Pr[\mathbf{G}_1^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge E_3 \wedge E_4].$$

The bound on the running time of algorithm A_5 is obvious. □

To end the proof, consider algorithm B playing in n-**COMDL** as follows: B samples $i^* \leftarrow [5]$ and then internally emulates A_{i^*}. Clearly, B is an algebraic algorithm running in time at most T (the running time of A_i, $1 \leq i \leq 5$). An application of the law of total probability yields

$$\begin{aligned}\Pr[n\text{-}\mathbf{COMDL}^{\mathsf{B}} = 1] &= \frac{1}{5} \sum_{i=1}^{5} \Pr[n\text{-}\mathbf{COMDL}^{\mathsf{A}_i} = 1] \\ &\geq \frac{1}{5}\left(1 - \frac{1}{p}\right) \cdot \Pr[\mathbf{G}_1^{\mathsf{A}} = 1] \\ &\geq \frac{1}{6} \cdot \Pr[\mathbf{G}_1^{\mathsf{A}} = 1] \geq \frac{1}{6}\left(\varepsilon' - \varepsilon_s - \frac{q_h}{p}\right),\end{aligned}$$

where the last equality comes from the soundness error of the NIZK proof of knowledge of discrete logarithm output by the adversary, one try for each random oracle query. Finally, in Appendix E.1 we elaborate on the simplifications made at the beginning of the proof, which completes our analysis. □

We conclude with a theorem on the aggregated unpredictability of SPURT's APVSS scheme. The proof mostly follows the lines of the above one with some modifications. For this, we observe that there are only two differences between SPURT's and OptRand's APVSS. (1) The former assumes an additional generator $\hat{h} \in \mathbb{G}_2$ (resulting in a total of four generators $g, \hat{g} \in \mathbb{G}_1$, $h, \hat{h} \in \mathbb{G}_2$) whose purpose being solely to help their security analysis, which is a reduction from the decisional bilinear Diffie-Hellman (DBDH) problem. (2) The NIZK proof $\pi = (c, r)$ of knowledge of $\alpha = f(0)$ is replaced by n *independently generated* knowledge-sound NIZK proofs $\pi_i = \{(c_i, r_i)\}$ for $i \in [n]$ of discrete logarithm equality of commitment C_i and encrypted share Y_i (thus obviating the need to compute n pairings for this task). Here, a challenge c_i, $i \in [n]$, is computed as the cryptographic hash $\mathsf{H}(C_i, g^{r_i} C_i^{c_i}, Y_i, h^{r_i} Y_i^{c_i})$ defined by the non-interactive Chaum-Pedersen Σ-protocol. For a proof and a formal description of their scheme, we refer to Appendix E.

Theorem 3.7. *If n-**COMDL** is (ε, T)-hard in the AGM and* DS *is $(\varepsilon_s, T_s, q_s)$-secure, then SPURT's* $\mathsf{APVSS_{DS}}$ *is $(\varepsilon', T', t, q_k, q_h)$-aggregated unpredictable in the AGM & ROM, where*

$$\varepsilon \geq \frac{\varepsilon' - \varepsilon_s}{6} - \frac{q_h}{6p}, \quad T \leq T' + T_s + O(n^2).$$

4 APPLICATION TO STATE-OF-THE-ART RANDOMNESS BEACONS

In this chapter, we discuss the adaptive security of the state-of-the-art randomness beacon protocols in their respective network models: OptRand [15] in the synchronous network model, and SPURT [34] in the partially synchronous network model. In Appendix B, we provide a detailed discussion (including a comparison table) on existing work of randomness beacons. We begin by formally defining a randomness beacon protocol.

Randomness Beacon. A randomness beacon is a distributed protocol that allows a system of n parties to generate a sequence of unpredictable and unbiased random values, one for each epoch. Each party P_i has a local log that is defined as a write-once array $\Sigma_i = (\Sigma_i[1], \Sigma_i[2], \ldots)$ with $\Sigma_i[\ell]$ being its beacon output at epoch $\ell \geq 1$. Initially, each value is set to $\perp$. We say that party P_i outputs a beacon value in epoch ℓ if it writes a value on $\Sigma_i[\ell]$. A secure randomness beacon has to satisfy the following properties: consistency, availability, bias-resistance, and d-unpredictability.

Definition 4.1 (d-Secure Randomness Beacon). Let $\mathcal{RB}$ be an epoch-based protocol executed by parties $P_1, \ldots, P_n$. We define the following security properties for $\mathcal{RB}$:

- *Consistency.* $\mathcal{RB}$ is (t, L)-consistent if the following holds whenever at most t parties are corrupted: if an honest party outputs a value $\sigma_\ell \in \{0,1\}^\lambda$ in epoch $\ell \in [L]$, then all honest parties output σ_ℓ in epoch ℓ.
- *Availability.* $\mathcal{RB}$ is (t, L)-available if the following holds whenever at most t parties are corrupted: for each $\ell \in [L]$, every honest party outputs a value $\sigma_\ell \in \{0,1\}^\lambda$ in epoch ℓ.

- *Bias-Resistance.* $\mathcal{RB}$ is (ε, T, t, L)-bias-resistant if it is (t, L)-available, (t, L)-consistent, and the following holds for all algorithms A, D s.t. A corrupts at most t parties and both A and D run in time at most T. Denote by $\Sigma_{\mathsf{A},L}$ the probability distribution induced by the outputs of an honest party in an execution of $\mathcal{RB}$ until epoch L with A as adversary. Then
$$\Big| \Pr_{\sigma \leftarrow \Sigma_{\mathsf{A},L}} [\mathsf{D}(\sigma) = 1] \; - \Pr_{u \leftarrow U_L} [\mathsf{D}(u) = 1] \Big| \leq \varepsilon,$$
where U_L denotes the uniform distribution over the L-times Cartesian product of $\{0, 1\}^\lambda$ with itself.
- *d-Unpredictability.* $\mathcal{RB}$ is $(\varepsilon, T, t, L, q_h, d)$-unpredictable if it is (t, L)-available, (t, L)-consistent, and for all $\ell \in [L]$ and algorithms A that run in time at most T and make at most q_h random oracle queries, the following experiment outputs 1 with probabillity at most ε :
  - *Offline Phase.* For all $i \in [n]$, run Keys on input (par, i) to generate keys $(pk_i, sk_i) \leftarrow \mathsf{Keys}(par, i)$. On input par and $\{pk_i\}_{i\in[n]}$, A returns an index set $C \subset [n]$ of initially corrupted parties along with updated public keys $\{\hat{pk}_j\}_{j\in C}$. Set $pk_j := \hat{pk}_j$ for all $j \in C$. Initiate an execution of $\mathcal{RB}$ with A controlling parties in C.
  - *Random Oracle Queries.* At any point of the experiment, A gets access to an oracle that answers queries of the following type: When A submits a query m, check if $H[m] = \perp$. If so, set $H[m] \leftarrow \{0, 1\}^\lambda$. Return $H[m]$.
  - *Online Phase.* Run $\mathcal{RB}$ with A. When A outputs a value (σ'_e, e) for an $e > \ell$, the experiment ends with output 0 in case there is an honest party that has output a value $\sigma_{\ell+1}$ for epoch $\ell + 1$. Continue the execution of $\mathcal{RB}$ for another $e - \ell$ epochs.
  - *Corruption Queries.* During the online phase, A may corrupt a party P_i by submitting an index $i \in [n] \setminus C$. In this case, return the internal state of P_i and set $C := C \cup \{i\}$. Henceforth, A controls P_i.
  - *Output Determination.* Return 1 if $|C| \leq t$, $e \geq \ell + d$, $L \geq e$, and $\sigma'_e = \sigma_e$. Otherwise, return 0.

We say that $\mathcal{RB}$ is a $(\varepsilon, T, t, L, q_h, d)$-secure randomness beacon protocol if it is (ε, T, t, L)-bias-resistant, $(\varepsilon, T, t, L, q_h, d)$-unpredictable, (t, L)-available, and (t, L)-consistent.

Discussion. We briefly elaborate on the security notions defined above. Consistency and availability guarantee that each honest party outputs the same value $\sigma_e \in \{0, 1\}^\lambda$ in each epoch $e \geq 1$. Bias-resistance guarantees that the beacon outputs are indistinguishable from uniformly random numbers. This property ensures that the adversary has no power in biasing the beacon output, even when controlling up to t parties in the system. On the other hand, this notion does not prohibit the adversary from learning the beacon output some epochs ahead of the honest parties. That is ensured by the notion of d-unpredictability, which states that the adversary does not learn the beacon output d epochs before the honest parties. Conversely, an adversary could predict the beacon output some epochs ahead of the honest parties, e.g. by corrupting the next t leaders whose previously committed values determine the next t beacon outputs as in GRandPiper [16] or HydRand [70], without having the power to bias it. In our notions, we introduce an epoch bound L upon which the randomness beacon protocol can run.

In Appendix C.2, we introduce the notion of a *weakly secure* randomness beacon to capture the properties of SPURT. Since the construction of SPURT allows parties to output a bot symbol $\perp_{\mathcal{RB}}$ whenever the leader of an epoch does not behave correctly, the protocol does not achieve full availability. Still, every n epochs the randomness beacon outputs at least $n - t$ proper non-$\perp_{\mathcal{RB}}$ values and thus has a form of weak availability. We adapt the other security notions of a randomness beacon accordingly.

4.1 OptRand's and SPURT's Beacon Design

We give a high-level description of the randomness beacons of interest, OptRand and SPURT. For more detailed descriptions, we refer the reader to Appendix F or their papers [15, 34]. Both protocols are built upon (respective) leader-based *state machine replication (SMR)* protocols. In leader-based SMR, parties run a protocol to agree on a public ledger. The protocol proceeds through *epochs*, where each epoch e has a designated *leader* L_e responsible for choosing the value to agree on for that epoch. In our setting, L_e will be instructed to gather and aggregate PVSS transcripts that other parties send to it at the beginning of epoch e.

The protocol rotates through leaders in round-robin fashion (or using a randomized scheduling) so that even a malicious leader cannot stall progress for more than one round. Whenever L_e is honest, parties are guaranteed to agree on a correct value for epoch e (where correctness here refers to that of the aggregate transcript L_e proposes). We stress that the details of these consensus protocols are immaterial for the ensuing discussion. However, it is important to note that while both OptRand and SPURT achieve only static security for their beacon constructions, their underlying SMR protocols are *adaptively* secure. We elaborate more on this below.

OptRand. The protocol employs their APVSS scheme and an optimistically responsive extension of RandPiper's adaptively secure leader-based SMR [16]. This gives a communication complexity of $O(n\ell + \lambda n^2)$ bits per consensus decision on a block of size ℓ bits. In each epoch $e \geq 1$, the leader L_e first collects $t + 1$ valid PVSS transcripts from other parties, aggregates them, and then puts the aggregate on the ledger. If L_e does not put anything on the ledger or the aggregate is invalid, parties blacklist L_e from future leader elections. Apart from this policy, parties adhere to a round-robin leader election. When the same party is elected as a leader $L_{e'}$ in epoch $e' > e + t$ the next time, parties take its previously published (valid) aggregate and reconstruct the secret S_e. The beacon output $O_{e'}$ for epoch e' is computed as hash $O_{e'} = \mathsf{H}(S_e)$. Finally, to ensure availability the first time a party is elected as the leader, the protocol relies on a setup where parties start with agreed-upon buffers $\mathcal{B}(P_i)$ for $i \in [n]$ that contain random PVSS transcripts each. Ignoring the pre-processing phase for buffers, OptRand outputs a randomness beacon value with a communication cost of $O(\lambda n^2)$ bits and optimal resilience $t < n/2$ in the synchronous setting.

SPURT. The protocol employs their APVSS scheme and a leader-based SMR protocol based on HotStuff [78]. Adaptive security of this SMR protocol follows directly from the adaptive security of HotStuff [63, 64, 78]. Furthermore, their SMR has a communication complexity of $O(\ell n^2)$ bits per consensus decision on a block of size ℓ bits. In each epoch $e \geq 1$, the leader L_e collects $t + 1$ valid PVSS transcripts from other parties, aggregates them and multicasts the aggregate. Additionally, L_e distributes other parts of the collection of $t+1$ transcripts among the parties via the private channels such that each non-leader party checks a disjoint part of the aggregation such that any subset of $t+1$ honest parties collectively checks the entire aggregation. For efficiency reasons, the SMR is only used for the hash of the aggregate. Again, parties adhere to a round-robin leader election. However, SPURT does not use a blacklisting strategy and a pre-processing phase for buffers and thus does not guarantee availability. When the same party is elected as a leader $L_{e'}$ in epoch $e' = e + n$ the next time, parties take its previously agreed-upon aggregate and reconstruct the secret S_e (in case the leader did not behave correctly, parties do not output a beacon value for that epoch). The randomness beacon value $O_{e'}$ is computed as $e(S_e, h)$. Spurt outputs a beacon with a communication cost of $O(\lambda n^2)$ bits and optimal resilience $t < n/3$ in the partially synchronous setting.

4.2 Security Analysis of OptRand and SPURT

We prove that the state-of-the-art randomness beacons OptRand and SPURT are indeed adaptively secure. In their respective papers, the authors only provide a security analysis against a much weaker static adversary. For this, we employ our results from the previous chapter. For our analysis, we consider the derived protocol SPURT+ which results from SPURT by hashing its final output (OptRand does hash the reconstructed secret at the end anyway). This is necessary, since our aggregated unpredictability notion allows the adversary to obtain partial information about the secret. Thus, by hashing the result, we obtain a truly random beacon output (in the random oracle model, which both protocols assume anyway). We provide a proof of the following theorem in Appendix F.1.

Theorem 4.2. *If the underlying* $\mathsf{APVSS_{DS}}$ *is* $(\varepsilon, T, t, q_k, q_h)$*-aggregated unpredictable in the AGM & ROM, then OptRand (SPURT+) is an* $(\varepsilon', T', t, L, q'_h, 1)$*-(weakly) secure randomness beacon protocol in the AGM & ROM, where*

$$\varepsilon \geq \frac{\varepsilon'}{L} - \frac{q'_h}{p}, \quad T \leq T' + O(Ln^2).$$

ACKNOWLEDGMENTS

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 507237585, and by the European Union, ERC-2023-STG, Project ID: 101116713. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] 2020. Drand - a distributed randomness beacon daemon,. (2020). https://github.com/drand/drand

[2] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. 2022. Bingo: Adaptively Secure Packed Asynchronous Verifiable Secret Sharing and Asynchronous Distributed Key Generation. Cryptology ePrint Archive, Report 2022/1759. (2022). https://eprint.iacr.org/2022/1759.

[3] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. 2021. Reaching Consensus for Asynchronous Distributed Key Generation. *CoRR* abs/2102.09041 (2021). arXiv:2102.09041 https://arxiv.org/abs/2102.09041

[4] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, and Ling Ren. 2018. Dfinity Consensus, Explored. Cryptology ePrint Archive, Report 2018/1153. (2018). https://eprint.iacr.org/2018/1153.

[5] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. 2005. Untraceable RFID Tags Via Insubvertible Encryption. In *ACM CCS 2005: 12th Conference on Computer and Communications Security*, Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels (Eds.). ACM Press, Alexandria, Virginia, USA, 92–101. https://doi.org/10.1145/1102120.1102134

[6] Sarah Azouvi, Patrick McCorry, and Sarah Meiklejohn. 2018. Winning the Caucus Race: Continuous Leader Election via Public Randomness. (2018). arXiv:cs.CR/1801.07965

[7] Renas Bacho and Julian Loss. 2022. On the Adaptive Security of the Threshold BLS Signature Scheme. In *ACM CCS 2022: 29th Conference on Computer and Communications Security*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM Press, Los Angeles, CA, USA, 193–207. https://doi.org/10.1145/3548606.3560656

[8] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. 2021. The One-More Discrete Logarithm Assumption in the Generic Group Model. In *Advances in Cryptology – ASIACRYPT 2021, Part IV (Lecture Notes in Computer Science)*, Mehdi Tibouchi and Huaxiong Wang (Eds.), Vol. 13093. Springer, Heidelberg, Germany, Singapore, 587–617. https://doi.org/10.1007/978-3-030-92068-5_20

[9] Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. 2020. CRAFT: Composable Randomness and Almost Fairness from Time. Cryptology ePrint Archive, Report 2020/784. (2020). https://eprint.iacr.org/2020/784.

[10] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. 2004. Security Proofs for Identity-Based Identification and Signature Schemes. In *Advances in Cryptology – EUROCRYPT 2004 (Lecture Notes in Computer Science)*, Christian Cachin and Jan Camenisch (Eds.), Vol. 3027. Springer, Heidelberg, Germany, Interlaken, Switzerland, 268–286. https://doi.org/10.1007/978-3-540-24676-3_17

[11] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. 2003. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology* 16, 3 (June 2003), 185–215. https://doi.org/10.1007/s00145-002-0120-1

[12] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the plain public-Key model and a general forking lemma. In *ACM CCS 2006: 13th Conference on Computer and Communications Security*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, Alexandria, Virginia, USA, 390–399. https://doi.org/10.1145/1180405.1180453

[13] Mihir Bellare and Adriana Palacio. 2002. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *Advances in Cryptology – CRYPTO 2002 (Lecture Notes in Computer Science)*, Moti Yung (Ed.), Vol. 2442. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 162–177. https://doi.org/10.1007/3-540-45708-9_11

[14] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby (Eds.). ACM Press, Fairfax, Virginia, USA, 62–73. https://doi.org/10.1145/168588.168596

[15] Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. 2022. OptRand: Optimistically responsive distributed random beacons. Cryptology ePrint Archive, Paper 2022/193. (2022). https://doi.org/10.14722/ndss.2023.24832 https://eprint.iacr.org/2022/193.

[16] Adithya Bhat, Nibesh Shrestha, Zhongtang Luo, Aniket Kate, and Kartik Nayak. 2021. RandPiper - Reconfiguration-Friendly Random Beacons with Quadratic Communication. In *ACM CCS 2021: 28th Conference on Computer and Communications Security*, Giovanni Vigna and Elaine Shi (Eds.). ACM Press, Virtual Event, Republic of Korea, 3502–3524. https://doi.org/10.1145/3460120.3484574

[17] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography (Lecture Notes in Computer Science)*, Yvo Desmedt (Ed.), Vol. 2567. Springer, Heidelberg, Germany, Miami, FL, USA, 31–46. https://doi.org/10.1007/3-540-36288-6_3

[18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable Delay Functions. In *Advances in Cryptology – CRYPTO 2018, Part I (Lecture Notes in Computer Science)*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10991. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 757–788. https://doi.org/10.1007/978-3-319-96884-1_25

[19] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *Advances in Cryptology – ASIACRYPT 2001 (Lecture Notes in Computer Science)*, Colin Boyd (Ed.), Vol. 2248. Springer, Heidelberg, Germany, Gold Coast, Australia, 514–532. https://doi.org/10.1007/3-540-45682-1_30

[20] Benedikt Bünz, Steven Goldfeder, and Joseph Bonneau. 2017. Proofs-of-delay and randomness beacons in Ethereum.

[21] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography. *Journal of Cryptology* 18, 3 (July 2005), 219–246. https://doi.org/10.1007/s00145-005-0318-0

[22] Antonio Cafure and Guillermo Matera. 2007. AN EFFECTIVE BERTINI THEOREM AND THE NUMBER OF RATIONAL POINTS OF A NORMAL COMPLETE INTERSECTION OVER A FINITE FIELD. *Acta Arithmetica* 130 (2007), 19–35.

[23] Jan Camenisch, Manu Drijvers, Timo Hanke, Yvonne-Anne Pignolet, Victor Shoup, and Dominic Williams. 2021. Internet Computer Consensus. Cryptology ePrint Archive, Report 2021/632. (2021). https://eprint.iacr.org/2021/632.

[24] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. 1999. Adaptive Security for Threshold Cryptosystems. In *Advances in Cryptology – CRYPTO'99 (Lecture Notes in Computer Science)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 98–115. https://doi.org/10.1007/3-540-48405-1_7

[25] Ignacio Cascudo and Bernardo David. 2017. SCRAPE: Scalable Randomness Attested by Public Entities. In *ACNS 17: 15th International Conference on Applied Cryptography and Network Security (Lecture Notes in Computer Science)*, Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi (Eds.), Vol. 10355. Springer, Heidelberg, Germany, Kanazawa, Japan, 537–556. https://doi.org/10.1007/978-3-319-61204-1_27

[26] Ignacio Cascudo and Bernardo David. 2020. ALBATROSS: Publicly AttestabLe BATched Randomness Based On Secret Sharing. In *Advances in Cryptology – ASIACRYPT 2020, Part III (Lecture Notes in Computer Science)*, Shiho Moriai and Huaxiong Wang (Eds.), Vol. 12493. Springer, Heidelberg, Germany, Daejeon, South Korea, 311–341. https://doi.org/10.1007/978-3-030-64840-4_11

[27] Fabrizio Catanese. 1992. Chow varieties, Hilbert schemes, and moduli spaces of surfaces of general type. *Journal of Algebraic Geometry* 1 (01 1992).

[28] David Chaum and Torben P. Pedersen. 1993. Wallet Databases with Observers. In *Advances in Cryptology – CRYPTO'92 (Lecture Notes in Computer Science)*, Ernest F. Brickell (Ed.), Vol. 740. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 89–105. https://doi.org/10.1007/3-540-48071-4_7

[29] Alisa Cherniaeva, Ilia Shirobokov, and Omer Shlomovits. 2019. Homomorphic Encryption Random Beacon. Cryptology ePrint Archive, Report 2019/1320. (2019). https://eprint.iacr.org/2019/1320.

[30] Kevin Choi, Arasu Arun, Nirvan Tyagi, and Joseph Bonneau. 2023. Bicorn: An optimistically efficient distributed randomness beacon. Cryptology ePrint Archive, Report 2023/221. (2023). https://eprint.iacr.org/2023/221.

[31] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. 1985. Verifiable secret sharing and achieving simultaneity in the presence of faults. *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)* (1985), 383–395.

[32] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. 1985. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Portland, Oregon, 383–395. https://doi.org/10.1109/SFCS.1985.64

[33] Elizabeth Crites, Chelsea Komlo, and Mary Maller. 2023. Fully Adaptive Schnorr Threshold Signatures. Cryptology ePrint Archive, Paper 2023/445. (2023). https://eprint.iacr.org/2023/445 https://eprint.iacr.org/2023/445.

