SoK: Dlog-based Distributed Key Generation

Renas Bacho
CISPA Helmholtz Center for Information Security
renas.bacho@cispa.de

Alireza Kavousi
University College London
a.kavousi@cs.ucl.ac.uk

***Abstract*—Distributed Key Generation (DKG) protocols are fundamental components of threshold cryptography, enabling key generation in a trustless manner for a range of cryptographic operations such as threshold encryption and signing. Of particular widespread use are DKG protocols for discrete-logarithm based cryptosystems. In this Systematization of Knowledge (SoK), we present a comprehensive analysis of existing DKG protocols in the discrete-logarithm setting, with the goal of identifying cryptographic techniques and design principles that facilitate the development of secure and resilient protocols. To offer a structured overview of the literature, we adopt a modular approach and classify DKG protocols based on their underlying network assumption and cryptographic tools. These two factors determine how DKG protocols manage secret sharing and reach consensus as their essential building blocks. We also highlight various insights and suggest future research directions that could drive further advancements in this area.**

1. Introduction

Distributed Key Generation (DKG) [66], [92] is a form of multi-party computation that allows a group of mutually distrustful parties to jointly generate a public-secret key pair. Each party then holds a share of the secret key such that a certain threshold t-out-of-n parties is necessary to reconstruct that key. DKG protocols are essential components for building up decentralized systems, as they enable to perform cryptographic operations such as signing and decryption without suffering from a single point of failure. As such, a DKG shifts the public-secret key pair generation from a single party to a set of n parties such that each of them holds a partial public-secret key pair for conducting cryptographic operations. At the same time, there is a single common public key, representing the whole set of n parties, that allows for efficient verification of the resulting output. Of particular widespread use are DKG protocols for discrete-logarithm based cryptosystems[1], on which we focus in this work. There is a large range of applications of DKG protocols across multiple domains, including threshold encryption and signing [67], [93], [98], Byzantine fault-tolerant consensus [65], [102], secure multi-party computation (MPC) [25], [68], distributed randomness beacons (DRB) [38], [42], [81], and more.

Abstractly speaking, a DKG protocol typically has the following three-phase structure:

- **Sharing Phase.** Each party chooses a random secret and shares it among all parties via a threshold secret sharing scheme. This phase can be interactive or non-interactive.
- **Agreement Phase.** Parties agree on a subset of parties whose secrets should be included for computing the final secret key. This phase is often interactive.
- **Key Derivation Phase.** Parties engage to derive their keys and terminate. This phase is often non-interactive.

In this paper, we synthesize existing efforts on the DKG literature with the goal of providing a clear, cohesive, and critical understanding of their underlying design and efficiency. Our framework comprehensively captures the DKG protocols by identifying two main ingredients in their design, namely *secret sharing* and *consensus*. Thus, we base our categorization on their underlying *network model* that crucially affects both mentioned components. To facilitate a modular approach, we further divide the protocols according to the way they handle secret sharing via threshold secret sharing tools. This allows us to observe and make connections between cryptographic and distributed system perspectives, leading to new insights and research gaps. We provide sufficient technical details to highlight the key ideas and challenges in designing DKG protocols while taking a bottom-up approach to make it accessible for general readers interested in this area. To the best of our knowledge, we are the first to present a holistic analysis of DKG protocols and their properties. We refer to Figure 1 for a pictorial overview of the DKG components and relevant properties.

Paper organization. In Section 2, we introduce the required background, including cryptographic and distributed systems primitives. In Section 3, we present the notion of a DKG protocol and explain our methodology of systematization. Following our methodology, in Sections 4 and 5, we then investigate DKG protocols in-depth under synchronous and asynchronous network models, respectively. Finally, in Section 7, we conclude with a discussion on further related properties and considerations. Throughout the paper, we present numerous insights and potential research directions labeled by “Insight” and “Gap”, respectively.

1. Distributing keys for public-key cryptography is generally considered for two settings: discrete-logarithm, and RSA modulus [26]. A few recent works also consider the post-quantum setting [53].

2025 IEEE Symposium on Security and Privacy (SP) | 979-8-3315-2236-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/SP61157.2025.00127

DOI 10.1109/SP61157.2025.00127

2. Background and Preliminaries

General notation. We denote the security parameter by λ. For a set X, we use $x \leftarrow\!\!\$\ X$ to denote that x is sampled uniformly at random from X. For a positive integer $x \in \mathbb{N}$, we define $[x] := \{1, \ldots, x\}$. Throughout the paper, we consider a network of n parties, denoted $\mathcal{P} := \{P_1, \ldots, P_n\}$, connected by point-to-point channels[2]; for mathematical formulations, we may also use $\{1, \ldots, n\}$ for $\mathcal{P}$. We assume a *Byzantine* adversary $\mathcal{A}$ who can corrupt up to t out of the n parties and may cause them to deviate from the protocol arbitrarily. We may refer to the non-faulty parties as *honest* and to the faulty parties as *corrupt* or *Byzantine*.