[34] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. 2022. Spurt: Scalable Distributed Randomness Beacon with Transparent Setup. In *2022 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 2502–2517. https://doi.org/10.1109/SP46214.2022.9833580

[35] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In *Advances in Cryptology – EUROCRYPT 2018, Part II (Lecture Notes in Computer Science)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.), Vol. 10821. Springer, Heidelberg, Germany, Tel Aviv, Israel, 66–98. https://doi.org/10.1007/978-3-319-78375-8_3

[36] Pierre Deligne. 1974. La conjecture de Weil : I. *Publications Mathématiques de l'IHÉS* 43 (1974), 273–307. http://eudml.org/doc/103930

[37] Yvo Desmedt and Yair Frankel. 1990. Threshold Cryptosystems. In *Advances in Cryptology – CRYPTO'89 (Lecture Notes in Computer Science)*, Gilles Brassard (Ed.), Vol. 435. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 307–315. https://doi.org/10.1007/0-387-34805-0_28

[38] Justin Drake. 2018. Minimal VDF randomness beacon. (2018). https://ethresear.ch/t/minimal-vdf-randomness-beacon/3566

[39] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. 2019. On the Security of Two-Round Multi-Signatures. In *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 1084–1101. https://doi.org/10.1109/SP.2019.00050

[40] Paul Feldman. 1987. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *28th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Angeles, CA, USA, 427–437. https://doi.org/10.1109/SFCS.1987.4

[41] Marc Fischlin and Nils Fleischhacker. 2013. Limitations of the Meta-reduction Technique: The Case of Schnorr Signatures. In *Advances in Cryptology – EUROCRYPT 2013 (Lecture Notes in Computer Science)*, Thomas Johansson and Phong Q. Nguyen (Eds.), Vol. 7881. Springer, Heidelberg, Germany, Athens, Greece, 444–460. https://doi.org/10.1007/978-3-642-38348-9_27

[42] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. 2014. On Tight Security Proofs for Schnorr Signatures. In *Advances in Cryptology – ASIACRYPT 2014, Part I (Lecture Notes in Computer Science)*, Palash Sarkar and Tetsu Iwata (Eds.), Vol. 8873. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C., 512–531. https://doi.org/10.1007/978-3-662-45611-8_27

[43] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. 2018. The Algebraic Group Model and its Applications. In *Advances in Cryptology – CRYPTO 2018, Part II (Lecture Notes in Computer Science)*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10992. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 33–62. https://doi.org/10.1007/978-3-319-96881-0_2

[44] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. 2020. Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model. In *Advances in Cryptology – EUROCRYPT 2020, Part II (Lecture Notes in Computer Science)*, Anne Canteaut and Yuval Ishai (Eds.), Vol. 12106. Springer, Heidelberg, Germany, Zagreb, Croatia, 63–95. https://doi.org/10.1007/978-3-030-45724-2_3

[45] William Fulton. 1998. Intersection Theory. *Springer New York, NY* XIII (June 1998), 470.

[46] Andreas Gathmann. 2021/2022. Algebraic Geometry (Class Notes). (2021/2022). https://agag-gathmann.math.rptu.de/class/alggeom-2021/alggeom-2021.pdf

[47] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. 1999. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *Advances in Cryptology – EUROCRYPT'99 (Lecture Notes in Computer Science)*, Jacques Stern (Ed.), Vol. 1592. Springer, Heidelberg, Germany, Prague, Czech Republic, 295–310. https://doi.org/10.1007/3-540-48910-X_21

[48] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. 2007. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *Journal of Cryptology* 20, 1 (Jan. 2007), 51–83. https://doi.org/10.1007/s00145-006-0347-3

[49] Rosario Gennaro, Darren Leigh, R. Sundaram, and William S. Yerazunis. 2004. Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices. In *Advances in Cryptology – ASIACRYPT 2004 (Lecture Notes in Computer Science)*, Pil Joong Lee (Ed.), Vol. 3329. Springer, Heidelberg, Germany, Jeju Island, Korea, 276–292. https://doi.org/10.1007/978-3-540-30539-2_20

[50] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. Cryptology ePrint Archive, Report 2017/454. (2017). https://eprint.iacr.org/2017/454.

[51] Jens Groth. 2006. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *Advances in Cryptology – ASIACRYPT 2006 (Lecture Notes in Computer Science)*, Xuejia Lai and Kefei Chen (Eds.), Vol. 4284. Springer, Heidelberg, Germany, Shanghai, China, 444–459. https://doi.org/10.1007/11935230_29

[52] Jens Groth. 2021. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339. (2021). https://eprint.iacr.org/2021/339.

[53] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. 2021. Aggregatable Distributed Key Generation. In *Advances in Cryptology – EUROCRYPT 2021, Part I (Lecture Notes in Computer Science)*, Anne Canteaut and François-Xavier Standaert (Eds.), Vol. 12696. Springer, Heidelberg, Germany, Zagreb, Croatia, 147–176. https://doi.org/10.1007/978-3-030-77870-5_6

[54] Runchao Han, Jiangshan Yu, and Haoyu Lin. 2020. RandChain: Decentralised Randomness Beacon from Sequential Proof-of-Work. Cryptology ePrint Archive, Report 2020/1033. (2020). https://eprint.iacr.org/2020/1033.

[55] Robin Hartshorne. 1977. *Algebraic Geometry*. Graduate Texts in Mathematics, Vol. 52. Springer. http://www.worldcat.org/oclc/2798099

[56] Somayeh Heidarvand and Jorge L. Villar. 2009. Public Verifiability from Pairings in Secret Sharing Schemes. In *SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography (Lecture Notes in Computer Science)*, Roberto Maria Avanzi, Liam Keliher, and Francesco Sica (Eds.), Vol. 5381. Springer, Heidelberg, Germany, Sackville, New Brunswick, Canada, 294–308. https://doi.org/10.1007/978-3-642-04159-4_19

[57] Stanislaw Jarecki and Anna Lysyanskaya. 2000. Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. In *Advances in Cryptology – EUROCRYPT 2000 (Lecture Notes in Computer Science)*, Bart Preneel (Ed.), Vol. 1807. Springer, Heidelberg, Germany, Bruges, Belgium, 221–242. https://doi.org/10.1007/3-540-45539-6_16

[58] Mahabir Prasad Jhanwar. 2011. A Practical (Non-interactive) Publicly Verifiable Secret Sharing Scheme. In *Information Security Practice and Experience (ISPEC)*, Vol. 6672. Springer, 273–287.

[59] Mahabir Prasad Jhanwar, Ayineedi Venkateswarlu, and Reihaneh Safavi-Naini. 2014. Paillier-based publicly verifiable (non-interactive) secret sharing. *Designs, Codes and Cryptography, Volume* 73, 2 (2014), 529–546.

[60] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Advances in Cryptology – CRYPTO 2017, Part I (Lecture Notes in Computer Science)*, Jonathan Katz and Hovav Shacham (Eds.), Vol. 10401. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 357–388. https://doi.org/10.1007/978-3-319-63688-7_12

[61] Rudolf Lidl and Harald Niederreiter. 1996. Finite fields and their applications.

[62] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008). https://bitcoin.org/bitcoin.pdf

[63] Kartik Nayak. 2023. https://decentralizedthoughts.github.io/2023-01-05-player-replaceability-II/. (2023). https://decentralizedthoughts.github.io/2023-01-05-

player-replaceability-II/

[64] Kartik Nayak. 2023. Player Replaceability - Towards Adaptive Security and Sub-quadratic Communication Simultaneously (Part I). (2023). https://decentralizedthoughts.github.io/2023-01-05-player-replaceability-I/

[65] Jonas Nick, Tim Ruffing, and Yannick Seurin. 2021. MuSig2: Simple Two-Round Schnorr Multi-signatures. In *Advances in Cryptology – CRYPTO 2021, Part I (Lecture Notes in Computer Science)*, Tal Malkin and Chris Peikert (Eds.), Vol. 12825. Springer, Heidelberg, Germany, Virtual Event, 189–221. https://doi.org/10.1007/978-3-030-84242-0_8

[66] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology – EUROCRYPT'99 (Lecture Notes in Computer Science)*, Jacques Stern (Ed.), Vol. 1592. Springer, Heidelberg, Germany, Prague, Czech Republic, 223–238. https://doi.org/10.1007/3-540-48910-X_16

[67] Pascal Paillier and Damien Vergnaud. 2005. Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log. In *Advances in Cryptology – ASIACRYPT 2005 (Lecture Notes in Computer Science)*, Bimal K. Roy (Ed.), Vol. 3788. Springer, Heidelberg, Germany, Chennai, India, 1–20. https://doi.org/10.1007/11593447_1

[68] Alexandre Ruiz and Jorge L. Villar. 2005. Publicly verifiable secret sharing from paillier's cryptosystem. In *WEWoRC 2005 – Western European Workshop on Research in Cryptology*, Christopher Wulf, Stefan Lucks, and Po-Wah Yau (Eds.). Gesellschaft für Informatik e.V., Bonn, 98–108.

[69] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar R. Weippl. 2021. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. In *ISOC Network and Distributed System Security Symposium – NDSS 2021*. The Internet Society, Virtual.

[70] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar R. Weippl. 2020. HydRand: Efficient Continuous Distributed Randomness. In *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 73–89. https://doi.org/10.1109/SP40000.2020.00003

[71] Berry Schoenmakers. 1999. A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic. In *Advances in Cryptology – CRYPTO'99 (Lecture Notes in Computer Science)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 148–164. https://doi.org/10.1007/3-540-48405-1_10

[72] Yannick Seurin. 2012. On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model. In *Advances in Cryptology – EUROCRYPT 2012 (Lecture Notes in Computer Science)*, David Pointcheval and Thomas Johansson (Eds.), Vol. 7237. Springer, Heidelberg, Germany, Cambridge, UK, 554–571. https://doi.org/10.1007/978-3-642-29011-4_33

[73] Victor Shoup. 1997. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology – EUROCRYPT'97 (Lecture Notes in Computer Science)*, Walter Fumy (Ed.), Vol. 1233. Springer, Heidelberg, Germany, Konstanz, Germany, 256–266. https://doi.org/10.1007/3-540-69053-0_18

[74] Markus Stadler. 1996. Publicly Verifiable Secret Sharing. In *Advances in Cryptology – EUROCRYPT'96 (Lecture Notes in Computer Science)*, Ueli M. Maurer (Ed.), Vol. 1070. Springer, Heidelberg, Germany, Saragossa, Spain, 190–199. https://doi.org/10.1007/3-540-68339-9_17

[75] Douglas R. Stinson and Reto Strobl. 2001. Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates. In *ACISP 01: 6th Australasian Conference on Information Security and Privacy (Lecture Notes in Computer Science)*, Vijay Varadharajan and Yi Mu (Eds.), Vol. 2119. Springer, Heidelberg, Germany, Sydney, NSW, Australia, 417–434. https://doi.org/10.1007/3-540-47719-5_33

[76] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris-Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. 2017. Scalable Bias-Resistant Distributed Randomness. In *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Jose, CA, USA, 444–460. https://doi.org/10.1109/SP.2017.45

[77] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. 2016. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Jose, CA, USA, 526–545. https://doi.org/10.1109/SP.2016.38

[78] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. 2019. HotStuff: BFT Consensus with Linearity and Responsiveness. In *38th ACM Symposium Annual on Principles of Distributed Computing*, Peter Robinson and Faith Ellen (Eds.). Association for Computing Machinery, Toronto, ON, Canada, 347–356. https://doi.org/10.1145/3293611.3331591

A WARM-UP: PVSS SCHEMES & PLAIN UNPREDICTABILITY

Publicly Verifiable Secret Sharing (PVSS). In a VSS scheme, a dealer distributes shares of a secret among a group of parties such that it can be reconstructed only if a threshold of these parties collaborate. In a PVSS scheme, any third party can verify the correctness of the sharing, thus avoiding the need for a complaint phase as in VSS schemes. In the following, we define a (non-interactive) PVSS scheme.

Definition A.1 (PVSS Scheme). Let $\hat{\mathbb{G}}$ be a cyclic group of prime order p specified by par. A (t, n)*-threshold PVSS scheme over* $\hat{\mathbb{G}}$ is a tuple of algorithms $\mathsf{PVSS} = (\mathsf{Keys}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Dist}, \mathsf{Ver}, \mathsf{Rec})$ with the following properties:

- Keys: The randomized *key generation algorithm* takes as input system parameters par and an identity index $i \in [n]$. It outputs a public key pk_i and a secret key sk_i.
- Enc: The randomized *encryption algorithm* that takes as input a public key pk_i and a message m. It outputs a ciphertext c.
- Dec: The deterministic *decryption algorithm* takes as input a secret key sk_i and a ciphertext c. It outputs a message m (optionally with a proof of correct decryption). We require that for all messages m,
$$\Pr[\mathsf{Dec}_{sk_i}(\mathsf{Enc}_{pk_i}(m)) = m] = 1.$$
- Dist: The randomized *secret sharing algorithm* takes as input public keys $pk_1, \ldots, pk_n$. It outputs a vector of encrypted shares $\vec{E} = (\mathsf{Enc}_{pk_1}(S_1), \ldots, \mathsf{Enc}_{pk_n}(S_n))$ and a proof π, where $S_1, \ldots, S_n$ are shares of a secret $S \in \hat{\mathbb{G}}$. We refer to $T := (\vec{E}, \pi)$ as a *PVSS transcript.*
- Ver: The deterministic *verification algorithm* takes as input public keys $pk_1, \ldots, pk_n$, and a PVSS transcript $T = (\vec{E}, \pi)$. It outputs 1 (accept) or 0 (reject). In the first case we call the transcript T *valid* (relative to $pk_1, \ldots, pk_n$); otherwise we call it *invalid.*
- Rec: The deterministic *reconstruction algorithm* takes as input $t + 1$ shares $S_1, \ldots, S_{t+1}$. It outputs a reconstructed secret $S \in \hat{\mathbb{G}}$. *In case* Rec *gets more than* $t + 1$ *shares as input, it takes the first lexicographical* $t + 1$.

When the parameters t, n, and $\hat{\mathbb{G}}$ are clear from the context, we will sometimes refer to a (t, n)-threshold PVSS scheme over $\hat{\mathbb{G}}$ simply as a PVSS scheme. Next, we define correctness and verifiability notions for a PVSS scheme:

Definition A.2 (Correctness and Verifiability). Let $\mathsf{PVSS} = (\mathsf{Keys}, \mathsf{Dist}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver}, \mathsf{Rec})$ be a (t, n)-threshold PVSS scheme over $\hat{\mathbb{G}}$. We define the following correctness and verifiability notions for PVSS.

- *Correctness (of PVSS).* We say that PVSS is *correct* if for all $(pk_1, sk_1), \ldots, (pk_n, sk_n) \in \mathsf{Keys}(par)$,
$$\Pr[\mathsf{Ver}(\{pk_i\}_i, \mathsf{Dist}(\{pk_i\}_i) = 1] = 1.$$
- *Public Verifiability (of Transcripts).* We say that PVSS is *publicly verifiable* if for all $(pk_1, sk_1), \ldots, (pk_n, sk_n) \in \mathsf{Keys}(par)$ and all $(\vec{E}, \pi)$ s.t. $\mathsf{Ver}(\{pk_i\}_i, (\vec{E}, \pi)) = 1$, there exists a unique $S \in \hat{\mathbb{G}}$ s.t.
$$\mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}_i)\}_{i \in \mathcal{I}}) = S \quad \forall \mathcal{I} \subset [n], |\mathcal{I}| = t + 1.$$

We introduce a new security notion for PVSS schemes. The notion of *unpredictability* prohibits an adversary controlling t parties from learning the secret by observing a transcript. This models a passive adversary who can observe distributions of transcripts, but does not contribute itself to the final secret. The notion of *aggregated unpredictability* introduced in the main body of the paper is a non-malleability kind of property specifically for aggregatable PVSS schemes. It prohibits an adversary controlling t parties from learning the secret of an aggregated transcript with at least one honest contribution, even if the adversary is allowed to contribute itself to the aggregate. This models an active adversary who can contribute to the final secret itself. In the following, we define the notion of unpredictability formally.

Definition A.3 (Unpredictability of PVSS Scheme). Let $\mathsf{PVSS} = (\mathsf{Keys}, \mathsf{Dist}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver}, \mathsf{Rec})$ be a (t, n)-PVSS scheme over $\hat{\mathbb{G}}$. For an algorithm A, define the *unpredictability* experiment $\mathbf{Pred}^{\mathsf{A}}_{\mathsf{PVSS},t}$ as follows:

- *Offline Phase.* For all $i \in [n]$, run Keys on input (par, i) to generate keys $(pk_i, sk_i) \leftarrow \mathsf{Keys}(par, i)$. On input par and $\{pk_i\}_{i \in [n]}$, A returns an index set $C \subset [n]$ of initially corrupted parties along with updated public keys $\{\hat{pk}_j\}_{j \in C}$. Set $pk_j := \hat{pk}_j$ for all $j \in C$.
- *Corruption Queries.* At any point of the experiment, A may submit an index $i \in [n] \setminus C$. In this case, return the secret key sk_i and update $C = C \cup \{i\}$. *If* A *is static, it submits an index set* $C' \subset [n] \setminus C$ *at the beginning of the experiment. Return the secret keys* $\{sk_i\}_{i \in C'}$ *and update* $C := C' \cup C$.
- *Random Oracle Queries.* At any point of the experiment, A gets access to an oracle that answers queries of the following type: When A submits a query m, check if $H[m] = \perp$. If so, set $H[m] \leftarrow \mathbb{Z}_p^*$ and return $H[m]$. Otherwise, return $H[m]$.
- *Challenge Phase.* Run Dist on input $pk_1, \ldots, pk_n$ to obtain the challenge transcript $T = (\vec{E}, \pi)$. Run A on input T.
- *Output Determination.* Let $S = \mathsf{Rec}(\{\mathsf{Dec}_{sk_i}(\vec{E}_i)\}_{i \le t+1})$. When A outputs an element $S^* \in \hat{\mathbb{G}}$, return 1 if $|C| \le t$ and $S^* = S$. Otherwise, return 0.

We say that PVSS is (ε, T, t, q_h)*-unpredictable* if for all algorithms A that run in time at most T, make at most q_h random oracle queries, $\Pr[\mathbf{Pred}^{\mathsf{A}}_{\mathsf{PVSS},t} = 1] \le \varepsilon$. Conversely, we say that A (ε, T, t, q_h)-*breaks unpredictability* of PVSS if it runs in time at most T, makes at most q_h random oracle queries, and $\Pr[\mathbf{Pred}^{\mathsf{A}}_{\mathsf{PVSS},t} = 1] > \varepsilon$.

A.1 Security Analysis of PVSS in the AGM, Schoenmakers' Scheme

In this section, we give a tight security reduction from the hardness of n-OMDL to the unpredictability of Schoenmakers' PVSS. In the following, let $\mathbb{G}$ be a cyclic group of prime order p with independent generators g, h. We denote by DL_g an oracle that on input $\xi = g^z \in \mathbb{G}$ returns the discrete logarithm z of ξ to base g.

The One-More Discrete Logarithm Assumption. A mathematical hardness assumption that finds wide-ranging application in modern cryptography is the *one-more discrete logarithm (OMDL) assumption* [11]. It is the foundation for the security analysis of identification protocols [10, 13, 49], blind signature [43, 44] and multi-signature schemes [12, 65], such as blind Schnorr signatures [28]. Beyond that, OMDL is also assumed for various impossibility results of certain classes of reductions [39, 41, 42, 67, 72]. Recently, it has also found applications in the context of the adaptive security of distributed protocols [2, 7, 33].

Definition A.4 (One-More Discrete Logarithm Problem). For an algorithm A and $n \in \mathbb{N}$, define the one-more discrete logarithm experiment n-$\mathbf{OMDL}^{\mathsf{A}}$ in $\mathbb{G}$ as follows:

- *Offline Phase.* Sample $(z_1, \ldots, z_n) \leftarrow \mathbb{Z}_p^n$ uniformly at random and set $\xi_i := g^{z_i} \in \mathbb{G}$ for all $i \in [n]$.
- *Online Phase.* Run A on input $(par, \xi_1, \ldots, \xi_n)$. In this phase, A gets access to the oracle DL_g.
- *Output Determination.* When A outputs $(z'_1, \ldots, z'_n)$, return 1 if (i) $z'_i = z_i$ for all $i \in [n]$, and (ii) DL_g was queried at most $n-1$ times during the online phase. Otherwise, return 0.

We say that the one-more discrete logarithm problem of degree n is (ε, T)-hard if for all algorithms A running in time at most T, $\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}} = 1] \leq \varepsilon$. Conversely, we say that an algorithm A (ε, T)-solves the one-more discrete logarithm problem of degree n if it runs in time at most T and $\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}} = 1] > \varepsilon$.

In the following, we provide an intuition for our proof of unpredictability. For this, we start with the observation that an adversary controlling t parties essentially has three options to successfully predict the secret $S \in \mathbb{G}$. Firstly, it learns an additional $(t+1)$-th decryption key controlled by an honest party, in which case it can derive S from enough decryptions of secret shares. Secondly, it breaks the underlying encryption scheme directly and thus obtains an additional secret share. Lastly, it finds the discrete logarithm e of the second generator $h \in \mathbb{G}$ to base g, in which case it can compute the secret $S = h^{\alpha}$ directly from the commitment g^{α}. The key idea of our reduction therefore is to embed the n-OMDL challenge ξ in the public keys $pk_1, \ldots, pk_n$ of parties, the polynomial $f \in \mathbb{Z}_p[X]$ chosen by the dealer, or the second generator $h \in \mathbb{G}$, a choice that remains hidden from the adversary. In the first case, we simulate by using the discrete logarithm oracle DL_g to answer corruption queries. In the second case, we simulate by using an honest-verifier zero knowledge simulation in the random oracle to generate the NIZK proofs of correctness of the sharing. In the third case, we simulate by an honest execution of the protocol. In either case, we solve the n-OMDL challenge ξ directly from the algebraic equation that comes from the secret prediction/forgery (with its algebraic representation) by the adversary. Overall, our reduction is tight and loses only a factor of 1/4. The running time of the reduction has only a quadratic overhead.