2.1. Network Models

The literature on DKG generally considers two types of network models: the *synchronous* and the *asynchronous*.[3] Abstractly speaking, the model defines to what extend the adversary can delay messages between honest parties.

Synchronous network. In a synchronous network [83], communication between parties proceeds in compute-send-receive rounds of a-priori known length $\Delta > 0$, called the *network delay*. When an honest party sends a message at the beginning of a round, the message is guaranteed to be received by the receiving party by the end of that round. In particular, messages sent by honest parties cannot be dropped from the network and are always delivered. Honest parties have local clocks that move at the same speed and they all start the protocol at the same time. In this model, the adversary can delay messages for at most Δ time.

Asynchronous network. In an asynchronous network [84], any message can be delayed arbitrarily under the constraint that messages sent between honest parties must *eventually* be delivered. In particular, honest parties' local clocks and processing speeds are not necessarily synchronized, and parties may start the protocol at different times. As a consequence, some honest parties may have terminated the protocol before other honest parties even started it. The difficulty in designing asynchronous protocols lies in the fact that it is, in general, not possible to distinguish a slow honest party from a silent Byzantine party.

For more details on the different network models, we refer the reader to the works on (Byzantine fault-tolerant) distributed algorithms by Lynch [87] and Davidson [47].

2.2. Consensus Primitives

Here, we define the consensus primitives that are commonly used in the design of DKG protocols.

2. For practical purposes, the point-to-point channels are often assumed to be authenticated and sometimes even secret.

3. The literature on distributed systems often also considers the *partially synchronous* network model [52]. However, we are aware of only one work on DKG in this particular setting [76] (see Section 6).

Byzantine broadcast. A *Byzantine broadcast (BBC)* protocol [49], often also called *synchronous broadcast*, allows a designated party P_s, called the *sender* or *dealer*, to consistently send a message to all parties in the network. Formally, let Π_{BBC} be a protocol executed by parties $P_1, \ldots, P_n$, where a designated sender P_s holds an input value v. Then, Π_{BBC} is a t-secure broadcast protocol if the following properties hold in the presence of up to t corrupt parties. (i) *Termination*: Every honest party P_i outputs a value v_i and terminates. (ii) *Agreement*: All honest parties output the same value, *i.e.*, $v_i = v_j$ for any two honest parties P_i and P_j. (iii) *Validity*: If the sender P_s is honest, then all honest parties output v, *i.e.*, $v_i = v$ for any honest party P_i.

Reliable broadcast. Similar to the above, a *reliable broadcast (RBC)* protocol [24] allows a designated sender P_s to consistently send a message to all parties in the network. In contrast to synchronous broadcast, it does not require full termination and instead has a weaker notion of termination tailored to asynchronous networks. Formally, let Π_{RBC} be a protocol executed by parties $P_1, \ldots, P_n$, where a designated sender P_s holds an input value v. Then, Π_{RBC} is a t-secure reliable broadcast protocol if the following properties hold in the presence of up to t corrupt parties. (i) *Termination*: If some honest party outputs a value, then every honest party eventually outputs a value. (ii) *Agreement*: All honest parties output the same value (if any). (iii) *Validity*: If the sender P_s is honest, then all honest parties output v.

Multivalued validated Byzantine agreement. A *multivalued validated Byzantine agreement (MVBA)* protocol [28] allows a set of parties, each holding an input value $v_i \in V$ from a value set V with $|V| \geq 2$, to agree on one common output value $v' \in V$ satisfying an external validity function $\mathsf{val}\colon V \to \{0,1\}$; a value $v \in V$ is called *externally valid* if $\mathsf{val}(v) = 1$. Formally, let Π_{MVBA} be a protocol executed by parties $P_1, \ldots, P_n$, where each party P_i holds an input value $v_i \in V$, and let $\mathsf{val} : V \to \{0,1\}$ be an external validity function. Then, Π_{MVBA} is a t-secure MVBA protocol if the following properties hold in the presence of up to t corrupt parties, and assuming every honest party's input value is externally valid. (i) *Termination*: Every honest party P_i terminates with an output value v'_i. (ii) *Agreement*: All honest parties output the same value v'. (iii) *External Validity*: All honest parties (that output any value) output an externally valid value.

Remark 1. There exists also the related concept of an *asynchronous common subset (ACS)* protocol [15], historically also called *agreement on a core set*. In an ACS protocol, each party P_i holds an input value v_i, and all honest parties terminate with a common subset $X \subseteq \{v_1, \ldots, v_n\}$ of input values of size $|X| \geq n - t$ containing at least $n - 2t$ values proposed by honest parties. Since most DKG protocols are built upon an MVBA protocol (as the consensus part), we just presented a formal definition for it. Further, we note that there are known transformations from MVBA to ACS [28].

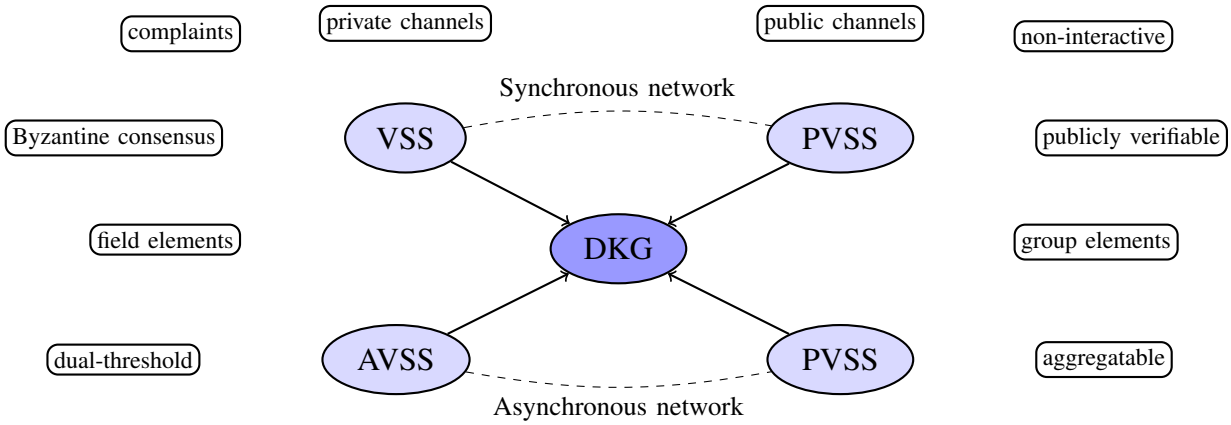

Figure 1: Overview of DKG ingredients and relevant properties.

2.3. Secret Sharing Primitives

Here, we define (threshold) secret sharing primitives that are commonly used in the design of DKG protocols.

Shamir secret sharing. The (t, n)-*threshold Shamir secret sharing* [97] allows to embed a secret $s \in \mathbb{Z}_p$ in the constant term of a polynomial $f(x) = s + a_1 x + \cdots + a_t x^t \in \mathbb{Z}_p[x]$, where other coefficients $a_1, \ldots, a_t \leftarrow\!\!\$\ \mathbb{Z}_p$ are chosen uniformly at random. The i-th share of the secret s is the evaluation point $f(i)$. Given $t+1$ distinct shares, one can efficiently reconstruct the polynomial f and the secret s via Lagrange interpolation. Further, s is information-theoretically hidden from an adversary that obtains up to t shares.

Verifiable secret sharing. A *verifiable secret sharing (VSS)* scheme [39] allows a designated dealer P_d to (t, n)-Shamir share a secret $s \in \mathbb{Z}_p$ among a set of parties in a verifiable manner, even in the presence of malicious parties. Formally, let Π_{VSS} be a protocol, consisting of a *sharing* phase and a *reconstruction* phase, executed by parties $P_1, \ldots, P_n$, where a designated dealer P_d holds a secret $s \in \mathbb{Z}_p$ as input. Then, Π_{VSS} is a t-secure VSS protocol if the following properties hold in the presence of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ corrupting up to t parties. (i) *Termination*: Every honest party P_i terminates the sharing phase and outputs a share s_i. (ii) *Completeness*: If some honest party outputs a share upon completing the sharing phase, then there exists a unique secret s' (potentially $\perp$) such that every honest party outputs a share of s' upon completing the sharing phase. Further, if P_d is honest, then $s' = s$. (iii) *Secrecy*: If P_d remains honest by the end of the sharing phase, then no information about the secret s is leaked to the adversary $\mathcal{A}$ during the sharing phase (other than the public information).

Asynchronous VSS. Similar to reliable broadcast, an *asynchronous VSS (AVSS)* scheme [31] can only satisfy a weaker notion of termination. Concretely, an asynchronous VSS protocol Π_{AVSS} satisfies the property of termination if the following holds: If some honest party terminates the sharing phase (and outputs a share), then all honest parties eventually terminate the sharing phase.