Theorem A.5. *If n-$\mathbf{OMDL}$ is (ε, T)-hard in the AGM, then Schoenmakers' PVSS is $(\varepsilon', T', t, q_h)$-unpredictable in the AGM & ROM, where*

$$\varepsilon \geq \frac{\varepsilon'}{4}, \quad T \leq T' + O(n^2).$$

Proof. Let A be an algebraic adversary that $(\varepsilon', T', t, q_h)$-breaks unpredictability of PVSS. In our proof, we assume that all parties are honest prior to the execution of PVSS. It is easy to adjust the proof to the case where the adversary has already corrupted some parties before the execution of the protocol. Let $\mathcal{C} \subset \mathcal{P} = \{P_1, \ldots, P_n\}$ be the dynamically changing set of corrupt parties and $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$ the set of honest parties. In particular, we assume that $\mathcal{C} = \{\}$ prior to the execution of the protocol. We consider the following game between a challenger and the adversary.

Game $\mathbf{G}$: This is the real game. The challenger runs PVSS on behalf of the honest parties and the designated dealer. In particular, it generates the system parameters $(\mathbb{G}, p, g, h)$, where $\mathbb{G}$ is a cyclic group of order p, and g and h are two independent generators of $\mathbb{G}$, and a uniformly at random chosen polynomial $P(X) = \alpha_0 + \alpha_1 X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ of degree t such that $h^{\alpha} = h^{P(0)}$ is the secret to be shared among all parties. Furthermore, the challenger generates the public-secret key pairs $(pk_i, sk_i) = (h^{x_i}, x_i)$ of the honest parties. Whenever A decides to corrupt a party P_i, the challenger honestly returns the internal state of that party, which consists of P_i's secret key $x_i = sk_i$, and sets $\mathcal{C} = \mathcal{C} \cup \{P_i\}$, $\mathcal{H} = \mathcal{H} \setminus \{P_i\}$. In addition, A gets full control over party P_i. Random oracle queries m_i are answered by sampling $r_i \leftarrow \mathbb{Z}_p$ uniformly at random and returning $H[m_i] = r_i$. The challenger broadcasts the commitments $C_i = g^{\alpha_i}$ for all $i \in [\![t]\!]$ and the encrypted shares $Y_i = pk_i^{P(i)}$ for all $i \in [n]$ using the public keys pk_i of parties. Let $X_i = g^{P(i)}$ for all $i \in [n]$, which can be computed by Lagrange interpolation in the exponent from the commitments C_j. Additionally, the challenger broadcasts Chaum-Pedersen non-interactive zero-knowledge (NIZK) proofs π_i of knowledge of the polynomial shares $P(i)$ for all $i \in [n]$ such that $X_i = g^{P(i)}$ and $Y_i = pk_i^{P(i)}$. The common challenge c for the proof is computed as the hash $\mathsf{H} : \mathbb{G} \to \mathbb{Z}_p$ of all elements $X_i, Y_i, g^{w_i}, pk_i^{w_i}$, $i \in [n]$, where $w_i \leftarrow \mathbb{Z}_p$ is chosen uniformly at random. The NIZK π consists of c and the n responses $s_i := w_i - P(i)c$ for $i \in [n]$. At the end of the game, A outputs a secret $\sigma^* \in \mathbb{G}$.

As A is an algebraic adversary, at the end of game $\mathbf{G}$ it returns a secret σ^* together with a representation

$$(\hat{a}, \hat{b}, a_1, \ldots, a_n, b_0, \ldots, b_t, c_1, \ldots, c_n)$$

of elements in $\mathbb{Z}_p$ such that

$$\sigma^* = g^{\hat{a}} \cdot h^{\hat{b}} \cdot pk_1^{a_1} \cdot \ldots \cdot pk_n^{a_n} \cdot C_0^{b_0} \cdot \ldots \cdot C_t^{b_t} \cdot Y_1^{c_1} \cdot \ldots \cdot Y_n^{c_n}.$$

Here, the representation is split (from left to right) into powers of the generators g, h, the public keys $pk_1, \ldots, pk_n$, the polynomial commitments $C_0, \ldots, C_t$, and the encrypted shares $Y_1, \ldots, Y_n$. In the following, let $e \in \mathbb{Z}_p$ denote the discrete logarithm of h to base g (i.e. $g^e = h$). Assuming the adversary wins the game $\mathbf{G}$ by outputting the secret $h^{P(0)} = g^{eP(0)}$ chosen by the challenger, the above equation is with $\alpha = P(0)$ equivalent to

$$e\alpha = \hat{a} + e\hat{b} + e\sum_{i=1}^{n} x_i a_i + \sum_{i=0}^{t} \alpha_i b_i + e\sum_{i=1}^{n} x_i P(i) c_i. \qquad (\spadesuit)$$

We define the following three events:

- E_1 defined by $\alpha = \hat{b} + \sum_{i=1}^{n} x_i a_i + \sum_{i=1}^{n} x_i P(i) c_i$.
- E_2 defined by the identity $1 = \sum_{i=1}^{n} x_i c_i$.

Let $\mathbb{G}$ be a cyclic group of prime order p and independent generators $g, h \in \mathbb{G}$. The protocol takes as input a tuple (g, X, h, Y) of group elements in $\mathbb{G}$ and outputs a NIZK proof of knowledge of an $\alpha \in \mathbb{Z}_p^*$ such that $X = g^\alpha$ and $Y = h^\alpha$ holds.

(1) Sample $s \leftarrow \mathbb{Z}_p^*$ uniformly at random and compute the challenge $c \in \mathbb{Z}_p^*$ as $c := \mathsf{H}(X, g^s, Y, h^s)$.
(2) Compute the response $r \in \mathbb{Z}_p^*$ as $r := s - \alpha c$ and output the proof of knowledge $\pi := (c, r)$.
(3) The NIZK proof of knowledge $\pi = (c, r)$ is valid if and only if equality $c = \mathsf{H}(X, g^r X^c, Y, h^r Y^c)$ holds.

Figure 4: Non-interactive zero-knowledge proof Dleq **for equality of discrete logarithms.**

Let $\mathbb{G}$ be a cyclic group of prime order p and independent generators $g, h \in \mathbb{G}$. Let (pk_i, sk_i) be the key pair of party P_i with $pk_i = h^{sk_i}$. The dealer P_L with key pair (pk_L, sk_L) wants to share secret h^α for an $\alpha \leftarrow \mathbb{Z}_p^*$. The Dist algorithm takes as input sk_L and public keys $pk_1, \dots, pk_n$. It outputs the transcript $T_L := \{C_i, Y_i, \pi\}_{i \in [n]}$ defined as follows.

(1) Choose a polynomial $f(X) = \alpha + \alpha_1 X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ of degree t uniformly at random.
(2) Publish commitments $C_i = g^{\alpha_i} \in Gr$ for $i \in [\![t]\!]$. Also publish encrypted shares $Y_i = pk_i^{f(i)} \in \mathbb{G}$ for $i \in [n]$. Let $X_i := \prod^{j \in [\![t]\!]} C_j^{i^j}$.
(3) Compute NIZK proofs $\pi_i := \mathsf{Dleq}(g, X_i, pk_i, Y_i)$ for $i \in [n]$ using the simultaneous challenge $c := \mathsf{H}(\{X_i, g^{w_i}, Y_i, pk_i^{w_i}\}_{i \in [n]})$. Compute the responses $r_i \in \mathbb{Z}_p^*$ as $r_i := w_i - f(i)c$ for $i \in [n]$, and publish the proof $\pi := (c, r_1, \dots, r_n)$.

The *transcript verification algorithm* Ver takes as input the public keys $pk_1, \dots, pk_n$ (including pk_L) and transcript T_L. It outputs 1 (accept) or 0 (reject).

(4) Compute $X_i := \prod^{j \in [\![t]\!]} C_j^{i^j}$ for $i \in [n]$. Also compute $g^{w_i} = g^{r_i} X_i^c$ and $pk_i^{w_i} = y_i^{r_i} Y_i^c$ for $i \in [n]$ using $\{X_i, Y_i, pk_i, r_i, c\}_{i \in [n]}$.
(5) Check that equality $c = \mathsf{H}(\{X_i, g^{w_i}, Y_i, pk_i^{w_i}\}_{i \in [n]})$ holds and thus the cumulated NIZK proof π verifies.
(6) If one of the above checks fails, output 0 (invalid transcript). Otherwise, output 1 (valid transcript).

Figure 5: Distribution protocol Dist **and transcript verification algorithm** Ver **of Schoenmakers' PVSS.**

On input the encrypted shares $Y_1, \dots, Y_n$, the decryption Dec and reconstruction Rec algorithms work as follows.

(1) Using sk_i, compute the secret share $S_i = h^{f(i)}$ from Y_i via extracting the root $S_i = Y_i^{1/sk_i}$. Also provide a proof of correct decryption $\theta_i := \mathsf{Dleq}(h, h^r, S_i, S_i^r)$ for an $r \leftarrow \mathbb{Z}_p^*$ sampled uniformly at random. Publish (S_i, θ_i).
(2) Upon receiving a secret share tuple (S_ℓ, θ_ℓ) from party P_ℓ, check that the proof θ_ℓ verifies. Otherwise, the secret share is invalid.
(3) Upon receiving $t + 1$ valid secret shares $S_j = h^{f(j)}$ from different parties, compute the secret $S = h^{f(0)}$ via Lagrange interpolation in the exponent and output.

Figure 6: Decryption Dec **and reconstruction** Rec **algorithms of Schoenmakers' PVSS.**

- E_3 defined by: There is an index $i \in \mathcal{H}$ s.t. $c_i \neq 0$.[4]

We have the following technical lemma.

Lemma A.6. *Let* $\mathbf{G}$ *and* E_i *for* $i = 1, 2, 3$ *be defined as above. Then there exist (algebraic) algorithms* A_j *for* $j \in [4]$ *playing in game* n-**OMDL** *that run in time at most* T *such that:*

$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_1} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \land \neg E_1],$$
$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_2} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \land E_1 \land \neg E_2],$$
$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_3} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \land E_1 \land E_2 \land \neg E_3],$$
$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_4} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \land E_1 \land E_2 \land E_3].$$

Moreover, $T \le T' + O(n^2)$.

Proof. Let $\xi = (\xi_1, \dots, \xi_n) \in \mathbb{G}^n$ with $\xi = g^{z_i}$ for $i \in [n]$ be the OMDL instance of degree n. Algorithms A_i for $i \in [4]$ have access to a (perfect) discrete logarithm oracle DL_g (to base g) which they can query at most $n - 1$ times. The algorithms A_i, $i \in [4]$, simulate game $\mathbf{G}$ as described in the following.

[4]At this stage, $\mathcal{H} \subset \mathcal{P}$ is the set of parties that remain honest at the end of the game.

Algorithm $\mathsf{A}_1(\xi, par)$: Algorithm A_1 works as follows. On input ξ, A_1 queries the discrete logarithm oracle DL_g on $\xi_2, \dots, \xi_n$ and gets $(z_2, \dots, z_n)$. It publishes the second generator h by setting $h = \xi_1$. In particular, it is $e = \mathsf{DL}_g(h) = z_1$. Furthermore, A_1 generates the public-secret key pairs of parties and the polynomial $P(X) \in \mathbb{Z}_p[X]$ honestly (by sampling $sk_i, \alpha_j \leftarrow \mathbb{Z}_p$ uniformly at random). Commitments X_i, encrypted shares Y_i, and NIZK proofs π are computed honestly and published. Random oracle queries m_i are answered honestly by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning $H[m_i] = r_i$. Corruption queries are answered by returning the secret key of the corresponding party. It is not hard to see that A_1's simulation of $\mathbf{G}$ is perfect.

Suppose that A_1 wins $\mathbf{G}$ and that event $\neg E_1$ happens. Equation (♠) is then equivalent to

$$e = \left(\hat{a} + \sum_{i=0}^{t} \alpha_i b_i\right) \left(\alpha - \hat{b} - \sum_{i=1}^{n} x_i a_i - \sum_{i=1}^{n} x_i P(i) c_i\right)^{-1},$$

since the second factor is non-zero. As a result, A_1 can efficiently compute $e = z_1$ and solve the OMDL instance. Overall, we obtain

$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_1} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \wedge \neg E_1].$$

The bound on the running time of A_1 (number of group operations and exponentiations) is obvious.

Algorithm $\mathsf{A}_2(\xi, par)$: Algorithm A_2 works on input $\xi_i = g^{z_i}$, $i \in [n]$, as follows. It samples $e \leftarrow \mathbb{Z}_p$ uniformly at random and publishes $h = g^e$. It generates the public-secret key pairs of parties honestly. It chooses the polynomial $P(X) = \alpha_0 + \alpha_1 X + \ldots + \alpha_t X^t$ such that $g^{\alpha_i} = \xi_{i+1}$ for all $i \in [\![t]\!]$. In particular, it is $\alpha_i = z_{i+1}$ for all $i \in [\![t]\!]$. The commitments $X_i = g^{\alpha_i} = \xi_i$ are published without the need of computation. The encrypted shares $Y_i = pk_i^{P(i)}$ are computed as $Y_i = (h^{P(i)})^{x_i}$ and published. Note that Lagrange interpolation in the exponent allows A_2 to compute $g^{P(i)}$ for any $i \in [n]$ from the commitments C_j, $j \in [\![t]\!]$. The NIZK proofs π are created via an honest-verifier zero-knowledge (HVZK) simulation using the random oracle. Corruption queries are answered by returning the secret key of the corresponding party. Random oracle queries m_i are answered honestly by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning $H[m_i] = r_i$. It is not hard to see that A_2's simulation of $\mathbf{G}$ is perfect.

Suppose that A_2 wins $\mathbf{G}$ and that event $E_1 \wedge \neg E_2$ happens. With the notation $P(X) = \alpha_0 + q(X)$ for polynomial $q(X) = \alpha_1 X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$, the equation defined by event E_1 then reduces to

$$\alpha = \hat{b} + \sum_{i=1}^{n} x_i a_i + \sum_{i=1}^{n} x_i q(i) c_i + \alpha \sum_{i=1}^{n} x_i c_i.$$

With the condition that $\neg E_2$ is satisfied, this equation is equivalent to

$$\alpha = \left(\hat{b} + \sum_{i=1}^{n} x_i a_i + \sum_{i=1}^{n} x_i q(i) c_i\right)\left(1 - \sum_{i=1}^{n} x_i c_i\right)^{-1}. \qquad (\blacklozenge)$$

Algorithm A_2 proceeds as follows. It queries the discrete logarithm oracle DL_g on $\xi_2, \ldots, \xi_n$ and obtains $(z_2, \ldots, z_n)$. In particular, it knows $\alpha_1 = z_2, \ldots, \alpha_t = z_{t+1}$ and thus $q(i)$ for all $i \in \mathbb{Z}_p$ (i.e. it knows the coefficients of the polynomial $q(X)$). From identity ($\blacklozenge$), A_2 can efficiently compute $z_1 = \alpha$ and solve the OMDL instance. Overall, we obtain

$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_2} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \wedge E_1 \wedge \neg E_2].$$

The bound on the running time is obvious, since A_2 has to compute the group elements $q^{P(i)}$ for all $i \in [n]$ by Lagrange interpolation in the exponent from the commitments, which results in additional $O(n^2)$ running time.

Algorithm $\mathsf{A}_3(\xi, par)$: Algorithm A_3 works on input $\xi_i = g^{z_i}$, $i \in [n]$, as follows. The simulation of the game $\mathbf{G}$ is identical to that of A_2; in particular, A_3 chooses $P(X) \in \mathbb{Z}_p[X]$ such that $g^{\alpha_i} = \xi_{i+1}$ for all $i \in [\![t]\!]$. The only difference lies in how A_3 extracts the solution to the OMDL instance after the game has finished. Again, A_3's simulation of $\mathbf{G}$ is perfect.

Suppose that A_3 wins $\mathbf{G}$ and that event $E_1 \wedge E_2 \wedge \neg E_3$ happens. With the same notation $P(X) = \alpha_0 + q(X)$ as before, the equations defined by events E_1 and E_2 then reduce to

$$0 = \hat{b} + \sum_{i=1}^{n} x_i a_i + \sum_{i=1}^{n} x_i q(i) c_i, \quad 1 = \sum_{i=1}^{n} x_i c_i, \qquad (\clubsuit)$$

where $c_i = 0$ for all $i \in \mathcal{H}$ by condition $\neg E_3$. Therefore,

$$1 = \sum_{i=1}^{n} x_i c_i = \sum_{i \in C} x_i c_i$$

implies that there is an index $\tilde{j} \in C$ such that $x_{\tilde{j}} c_{\tilde{j}} \neq 0$. The first equation in ($\clubsuit$) is then equivalent to

$$q(\tilde{j}) = -\frac{1}{x_{\tilde{j}} c_{\tilde{j}}} \cdot \left(\hat{b} + \sum_{i=1}^{n} x_i a_i + \sum_{i \in C \setminus \{\tilde{j}\}} x_i q(i) c_i\right).$$

Algorithm A_3 proceeds as follows. It queries the discrete logarithm DL_g on $\xi_1, \xi_{t+2}, \ldots, \xi_n$ and $g^{q(i)}$ for all $i \in C \setminus \{\tilde{j}\}$. Note that since $g^{P(X)} = g^{\alpha_0} \cdot g^{q(X)} = \xi_1 \cdot g^{q(X)}$, algorithm A_3 can compute (and hence query) $g^{q(i)}$ for any $i \in \mathbb{Z}_p$. W.l.o.g. we may assume that $|C| = t$ (otherwise, A_3 simply simulates, for itself, $t - |C|$ corruption queries for random parties from $\mathcal{H}$). As a result, A_3 can compute $q(\tilde{j})$ and has finally knowledge of $t+1$ points in the range of $[n]$ on the polynomial $q(X)$ of degree t. In particular, A_3 knows the coefficients of q, i.e. $\alpha_1 = z_2, \ldots, \alpha_t = z_{t+1}$. From previous oracle queries it knows $z_1, z_{t+2}, \ldots, z_n$ and thus solves the OMDL instance with $1 + (n - t - 1) + (t - 1) = n - 1$ queries to DL_g. Overall, we obtain

$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_3} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge \neg E_3].$$

The bound on the running time of algorithm A_3 is obvious from the previous case.

Algorithm $\mathsf{A}_4(\xi, par)$: Algorithm A_4 works on input $\xi_i = g^{z_i}$, $i \in [n]$, as follows. It samples $e \leftarrow \mathbb{Z}_p$ uniformly at random and publishes $h = g^e$. It generates the polynomial $P(X) \in \mathbb{Z}_p[X]$ honestly by sampling $\alpha_i \leftarrow \mathbb{Z}_p$ for all $i \in [\![t]\!]$ uniformly at random. It chooses party P_j's public key as $pk_j = \xi_j$ for all $j \in [n]$. In particular, it is $x_j = sk_j = z_j$ for all $j \in [n]$. Commitments X_i, encrypted shares Y_i, and NIZK proofs π are computed honestly and published (which is possible, since the polynomial $P(X)$ is completely known to A_4). Random oracle queries m_i are answered honestly by sampling $r_i \leftarrow \mathbb{Z}_p$ and returning $H[m_i] = r_i$. Corruption queries are answered with the help of the discrete logarithm oracle DL_g. A corruption query on party P_j is answered by computing $\mathsf{DL}_g(pk_j)/e$ and returning the secret key sk_j (note that $pk_j = h^{sk_j}$ and the oracle returns the discrete logarithm to base g so that we have to divide the result by e afterwards to get sk_j). It is not hard to see that A_4's simulation of $\mathbf{G}$ is perfect.

Suppose that A_4 wins $\mathbf{G}$ and that event $E_1 \wedge E_2 \wedge E_3$ happens. The equation $1 = \sum_{i=1}^{n} x_i c_i$ defined by event E_2 together with condition E_3 imply that there is an index $\tilde{i} \in \mathcal{H}$ such that

$$x_{\tilde{i}} = \frac{1}{c_{\tilde{i}}} \sum_{i \neq \tilde{i}} x_i c_i. \qquad (\heartsuit)$$

Algorithm A_4 proceeds as follows. It queries the discrete logarithm oracle DL_g on ξ_j for all $j \in \mathcal{H} \setminus \{\tilde{i}\}$ and obtains x_j for all $j \in \mathcal{H} \setminus \{\tilde{i}\}$. Therefore, A_4 has knowledge of x_j for all $j \in \mathcal{H} \setminus \{\tilde{i}\} \cup C = \mathcal{P} \setminus \{\tilde{i}\}$ and computes the remaining value $x_{\tilde{i}}$ by the above identity ($\heartsuit$). As a result, it solves the OMDL instance with $n - 1$ oracle queries. Overall, we obtain

$$\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_4} = 1] = \Pr[\mathbf{G}^{\mathsf{A}} = 1 \wedge E_1 \wedge E_2 \wedge E_3].$$

The claim on the running time of A_4 is clear. □

To end the proof, consider algorithm B playing in n-**OMDL** as follows: B samples $i^* \leftarrow [4]$ and then internally emulates algorithm A_{i^*}. Clearly, B is an algebraic algorithm running in time at most T (the running time of A_i, $1 \leq i \leq 4$). An application of the law of total probability yields

$$\begin{aligned}\Pr[n\text{-}\mathbf{OMDL}^{\mathsf{B}} = 1] &= \sum_{i=1}^{4} \Pr[n\text{-}\mathbf{OMDL}^{\mathsf{B}} = 1 \mid i^* = i] \cdot \Pr[i^* = i] \\ &= \frac{1}{4} \sum_{i=1}^{4} \Pr[n\text{-}\mathbf{OMDL}^{\mathsf{A}_i} = 1] \\ &\geq \frac{1}{4} \Pr[\mathbf{G}^{\mathsf{A}} = 1] = \frac{\varepsilon'}{4}.\end{aligned}$$

□

Remark A.1. By breaking down our above proof, we find that it can be adapted to obtain a security reduction (with the same parameters for tightness) from the plain discrete logarithm problem assuming a weaker static adversary. The main idea being to embed the discrete logarithm challenge $\xi \in \mathbb{G}$ into the public keys $\{pk_i\}_{i \in \mathcal{H}}$ of honest parties (which is fixed from the very beginning in the static corruption model) via $pk_i = \xi^{u_i} g^{v_i}$ for uniformly random $u_i, v_i \leftarrow \mathbb{Z}_p$, into the polynomial $f(X) = \alpha_0 + \alpha X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ chosen by the dealer via $g^{\alpha_i} = \xi^{u_i} g^{v_i}$ for uniformly random $u_i, v_i \leftarrow \mathbb{Z}_p$, or into the second generator $h \in \mathbb{G}$ via $h = \xi$. Using this technique, the above proof can be adapted accordingly for the static case to reduce the security from the plain discrete logarithm problem.