Remark 2. There are recent works [46], [103] that call such schemes an *asynchronous complete secret sharing (ACSS)* and use the term AVSS for schemes with a weaker notion of completeness that only guarantees that honest parties can reconstruct the secret, but not that all honest parties output shares upon completing the sharing phase. In this work, we stick to the historical notion of AVSS, where (strong) completeness is explicitly required. We choose to do so, as all existing asynchronous DKG protocols employ an AVSS scheme *with* (strong) completeness.

Publicly verifiable secret sharing. An important variant of threshold secret sharing is the one with *public verifiability*, called *publicly verifiable secret sharing (PVSS)* [100]. Such a scheme outputs a single transcript that allows verification of correct sharing by any third party, and not only the participating parties. Technically, a PVSS scheme achieves this property by encrypting the shares $\mathbf{E} = (\hat{s}_1, \ldots, \hat{s}_n)$ and generating a validity proof π showing that the encrypted shares are all valid with respect to a degree-t polynomial. Apart from completeness, a PVSS must satisfy the following two security properties. (i) *Soundness*: If the proof π verifies, then the values $\hat{s}_i$ are indeed a valid sharing of some secret s embedded in a degree-t polynomial.[4] (ii) *Secrecy*: If the dealer remains honest by the end of the sharing phase, then no information about the secret s is leaked to $\mathcal{A}$ during the sharing phase (other than the public information). Further, if a PVSS scheme allows to homomorphically combine two transcripts for polynomials f_1 and f_2 into a single transcript for their sum $f_1 + f_2$ (while preserving public verifiability), then it is called an *aggregatable PVSS (APVSS)* [12], [72].

3. Distributed Key Generation

In this section, we first give a formal definition for DKG following the literature and then elaborate on our

4. The proof also certifies the verification of the decrypted shares in the reconstruction phase, which we implicitly assume.

methodology of systematization.

3.1. Formal Definition

A *Distributed Key Generation (DKG)* protocol for discrete logarithm amounts to a (t, n)-threshold Shamir sharing of a uniformly random secret key $\mathsf{sk} \in \mathbb{Z}_p$ among a set of n parties $P_1, \ldots, P_n$, where each party P_i outputs a secret key share sk_i of sk, the public key $\mathsf{pk} := g^{\mathsf{sk}}$, and a vector of public key shares $\mathbf{Y} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, where $\mathsf{pk}_i := g^{\mathsf{sk}_i}$ for all $i \in [n]$.

The natural way to formally define security of a DKG protocol is via a simulation-based approach where an appropriate ideal functionality is defined. To this end, we adopt the one recently proposed by Katz [79] for a *fully secure* DKG protocol. A DKG protocol Π_{DKG} t-securely realizes the ideal functionality $\mathcal{F}_{\mathsf{DKG}}$ (cf. Figure 2) if for any PPT adversary $\mathcal{A}$ corrupting up to t parties, there exists a PPT simulator $\mathcal{S}$ such that no PPT distinguisher can distinguish interacting with parties in Π_{DKG} and the functionality $\mathcal{F}_{\mathsf{DKG}}$. Intuitively, this definition captures the commonly used correctness and secrecy requirement proposed by Gennaro *et al.* [66], [67]. Further, the functionality $\mathcal{F}_{\mathsf{DKG}}$ also captures *robustness*, *i.e.*, it always completes successfully with honest parties outputting valid shares.

Definition 1 (Secrecy)**.** The DKG protocol Π_{DKG} is t-secret if for any PPT adversary $\mathcal{A}$ corrupting up to t parties, there exists a PPT simulator $\mathcal{S}$ that on input a uniformly random element $y \in \mathbb{G}$, produces a view which is indistinguishable from $\mathcal{A}$'s view in a run of Π_{DKG} that terminates with output y as the public key.

Definition 2 (Robustness)**.** The protocol Π_{DKG} is t-robust if the following properties hold in the presence of up to t corrupt parties. Every honest party P_i outputs the same tuple $(\mathsf{pk}, \mathbf{Y})$, and its secret key share sk_i satisfies $\mathsf{pk}_i = g^{\mathsf{sk}_i}$. Further, for any two sets $I, J \subseteq \{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$, each having at least $t + 1$ honest shares, a unique secret key sk can be reconstructed where $\mathsf{pk} = g^{\mathsf{sk}}$.

Remark 3. There are recent works that introduce weaker notions of security for DKG, such as *key-expressability* [72] or *oracle-aided algebraic simulatability* [11], and show that these are sufficient for secure applications such as threshold BLS or threshold Schnorr signing. We recap these notions in Appendix A and briefly discuss them in Section 6.

3.2. Systematization Methodology

DKG protocols have seen considerable attention from the research community for decades. Recent research on DKG protocols focuses on improving their security and scalability, particularly due to the need for trust distribution in the blockchain and web3 settings. We base our systematization on a distinguishing factor in developing DKG protocols which is their underlying network model, affecting how they reach consensus. Then, we give an overview on their design and complexity according the way they handle secret sharing and provide a critical analysis of the techniques and challenges they pose.