A.2 Application to Randomness Beacons

We discuss the adaptive security of the previous state-of-the-art randomness beacon protocol from the literature, GRandPiper [16] in the synchronous network model. The protocol employs an unspecified PVSS scheme in its design. Thus by instantiating their generic construction with Schoenmakers' PVSS, this results in the adaptive security of the randomness beacon. We briefly discuss the randomness beacon.

GRandPiper. The protocol employs an unspecified PVSS scheme (any secure PVSS scheme can be used) and a leader-based SMR protocol with a communication cost of $O(n\ell + \lambda n^2)$ bits per consensus decision on a block of size ℓ. In each epoch $e \geq 1$, the leader L_e puts a randomly sampled PVSS transcript on the ledger. If L_e does not put anything on the ledger or the transcript is invalid, parties blacklist L_e from future leader elections. Apart from that, parties adhere to a randomized election with blacklisting. Concretely, the leader for epoch e' is chosen based on the beacon output $O_{e'-1}$ and by removing the leaders $L_{e'-1}, \ldots, L_{e'-t}$ of the previous t epochs. Incorrectly behaving leaders are blacklisted from future leader elections. When the same party is elected as a leader $L_{e'}$ in epoch $e' > e + t$ the next time, parties take its previously published (valid) transcript and reconstruct the secret S_e. The beacon output $O_{e'}$ for epoch e' is computed as hash $O_{e'} = \mathsf{Hash}(O_{e'-1}, \ldots, O_{e'-t}, S_e)$. Finally, to ensure availability the first time a party is elected as the leader, the protocol relies on a setup where parties start with agreed-upon buffers $\mathcal{B}(P_i)$ for $i \in [n]$ that contain random PVSS transcripts each. Ignoring the pre-processing phase for the buffers, GRandPiper outputs a randomness beacon value with a communication cost of $O(\lambda n^2)$ bits and optimal resilience in the synchronous setting. However, we note that even if the underlying PVSS scheme was adaptively secure, the randomness beacon remains only $(t+1)$-unpredictable, since an adaptive adversary can predict the next t beacon values (by sequentially computing the next beacon value and corrupting the next leader). The reason for that is that single transcripts are proposed by leaders and not aggregated transcripts.

Theorem A.7. *If Schoenmakers'* PVSS *is* (ε, T, t, q_h)*-unpredictable in the AGM & ROM, then GRandPiper is an* $(\varepsilon', T', t, L, q_h', t{+}1)$*-secure randomness beacon protocol in the AGM & ROM, where*

$$\varepsilon \geq \frac{\varepsilon'}{L} - \frac{q_h'}{p}, \quad T \leq T' + O(Ln^2).$$

Proof. The proof follows along the same lines as the proofs for OptRand and SPURT in Appendix F.1 (for Theorem 4.2). □

B RELATED WORK ON RANDOMNESS BEACONS

In this appendix, we discuss existing works on distributed randomness beacon protocols from the literature.

Randomness Beacons. Over the years, numerous randomness beacon protocols have been proposed, each having its own set of strengths and weaknesses, depending on factors such as the network model, setup assumptions, desired efficiency and security properties, and reliance on cryptographic tools or specialized hardware. We briefly categorize several notable randomness beacon protocols found in the literature:

- DKG-based protocols: These randomness beacon protocols employ threshold cryptography to generate random values. More specifically, the randomness beacon output is a unique threshold signature on the hash of the epoch number. Typically, the threshold BLS signature is used in conjunction with an initial distributed key generation (DKG) phase. Most protocols in this category achieve a communication cost of $O(\lambda n^2)$ bits per beacon output when ignoring the setup phase. A significant drawback of these protocols is their reliance on a DKG, which implies an expensive setup phase and limited reconfiguration guarantees.
- VDF/PoW-based protocols: These randomness beacon protocols employ verifiable delay functions (VDFs) or Proof-of-Work (PoW) to generate random values. VDFs are functions that require a certain amount of time to compute but can be verified quickly. While VDF-based protocols can offer high efficiency, they require specialized hardware to compute the VDF, which might not be accessible to all parties. The same applies to PoW-based protocols, which have a very high computation cost. These protocols utilize heavy and expensive cryptographic or hardware tools.
- Other protocols: These randomness beacon protocols do not rely on a DKG and do not utilize heavy cryptographic tools such as VDFs. Most of these protocols employ a PVSS or a VSS to generate random values. As such, they are reconfiguration-friendly, efficient, and simple. With some exceptions, these protocols have cubic or even quartic communication cost.

Table 1: Comparison table of existing distributed randomness beacon protocols.

Protocol	Network	Resil.	Adap. Adv.	Unpred.	Bias-res.	Commun.	Crypto. Primit.	Assump.	Setup
Cachin et al. [21]	*async*	1/3	✗	✓	✓	$O(\lambda n^2)$	Uniq. Thresh. Signature	CDH	*DKG*
RandHerd [76]	*async*	1/3	✗	✓	✓	$O(\lambda c^2 \log(n))$	PVSS & Thresh. Schnorr	DL	*DKG*
Dfinity [4, 23]	*part sync*	1/3	✗	✓	✓	$O(\lambda n^2)$	Thresh. BLS	CDH	*DKG*
Herb [29]	*sync*	1/3	✗	✓	✓	$O(\lambda n^3)$	Thresh. ElGamal	DDH	*DKG*
Drand [1]	*sync*	1/2	✗	✓	✓	$O(\lambda n^2)$	Thresh. BLS	CDH	*DKG*
Algorand [50]	*part sync*	1/3	✗	$\Omega(t)$	✗	$O(\lambda cn)$	VRF	VRF	*CRS*
SPURT [34]	*part sync*	1/3	✗	✓	✓	$O(\lambda n^2)$	PVSS & Pairing	DBS	*CRS*
HydRand [70]	*sync*	1/3	✗	$t+1$	✓	$O(\lambda n^2)$	PVSS	DDH	*CRS*
Proof-of-Work [62]	*sync*	1/2	✗	$\Omega(t)$	✗	$O(\lambda n)$	Hash function	PoW	*CRS*
GRandPiper [16]	*sync*	1/2	✗	$t+1$	✓	$O(\lambda n^2)$	PVSS	SXDH	*q-SDH*
OptRand [15]	*sync*	1/2	✗	✓	✓	$O(\lambda n^2)$	PVSS & Pairing	SXDH	*q-SDH*
RandShare [76]	*async*	1/3	✓	✓	✓	$O(\lambda n^4)$	VSS	DL	*CRS*
RandChain [54]	*sync*	1/2	✓	$O(\lambda)$	✓	$O(\lambda n^2)$	Seq. PoW & Naka. Cons.	VDF & PoW	*CRS*
RandRunner [69]	*sync*	1/2	✓	$t+1$	✓	$O(\lambda n^2)$	Trapdoor VDF	tVDF & DL	*CRS*
BRandPiper [16]	*sync*	1/2	✓	✓	✓	$O(\lambda f n^2)$	VSS	SXDH	*q-SDH*

We explain the table. **Resil.** denotes the Byzantine resilience threshold. **Adap. Adv.** denotes adaptive adversary. **Unpred.** denotes unpredictability. **Bias.-res.** denotes bias-resistance. **Comm.** denotes communication complexity in bits. In RandHerd and Algorand, c denotes the average size of a randomly chosen committee. In BRandpiper, $f \leq t$ denotes the actual number of faults in the system. **Crypto. Primit.** denotes the cryptographic primitives in usage. **Assump.** denotes the underlying hardness assumption for the security proof. **Setup** denotes the setup assumption. We note that all protocols achieve availability.

Protocols in the first category include [1, 4, 21, 23, 29, 76]. Most of these works rely on the same idea from Cachin et al., in which a randomness beacon value in epoch $e \geq 1$ is computed as a unique threshold signature on $\mathsf{Hash}(e)$. In more detail, each party P_i generates a partial signature $\sigma_e^{(i)}$ on the message $\mathsf{Hash}(e)$ and sends this share to every other party. Upon collecting $t+1$ of these partial signatures, any party can locally compute the beacon output as a full signature σ_e. Cachin et al. works in asynchrony and neither specifies the threshold signature nor the DKG. Dfinity works in partial synchrony and uses threshold BLS together with a non-interactive DKG [52]. Drand [1] works in synchrony and uses threshold BLS together with Gennaro et al.'s DKG [48]. Other threshold signatures are also employed: Herb works in synchrony and uses the threshold ElGamal signature [37], and RandHerd works in asynchrony and uses threshold Schnorr signatures with collective signing [75, 77]. For setup, all these protocols incur at least a communication cost of $O(\lambda n^3)$ bits. Once the setup phase is finished, the protocols achieve an improved communication cost of $O(\lambda n^2)$ bits per beacon output. The protocols in this category are unpredictable and bias-resistance, but security is only guaranteed against a static adversary. The most significant drawback of the protocols in this category is that they do not support efficient reconfiguration. In case a party is removed from the network and another one joins, the DKG must be run again among all parties. In particular, these protocols are not well-suited for public permissionless blockchain environments.

Protocols in the second category include [9, 20, 26, 30, 38, 54, 69]. These protocols rely on computationally expensive tools such as Proof-of-Work (PoW) [62] and Verifiable Delay Functions (VDF) [18, 38] that require specialized hardware and rely on strong and new assumptions about verifiable time-lock puzzles. Protocols in this category are highly energy-consuming. These works essentially rely on the same idea, in which it takes a certain amount of energy (in the case of PoW) or time (in the case of VDF) to compute the next block or beacon output. VDF-based protocols rely on the assumption that the adversary has no advantage over the honest parties in computing the VDF faster. PoW-based protocols rely on the assumption that the adversary has less computational hash power than the honest parties. RandRunner [69] works in synchrony and uses a trapdoor VDF. Such a trapdoor VDF can generate unique function values efficiently with the knowledge of the trapdoor, but takes some high specified time T otherwise. RandRunner is bias-resistant against an adaptive adversary and has a communication cost of $O(\lambda n^2)$ bits per beacon output. However, it only achieves $(t+1)$-unpredictability, since an adaptive adversary can simply corrupt the next t leaders and thus learn the beacon values for the next t epochs. Other protocols in this category such as RandChain [54] that uses a combination of sequential PoW, Nakamoto Consensus and VDF even achieve a communication cost of $O(\lambda n)$ bits per beacon output while providing security against an adaptive adversary. Besides the high energy costs another drawback is that the beacon output is only guaranteed to be 1/5-fair (1-fairness means that each party in the system controls comparable power on deciding random outputs). Apart from that, it also suffers from problems concerning blockchain-oriented attacks.

Protocols in the third category include [6, 15, 16, 25, 34, 35, 50, 54, 60, 69, 70, 76]. These protocols are reconfiguration-friendly and do not rely on heavy tools such as PoW or VDF. Most notably are HydRand [70], SPURT [34], GRandPiper [16], BRandPiper [16], and OptRand [15]. We briefly elaborate on some of them. SPURT works in partial synchrony, achieves bias-resistance and unpredictability against a static adversary, and has a communication cost of $O(\lambda n^2)$ bits per beacon output. It relies on a pairing-based PVSS scheme and a modified version of the HotStuff SMR protocol [78]. Since it relies on a PVSS scheme, it does not provide security against an adaptive adversary. GRandPiper is a protocol from the RandPiper family of randomness beacons. It works in synchrony, achieves bias-resistance against a static adversary, and has a communication cost of $O(\lambda n^2)$ bits per beacon output. It relies on a PVSS scheme and the RandPiper SMR protocol. However, against an adaptive adversary it only provides $(t+1)$-unpredictability (apart from the problem of adaptive security of the PVSS scheme) and even against a static adversary it achieves only $\min(\lambda, t)$-unpredictability. HydRand works in synchrony with sub-optimal Byzantine resilience threshold $t < n/3$ and achieves a communication cost of $O(\lambda n^2)$ bits. Similar to GRandPiper, it only provides bias-resistance and $(t+1)$-unpredictability against a static adversary. It relies on the Scrape PVSS scheme and the repeated execution of a Byzantine agreement protocol. Since it uses a PVSS scheme, it also suffers from the existence of a security proof against adaptive adversaries. BRandPiper is a self-claimed better version of GRandPiper from the family of RandPiper protocols. It works in synchrony with optimal-resilience threshold and provides complete security (i.e. unpredictability and bias-resistance) against an adaptive adversary. It achieves a communication cost of $O(\lambda f n^2)$ bits per beacon output, where $f \leq t$ is the actual number of faults in the system. It relies on an efficient VSS scheme and the RandPiper SMR protocol. In BRandPiper, the leader of an epoch shares n secrets at once and for the beacon output parties reconstruct a random value accumulating secrets from $t+1$ different (previous) leader parties. The other only protocol in this category that provides complete security against an adaptive adversary is RandShare [76]. It works in asynchrony with optimal-resilience threshold, and has a worst-case communication cost of $O(\lambda n^4)$ bits per beacon output with best-case communication of $O(\lambda n^3)$ bits per beacon output. It relies on a VSS scheme and a Byzantine agreement protocol. In RandShare, parties run n independent Byzantine agreement instances in parallel, from which the beacon value is then computed.

C FORMAL DEFINITIONS AND SECURITY NOTIONS

In this appendix, we formally define syntax and security notions of the primitive used in the paper.

C.1 Cryptographic Notions

Digital Signature Scheme. A digital signature scheme provides a user with a verification-signing key pair (vk, dk), where the signing key is only known to the user but the verification key is public. The signing key allows the user to sign any message of its choice, while any third party that knows vk can verify that the message was indeed signed by that particular user. We formally define this as follows.

Definition C.1 (Digital Signature Scheme). A digital signature scheme is a tuple of algorithms $\mathsf{DS} = (\mathsf{SKey}, \mathsf{Sign}, \mathsf{Ver})$ with the following properties:

- SKey: The randomized *key generation algorithm* takes as input system parameters par and an identity index $i \in [n]$. It outputs a verification key vk_i and a signing key dk_i.
- Sign: The possibly randomized *signing algorithm* takes as input a signing key dk_i and a message m. It outputs a signature σ. We also write $\langle m \rangle_i$ to denote the message-signature pair (m, σ) where $\sigma \leftarrow \mathsf{Sign}(dk_i, m)$.
- Ver: The deterministic *verification algorithm* takes as input a verification key vk_i, a message m, and a signature σ. It outputs 1 (accept) or 0 (reject). In the first case we call the signature σ *valid* (relative to vk_i); otherwise we call it *invalid*.

For a secure digital signature scheme, we require that an adversary (not knowing the signing key) cannot create a signature on a new message, even after obtaining many signatures on messages of its choice.

Definition C.2 (Unforgeability Under Chosen Message Attack). Let $\mathsf{DS} = (\mathsf{SKey}, \mathsf{Sign}, \mathsf{Ver})$ be a digital signature scheme. For an algorithm A, define the unforgeability under chosen message experiment $\mathbf{UF\text{-}CMA}^{\mathsf{A}}_{\mathsf{DS}}$ as follows:

- *Offline Phase.* Run SKey on input (par, i) to obtain a key pair (vk_i, dk_i). Run A on input (par, vk_i). Initialize $\mathcal{M} := \emptyset$.
- *Signing Oracle Queries.* At any point of the experiment, A gets access to an oracle that answer queries of the following type: When A submits a message m, return $\sigma \leftarrow \mathsf{Sign}(dk_i, m)$ and update $\mathcal{M} := \mathcal{M} \cup \{m\}$.
- *Output Determination.* When A outputs a message m^* and a signature σ^*, do the following. If $\mathsf{Ver}(vk_i, m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{M}$, return 1. Otherwise, return 0.

We say that DS is (ε, T, q_s)*-unforgeable under chosen message attacks (UF-CMA)* if for all algorithms A that run in time at most T and make at most q_s signing oracle queries, $\Pr[\mathbf{UF\text{-}CMA}^{\mathsf{A}}_{\mathsf{DS}} = 1] \leq \varepsilon$. Conversely, we say that A (ε, T, q_s)*-breaks unforgeability* of DS under chosen message attacks if it runs in time at most T, makes at most q_s signing oracle queries, and $\Pr[\mathbf{UF\text{-}CMA}^{\mathsf{A}}_{\mathsf{DS}} = 1] > \varepsilon$.

Non-Interactive Zero-Knowledge Proof. A zero-knowledge proof is an interactive protocol between a prover Prove and a verifier Ver which enables the prover to convince the verifier that a statement x is in an NP language $\mathcal{L}$ without leaking any information besides the fact that the statement is true. A non-interactive zero-knowledge (NIZK) proof is a class of zero-knowledge proofs, where no interaction is required. The prover Prove outputs only one message, called a proof, which convinces the verifier Ver of the truth of the statement.

Definition C.3 (Non-Interactive Proof System). Let $\mathcal{R}$ be an efficiently decidable binary relation and let $\mathcal{L}$ be the corresponding language. For pairs $(x, w) \in \mathcal{R}$, we call x the statement and w the witness. $\mathcal{L}$ consists of statements in $\mathcal{R}$. A non-interactive proof system for $\mathcal{R}$ is a tuple of probabilistic polynomial-time algorithms $\Sigma = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ with the following properties:

- Gen: This is a randomized parameter generation algorithm that takes as input the security parameter λ and the relation $\mathcal{R}$. It outputs public parameters par. All other algorithms implicitly take par as input.
- Prove: This is a possibly randomized proving algorithm that takes as input a statement x, and a witness w. It outputs a proof π.
- Ver: This is a deterministic verification algorithm that takes as input a statement x, and a proof π. It outputs 1 (accept) if the proof is valid and 0 (reject) otherwise.

We continue with the standard security notions for a non-interactive zero-knowledge proof [51]: perfect completeness, zero-knowledge, and simulation-soundness. For this let $\mathcal{R}$ and $\mathcal{L}$ be as above.

Definition C.4 (Perfect Completeness). Let $\Sigma = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ be a non-interactive proof system as defined above. For an algorithm A, define the perfect completeness experiment $\mathbf{PerfComp}_{\Sigma}^{\mathsf{A}}$ as follows:

- *Offline Phase.* Run the parameter generation algorithm on input λ to get $par \leftarrow \mathsf{Gen}(\lambda)$. Run A on input par.
- *Online Phase.* When A outputs (x, w), compute the proof $\pi \leftarrow \mathsf{Prove}(x, w)$.
- *Output Determination.* Return 1 if $(x, w) \in \mathcal{R}$ and $\mathsf{Ver}(x, \pi) = 1$. Otherwise, return 0.

We say that Σ has perfect completeness if for all algorithms A, $\Pr[\mathbf{PerfComp}_{\Sigma}^{\mathsf{A}} = 1] = 1$.

Definition C.5 (Zero-Knowledge). Let $\Sigma = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ be a non-interactive proof system as defined above. Let $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ be a pair of PPT algorithms (called the simulator). Furthermore, let algorithm Sim' be such that $\mathsf{Sim}'(par, \tau, x, w) = \mathsf{Sim}_2(par, \tau, x)$ if $(x, w) \in \mathcal{R}$ and $\mathsf{Sim}'(par, \tau, x, w) = 0$ otherwise. For an algorithm A, we define the zero-knowledge advantage of A as

$$\begin{aligned}\mathbf{Adv\text{-}zk}_{\Sigma}^{\mathsf{A},\mathsf{Sim}} = |\Pr[par \leftarrow \mathsf{Gen}(\lambda) : \mathsf{A}^{\mathsf{Prove}(par,\cdot,\cdot)}(par) = 1] \\ - \Pr[(par, \tau) \leftarrow \mathsf{Sim}_1(\lambda) : \mathsf{A}^{\mathsf{Sim}'(par,\tau,\cdot,\cdot)}(par) = 1]|.\end{aligned}$$

We say that Σ is zero-knowledge if there exists a simulator Sim as above such that for all non-uniform PPT algorithms A, its zero-knowledge advantage is negligible in λ.

Definition C.6 (Simulation-Soundness). Let $\Sigma = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ be a non-interactive proof system as defined above. Let $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ be a pair of PPT algorithms. For an algorithm A, we define the simulation-soundness advantage of A as

$$\begin{aligned}\mathbf{Adv\text{-}ss}_{\Sigma}^{\mathsf{A},\mathsf{Sim}}(\lambda) = \Pr[(par, \tau) \leftarrow \mathsf{Sim}_1(\lambda), (x, \pi) \leftarrow \mathsf{A}^{\mathsf{Sim}_2(par,\tau,\cdot)}(par) : \\ w \leftarrow \mathsf{Ver}(par, x, \pi) = 1, (x, \pi) \notin Q, x \notin \mathcal{L}],\end{aligned}$$

where Q is the list of simulation queries and responses. We say that Σ has simulation-soundness if there exists a simulator Sim as above such that for all non-uniform PPT algorithms A, its simulation-soundness advantage is negligible in λ.

C.2 Consensus Notions

Weakly Secure Randomness Beacon. We weaken the definition of a secure randomness beacon to capture the properties of SPURT. Since the construction of SPURT allows the parties to output a bot symbol $\bot_{\mathcal{RB}}$ whenever the leader of an epoch does not behave correctly, every n epochs the protocol only given $n - t$ truly random outputs $\sigma_i \in \{0, 1\}^{\lambda}$. As a consequence, we have to weaken the security notions of a randomness beacon accordingly.

Definition C.7 (d-Weakly Secure Randomness Beacon). Let $\mathcal{RB}$ be an epoch-based protocol executed by parties $P_1, \ldots, P_n$. We define the following security properties for $\mathcal{RB}$:

- *Weak Consistency.* $\mathcal{RB}$ is (t, L)-weakly consistent if the following holds whenever at most t parties are corrupted: if an honest party outputs a value $\sigma \in \{0, 1\}^{\lambda} \cup \{\bot_{\mathcal{RB}}\}$ in epoch $\ell \in [L]$, then all honest parties output σ in epoch ℓ.
- *Weak Availability.* $\mathcal{RB}$ is (t, L)-weakly available if the following holds whenever at most t parties are corrupted: for each $\ell \in [L]$, every honest party outputs a value $\sigma_\ell \in \{0, 1\}^{\lambda} \cup \{\bot_{\mathcal{RB}}\}$ in epoch ℓ. Furthermore, the sequence $\sigma_1, \sigma_2, \ldots$ of beacon outputs has the following property: for all $\ell \in [\![L - n]\!]$, any n consecutive values $\sigma_{\ell+1}, \ldots, \sigma_{\ell+n}$ contain at most t values $\bot_{\mathcal{RB}}$.
- *Weak Bias-Resistance.* $\mathcal{RB}$ is (ε, T, t, L)-weakly bias-resistant if it is (t, L)-weakly available, (t, L)-weakly consistent, and the following holds for all algorithms A, D such that A corrupts at most t parties and both A and D run in time at most T. Denote by $\Sigma_{\mathsf{A},L}$ the probability distribution induced by the outputs of an honest party in an execution of $\mathcal{RB}$ until epoch L with A as adversary. And denote by U_L the uniform distribution over the L-times Cartesian product of $\{0, 1\}^{\lambda} \cup \{\bot_{\mathcal{RB}}\}$ with itself. Then
$$\left|\Pr[\mathsf{D}(\sigma) = 1] \;\; - \Pr[\mathsf{D}(u) = 1]\right| \leq \varepsilon,$$
where the probabilities are taken over all $(\sigma, u) \in \Sigma_{\mathsf{A},L} \times U_L$ such that $u_i = \bot_{\mathcal{RB}}$ whenever $\sigma_i = \bot_{\mathcal{RB}}$ for $i \in [L]$. *This captures the condition that u is sampled from a distribution with $\bot_{\mathcal{RB}}$ at all points where $\sigma \leftarrow \Sigma_{\mathsf{A},L}$ also has $\bot_{\mathcal{RB}}$.*
- *d-Weak Unpredictability.* $\mathcal{RB}$ is $(\varepsilon, T, t, L, d)$-weakly unpredictable if it is (t, L)-weakly available, (t, L)-weakly consistent, and the following holds for all algorithms A that corrupt at most t parties and run in time at most T: A's advantage in the d-weak unpredictability experiment defined hereafter is at most ε.

We say that $\mathcal{RB}$ is a $(\varepsilon, T, t, L, d)$-weakly secure randomness beacon protocol if it is (ε, T, t, L)-weakly bias-resistant, $(\varepsilon, T, t, L, d)$-weakly unpredictable, (t, L)-weakly available, and (t, L)-weakly consistent.

Definition C.8 (d-Weak Unpredictability for $\mathcal{RB}$). Let $\mathcal{RB}$ be an epoch-based protocol as defined above. For an algorithm A and an $\ell \in [L]$, define the d-weak unpredictability experiment $d\text{-}\mathbf{wUnpred}_{\mathcal{RB},t}^{\mathsf{A},\ell}$ as follows:

- *Offline Phase.* Initialize sets $C = \{\}$ and $\mathcal{H} = \mathcal{P} \setminus C$. Run the parameter generation algorithm on input λ to obtain par. Run A on input par.
- *Corruption Queries.* At any point of the experiment, A may corrupt a party P_i by submitting an index $i \in \mathcal{P}$. In this case, return the internal state of P_i and set $\mathcal{H} = \mathcal{H} \setminus \{i\}$, $C = C \cup \{i\}$. Henceforth, A controls P_i.
- *Online Phase I.* Run $\mathcal{RB}$ for ℓ epochs until the first honest party outputs value σ_ℓ. *Note that* A *may also participate in the protocol through the corrupt parties.* Let $\sigma = (\sigma_1, \ldots, \sigma_\ell)$ denote its output values and T_ℓ its protocol transcript up to that point. Run A on input (T_ℓ, σ).
- *Online Phase II.* When A outputs a value (σ'_e, e) for an $e > \ell$, run $\mathcal{RB}$ for $e - \ell$ further epochs to obtain the output values $(\sigma_{\ell+1}, \ldots, \sigma_e)$. Again, A may participate in the execution of $\mathcal{RB}$ through the corrupt parties.
- *Output Determination.* Return 1 if $|C| \leq t$, $e \geq \ell + d$, $L \geq e$, and $\sigma'_e = \sigma_e \neq \bot_{\mathcal{RB}}$. Otherwise, return 0.

Define the advantage of A in the above experiment as

$$\mathbf{Adv}_{\mathcal{RB}}^{\mathsf{A}} = \Pr[d\text{-}\mathbf{wUnpred}_{\mathcal{RB},t}^{\mathsf{A},\ell} = 1],$$

where the probability is taken over all possible executions of $\mathcal{RB}$ and all $\ell \leq L$.

D ON THE HARDNESS OF CO-OMDL IN THE GENERIC GROUP MODEL

In this appendix, we provide a proof for the hardness of the Co-OMDL assumption in the generic group model (GGM). We assume that the groups are equipped with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In that case the adversary has access to an additional pairing evaluation oracle that on input $(u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$ returns $e(u, v) \in \mathbb{G}_T$. Our proof is inspired from the original proof for the hardness of the standard OMDL assumption in the GGM of Bauer et al. [8].

D.1 Do we really need the Co-OMDL Assumption? An Heuristic Argument

We give two examples from the literature (including this work) and explain why the ordinary OMDL assumption does not suffice anymore to provide adaptive security in the context of asymmetric pairings. We believe this observation to be of high significance, since actual implementations of pairing-based schemes do use asymmetric pairings for efficiency reasons.

Asymmetric PVSS Schemes. In our warm-up chapter, we gave a reduction from plain unpredictability of Schoenmakers' PVSS (which uses a group $\mathbb{G}$ with generators g, h and no pairing is involved) to the hardness of n-OMDL. In the setting of Type 3 asymmetric pairing, our reduction cannot rely on the n-OMDL assumption anymore. We explain the reason for that. Summarizing our proof for Schoenmakers' PVSS in Appendix A.1, the reduction embeds the OMDL challenge ξ in the second generator h, the public keys $pk_1, \ldots, pk_n$ of parties, or the degree-t polynomial $f \in \mathbb{Z}_p[X]$ chosen by the dealer/challenger (in the plain unpredictability notion, there is no aggregation involved). For the simulation to work, it needs to generate the commitments $g^{f(i)}$ and encrypted shares $pk_i^{f(i)}$ for $i \in [n]$. In the case where the reduction embeds ξ into the polynomial f, it relies on the knowledge of the discrete logarithm of h to base g to compute the elements $h^{f(i)}$ and from that generate the encrypted shares. This strategy also works for symmetric (Type 1) pairings and Type 2 pairings where there is an efficient way to map between g and h, even if these elements live in different source groups. However, for Type 3 pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, there is no such connection between generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ from different groups. In particular, the reduction is not able to compute $h^{f(i)}$ from $g^{f(i)}$ (and vice versa) and thus fails. This issue can be resolved by relying instead on the Co-OMDL assumption that simultaneously provides challenges with the same exponents in both source groups.

Asymmetric Threshold BLS. We recall the threshold BLS signature scheme. Given a symmetric pairing $e' : \mathbb{G} \times \mathbb{G} \to \mathbb{G}'$ and a random oracle $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$, a threshold BLS signature share σ_i from party P_i on a message m is computed as $\sigma_i = \mathsf{H}(m)^{sk_i}$. Verification of σ_i is done by checking the equality $e'(g, \sigma_i) = e'(pk_i, \mathsf{H}(m))$. Now, the security reduction of Bacho and Loss [7] embeds the OMDL instance either in the public keys $pk_1, \ldots, pk_n$ of parties or in the answers $\mathsf{H}(m)$ to random oracle queries. In the first case, signature shares σ_i on a message m are simulated by sampling $r \leftarrow \mathbb{Z}_p^*$ uniformly at random and returning $\mathsf{H}(m)^{sk_i} = (g^{sk_i})^r = pk_i^r$. In the asymmetric version of the threshold BLS scheme, the underlying pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is of Type 3 with public keys $pk_i \in \mathbb{G}_1$ and hash values $\mathsf{H}(m) \in \mathbb{G}_2$. Since there is no connection between the two source groups, signature shares cannot be simulated as before if the secret keys are unknown to the reduction. More concretely, a signature share on m is $\sigma_i = \mathsf{H}(m)^{sk_i} = h^{sk_i r}$ for an $r \in \mathbb{Z}_p^*$ and the generator $h \in \mathbb{G}_2$. Obviously, there is no way to compute this value unless h^{sk_i} is known. However, the public keys g^{sk_i} are elements in $\mathbb{G}_1$ and do not help in the Type 3 setting. Consequently, the simulation fails. This issue could potentially be resolved by relying instead on the Co-OMDL assumption that simultaneously provides challenges with the same exponents in both source groups.

D.2 Proof of Hardness of Co-OMDL in GGM

THEOREM D.1. *Let* A *be an adversary that (ε, T)-solves the COMDL problem of degree n in the generic group model, making at most m group operation queries and q_e pairing evaluation queries. Then*

$$\varepsilon \leq \frac{2m^2}{p - m^2} + \frac{4(m + q_e)^2}{p - 4(m + q_e)^2} + \frac{1}{p}.$$

PROOF. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing of prime order p cyclic groups with generators $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $e(g, h) \in \mathbb{G}_T$. And let A be an adversary that tries to break n-**COMDL**$^{\mathsf{A}}$ defined for $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We consider the generic group model (GGM). In the GGM, the adversary does not see actual group elements of the form $g^x \in \mathbb{G}_1$ with $x \in \mathbb{Z}_p$, but encodings $\Xi_1(x)$ of them, where $\Xi_1 : \mathbb{Z}_p \to \{0,1\}^{\log(p)}$ is a random injective function corresponding to the group $\mathbb{G}_1$. The same holds true for the groups $\mathbb{G}_2$ and $\mathbb{G}_T$ with encoding functions Ξ_2 and Ξ_T, respectively. Furthermore, the adversary cannot compute the encoding of $g^x \cdot g^y$ from encodings of g^x and g^y. Instead, it is provided with an group operation oracle that on input the pair $(\Xi_1(x), \Xi_1(y))$ returns $\Xi_1(x + y)$. The same holds true for the group $\mathbb{G}_2$ and $\mathbb{G}_T$ with its encoding functions. Additionally, the adversary can query the pairing on evaluations. That is, on input a pair $(\Xi_1(a), \Xi_2(b))$ which corresponds to (g^a, h^b), the oracle returns $\Xi_T(ab)$ which corresponds to $e(g, h)^{ab} \in \mathbb{G}_T$.

In their proof, the authors of [8] observed the following. In general GGM proofs, the challenger replaces the secrets (that the adversary is supposed to find) by indeterminates. For instance in the case of the discrete logarithm problem, the challenger provides the adversary with an encoding $\Xi(X)$ of a group element g^X where $X \in \mathbb{Z}_p[X]$ is an undefined variable instead of an actual value $x \in \mathbb{Z}_p$. As long as the simulation succeeds, the adversary cannot distinguish between g^X and an actual group element g^x. The idea is then, after the adversary outputs a solution $x' \in \mathbb{Z}_p$ to the discrete logarithm problem g^X, the challenger chooses an actual value $x \leftarrow \mathbb{Z}_p$ uniformly at random to replace X with x. However, an issue arises. During the game the adversary queried the group operation oracle GC to compute several group elements, e.g. $g^X \cdot g^X, g \cdot g^X, (g^X)^7 \cdot g^5, \ldots$ (note that the adversary can only query the oracle GC on input two already computed group elements A, B along with a bit $b \in \{-1, 1\}$ to compute $A \cdot B^b$, but we simply assume the adversary already computed the elements $(g^X)^7, g^5$ etc.). More general, we assume that the adversary computed the group elements $g^{P_i(X)}$ for $i \in [m]$ (where m denotes the total number of group operation oracle queries) for polynomials $P_i \in \mathbb{Z}_p[X]$ of degree 1. Obviously, different polynomials $P_i \neq P_j$ correspond to different group elements and thus different encodings $\xi_i \neq \xi_j$ that the challenger provided the adversary with. However, if the variable X is set to be defined $X := x$ for the actual (uniformly random) value $x \in \mathbb{Z}_p$, it could happen that there are polynomials $P_i \neq P_j$ such that $P_i(x) = P_j(x)$. In that case, the adversary detects incorrect behavior and thus the simulation fails. However, by the well-known Schwartz-Zippel lemma, for a fixed pair (P_i, P_j) of polynomials $P_i \neq P_j$, the probability that $P_i(x) = P_j(x)$ is the probability that $(P_i - P_j)(x) = 0$, which is at most $1/p$ by Schwartz-Zippel, since $x \leftarrow \mathbb{Z}_p$ is chosen uniformly at random (after the polynomials P_i, P_j are defined). As a consequence, when considering all $O(m^2)$ pairs of polynomials (P_i, P_j) with $(i, j) \in [m]^2$ and $i \neq j$, Schwartz-Zippel tells us that the simulation fails with probability at most m^2/p which is negligible. This is the original proof idea of Shoup [73].

However, Bauer et al. observed that this technique does not suffice anymore in the OMDL game. Again, the strategy is to replace the n-OMDL challenge $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ with independent indeterminates $\vec{X} = (X_1, \ldots, X_n) \in \mathbb{Z}_p[X_1, \ldots, X_n]$ from the n-dimensional ring $\mathbb{Z}_p[X_1, \ldots, X_n]$. In contrast to the discrete logarithm (DL) game, the adversary has access to a discrete logarithm oracle DL_g that it can query on (encodings $\xi_i = \Xi(a_i)$ of) group elements $A = g^{a_i}$ to obtain $a_i \in \mathbb{Z}_p$. In particular, if it queries the oracle DL_g on e.g. a challenge element g^{X_i}, the challenger is forced to return an actual value $x_i \in \mathbb{Z}_p$ and thus the adversary gains (partial) information on the challenge $\vec{x}$. Hence, the adversary can choose the polynomials P_i (after obtaining x_i) dependent on x_i and for this reason the Schwartz-Zippel lemma does not apply anymore, since the polynomials $P_1, \ldots, P_m$ may not be independent from $(x_1, \ldots, x_n)$ anymore. However, the idea is that even though the adversary gains information on $\vec{x}$ via its DL_g queries, the total information can only encode a space of dimension $q < n$ in $\mathbb{Z}_p^n$ where q is the total number of queries to DL_g (when we think of one DL_g query giving the adversary one information on $\vec{x}$). Let us informally encode this information in the space $\mathcal{L} \subset \mathbb{Z}_p^n$. Following this, the intuition is that, when sampling $\vec{x}$ uniformly at random (when the adversary queries DL_g on a newly g^{X_i}, the challenger simply returns a uniformly random $x_i \leftarrow \mathbb{Z}_p$), there is still one dimension in the vector left that is completely independent from all the polynomials $P_1, \ldots, P_m$ and thus by (some form of) Schwartz-Zippel the probability that a collision appeared somewhere outside of the space $\mathcal{L}$ should be at most $1/p$. Indeed, the authors provide a technical lemma that exactly captures the above intuition and can be seen as some form of the Schwartz-Zippel lemma that deals with polynomials of degree 1.

We will follow that idea. However, in our case, since there is a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ involved, we have to deal with polynomials of degree 2 that appear in the target group $\mathbb{G}_T$. Precisely, when the adversary obtains encodings of polynomials $P(\vec{X})$ via the group operation oracle GC_1 in the group $\mathbb{G}_1$, $Q(\vec{X})$ via GC_2 in $\mathbb{G}_2$ (recall that $\vec{X} = (X_1, \ldots, X_n)$ is the list of indeterminates), then it can query the pairing oracle on $(g^{P(\vec{X})}, h^{Q(\vec{X})})$ to obtain $e(g^{P(\vec{X})}, h^{(\vec{X})}) = e(g, h)^{(PQ)(\vec{X})}$ where PQ is now a polynomial of degree 2. We will deal with this issue by carefully keeping track of three different lists, each for one group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, for the simulation to go through. We also provide a generalization of their technical lemma to incorporate polynomials of degree 2. However, the structure and simulation of our proof follows the one of Bauer et al. with some adaptions to deal with the pairing. Our proof progresses through a series of games that we describe now. In the following, we write $E_P := \{0,1\}^{\log(p)}$ for the space of strings and we let generators be $g_1 := g \in \mathbb{G}_1$, $g_2 := h \in \mathbb{G}_2$, and $g_T := e(g, h) \in \mathbb{G}_T$. For a set $L = \{P_1, \ldots, P_q\} \subset \mathbb{Z}_p[X_1, \ldots, X_n]$ of polynomials in n variables, we write $\mathsf{Span}(L)$ for the $\mathbb{Z}_p$-subspace spanned by $P_1, \ldots, P_q$, that is $\mathsf{Span}(L) := \{\alpha_1 P_1 + \ldots + \alpha_q P_q \mid \alpha_i \in \mathbb{Z}_p\}$.

Game$_0$: This is the real game in the generic group model. The challenger C initializes the challenge counter $n := 0$, the discrete logarithm oracle counter $q := 0$, a group operation oracle counter $c_j := 0$ for the group $\mathbb{G}_j$ for each group index $j \in \{1, 2, T\}$, the pairing evaluation oracle counter $q_e := 0$, the challenge vector $\vec{x} := ()$, and the field element $a_{0,j} := 1$ corresponding to the generator $g_j \in \mathbb{G}_j$ for each $j \in \{1, 2, T\}$. The adversary has access to an OMDL challenge oracle Chal, a discrete logarithm oracle DL_g in the group $\mathbb{G}_1$, a group operation oracles GC_j in the group $\mathbb{G}_j$ for each $j \in \{1, 2, T\}$, and a pairing oracle $e(-,-)$. For $j \in \{1, 2, T\}$, the challenger C has an encoding function Enc_j that implements an injective random function $\Xi_j : \mathbb{Z}_p \to E_p$ to return encodings of group elements (in the generic group model). For this, the challenger initializes lists $\mathcal{M}_j$ for $j \in \{1, 2, T\}$ that keeps track of already assigned encodings $\xi_{i,j} := \Xi_j(a_i)$ for a field element $a_i \in \mathbb{Z}_p$ corresponding to the group element $g_j^{a_i} \in \mathbb{G}_j$. Each time the function Enc_j is called on a field element $a_i \in \mathbb{Z}_p$, it returns $\xi_{i,j}$ if a_i is already assigned an element $\xi_{i,j}$. Otherwise, it returns a uniformly random $\xi_{i,j} \leftarrow E_p \setminus \{\xi_{k,j}\}_{k<i}$ and updates $\mathcal{M}_j := \mathcal{M}_j \sqcup \{(\xi_{i,j}, a_i)\}$ where $\sqcup$ means that the pair $(\xi_{i,j}, a_i)$ is appended to the list $\mathcal{M}_j$ with associated index i. For each $j \in \{1, 2, T\}$, these lists $\mathcal{M}_j$ are initially empty and the challenger inserts uniformly at random chosen $\xi_{0,j} \leftarrow E_p$ strings for to represent the generators $g_j \in \mathbb{G}_j$, i.e. it updates $\mathcal{M}_j := \mathcal{M}_j \sqcup \{(\xi_{0,j}, 1)\}$ for each j. By abuse of notation, we may sometimes write $a_i \in \mathcal{M}_j$ or $\xi_{i,j} \in \mathcal{M}_j$ to mean $(\xi_{i,j}, a_i) \in \mathcal{M}_j$. If the adversary calls the group operation oracle GC_j for an $j \in \{1, 2, T\}$ on input (ξ, ξ', b) where $b \in \{0,1\}$ is a bit (which indicates if the actual group operation $A \cdot A'$ should be computed or the inverse $A \cdot A^{-1}$), the challenger returns $\perp$ if $\xi \notin \mathcal{M}_j$ or $\xi' \notin \mathcal{M}_j$ (because the group operation can only be queried on already known group elements). Otherwise, it updates the group operation counter $c_j := c_j + 1$ and returns $\mathsf{Enc}_j(a_k)$ where $a_k := a_i + (-1)^b a_i'$ where $k := c_j$ and a_i, a_i' are the representatives for ξ, ξ', respectively. This captures the idea that the challenger keeps track of already returned (encodings of) group elements so that it does not assign different values $\xi_{i,j} \neq \xi_{i,j}'$ to the same group element. Further, if the adversary calls the pairing oracle on input $(\xi_{i,1}, \xi_{j,2})$ corresponding to the pair $(g_1^{a_i}, g_2^{a_j}) \in \mathbb{G}_1 \times \mathbb{G}_2$, the challenger returns $\perp$ if $\xi_{i,1} \notin \mathcal{M}_1$ or $\xi_{j,2} \notin \mathcal{M}_2$ (one of the two input elements are not assigned and thus unknown to the adversary at this point). Otherwise, it updates $c_T := c_T + 1$ and returns $\mathsf{Enc}_T(a_i a_j)$ where the value $a_i a_j \in \mathbb{Z}_p$ is considered modulo p as usual. If the adversary calls the challenge oracle Chal, update the challenge counter $n := n+1$, sample a challenge $x_n \leftarrow \mathbb{Z}_p$ uniformly at random, update the group operation counters $c_1 := c_1 + 1$ and $c_2 := c_2 + 1$ (since the challenge is provided in both source groups $\mathbb{G}_1$ and $\mathbb{G}_2$), return $\mathsf{Enc}_1(a_{c_1})$ and $\mathsf{Enc}_2(a_{c_2})$ where $a_{c_1} := x_n$ and $a_{c_2} := x_n$, and update the challenge vector $\vec{x} \sqcup \{x_n\}$ (append the

element x_n at the end of the vector $\vec{x}$). This captures the idea that the challenger samples a new random challenge x_n (resulting in the total challenge $(x_1, \ldots, x_n)$ up to that point), and returns (an encoding of) the corresponding group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. Finally, if the adversary calls the discrete logarithm oracle DL_g on input ξ (which operates in the group $\mathbb{G}_1$ only), return $\perp$ if $\xi \notin \mathcal{M}_1$ (that is, the element ξ is unknown to the adversary or not in the group $\mathbb{G}_1$). Otherwise, update the discrete logarithm oracle counter $q := q + 1$, set $v := a_i$ where $i = \min_{k \in [\![c_1]\!]}\{\xi = \xi_{k,1}\}$, and return v. This captures the idea that the challenger returns the to ξ already assigned value a_i.