$\mathcal{F}_{\mathsf{DKG}}$

- Sample $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$ uniformly at random, and then compute $\mathsf{pk} := g^{\mathsf{sk}} \in \mathbb{G}$.
- Generate (t, n)-threshold Shamir shares $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$ of sk. Set $\mathsf{pk}_i := g^{\mathsf{sk}_i}$ for $i \in [n]$, and $\mathbf{Y} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$.
- Send sk_i and $(\mathsf{pk}, \mathbf{Y})$ to each party P_i for $i \in [n]$. Further, send $(\mathsf{pk}, \mathbf{Y})$ to the adversary.

Figure 2: The ideal functionality $\mathcal{F}_{\mathsf{DKG}}$

Synchronous DKG. The typical approach for protocols in this category involves an initial sharing phase where n parties each distribute their random secrets simultaneously. This is followed by an agreement phase, during which all honest parties agree on a common set of qualified parties whose secrets were correctly distributed. The DKG secret is then the sum of all the correctly shared secrets and each party's partial secret key is the sum of all the corresponding shares. The common public key and the parties' partial public keys are derived respectively. Almost all synchronous DKG protocols in the literature employ broadcast channels over several rounds, which are commonly implemented via efficient Byzantine broadcast [49] or Byzantine agreement [4] protocols. We provide a summary of existing synchronous DKG protocols with their relevant properties in Table 1.

Asynchronous DKG. Conceptually, the protocols in this setting follow the same paradigm of sharing secrets in parallel and agreeing on a common set for key derivation. However, the asynchronous communication model introduces unique challenges that directly affect the design of DKG protocols. The primary difficulty lies in the agreement phase, which is usually implemented by an MVBA (or ACS) protocol. By the famous FLP impossibility result [60], we know that reaching agreement under asynchronous communication necessitates randomness, often in the form of *shared/common* randomness. Typically, common randomness is generated from a threshold key setup, which itself requires a DKG (or a strong form of trusted setup), creating a circularity. Tackling this circularity is in general what underlies the difficulty and differences in asychronous DKG constructions. We also provide a summary of existing asynchronous DKG protocols in Table 3.

4. Synchronous DKG

We classify synchronous DKG protocols based on how they implement the sharing phase. The two popular secret sharing tools are VSS and PVSS, each of which has its own specific considerations outlined as follows.

TABLE 1: Comparison of existing synchronous DKG protocols.