The following game is where we (acting as the challenger) introduce polynomials and the set of polynomials $L \subset \mathbb{Z}_p[X_1, \ldots, X_n]$ that encodes the information the adversary gets through the discrete logarithm oracle queries. Initially, the set L is empty. For instance, if the adversary queries DL_g the first time on input $\Xi_1(X_1)$ corresponding to the group element g^{X_1}, we return a uniformly random $x_1 \leftarrow \mathbb{Z}_p$ and update $L := L \cup \{X_1 - x_1\}$. If the adversary queries DL_g the second time on input $\Xi_1(3X_1 + 4)$, we are supposed to return the element $3x_1 + 4$. For this, we simply consider the $\mathbb{Z}_p$-subspace $\mathsf{Span}(1, L) \subset \mathbb{Z}_p^n[X_1, \ldots, X_n]$ spanned by 1 and the polynomials in L. The composition $3X_1 + 4 = (3x_1 + 4) + 3(X_1 - x_1)$ lets us return the value $3x_1 + 4$ if DL_g is queried on $3X_1 + 4$. This strategy just ensures that we do not sample a new uniform value $x_2 \leftarrow \mathbb{Z}_p$ to answer the discrete logarithm query on $\Xi_1(3X_1 + 4)$. In that case, the adversary would detect incorrect behavior directly. Recollecting the idea of the GGM proof for the discrete logarithm problem, the simulation aborts when there is a pair $P_i \neq P_j$ such that $P_i(\vec{x}) = P_j(\vec{x})$. For the OMDL problem, the simulation of Bauer et al. allow the existence of pairs $P_i \neq P_j$ for which the current knowledge of $\vec{x}$ allows to deduce $P_i(\vec{x}) = P_j(\vec{x})$ (this is necessary, since the adversary adaptively obtains information on $\vec{x}$ and thus can construct such polynomials). However, if $P_i - P_j \notin \mathsf{Span}(L)$ and still $P_i(\vec{x}) = P_j(\vec{x})$, then the simulation should abort. On the other hand, if $P_i - P_j \in \mathsf{Span}(L)$, then obviously $P_i(\vec{x}) = P_j(\vec{x})$. Hence, the event $P_i(\vec{x}) = P_j(\vec{x})$ can be phrased as $P_i - P_j \in \mathsf{Span}(L)$ in the OMDL game. As a further consequence, if a new polynomial P_j (which replaces the scalar a_j that corresponds to the group element g^{a_j}) is getting encoded, their simulation sets $\xi_j := \xi_i$ if there is an index $i \in [\![j - 1]\!]$ such that $P_j - P_i \in \mathsf{Span}(L)$ (since this corresponds to the event $P_i(\vec{x}) = P_j(\vec{x})$ in the plain discrete logarithm game). On the other hand, the simulation fails if there is a pair of polynomials $P_i \neq P_j$ such that $P_i - P_j \in \mathsf{Span}(L)$ but $\xi_i \neq \xi_j$. Since we have now three groups and a pairing, this argument adapts slightly. For this, we let $P_{1,1}, \ldots, P_{m_1,1}$ be the polynomials the adversary constructs in the group $\mathbb{G}_1$, and $P_{1,2}, \ldots, P_{m_2,2}$ the ones that it constructs in $\mathbb{G}_2$, and $P_{1,T}, \ldots, P_{m_T,T}$ the ones that it obtains in $\mathbb{G}_T$. Note that the $P_{i,1}$ and $P_{i,2}$ are polynomials of degree 1, but the $P_{i,T}$ are polynomials of degree 1 or 2 because of the pairing. In this case, the simulation fails if there is an $\ell \in \{1, 2, T\}$ with a pair of polynomials $P_{i,\ell} \neq P_{j,\ell}$ such that $P_{i,\ell} - P_{j,\ell} \in \mathsf{Span}(L)$ but $\xi_{i,\ell} \neq \xi_{j,\ell}$. This will be the idea for the final game. But before that, we will formally define an intermediate game that tells the simulation to abort when it finds a collision among some $P_{i,\ell} \neq P_{j,\ell}$ evaluated at $\vec{x}$ while $P_{i,\ell} - P_{j,\ell} \notin \mathsf{Span}(L)$. For this game, we will derive a technical lemma that bounds the probability of simulation failure and thus bounds the statistical distance between $\mathbf{Game}_0$ and $\mathbf{Game}_1$. Afterwards, we will define the final game and show that the intermediate and final game are equally distributed. Having obtained that, we upper-bound the probability of the adversary in winning the final game.

Game$_1$: This is the intermediate game. We introduce polynomials $P_{i,j}$ that replace the scalars a_i, the set of polynomials $L \subset \mathbb{Z}_p[X_1, \ldots, X_n]$ that encodes the information the adversary gets through the discrete logarithm oracle queries, and the abort condition that tells the simulation to abort if it finds a collision among a pair of distinct polynomials whose difference does not lie in the span $\mathsf{Span}(L)$ of L. Apart from that, there is no difference to the previous game. For this reason, we will keep the following description short. The challenger C initializes as usual $n := 0$, $q := 0$, $c_j := 0$ for each $j \in \{1, 2, T\}$, $q_e := 0$, $\vec{x} := ()$. Additionally, the challenger initializes polynomials $P_{0,j} := 1$ for $j \in \{1, 2, T\}$ instead of scalars $a_{0,j}$ and an initially empty set $L := \emptyset$. The adversary has access to the oracles Chal, DL_g, GC_j for $j \in \{1, 2, T\}$, and $e(-,-)$. For the encoding functions Enc_j, the challenger has its bookkeeping lists $\mathcal{M}_j$ for $j \in \{1, 2, T\}$ that keep track of already assigned encodings $\xi_{i,j} := \Xi_j(P_{i,j})$ for a polynomial $P_{i,j} \in \mathbb{Z}_p[X_1, \ldots, X_n]$ corresponding to the group element $g_j^{P_{i,j}} \in \mathbb{G}_j$. Additionally, the challenger C checks for each new request $\mathsf{Enc}_j(P_{\ell,j})$ if there exists an already assigned polynomials $P_{i,j}$ with $i \in [\![\ell - 1]\!]$ such that $P_{\ell,j} - P_{i,j} \in \mathsf{Span}(L)$. In that case, it sets $\xi_{\ell,j} := \xi_{i,j}$. (Otherwise, it sets $\xi_{\ell,j} \leftarrow E_p \setminus \mathcal{M}_j$ to a uniformly random string that has not already been assigned). If the adversary calls the challenge oracle Chal, update $n := n + 1$, sample a new indeterminate X_n and a value $x_n \leftarrow \mathbb{Z}_p$ uniformly at random (this value should help the simulation to abort directly when it finds a collision), update $c_1 := c_1 + 1$ and $c_2 := c_2 + 1$ (since the challenge is provided in both source groups $\mathbb{G}_1$ and $\mathbb{G}_2$), return $\mathsf{Enc}_1(P_{c_1,1})$ and $\mathsf{Enc}_2(P_{c_2,2})$ where $P_{c_1,1} := X_n$ and $P_{c_2,2} := X_n$, and update $\vec{x} \sqcup \{x_n\}$. On the other hand, if there is a polynomial $P_{i,j}$ with $i \in [\![\ell - 1]\!]$ such that $P_{\ell,j} - P_{i,j} \notin \mathsf{Span}(L)$ but with a collision $P_{\ell,j}(\vec{x}) = P_{i,j}(\vec{x})$, then the simulation aborts the game. We will after the definition of this game bound the probability that this happens. For $j \in \{1, 2\}$ with groups $\mathbb{G}_1, \mathbb{G}_2$, the polynomials are of degree 1 so that the technical lemma of Bauer et al. applies. However, in the case of $\mathbb{G}_T$, these polynomials can be of degree 2, so that we have to come up with a new lemma that captures also quadratic polynomials. To upper-bound the final probability that a collision occurs in any of the three groups, we simply sum up the individual probabilities. Continuing with the game, if the adversary calls the group operation oracle GC_j for an $j \in \{1, 2, T\}$ on input (ξ, ξ', b), the challenger updates $c_j := c_j + 1$ and returns $\mathsf{Enc}_j(P_{k,j})$ where $P_{k,j} := P_{i,j} + (-1)^b P_{i',j}$ where $k := c_j$ and $P_{i,j}, P_{i',j}$ are the representatives for ξ, ξ', respectively. (In case one of the inputs is invalid/not already assigned, the challenger returns $\perp$ as usual). If the adversary calls the pairing oracle on input $(\xi_{i,1}, \xi_{j,2})$, the challenger returns $\perp$ or it updates $c_T := c_T + 1$ and returns $\mathsf{Enc}_T(P_{i,1}P_{j,2})$. We point out that this is exactly the point where the quadratic polynomials appear. Finally, if the adversary calls the discrete logarithm oracle DL_g on input ξ (which operates in the group $\mathbb{G}_1$ only), return $\perp$ if the input is invalid and otherwise do the following. Let $i \in [\![c_1]\!]$ be such that $\xi = \xi_{i,1}$ with corresponding polynomial $P_{i,1}$. Set $v := P_{i,1}(\vec{x})$. If $P_{i,1} \in \mathsf{Span}(1, L)$, then let $P_{i,1} = \alpha_0 + \alpha_1 Q_1 + \ldots + \alpha_{q-1} Q_{q-1}$ be the decomposition of $P_{i,1}$ in the $\mathbb{Z}_p$-subspace $\mathsf{Span}(1, L)$, where the coefficients are $\alpha_k \in \mathbb{Z}_p$, and update $v := \alpha_0$. Return v, set $Q_q := P_{i,1} - v$ and $L := L \cup \{Q_q\}$. This captures the idea that the challenger should be consistent with its answers to discrete logarithm oracle queries in case the adversary queries for redundant information (if A asks for the discrete logarithm of X_1 to obtain x_1 and then asks for $3X_1$, the challenger should consistently return $3x_1$).

In the following, we will upper-bound the probability that the simulation aborts the game. That is, we will bound the probability that for an $j \in \{1, 2, T\}$ there is a pair of polynomials $P_{\ell,j} \neq P_{i,j}$ such that $P_{\ell,j} - P_{i,j} \notin \mathsf{Span}(L)$ but $P_{\ell,j}(\vec{x}) = P_{i,j}(\vec{x})$. For this, we will need two lemmas. We introduce some notation. For a polynomial $P \in \mathbb{Z}_p[X_1, \ldots, X_n]$, we let the corresponding calligraphic $\mathcal{P}$ denote the zero set of the polynomial in $\mathbb{Z}_p^n$, i.e. $\mathcal{P} := \{\vec{x} \in \mathbb{Z}_p^n \mid P(\vec{x}) = 0\}$.

Lemma D.2. *Let $D_1, \ldots, D_m, Q_1, \ldots, Q_{q+1} \in \mathbb{Z}_p[X_1, \ldots, X_n]$ be polynomials of degree 1. And let*

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

be the set of points at which all polynomials Q_i vanish but none of the polynomials D_i do. Assume that $\mathcal{Q}_{q+1} \cap C \neq \emptyset$ and that $(\mathcal{Q}_1 \cap \ldots \cap \mathcal{Q}_q) \nsubseteq \mathcal{Q}_{q+1}$. If $\vec{x} \in \mathbb{Z}_p^n$ is sampled uniformly at random from the set C, then

$$\frac{p-m}{p^2} \le \Pr[Q_{q+1}(\vec{x}) = 0] \le \frac{1}{p-m}.$$

Proof. See Bauer et al. [8] on page 9. □

Lemma D.3. *Let $Q_1, \ldots, Q_q \in \mathbb{Z}_p[X_1, \ldots, X_n]$ be polynomials of degree 1, and let $D_1, \ldots, D_m, Q_{q+1} \in \mathbb{Z}_p[X_1, \ldots, X_n]$ be polynomials of degree $d_i \in \{1, 2\}$. And let*

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

be the set of points at which all polynomials Q_i vanish but none of the polynomials D_i do. Assume that $\mathcal{Q}_{q+1} \cap C \neq \emptyset$ and that $(\mathcal{Q}_1 \cap \ldots \cap \mathcal{Q}_q) \nsubseteq \mathcal{Q}_{q+1}$. If $\vec{x} \in \mathbb{Z}_p^n$ is sampled uniformly at random from the set C, then

$$\frac{p-4m}{p^2} \le \Pr[Q_{q+1}(\vec{x}) = 0] \le \frac{4}{p-4m}.$$

Proof. Since $\vec{x}$ is sampled uniformly at random from the above non-empty set C, it is $\Pr[\vec{x} \in \mathcal{Q}_{q+1}] = |\mathcal{Q}_{q+1} \cap C|/|C|$. In the following, we bound both these set sizes. We begin with C. In the following, we write $\mathbb{F}_p := \mathbb{Z}_p$ for the field of p elements.

We let $\mathcal{Q} := \cap_{i \in [q]} \mathcal{Q}_i$ be the affine space with dimension d (note that the polynomials $Q_1, \ldots, Q_q$ are of degree 1). By assumption we know that $\mathcal{Q}$ is non-empty, otherwise C would be empty. In particular, $\mathcal{Q}$ contains p^d elements. We write the set C as

$$C = \mathcal{Q} \setminus \Big(\bigcup_{i \in [m]} (\mathcal{Q} \cap \mathcal{D}_i) \Big).$$

For an $i \in [m]$, we want to upper-bound the size of the set $\mathcal{Q} \cap \mathcal{D}_i$ and at the end sum up over all $i \in [m]$ to obtain a bound on C. For this, we fix an $i \in [m]$ and consider $\mathcal{Q} \cap \mathcal{D}_i$. There are two cases two consider: (i) the polynomial D_i is of degree 1, and (ii) the polynomial D_i is of degree 2. The first case is easy to handle, which we do now. Obviously, $\mathcal{Q}$ cannot be contained in $\mathcal{D}_i$, otherwise we would get $C = \emptyset$. Therefore, either $\mathcal{Q} \cap \mathcal{D}_i = \emptyset$ or $\mathcal{Q} \cap \mathcal{D}_i$ is of dimension $d-1$ (since D_i defines a hyperplane in $\mathbb{F}_p^n$). In both cases, the space contains at most p^{d-1} elements. And as a result, we obtain by summing over all $i \in [m]$ the bound

$$p^d - mp^{d-1} \le |C| \le p^d. \tag{1}$$

The second case (ii) is more complicated, since D_i does not define a hyperplane anymore but a hypersurface of degree 2 and linear algebraic methods do not apply anymore (as was done above for the case where D_i is of degree 1). Additionally, since D_i defines a multivariate polynomial of degree 2, its vanishing set intersection with $\mathcal{Q}$ can contain more points if D_i was of degree 1. As a result, the union over the sets $\mathcal{Q} \cap \mathcal{D}_i$ could be close to $\mathcal{Q}$ so that their set-theoretic difference C could be very small (e.g. of size 1). This, however, would result in a lower-bound $|C| \ge 1$ which would have a devastating effect on our probability bound, since that would yield $\Pr[\vec{x} \in \mathcal{Q}_{q+1}] = |\mathcal{Q}_{q+1} \cap C|/|C| \le 1$, giving us no way to properly bound the probability of simulation failure in our security game. Our intuition, however, tells us that the set $\mathcal{Q} \cap \mathcal{D}_i$ for a degree 2 polynomial D_i should only be about twice (or some other constant factor) as big as if D_i was of degree 1. Indeed, we back up this intuition by escaping the realm of linear algebra to enter the realm of algebraic geometry. In the following, let D_i be a polynomial of degree 2 and let $\overline{\mathbb{F}}_p$ be the algebraic closure of the field $\mathbb{F}_p$. We will pass to the n-dimensional projective space $\mathbb{P}^n(\overline{\mathbb{F}}_p)$ over the field $\overline{\mathbb{F}}_p$ to analyze the zero set $\mathcal{Q} \cap \mathcal{D}_i$. In order to do so, the polynomials $Q_1, \ldots, Q_q$ and D_i are homogenized and considered in $\overline{\mathbb{F}}_p[X_1, \ldots, X_n, X_0]$ where X_0 is a new variable that homogenizes the polynomials. Now we can do algebraic geometry. We denote by $d \le n-q$ the dimension of the variety $\mathcal{Q}$ which is simply a complete intersection of hyperplanes. Again, $\mathcal{Q}$ cannot be contained in $\mathcal{D}_i$, otherwise we would get $C = \emptyset$. Therefore, either it is $\mathcal{Q} \cap \mathcal{D}_i = \emptyset$ or $\mathcal{Q} \cap \mathcal{D}_i$ defines a projective variety of dimension $d-1$. The reason for the latter is: Since $\mathcal{Q} \nsubseteq \mathcal{D}_i$ and $\mathcal{Q} \cap \mathcal{D}_i \neq \emptyset$, the set $\mathcal{Q} \cap \mathcal{D}_i$ is a proper closed subvariety of $\mathcal{Q}$ and thus of dimension $< d$ (see any textbook on algebraic geometry, e.g. Hartshorne [55]). On the other hand, since any new equation cuts out at most one dimension, and $\mathcal{D}_i$ is defined by one equation, the intersection $\mathcal{Q} \cap \mathcal{D}_i$ drops by at most one dimension (can also see Gathmann [46]). Since we want to upper bound the size of $\mathcal{Q} \cap \mathcal{D}_i$, we ignore the case $\mathcal{Q} \cap \mathcal{D}_i = \emptyset$ of empty intersection (anyway, this gives the trivial bound $|\mathcal{Q} \cap \mathcal{D}_i| \le 0$). An important tool in the study of $\overline{\mathbb{F}}_p$-rational points on an algebraic variety is Bezout's inequality [27, 45] which states that for two (projective) varieties V and W, the following inequality holds:

$$\deg(V \cap W) \le \deg(V) \cdot \deg(W).$$

For our variety $\mathcal{Q} \cap \mathcal{D}_i$ this gives a degree of at most 2, since $\mathcal{Q}$ is defined by polynomials of degree 1 and $\mathcal{D}_i$ by a polynomial of degree 2. A long line of works in mathematics has studied estimates on the number of $\mathbb{F}_p$-rational points on a projective variety $V \subset \mathbb{P}^n$, which even goes back to the fundamental work of Deligne [36]. Extending classical results [61], Cafure and Matera [22] find an elementary bound on this number depending on the degree δ and dimension r of V.

Theorem D.4. *Let $V \subset \mathbb{P}^n(\overline{\mathbb{F}}_p)$ be a projective variety of dimension r and degree δ. Then the following estimate on the number $V(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points on V holds:*

$$|V(\mathbb{F}_p)| \leq \delta \cdot |\mathbb{P}^r(\mathbb{F}_p)| = \delta(p^r + \ldots + p + 1).$$

In our case, we have the degree $\delta = \deg(Q \cap \mathcal{D}_i) \leq 2$ and the dimension $r = \dim(Q \cap \mathcal{D}_i) = d - 1$. Given the above bound, we find the resulting bound over $\mathbb{F}_p$,

$$|Q \cap \mathcal{D}_i| \leq \delta(p^{d-1} + \ldots + p + 1) \leq 2\delta p^{d-1} \leq 4p^{d-1}.$$

As a result, we obtain by summing over all $i \in [m]$ the bound

$$p^d - 4mp^{d-1} \leq |C| \leq p^d. \tag{2}$$

Next, we want to bound the size $|Q_{q+1} \cap C|$. We define the intersection set $Q' := Q \cap Q_{q+1}$. Again, there are two cases two consider: (i) the polynomial Q_{q+1} is of degree 1, and (ii) the polynomial Q_{q+1} is of degree 2. The first case is easy to handle, which we do now. By assumption $Q \not\subseteq Q_{q+1}$, and therefore Q_{q+1} cuts out one dimension of Q, i.e. $\dim(Q') = d - 1$ where as before $d := \dim(Q)$. For a fixed $i \in [m]$, using the same techniques as before applied to Q' now (instead of Q), we find the bound

$$p^{d-1} - 4mp^{d-2} \leq |Q_{q+1} \cap C| \leq p^{d-1}. \tag{3}$$

In the second case, we directly pass to the projective space $\mathbb{P}^n$ over the algebraic closure $\overline{\mathbb{F}}_p$. The only difference now is that we directly apply Theorem D.4 to the variety Q' (which is now of degree 2 by Bezout) and then to the varieties $Q' \cap \mathcal{D}_i$ (which is now of degree 4 by Bezout). With the updated degrees of the respective varieties, the same calculation as before gives

$$4p^{d-1} - 8mp^{d-2} \leq |Q_{q+1} \cap C| \leq 4p^{d-1}. \tag{4}$$

Taking all bounds (1)-(4) together, we finally obtain

$$p^d - 4mp^{d-1} \leq |C| \leq p^d,$$
$$p^{d-1} - 4mp^{d-2} \leq |Q_{q+1} \cap C| \leq 4p^{d-1}.$$

Combining these identities, we obtain

$$\frac{p^{d-1} - 4mp^{d-2}}{p^d} \leq \frac{|Q_{q+1} \cap C|}{|C|} \leq \frac{4p^{d-1}}{p^d - 4mp^{d-1}}$$
$$\iff \quad \frac{p - 4m}{p^2} \leq \frac{|Q_{q+1} \cap C|}{|C|} \leq \frac{4}{p - 4m}.$$

This completes the proof of our technical lemma. □

We continue with our sequence of games. We now compare the statistical distance of **Game**$_0$ and **Game**$_1$. Recall that the simulation in **Game**$_1$ aborts when it finds a collision among a pair of distinct polynomials in one of the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. More concretely, let us for $j \in \{1, 2, T\}$ define the event F_j as: There exists an $i \in [\![c_j - 1]\!]$ such that $P_{c_j,j}(\vec{x}) = P_{i,j}(\vec{x})$ and $P_{c_j,j} - P_{i,j} \notin \mathsf{Span}(L)$. Since the group operation oracles are called at most m times, we get

$$\begin{aligned}\mathbf{Adv}^{\mathsf{A}}_{\mathbf{Game}_0} \leq &\mathbf{Adv}^{\mathsf{A}}_{\mathbf{Game}_1} + m\Pr[F_1] + m\Pr[F_2] \\ &+ (m + q_e)\Pr[F_3]. \end{aligned} \tag{♠}$$

We upper-bound the probability $\Pr[F_j]$ that event F_j happens. Before a call to Enc_j, the oracle defines $P_{c_j,j}$. We fix $j \in \{1, 2, T\}$ and consider a fixed $i \in [\![c_j - 1]\!]$ and let $P_j := P_{c_j,j} - P_{i,j}$. We want to use our two technical lemmas to bound the probability that $P_j(\vec{x}) = 0$ while $P_j \notin \mathsf{Span}(L)$. Therefore, we let $L := \{Q_1, \ldots, Q_q\}$ with the Q_i's being defined as in **Game**$_1$ and $Q_{q+1} := P_j$. According to Bauer et al., we further observe that the adversary A knows $P_{i_1,j}(\vec{x}) \neq P_{i_2,j}(\vec{x})$ when $\xi_{i_1,j} \neq \xi_{i_2,j}$. By writing $D_{\vec{i}} := P_{i_1,j} - P_{i_2,j}$ for $\vec{i} \in I := \{(i_1, i_2) \in [\![c_j - 1]\!]^2 \mid \xi_{i_1,j} \neq \xi_{i_2,j}\}$, we know that A knows $D_{\vec{i}}(\vec{x}) \neq 0$. Using previously defined notation, we get that

$$\vec{x} \in C := \Big(\bigcap_{i \in [q]} Q_i\Big) \setminus \Big(\bigcup_{i \in I} \mathcal{D}_i\Big).$$

The same argumentation as in [8] on page 17ff allows us to use our technical lemmas. Since for $j \in \{1, 2\}$ in the group $\mathbb{G}_j$ all appearing polynomials are of degree 1, we can directly apply the more straightforward Lemma D.2. In the case $j = T$ for the group $\mathbb{G}_T$ the appearing polynomials can be of degree 2, so that we need to apply the more complicated Lemma D.3. We will now apply these lemmas. For $j \in \{1, 2\}$, we obtain

$$\Pr[P_j(\vec{x}) = 0] \leq \frac{1}{p - m^2},$$

since there are at most m^2 elements in the set I which define the polynomials $D_{\vec{i}}$. Since we fixed an $i \in [\![c_j - 1]\!]$ to define P_j, we get the following when running over all such $i \leq m$:

$$\Pr[F_1] \leq \frac{m}{p - m^2}, \quad \Pr[F_2] \leq \frac{m}{p - m^2}.$$

On the other hand, for $j = T$ in the group $\mathbb{G}_T$ with quadratic polynomials, we obtain

$$\Pr[P_j(\vec{x}) = 0] \leq \frac{4}{p - 4(m + q_e)^2},$$

since the queries to the pairing operation give additional q_e polynomials (i.e. the set I is of size $(m + q_e)^2$). Running over all possible $i \leq (m + q_e)$, we obtain

$$\Pr[F_T] \leq \frac{4(m + q_e)}{p - 4(m + q_e)^2}.$$

Combining all these identities, (♠) then reduces to

$$\mathbf{Adv}^{\mathsf{A}}_{\mathbf{Game}_0} \leq \mathbf{Adv}^{\mathsf{A}}_{\mathbf{Game}_1} + \frac{2m^2}{p - m^2} + \frac{4(m + q_e)^2}{p - 4(m + q_e)^2}.$$

We end with the final game **Game**$_2$ which is described as **Game**$_4$ in Bauer et al.'s proof. In this game, we completely get rid of actual values for the challenge $\vec{x} = (x_1, \ldots, x_n)$ and only use polynomials. The only use of $\vec{x}$ in the previous game **Game**$_1$ was in the abort conditions in the encoding function and how to sample the answer v to an discrete logarithm query. However, as already noted previously, we can transfer these events to the discrete logarithm oracle by letting the simulation to abort when it finds a collision among a pair of polynomials $P_{i,\ell} \neq P_{j,\ell}$ such that $P_{i,\ell} - P_{j,\ell} \in \mathsf{Span}(L)$ while $\xi_{i,\ell} \neq \xi_{j,\ell}$ for an $\ell \in \{1, 2, T\}$. As observed by Bauer et al., the abort conditions in **Game**$_1$ and **Game**$_2$ are indeed equivalent from the view of the adversary. Furthermore, the authors also show why it is possible to sample $v \leftarrow \mathbb{Z}_p$ uniformly at random when there is no $\vec{x}$ involved anymore. Therefore, we can define the final game now.

Game$_2$: This is the final game. We will keep the following description short (since there is barely a difference to the previous game). The challenger C initializes as usual $n := 0$, $q := 0$, $c_j := 0$ for each $j \in \{1, 2, T\}$, $q_e := 0$, $\vec{x} := ()$. Additionally, the challenger initializes polynomials $P_{0,j} := 1$ for $j \in \{1, 2, T\}$ and $L := \emptyset$. The adversary has access to the oracles Chal, DL_g, GC_j for $j \in \{1, 2, T\}$, and $e(-,-)$. For the encoding functions Enc_j, the challenger has its bookkeeping lists $\mathcal{M}_j$ for $j \in \{1, 2, T\}$ that keep track of already assigned encodings $\xi_{i,j} := \Xi_j(P_{i,j})$ for a polynomial $P_{i,j}$. The challenger C checks for each new request $\mathsf{Enc}_j(P_{\ell,j})$ if there exists an already assigned polynomials $P_{i,j}$ with $i \in [\![\ell - 1]\!]$ such that $P_{\ell,j} - P_{i,j} \in \mathsf{Span}(L)$. In that case, it sets $\xi_{\ell,j} := \xi_{i,j}$. (Otherwise, it sets $\xi_{\ell,j} \leftarrow E_p \setminus \mathcal{M}_j$ to a uniformly random string that has not already been assigned). If the adversary calls the challenge oracle Chal, update $n := n + 1$, sample a new indeterminate X_n, update $c_1 := c_1 + 1$ and $c_2 := c_2 + 1$, return $\mathsf{Enc}_1(P_{c_1,1})$ and $\mathsf{Enc}_2(P_{c_2,2})$ where $P_{c_1,1} := X_n$ and $P_{c_2,2} := X_n$. Furthermore, if the adversary calls the group operation oracle GC_j for an $j \in \{1, 2, T\}$ on input (ξ, ξ', b), the challenger updates $c_j := c_j + 1$ and returns $\mathsf{Enc}_j(P_{k,j})$ where $P_{k,j} := P_{i,j} + (-1)^b P_{i',j}$ where $k := c_j$ and $P_{i,j}, P_{i',j}$ are the representatives for ξ, ξ', respectively. (In case one of the inputs is invalid/not already assigned, the challenger returns $\perp$ as usual). If the adversary calls the pairing oracle on input $(\xi_{i,1}, \xi_{j,2})$, the challenger returns $\perp$ or it updates $c_T := c_T + 1$ and returns $\mathsf{Enc}_T(P_{i,1}P_{j,2})$. Finally, if the adversary calls the discrete logarithm oracle DL_g on input ξ (which operates in the group $\mathbb{G}_1$ only), return $\perp$ if the input is invalid and otherwise do the following. Let $i \in [\![c_1]\!]$ be such that $\xi = \xi_{i,1}$ with corresponding polynomial $P_{i,1}$. Sample $v \leftarrow \mathbb{Z}_p$ uniformly at random. If $P_{i,1} \in \mathsf{Span}(1, L)$, then let $P_{i,1} = \alpha_0 + \alpha_1 Q_1 + \ldots + \alpha_{q-1} Q_{q-1}$ be the decomposition of $P_{i,1}$ in the $\mathbb{Z}_p$-subspace $\mathsf{Span}(1, L)$ and update $v := \alpha_0$. Set $Q_q := P_{i,1} - v$ and $L := L \cup \{Q_q\}$. If there is a pair of polynomials $P_{i,\ell} \neq P_{j,\ell}$ for $(i, j) \in [\![c_\ell]\!]^2$ such that $P_{i,\ell} - P_{j,\ell} \in \mathsf{Span}(L)$ and $\xi_{i,\ell} \neq \xi_{j,\ell}$ for $\ell \in \{1, 2, T\}$, then abort the game. Otherwise, return the value v.

To end the proof, we bound the probability that the adversary wins the final game. To this end, we show that there is a component of $\vec{x}$ that is sampled uniformly at random after the adversary outputs its solution $\vec{y}$ to the COMDL game. After A outputs $\vec{y}$, the set L is of size q and thus $\dim(\mathsf{Span}(L)) \leq q < n$. Therefore, $\mathsf{Span}(1, L)$ is of dimension $\leq q{+}1 \leq n$. On the other hand, $\dim(\mathsf{Span}(X_1, \ldots, X_n)) = n$ while $1 \notin \mathsf{Span}(X_1, \ldots, X_n)$. Having said that, we obtain

$$\mathsf{Span}(X_1, \ldots, X_n) \nsubseteq \mathsf{Span}(1, L).$$

In particular, there is an index $i \in [n]$ such that $X_i \notin \mathsf{Span}(1, L)$. Let $i \in [n]$ be the smallest such index. The discrete logarithm oracle DL_g outputs a value x_i uniformly at random when queried on $\xi_{j_i,1}$. However, this x_i is sampled uniformly at random after the i-th component of $\vec{y}$ output by the adversary A. Hence, $\Pr[\vec{x} = \vec{y}] \leq 1/p$. Finally, this yields the winning probability

$$\mathsf{A}^{\mathsf{A}}_{\mathbf{Game}_2} \leq \frac{1}{p}.$$

Taking together the obtained distances between sequential games, gives us the desired final bound

$$\varepsilon \leq \frac{2m^2}{p - m^2} + \frac{4(m + q_e)^2}{p - 4(m + q_e)^2} + \frac{1}{p}.$$

□

E SECURITY ANALYSIS OF APVSS SCHEMES

In this appendix, we provide the full security proof for SPURT's APVSS from Chapter 3. Additionally, we provide a formal description of their scheme which is given in Figure 7. Finally, we also elaborate on the simplifications made in our security proof in Chapter 3.2 for OptRand's APVSS. To recapitulate, there are only two differences to OptRand's APVSS: (i) the former uses an additional generator $\hat{h} \in \mathbb{G}_2$, and (ii) the former uses n NIZKs to prove correctness of the transcript.

E.1 On the Security Proof of OptRand's APVSS

In the following, we justify our simplifications made at the beginning of the proof given in 3.2. The first point was to assume that there is contribution in the aggregated transcript from precisely one corrupt party. If we more generally assume contribution from $f \leq t$ corrupt parties, then the corresponding NIZKs yield f pairs of equations for $\alpha'_1, \ldots, \alpha'_f$ (instead of only α') as given in the identities (2') and (†):

$$\alpha'_j c'_j = r'_j - a'_j - \ell b'_j - \sum_{i=1}^{n} P(i) c'_{i,j}, \quad \alpha'_j = a^\dagger_j + \ell b^\dagger_j + \sum_{i=1}^{n} P(i) c^\dagger_{i,j}$$

for all $j \in [f]$. By summation of these equations over $j \in [f]$ and observing that the last queried challenge (w.l.o.g. the adversary queries and gets $c'_1, \ldots, c'_f$ in ascending order) is truly independent from all the algebraic coefficients output by A for all previously queried challenges, we can reduce to the single-challenge case that we already considered because $c'_1 + \ldots + c'_f$ is now completely independent from the algebraic coefficients on the right-hand side of the resulting equation. The second point was to embed the challenge in only one answer to the transcript queries and that the adversary picks this transcript for his aggregate (which actually only happens with probability at most $1/q_k$). We resolve this with the trick of embedding this instance re-randomized into all transcript. That is, whenever in a simulation that embeds $\xi_1, \ldots, \xi_{t+1}$ into the polynomial f_1, we instead embed $\xi^{(i)}_1, \ldots, \xi^{(i)}_{t+1}$ into polynomial f_i for $i \in [q_k]$, where $\xi^{(i)}_j := \xi_j^{u_{i,j}} \cdot g^{v_{i,j}}$ for uniformly at random chosen $u_{i,j}, v_{i,j} \leftarrow \mathbb{Z}^*_p$ for $j \in [t + 1]$ and $i \in [q_k]$. The initial algebraic equation coming from the forgery of the adversary is then identical with polynomial $P = f_1$ replaced by $f := \sum_{i=1}^{q_k} f_i$. The corresponding coefficients are then not $\mathsf{DL}_g(\xi_i) = z_i$ (for $i \in [t + 1]$), but they are $z'_i := \sum_{j=1}^{q_k} u_{i,j} z_i + v_{i,j}$. Still, the reduction can solve for the z'_i in each scenario and thus compute the z_i (since it chose the values $u_{i,j}, v_{i,j}$ by itself). This justifies our simplifications and completes our proof.

E.2 Security Analysis of SPURT's APVSS

We give a proof for Theorem 3.7 which states the following.

THEOREM E.1. *If n-**COMDL** is (ε, T)-hard in the AGM and DS is $(\varepsilon_s, T_s, q_s)$-secure, then SPURT's $\mathsf{APVSS}_{\mathsf{DS}}$ is $(\varepsilon', T', t, q_k, q_h)$-aggregated unpredictable in the AGM & ROM, where*

$$\varepsilon \geq \frac{\varepsilon' - \varepsilon_s}{6} - \frac{q_h}{6p}, \quad T \leq T' + T_s + O(n^2).$$

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric pairing and independent generators $g, \hat{g} \in \mathbb{G}_1$ and $h, \hat{h} \in \mathbb{G}_2$. Let (pk_i, sk_i) be the key pair of party P_i with $pk_i = h^{sk_i}$. The dealer P_L with key pair (pk_L, sk_L) wants to share secret $e(\hat{g}, h^\alpha)$ for an $\alpha \leftarrow \mathbb{Z}_p^*$. The ADist algorithm takes as input sk_L and public keys $pk_1, \ldots, pk_n$. It outputs the transcript $T_L := \{C_i, Y_i, \pi_i\}_{i \in [n]}$ defined as follows.

(1) Choose a polynomial $f(X) = \alpha + \alpha_1 X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ of degree t uniformly at random.
(2) Publish commitments $C_i = g^{f(i)} \in \mathbb{G}_1$ for $i \in [n]$. Also publish encrypted shares $Y_i = pk_i^{f(i)} \in \mathbb{G}_2$ for $i \in [n]$.
(3) Compute NIZK proofs $\pi_i = \mathsf{Dleq}(g, C_i, pk_i, Y_i)$ of discrete logarithm equality for $i \in [n]$. Publish π_i for $i \in [n]$.

The *transcript verification algorithm* Ver takes as input the public keys $pk_1, \ldots, pk_n$ (including pk_L) and transcript T_L. It outputs 1 (accept) or 0 (reject). Let $\mathcal{LC}$ be the linear code as defined in General Notation 2 and let $\mathcal{LC}^\perp$ be its dual code.

(4) Check that $e(g, Y_i) = e(C_i, pk_i)$ for all $i \in [n]$. Sample a random codeword $(v_1, \ldots, v_n) \in \mathcal{LC}^\perp$ and check that $C_1^{v_1} \cdot \ldots \cdot C_n^{v_n} = 1$.
(5) Check that the NIZK proofs $\pi_1, \ldots, \pi_n$ verify using public keys $pk_1, \ldots, pk_n$ and H.
(6) If one of the above checks fails, output 0 (invalid transcript). Otherwise, output 1 (valid transcript).

Figure 7: Aggregatable distribution protocol ADist **and transcript verification algorithm** Ver **of SPURT's APVSS.**

On input the encrypted shares $Y_1, \ldots, Y_n$, the decryption Dec and reconstruction Rec algorithms work as follows.

(1) Using sk_i, compute the secret share $S_i = h^{f(i)}$ from Y_i via extracting the root $S_i = Y_i^{1/sk_i}$. Publish the decryption S_i.
(2) Upon receiving a secret share S_ℓ from party P_ℓ, check that $e(C_\ell, h) = e(g, S_\ell)$. Otherwise, the secret share is invalid.
(3) Upon receiving $t + 1$ valid secret shares $S_j = h^{f(j)}$ from different parties, compute $S = h^{f(0)}$ via Lagrange interpolation in the exponent. Finally, the secret is computed as $e(\hat{g}, S) \in \mathbb{G}_T$ and output.

Figure 8: Decryption Dec **and reconstruction** Rec **algorithms of SPURT's APVSS.**

We demonstrate aggregation for the first $t + 1$ parties $P_1, \ldots, P_{t+1}$. The algorithm Agg takes as input the individual parties' transcripts $\{C_{i,j}, Y_{i,j}, \pi_{i,j}\}_{i \in [n]}$ for party indices $j \in [t + 1]$ and outputs an aggregated transcript $AT := \{C_i, Y_i\}_{i \in [n]}$. In the following, let $\mu_1, \ldots, \mu_{t+1}$ denote the Lagrange coefficients for the set $[t + 1]$ at the point $x = 0$, i.e. $\mu_i := \prod_{j \in [t+1] \setminus \{i\}} j/(j - i)$ for $i \in [t + 1]$.

(1) For $i \in [n]$, compute $C_i := C_{i,1} \cdot \ldots \cdot C_{i,t+1}$ and $Y_i := Y_{i,1} \cdot \ldots \cdot Y_{i,t+1}$. Publish the aggregated transcript $AT := \{C_i, Y_i\}_{i \in [n]}$.

The *aggregation transcript verification algorithm* AVer takes as input public keys $pk_1, \ldots, pk_n$ and an aggregated transcript $AT := \{C_i, Y_i\}_{i \in [n]}$ as above. It outputs 1 (valid aggregated transcript) or 0 (invalid aggregated transcript).

(2) Check as usual that $\{C_i, Y_i\}_{i \in [n]}$ verifies using the pairing. If this checks fails, output 0 (invalid aggregated transcript). Otherwise, output 1 (valid aggregated transcript).

Note on aggregation verification. The aggregated transcript AT does not include any NIZK proofs in contrast to OptRand's scheme. However, to ensure security SPURT introduces a novel *collective verification* mechanism in which the set of elements of the individual transcripts $T_1, \ldots, T_{t+1}$ is distributed among all parties. This allows the parties to collectively verify that AT is indeed an aggregation of single transcripts $T_1, \ldots, T_{t+1}$.

Figure 9: Aggregation algorithm Agg **and aggregation transcript verification algorithm** AVer **of SPURT's APVSS.**

In the following, we explain how the proof differs from the proof of OptRand's APVSS. For this, we observe that there are only two differences between SPURT's and OptRand's APVSS. (1) The former assumes an additional generator $\hat{h} \in \mathbb{G}_2$ (resulting in a total of four generators $g, \hat{g} \in \mathbb{G}_1$, $h, \hat{h} \in \mathbb{G}_2$) whose purpose being solely to help their security analysis, which is a reduction from the decisional bilinear Diffie-Hellman (DBDH) problem. In particular, their PVSS scheme itself makes no use of this additional generator besides in the system parameter generation. (2) The NIZK proof $\pi = (c, r)$ of knowledge of $\alpha = f(0)$ is replaced by n *independently generated* knowledge-sound NIZK proofs $\pi_i = \{(c_i, r_i)\}_{i \in [n]}$ of discrete logarithm equality of commitment C_i and encrypted share Y_i for $i \in [n]$ (thus obviating the need to compute n pairings for this task). Here, a challenge c_i, $i \in [n]$, is computed as the cryptographic hash $\mathsf{H}(C_i, g^{r_i} C_i^{c_i}, Y_i, h^{r_i} Y_i^{c_i})$ defined by the non-interactive Chaum-Pedersen Σ-protocol. In particular, challenge c_i depends only on the pair (C_i, Y_i) and does not establish a connection to the whole transcript $\{C_i, Y_i, \ldots\}_{i \in [n]}$. In contrast to that, in Schoenmakers' PVSS scheme there is only one challenge c for all n NIZKs that is computed as hash of the whole transcript. The reason for SPURT's choice of using n separate challenges is that the design of their randomness beacon would not work otherwise, as we will see in the next chapter.

This choice, however, comes with a subtle nuance in the security analysis of SPURT's APVSS compared to the one from OptRand that we explain now. An argument in the analysis of OptRand's APVSS involves the observation that the challenge $c = \mathsf{H}(g^r \zeta^{-c}, g^\alpha)$ for the proof of knowledge of α is completely independent from the algebraic coefficients for g^α chosen by the adversary. In the case of SPURT's APVSS, the algebraic adversary could choose the algebraic coefficients for a tuple (C_i, Y_i) (that is input into the random oracle H to obtain the corresponding challenge c_i) dependent from previously obtained challenges c_j for different (C_j, Y_j), $j \neq i$, so that our above argument does not apply directly anymore. We solve this issue by regarding the on H last queried tuple (C_ℓ, Y_ℓ) which gives a challenge c_ℓ that is truly independent from the algebraic coefficients for all previously queried tuples (C_i, Y_i) chosen by the adversary. This allows us by summation over the algebraic equations coming from the n NIZKs to reduce to the single-challenge NIZK case as is given in OptRand's APVSS security analysis. Together with the fact that the additional generator $\hat{h} \in \mathbb{G}_2$ in SPURT's APVSS is an auxiliary element used only in their security analysis, allows us in the algebraic group model to transform the resulting algebraic equations to the algebraic equations that arose in the analysis of OptRand's APVSS, thus getting SPURT's APVSS aggregated unpredictability.

Proof. In the following, we explain how to adapt the proof for OptRand's PVSS into one for SPURT's PVSS. To this end, we first recapitulate the differences between these two designs. (1) SPURT's PVSS assumes an additional generator $\hat{h} \in \mathbb{G}_2$ (resulting in a total of four generators $g, \hat{g} \in \mathbb{G}_1$, $h, \hat{h} \in \mathbb{G}_2$) which the scheme itself makes no use of besides in the system parameter generation (its purpose is solely to help their security analysis, which is a reduction from the decisional bilinear Diffie-Hellman (DBDH) problem). (2) The NIZK proof $\pi = (c, r)$ of knowledge of $\alpha = f(0)$ in OptRand's PVSS is replaced in SPURT's PVSS by n independently generated knowledge-sound NIZK proofs $\pi_i = \{(c_i, r_i)\}_{i\in[n]}$ of discrete logarithm equality of commitment C_i and encrypted share Y_i for $i \in [n]$. Here, a challenge c_i for $i \in [n]$ is computed as the cryptographic hash $\mathsf{H}(C_i, g^{r_i}C_i^{c_i}, Y_i, h^{r_i}Y_i^{c_i})$ defined by the non-interactive Chaum-Pedersen Σ-protocol. In particular, challenge c_i depends only on the pair (C_i, Y_i) and does not establish a connection or dependence to other elements of the transcript $\{C_i, Y_i, \pi\}_{i\in[n]}$. In contrast to that, in Schoenmakers' PVSS scheme there is only one challenge c for all n NIZKs simultaneously that is computed as the hash of the whole transcript.