Protocol	Year	Corruption	Communication	Rounds	Field	Sharing	Secrecy	Miscellaneous
Pedersen [93]	*1991*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$3 \cdot \mathsf{BC}_n$	✓	VSS	✗	signatures
Gennaro *et al.* [66]	*1999*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$5 \cdot \mathsf{BC}_n$	✓	VSS	✓	no signatures
Canetti *et al.* [30]	*1999*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$5 \cdot \mathsf{BC}_n$	✓	VSS	✓	erasures, adaptive
Jarecki *et al.* [75]	*2000*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$10 \cdot \mathsf{BC}_n$	✓	VSS	✓	no erasures, adaptive
Fouque-Stern [61]	*2001*	$t < n$	$n \cdot \mathsf{BC}_n(\lambda n)$	$1 \cdot \mathsf{BC}_n$	✓	PVSS	✗	non-interactive, heavy NIZKs, DCR
Canny-Sorkin [32]†	*2004*	$t < \epsilon n/2$	$n \cdot \mathsf{BC}_\ell(\lambda \ell) + n^2 \lambda \ell$	$5 \cdot \mathsf{BC}_\ell$	✓	VSS	✓	committees, $\ell \in \Theta(\kappa \log n/(1-\epsilon)^2)$
Abe-Fehr [2]	*2004*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$5 \cdot \mathsf{BC}_n$	✓	VSS	✓	no erasures, adaptive
Kate *et al.* [78]†	*2010*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda + \delta \lambda n)$	$3 \cdot \mathsf{BC}_n$	✓	VSS	✗	powers-of-tau (PoT), $\delta \in \{0,1\}$
Neji *et al.* [89]	*2016*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$2 \cdot \mathsf{BC}_n$	✓	PVSS	✓	public key $\mathsf{pk} = h^{\mathsf{sk}}$ for $h \neq g$
Gurkan *et al.* [72]†	*2021*	$t < \log n$	$n\mathsf{BC}_n(\lambda) + \ell^2 \mathsf{BC}_n(\lambda n)$	$\ell \cdot \mathsf{BC}_n$	✗	APVSS	✗	gossiping, $\ell \in \Theta(\kappa \log n)$
Groth [70]	*2021*	$t < n$	$n \cdot \mathsf{BC}_n(\lambda^2 n)$	$1 \cdot \mathsf{BC}_n$	✓	PVSS	✗	non-interactive, heavy NIZKs
Cascudo *et al.* [37]	*2021*	$t < n/2$	$n \cdot \mathsf{BC}_n(\lambda n)$	$4 \cdot \mathsf{BC}_n$	✓	PVSS	✓	efficient complaints
Shrestha *et al.* [99]	*2021*	$t < n/2$	$\lambda n^2(t+1)$	$7(t+1)$	✓	VSS	✓	PoT, no broadcasts, signatures
Shrestha *et al.* [99]†	*2021*	$t < n/2$	$\lambda n^3(r+1)$	$29 + 9r$	✓	VSS	✓	as above, randomized with $r \geq 2$
Kate *et al.* [77]	*2023*	$t < n$	$n \cdot \mathsf{BC}_n(\lambda n)$	$1 \cdot \mathsf{BC}_n$	✓	PVSS	✗	non-interactive, heavy NIZKs, CL
Cascudo-David [35]	*2023*	$t < n$	$n \cdot \mathsf{BC}_n(\lambda n)$	$1 \cdot \mathsf{BC}_n$	✓	PVSS	✗	non-interactive, heavy NIZKs, CL
Feng *et al.* [58]	*2023*	$t < n/2$	$\kappa \cdot \mathsf{BC}_n(\lambda n) + n\lambda\kappa$	$2 \cdot \mathsf{BC}_n$	✓	VSS	✗	coin-aided, erasures, signatures
Bacho *et al.* [10]	*2023*	$t < n/2$	$\mathsf{BC}_n(\lambda n) + n^2 \lambda \log n$	$4 \cdot \mathsf{BC}_n$	✗	APVSS	✗	recursive design
Feng *et al.* [57]	*2024*	$t < n/2$	$\sqrt{n} \cdot \mathsf{BC}_n(\lambda n)$	$2 \cdot \mathsf{BC}_n$	✗	APVSS	✗	network partitioning
Feng *et al.* [57]	*2024*	$t < n/2$	$\sqrt{n} \cdot \mathsf{BC}_n(\lambda n \kappa)$	$2 \cdot \mathsf{BC}_n$	✓	PVSS	✗	network partitioning

In **Communication**, we give the communication cost in bits and omit multiplicative constants. Here, $\mathsf{BC}_a(b)$ denotes the cost of one party broadcasting b bits to a parties. In **Rounds**, we omit additive constants (independent of n). Here, BC_a denotes the number of rounds of one broadcast among a parties (which is at least linear in n due to known lower bounds [48]). The **Secrecy** column means *full secrecy* (cf. Definition 1), and **Miscellaneous** lists further important properties or assumptions. Here, DCR denotes the decisional composite residuosity assumption, and CL denotes certain class group assumptions. Throughout, κ is a statistical security parameter which *could* potentially be smaller than λ for some applications. † In [32], $\epsilon < 1$ is a parameter that can be chosen. In [78], it is $\delta = 0$ only in the failure-free case (without any Byzantine parties). In [72], the protocol can tolerate up to $n/2 - \log n$ additional crash faults. In [99], the randomized protocol terminates in epoch $r \geq 2$ with probability $1 - 2^{-r}$ (thus, in expectation $r = 2$ is enough).

4.1. Using VSS

Design. The first synchronous DKG protocol is due to Pedersen [93], although the concept was already introduced by Ingemarsson and Simmons [74]. The protocol works by having each party P_i share a random secret $s_i \in \mathbb{Z}_p$ with others through Feldman's verifiable secret sharing (VSS) [54]. The secret key sk is then equal to the sum of the set $I \subseteq [n]$ of at least $t+1$ correctly shared secrets agreed upon by honest parties. The public key pk is derived respectively due to the property of the underlying secret sharing that reveals $\{g^{s_i}\}_{i \in [n]}$. This design rational is the basis for many of the subsequent efforts in developing DKG protocols.

Despite its efficiency and the simplicity of the design, Gennaro *et al.* [66] showed the insecurity of the Pedersen DKG due to the ability of the adversary to manipulate the generated key-pair, failing to guarantee the uniform distribution of the output. More precisely, a (rushing) adversary[5] can manipulate the set of parties whose contributions are considered for computing the DKG secret/public key after viewing their respective public values. Consequently, the protocol does not offer a simulation-based security given that the simulator cannot efficiently[6] decide on the set of honest parties while generating an indistinguishable transcript for the protocol. The basic idea behind [66] is to make the sharing algorithm information-theoretically secure, so that no information is revealed for the adversary to be able to manipulate the set of qualified parties accordingly. This is done using Pedersen VSS [92] in the sharing step, which inures an additional round of communication (via broadcast channel) to enable public key extraction. Note that original Pedersen DKG [93] requires three rounds of broadcast for sending the commitments and for handling complaints, where a party may raise against some dealer upon receiving an invalid share or no share at all. This number of rounds seem to be inherent for the typical complaint-based DKG protocols, as pointed out by Katz [79]. It turns out, however, the Pedersen DKG is sufficient to securely instantiate those threshold cryptosystems where the only security guarantee is the inability of the adversary to learn the secret key by a direct reduction to the hardness of Discrete Logarithm (DL) problem in the random oracle model [66], [67]. A well-known candidate is the threshold Schnorr signature scheme that is proven to be unforgeable [67].

> **Insight 1.** The secure variant of Pedersen DKG allows an efficient security reduction to its centralized counterpart via simulation while the original variant can only be proved secure by a direct reduction to the hardness of a DL problem to get around the simulation.

Very recently, Katz [79] showed the impossibility of having a fully secure (*i.e.*, statically unbiased) DKG with only one round. The impossibility is proved by making use of a one-round DKG to build up a coin tossing protocol, where its output (0 or 1) depends on whether the DKG public key falls in each of two balanced partition of the underlying cyclic group $\mathbb{G}$. It turns out a rushing adversary can always bias the output of the resulting coin tossing

5. A rushing adversary may choose to speak last, after observing all the other messages.

6. The set of possible choices for the simulator is of exponential size.

protocol, implying the impossibility result. Interestingly, this result holds regardless of the number of corrupt parties, any setup or idealized assumption (*e.g.*, random oracle), or even assuming a weaker variant without robustness. Given that a fully secure DKG might be hard to realize efficiently, Gurkan *et al.* [72] proposed a weaker notion for DKG called *security preserving* that relates the security of the DKG to the primitive built on top of it. More accurately, a DKG is security-preserving if it preserves the security of the threshold primitive such that breaking its security leads to breaking the security of the original (plain) primitive. Under this weaker notion, Pedersen DKG can be used securely in any setting with the cryptographic scheme being *rekeyable*[7] such as BLS signature or Elgamal encryption.

Gap 1. It would be interesting to explore the security of a threshold cryptosystem built upon a (biased) one-round DKG inspired by the previous results by Gennaro *et al.* [67] on the (biased) Pedersen DKG [93].

Almost all synchronous DKG protocols deploy a broadcast channel as their underlying consensus primitive, where all parties run parallel invocations over multiple rounds to agree on a set of correctly shared secrets. The extensive use of broadcast channels often is a major bottleneck for adopting DKGs in practice due to the imposed latency and the challenges in realization. An exception is the work of Shrestha *et al.* [99], proposing a synchronous DKG (almost) without broadcasts. The basic idea is to replace the broadcast with *multicast* and weaker consensus variants [55], [80], causing parties to have different views on the set of secrets correctly shared in the sharing phase. Given that the DKG requires (honest) parties to agree on a common set of secrets used for key derivation, the protocol deploys an MVBA invocation that takes long messages (*i.e.*, views) as inputs and outputs the common set. So, this is the only place where an instance of Byzantine consensus comes into play.

Although the traditional design for DKGs have all parties involved act as dealers to share their secrets, the involvement of only one honest dealer/contribution should be enough to ensure the secrecy (and robustness) of the protocol. This is the basic observation behind a few recent works such as [14], [57]. This insight allows choosing a small group of parties in $\mathcal{P}$ as the set of dealers contributing to the DKG output while making anytrust assumption [101], *i.e.*, only one of the parties needs to be honest without knowing which one. Given that the size of such group could be small (compared to the size of $\mathcal{P}$), this approach is vulnerable to adaptive corruption, *i.e.*, the adaptive adversary could corrupt the entire group upon detection. A common method to tackle this issue is via *self-selection* to let each party first figure its membership in the group before they speak up using random sortition [57], [69]. The seed for the sortition is a *common coin* where is publicly available and everybody agrees on. Note that the (security) requirement for such a common coin is less stringent than for a DKG, allowing it to be derived efficiently in various ways [81].

Insight 2. Interestingly, one can build a DKG protocol in the synchronous setting even for the corruption threshold of $t < n$, as only one honest contribution is enough to ensure secrecy. However, an honest majority assumption is typically made to ensure robustness for the resulting application such as threshold signing or decryption.

Gap 2. Following the above insight, one can explore if it is possible to have secure and robust applications for such (dishonest-majority) DKG protocols or if there is an inherent barrier as for coin-flipping [23].