Let A be an algebraic adversary that $(\varepsilon', T', t, q_k, q_h)$-breaks aggregated unpredictability of PVSS. As in the previous proof, we make some simplifications. First, we assume that all parties are honest prior to the execution of PVSS. Second, we assume that there is contribution from exactly one corrupt party in the aggregated transcript. Third, we embed the COMDL instance in only the first answer to the transcript queries. The justification of these three points is analog to the one in the proof for OptRand's PVSS. The adversary wins the aggregated unpredictability game if it can predict the secret of the aggregation that it outputs at the end. In the following, let $C \subset \mathcal{P} = \{P_1, \dots, P_n\}$ be the dynamic set of corrupt parties and $\mathcal{H} = \mathcal{P} \setminus C$ the set of honest parties. The game between a challenger and the adversary is the same as in the previous proof with the following modifications as described above: (1) the system parameters are generated as $(\mathbb{G}_1, \mathbb{G}_2, p, g, \hat{g}, h, \hat{h})$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an asymmetric pairing of cyclic groups of prime order p with independent generators $g, \hat{g} \in \mathbb{G}_1$ and $h, \hat{h} \in \mathbb{G}_2$, and (2) a transcript $\{C_i, Y_i\}_{i\in[n]}$ is generated with n NIZKs proofs $\{\pi_i\}_{i\in[n]}$. At the end of the game, A outputs an aggregated transcript AT along with a secret $\sigma^* \in \mathbb{G}_T$. As A is an algebraic adversary, it returns the secret σ^* together with a representation

$$\left(a, b, \{c_i\}_{i=1}^n, \{d_i\}_{i=1}^n, \{e_i\}_{i=1}^n, \{f_i, u_i\}_{i=1}^n, \{v_{i,j}\}_{i,j=1}^n, \{w_{i,j}\}_{i,j=1}^n, \{v_i\}_{i=1}^{n+2}\right)$$

of elements in $\mathbb{Z}_p$ such that

$$\begin{aligned}\sigma^* &= e(g,h)^a \cdot e(\hat{g},h)^b \cdot \prod_{i=1}^n e(C_i,h)^{c_i} \cdot \prod_{i=1}^n e(g,pk_i)^{d_i} \\ &\cdot \prod_{i=1}^n e(g,Y_i)^{e_i} \cdot \prod_{i=1}^n e(\hat{g},pk_i)^{f_i} \cdot \prod_{i=1}^n e(\hat{g},Y_i)^{u_i} \\ &\cdot \prod_{i,j=1}^n e(C_i,pk_j)^{v_{i,j}} \cdot \prod_{i,j=1}^n e(C_i,Y_j)^{w_{i,j}} \\ &\cdot e(g,\hat{h})^{v_1} \cdot e(\hat{g},\hat{h})^{v_2} \cdot \prod_{i=1}^n e(C_i,\hat{h})^{v_{i+2}}. \qquad (\spadesuit)\end{aligned}$$

All our simulations will generate $\hat{h} \in \mathbb{G}_2$ honestly by sampling $\ell' \leftarrow \mathbb{Z}_p$ uniformly at random and setting $\hat{h} = h^{\ell'}$. By replacing the elements $e(X, \hat{h})$ with $e(X, h)^{\ell'}$ for $X \in \{g, \hat{g}, C_1, \dots, C_n\}$, the above equation reduces to the initial equation coming from the adversary's forgery obtained in OptRand's PVSS proof. The only difference is that the coefficients for $e(X, h)$ are shifted by ℓ' which sets no problem, since the reduction knows the value ℓ'. The other modification to OptRand's PVSS proof comes from the n NIZKs output by the adversary corresponding to the transcript of the contribution of the corrupt party (for the reduction, it simply simulates the n NIZKs for a transcript via n-time honest-verifier zero-knowledge (HVZK) simulation instead of one time). Let the corresponding equations be the following.

$$f'(j) = \tilde{a}_j + \ell\tilde{b}_j + \sum_{i=1}^n P(i)\tilde{c}_{j,i}/c'_j \quad \forall j \in [n], \qquad (\heartsuit)$$

which come directly from the NIZKs $\pi'_j = (c'_j, r'_j)$ output by the adversary with degree-t polynomial $f' \in \mathbb{Z}_p[X]$. From the input to the random oracle, there are also the equations

$$f'(j) = a_j^\dagger + \ell b_j^\dagger + \sum_{i=1}^n P(i)c_{j,i}^\dagger, \quad \forall j \in [n]. \qquad (\dagger)$$

Without loss of generality we assume that the adversary queries the random oracle on the corresponding group elements (C_j, Y_j) in ascending order starting from $j = 1$ up to $j = n$. Since we assume that the adversary wins the game, its output transcript is valid and thus f' is a polynomial of degree t. Therefore, once the adversary queried the random oracle on the group elements (C_j, Y_j) for $j \in [t + 1]$, the other elements (C_i, Y_i) for $i > t + 1$ can be computed (by the adversary) via Lagrange interpolation in the exponent and thus their algebraic representations won't give more information than needed. So we only consider the first $t + 1$ equations. An argument in the analysis of OptRand's PVSS involved the observation that the challenge $c = \mathsf{H}(g^r\zeta^{-c}, g^\alpha)$ for the proof of knowledge of α is completely independent from the algebraic coefficients for g^α chosen by the adversary. In the above case, however, the algebraic adversary could choose the algebraic coefficients for a tuple (C_i, Y_i) dependent from previously obtained challenges c_j for different (C_j, Y_j), $j < i$, so that the previous argument does not apply directly anymore. We solve this issue by regarding the last queried tuple (C_{t+1}, Y_{t+1}) which gives a challenge c_{t+1} that is truly independent from the algebraic coefficients for all previously queried tuples (C_i, Y_i), $i < t+1$ chosen by the adversary. Let $\mu_1, \dots, \mu_{t+1}$ be the Lagrange coefficients for the set $[t + 1]$ at the point $x = 0$. By Lagrange interpolation (and summation), the equations (♥) and (†) with the notation $\alpha' = f'(0)$ reduce to

$$\alpha' = \tilde{a} + \ell\tilde{b} + \sum_{i=1}^{n} P(i)(\tilde{c}_{i,1} + \tilde{c}_{i,2}/c'_{t+1}),$$

$$\alpha' = a^{\dagger} + \ell b^{\dagger} + \sum_{i=1}^{n} P(i)c_i^{\dagger},$$

with appropriately defined coefficients. In OptRand's PVSS proof, event E_2 was defined by $(\tilde{c}_1, \dots, \tilde{c}_n)/c' \in \mathbb{Z}_p^n$ being in the kernel of the linear map V. We replace this condition by E_2' defined by

$$(\tilde{c}_{1,1} + \tilde{c}_{1,2}/c'_{t+1}, \dots, \tilde{c}_{n,1} + \tilde{c}_{n,2}/c'_{t+1}) \in \mathbb{Z}_p^n$$

being in the kernel of V. Since the coefficients $(\tilde{c}_{i,1}, \tilde{c}_{i,2})$ for $i \in [n]$ are independent from c'_{t+1}, the same argumentation as in the previous proof applies and there is no other difference. With the same simulations and same extraction of solutions to the OMDL challenge, the proof goes through. This completes our proof. □

Remark E.1. By breaking down our above proof, we find that it can be adapted to obtain a security reduction (with the same parameters for tightness) from the plain discrete logarithm problem assuming a weaker static adversary. The main idea being to embed the discrete logarithm challenge $\xi \in \mathbb{G}$ into the public keys $\{pk_i\}_{i\in\mathcal{H}}$ of honest parties (which is fixed from the very beginning in the static corruption model) via $pk_i = \xi^{u_i} g^{v_i}$ for uniformly random $u_i, v_i \leftarrow \mathbb{Z}_p$, into the polynomial $f_1(X) = \alpha_0 + \alpha X + \ldots + \alpha_t X^t \in \mathbb{Z}_p[X]$ chosen by the simulator via $g^{\alpha_i} = \xi^{u_i} g^{v_i}$ for uniformly random $u_i, v_i \leftarrow \mathbb{Z}_p$, or into the second generator $\hat{g} \in \mathbb{G}$ via $\hat{g} = \xi$. Using this technique, the above proof can be adapted accordingly for the static case to reduce the security from the plain co-discrete logarithm problem (i.e. our asymmetric version of the discrete logarithm problem). The same is true for the security proof of OptRand's PVSS.

F SECURITY ANALYSIS AND DESCRIPTION OF THE OPTRAND AND SPURT RANDOMNESS BEACONS

In this appendix, we provide formal descriptions of the randomness beacons OptRand (Figure 10) and SPURT (Figure 11) in an abstracted manner. We do this for two reasons: (i) the abstractions that we provide, capture the most important building blocks of the beacons that are sufficient to understand them, and (ii) the actual protocols contain a lot of involved consensus parts.

Network Models. In a *synchronous network* as in OptRand, honest parties have local clocks that move at the same speed. Protocols proceed in rounds of fixed and a-priori known length Δ and parties start executing the protocol within Δ time from each other. Here, Δ corresponds to an upper known bound on the *network delay*: hen an honest party P sends a message at time τ, the message is guaranteed to be delivered by time $\tau + \Delta$. In particular, messages sent by honest parties cannot be dropped from the network and are always delivered. Thus, messages sent at the beginning of a round r are guaranteed to be delivered by the end of round r. However, the adversary has full control of the network subject to the above constraints and may deliver some messages much faster, i.e. within time $\delta \ll \Delta$. In addition, the adversary is *rushing* and may pick its messages *after* seeing the honest parties' messages within a round. In SPURT, the network is *partially synchronous*. This means that the network can have unbounded message delays which are under full control of the adversary (however, messages of honest parties may not be dropped). However, there is an unknown *global stabilization time (GST)*, which occurs *eventually*. After GST, the network behaves synchronously with parameter Δ as above.

F.1 Security Proof for the OptRand and SPURT Randomness Beacons

Here, we provide a full proof of Theorem 4.2.

Proof. In the following, we provide a security analysis of the randomness beacons of interest. There are four security notions to consider: consistency, availability, unpredictability, and bias-resistance. Regarding the first two notions, their validity follows directly from the analysis of the randomness beacon protocols in their respective papers [15, 34] by additionally noting that the underlying SMR protocols are already proven adaptively secure. The defining reason is that consistency and availability (also called *safety* and *liveness* in the context of SMR protocols) are exclusively part of the consensus layer and not affected by the security of the underlying APVSS scheme. For the last two notions, we provide a reduction to the aggregated unpredictability of the underlying APVSS schemes (which themselves reduce to the hardness of n-COMDL as seen before). To provide an intuition for the latter two notions, we have the following. Regarding 1-unpredictability of the randomness beacons, we observe: until reconstruction of a secret S_e in epoch e, the hash $\mathsf{H}(S_e)$ is uniformly random for the adversary A that corrupts up to t parties and does not break aggregated unpredictability of the underlying APVSS. Therefore, the inequality for the parameter ε in the theorem directly follows from the results on the underlying APVSS schemes from Chapter 3; the loss in $1/L$ comes from the reduction now guessing which of the at most L beacon outputs the adversary can predict. This implies 1-unpredictability. To argue bias-resistance, we observe that according to the above $\mathsf{H}(S_e)$ is uniformly random to A up to reconstruction of S_e, at which point it is fixed in the view of all honest parties and can no longer be altered. The algorithm D gets strictly less information than A about all output beacon values (since D only gets these values, but no information about the protocol execution). Therefore, the uniformity of $\mathsf{H}(S_e)$ for A prior to reconstruction of S_e implies uniformity for D of $\mathsf{H}(S_e)$ under the same assumptions. In the following, we denote by $\mathcal{RB}$ the DRB of consideration, i.e. $\mathcal{RB} \in \{OptRand, SPURT\}$.

- (t, L)-consistency and (t, L)-availability. From previous work [16, 64, 78], we know that the randomness beacon $\mathcal{RB}$ builds upon an already adaptively secure SMR (state machine replication) protocol. As a consequence, the analysis on (the weak variants of) consistency and availability remains the same as in their respective papers [15, 34], since the security of the consensus layer is not affected by the security of the underlying APVSS scheme.
- $(\varepsilon', T', t, L, q_h, 1)$-unpredictability. Let A be an algebraic adversary that breaks $(\varepsilon', T', t, L, q_h, 1)$-unpredictability of the randomness beacon protocol $\mathcal{RB}$. In particular, there is an epoch $\ell \in [L]$ in which A outputs a prediction (σ_e', e) for

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the distributed system of n parties. All parties run the underlying (responsive) state machine replication protocol SMR. The beacon protocol starts with epoch $e = 1$.

- **Setup Phase.** Set $e = 1$ and let $\mathcal{G} := \{P_1, \ldots, P_n\}$ denote the dynamic set of potentially good leaders. Parties agree on a buffer $\mathcal{B}(P_i) := \{AT_i\}$ of a valid random PVSS transcript for each $i \in [n]$.
- **Leader Election.** Use a round-robin leader election with respect to $\mathcal{G}$. If an epoch leader L_e fails to publish a valid aggregated PVSS transcript on SMR, blacklist it from future leader elections by setting $\mathcal{G} := \mathcal{G} \setminus \{L_e\}$.
- **Leader Proposal.** Upon receiving $t + 1$ valid PVSS transcripts from other parties at the beginning of epoch e, the leader L_e creates an aggregated PVSS transcript AT_e and publishes it on SMR.
- **Buffer Update.** Upon observing a valid aggregated transcript AT_e published by the leader L_e of some epoch e, update the buffer $\mathcal{B}(L_e) := \mathcal{B}(L_e) \cup \{AT_e\}$. At the end of epoch e, if no valid aggregated transcript was proposed for epoch $e - t$ by leader L_{e-t}, remove L_{e-t} from future leader elections by setting $\mathcal{G} := \mathcal{G} \setminus \{L_{e-t}\}$.
- **Reconstruction Phase.** Upon entering epoch e with leader L_e, pick the aggregated PVSS transcript AT_e from $\mathcal{B}(L_e)$ and run the PVSS reconstruction protocol on AT_e among all parties. Additionally, update $\mathcal{B}(L_e) := \mathcal{B}(L_e) \setminus \{AT_e\}$.
- **Output Generation.** Upon reconstruction of the secret S_e in epoch e, output the beacon value $\rho_e = \mathsf{Hash}(S_e) \in \{0,1\}^\lambda$.

On the Setup Phase. In order to agree on n valid random PVSS transcripts in their buffers, parties can run the SMR protocol for an initial $2n$ epochs, after which the first epoch $e = 1$ of the actual randomness beacon starts. This ensures that by the time the beacon protocol starts at least n valid aggregated transcripts are available to every party.

Figure 10: Distributed randomness beacon protocol OptRand described from the view of party P_i.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the distributed system of n parties. All parties run the underlying (responsive) state machine replication protocol SMR. The beacon protocol starts with epoch $e = 1$.

- **Setup Phase.** There is no setup phase. As a result, the protocol does not guarantee full availability.
- **Leader Election.** Use a deterministic round-robin leader election with respect to $\mathcal{P}$.
- **Leader Proposal.** Upon receiving $t + 1$ valid PVSS transcripts $T_1, \ldots, T_{t+1}$ from other parties at the beginning of epoch e, the leader L_e creates an aggregated PVSS transcript AT_e that consists of the commitments and encrypted shares only. It publishes $\mathsf{Hash}(AT_e)$ on SMR, multicasts AT_e, and additionally sends a part $(AT_e)_i$ to party P_i (see bottom explanation).
- **Buffer Update.** Upon observing a valid aggregated transcript AT_e published by the leader L_e of some epoch e, update the buffer $\mathcal{B}(L_e) := \mathcal{B}(L_e) \cup \{AT_e\}$. At the end of epoch e, if no valid aggregated transcript was proposed for epoch $e - t$ by leader L_{e-t}, output $\bot_{\mathcal{RB}}$ as beacon output for that epoch.
- **Output Generation.** Upon entering epoch e with leader L_e, pick the aggregated PVSS transcript AT_e from $\mathcal{B}(L_e)$ and run the PVSS reconstruction protocol on AT_e among all parties to obtain the secret S_e. Output the beacon value $\rho_e = S_e \in \{0,1\}^\lambda$. Additionally, update $\mathcal{B}(L_e) := \mathcal{B}(L_e) \setminus \{AT_e\}$.

On the Leader Proposal and Aggregation. The aggregated transcript AT_e consists only of the commitments and encrypted shares, the NIZK proofs are ignored for AT_e. However, the leader also sends $(AT_e)_i$ to party P_i which is defined as the i-th part of the collection $\{T_1, \ldots, T_{t+1}\}$ of individual transcripts including the NIZKs. Via quorum certificates the parties can collectively verify that the aggregate AT_e is indeed a valid aggregation of the single transcripts $T_1, \ldots, T_{t+1}$.

Figure 11: Distributed randomness beacon protocol SPURT described from the view of party P_i.

an $e \in [\ell + 1, L]$ such that $\sigma'_e = \sigma_e$ where σ_e is the randomness beacon output for epoch e. We use algorithm A and its prediction to build an algebraic reduction R that breaks aggregated unpredictability of the underlying APVSS. For this, we simulate an execution of $\mathcal{RB}$ for the adversary A with the help of the oracles provided by the aggregated unpredictability game for APVSS. In the following, let $C \subset \mathcal{P} := \{P_1, \ldots, P_n\}$ be the dynamically changing set of corrupt parties and $\mathcal{H} := \mathcal{P} \setminus C$ the set of honest parties. We consider the following game of aggregated unpredictability of APVSS between R (with black-box access to the randomness beacon predictor A) and a challenger C.

Reduction R: In the following, we describe the workings of the reduction R. The challenger C provides R with public keys $\{pk_i\}_{i \in [n]}$ along with system parameters *par*. Before starting the simulation of the randomness beacon $\mathcal{RB}$ with A (having full control over parties in C), algorithm R makes a guess which of the at most L beacon outputs the adversary will predict. For this, the reduction samples a number $e^* \leftarrow [L]$ uniformly at random from the set $[L]$. Then, R begins the offline phase of an execution of $\mathcal{RB}$ and runs A on input *par* and $\{pk_i\}_{i \in [n]}$. A returns an index set $C \subset [n]$ of initially corrupted parties along with updated public keys $\{\hat{pk}_j\}_{j \in C}$, which R forwards to C. The challenger sets $pk_j := \hat{pk}_j$ for all $j \in C$. Subsequently, R begins an execution of $\mathcal{RB}$ with A controlling parties in C. At any point of the execution, A may corrupt a party P_i by submitting an index $i \in [n] \setminus C$. In this case, R forwards this corruption query to C who returns the secret key sk_i of P_i to R. Upon receiving this secret key, R forwards it to A and gives A also full control over P_i. Additionally, at any point of the execution of $\mathcal{RB}$, A may query the random oracle H on input m_i. In this case, R checks if $H[m_i] = \perp$. If so, it samples $r_i \leftarrow \{0,1\}^\lambda$ uniformly at random and returns $H[m_i] := r_i$. Now, for all epochs $\ell \in [L] \setminus \{e^*\}$, algorithm R simulates an honest execution of $\mathcal{RB}$ on behalf of all honest parties. For this, R samples on behalf of each honest party $P_i \in \mathcal{H}$ a polynomial $f_{i,\ell} \leftarrow \mathbb{Z}_p[X]$ of degree t uniformly at random, runs the aggregated distribution protocol ADist on it, and sends the corresponding PVSS transcript $T_{i,\ell}$ to the leader L_ℓ of the epoch. All other instructions of the randomness beacon protocol are also executed honestly. For epoch $e^* \in [L]$, however, R does the following. On behalf of all honest parties $P_i \in \mathcal{H}$, it queries the transcript oracle provided by C. More precisely, it submits (givePVSS, i) for all $i \in \mathcal{H}$, upon which C returns PVSS transcripts T_{i,e^*}. Algorithm R uses these transcripts to simulate the behavior of honest parties in the execution of $\mathcal{RB}$ with A. Denote the aggregated transcript of that epoch e^* by AT_{e^*}. We define the event E by $e = e^*$. Obviously, we have $\Pr[E] = 1/L$, since $e^* \in [L]$ is sampled uniformly at random and unknown to A. In case event E does not happen, R aborts the game with C and the execution of $\mathcal{RB}$. Note that this happens with probability $1 - \Pr[E] = 1 - 1/L$. Otherwise, A outputs with probability ε' a successful prediction (σ'_e, e) such that $\sigma'_e = \sigma_{e*}$ before the reconstruction of AT_{e^*}. Since $\sigma_{e*} = \mathsf{H}(S_{e^*})$, where S_{e^*} is the reconstructed secret of the aggregated transcript AT_{e^*}, algorithm A must have queried the random oracle H on S_{e^*} before outputting its prediction. As a consequence, it must have submitted S_{e^*} to R before outputting σ_{e^*}. Following the prediction output by A, R aborts the execution of $\mathcal{RB}$ with A and submits (AT_{e^*}, S_{e^*}) as a solution to C for the aggregated unpredictability game. R's success probability in breaking aggregated unpredictability of APVSS is then given by

$$\begin{aligned}\varepsilon &= \Pr[\mathbf{AggPred}^{\mathsf{R}}_{\mathsf{APVSS}} = 1]\\ &\geq \Pr[E] \cdot \Pr[\mathbf{Unpred}^{\mathsf{A}}_{\mathcal{RB}} = 1] - \frac{q'_h}{2^\lambda}\\ &\geq \frac{1}{L} \cdot \varepsilon' - \frac{q'_h}{2^\lambda} = \frac{\varepsilon'}{L} - \frac{q'_h}{p},\end{aligned}$$

where the negative summand $q'_h/2^\lambda$ (which is equal to q'_h/p, since p has λ-bit size) comes from the fact that the hash $\mathsf{H}(S_{e^*}) \in \{0,1\}^\lambda$ could occur more than once in the list of q'_h queried hashes by A, in which case R cannot recover S_{e^*}. Regarding the running time bound, R has to simulate each of the at most L epochs which comes with a computational overhead of $L \cdot O(n^2)$.

- (ε', T', t, L)-bias-resistance. We begin with the observation that D gets strictly less information than A about the beacon outputs (since D only gets these output values and no other information about the $\mathcal{RB}$ protocol execution). Now, our previous analysis on the 1-unpredictability of the randomness beacon output for A, however, implies that even A cannot distinguish the beacon output before reconstruction from uniform with probability better than ε'. As a result, D's success probability in distinguishing the beacon outputs from uniform is bounded by A's success probability ε' in predicting the beacon output. Thus yielding the desired (ε', T', t, L)-bias-resistance property.

□