Complexity. In the Pedersen DKG, the sharing creates an $O(n)$-sized message for each party to broadcast. The common paradigm of conducting DKG via n parallel instances of VSS leads to a $O(\lambda n^4)$ communication complexity per key, given the implementation of a broadcast channel in synchrony through Byzantine broadcast [49] in the plain model.[8] However, it is possible to achieve a cubic communication complexity using an optimal Byzantine broadcast [48]. Breaking the inherent linear round complexity in the synchronous DKGs is possible by adopting randomized consensus protocols [80] with expected constant rounds. It turns out, however, running n parallel invocations of such protocols require expected $O(\log n)$ rounds for termination [40]. One could also opt for the asynchronous counterpart known as reliable broadcast [24], with the optimal communication cost of $O(n|m| + n^2)$ and a constant round complexity [44], where m is the size of the message. Sampling random committees for dealing is a common approach to break the cubic communication cost of a DKG [58], due to lowering the number of dealers from $O(n)$ to $O(c)$ where c is a constant. This, however, demands a shared randomness (*i.e.*, common coin) in the first place to allow secure sampling. The idea of having each party talk to only a subset of small group of parties (*e.g.*, $\log n$) was previously explored by Canny and Sorkin [32] to design DKG with poly-logarithmic communication and computation per party. However, they also need to make some trust assumption on how to obtain some random seed in the first place. The introduction of the constant-sized polynomial commitment schemes reduced the size of the message to broadcast from $O(\lambda n)$ to $O(\lambda)$, leading to DKG protocols with $O(\lambda n^2)$ communication cost per party at the cost of requiring trusted setup [78]. Since VSS relies on polynomial commitments, any innovation on polynomial commitment schemes lead to an improvement on a VSS, and subsequently, on a DKG.

As mentioned, the typical complaint-based DKG such as Pedersen [93] requires at least three rounds of broadcast for sharing, complaining, and resolving the complaint. A common technique to reduce the number of broadcast rounds from three to two is to avoid posting complaints on broadcast channel and only have dealer broadcast a vote-

7. Rekeyablity refers to possibility of transferring an object (signature or ciphertext) to another one with a related secret key.

8. That is, without assuming trusted setup or threshold signatures.

certificate upon getting $t+1$ votes for resolving a complaint [19]. A relevant technique was recently introduced by Das et at. [45] to design a simplified VSS. The core idea is to let dealer send the polynomial commitment and the share *privately* to each party and upon receiving an acknowledgment, broadcast an $O(n)$-sized message containing the polynomial commitment, the acknowledgments, and shares of any unresponsive parties.

Insight 3. Assuming the presence of an external broadcast channel like a blockchain [95], the VSS proposed in [45] essentially has a star-like communication pattern, where only the dealer communicates with parties and there is no inter-communication, unlike the common all-to-all communication model.

Both aforementioned techniques, however, require interactions between dealer and parties to resolve any complaints. The notion of *publicly verifiable complaint* was recently proposed by Feng *et al.* [58] that essentially make this non-interactive. That is, the dealer broadcasts the set of all encrypted shares under the parties' public keys (together with polynomial commitment). If a decrypted share is invalid, the party can generate a NIZK proof of correct decryption which together with the share can serve as a verifiable complaint[9] with only a single complaint being enough to disqualify the dealer. As we discuss later, there is a subtle difference between this approach and deploying PVSS in the sharing phase which could circumvent any use of complaints.

4.2. Using PVSS

Design. The VSS protocol allows verification only for the participating parties and inherently relies on some form of complaint in one way or another. This inevitably affects its usability in scenarios where interaction is not desirable, motivating the use of publicly verifiable secret sharing (PVSS) [96], [100]. An immediate consequence of public verification is avoiding any complaints as anyone, not only the corresponding party, can check if the dealer has sent the valid shares. Note that a straightforward way to add public verifiability to VSS-based protocols is to add an additional round where parties wait to receive $n-t$ signatures from others to form a certificate [45]. However, PVSS schemes offer this property in the *sharing phase* by having dealer broadcast encrypted shares with associated validity proofs, obviating the need to make an honest majority assumption on the shareholders to validate the certificate. There exists a number of works that deploy PVSS to build publicly verifiable DKGs such as [10], [37], [57], [61], [70], [72]. *Non-interactive* DKGs, such as [61], [70], follow the same blueprint of Pedersen [93], except having parties deal their secrets via PVSS. That is, each dealer creates a local sharing transcript containing the encrypted shares under the public key of the respective parties and non-interactive zero-knowledge (NIZK) proof that certifies the validity of the encrypted shares. Finally, the dealer sends the transcript over a broadcast channel and the DKG keys are derived from the valid transcripts (that is agreed by everyone). Given that the protocol has only one-round, a rushing adversary can wait to see other parties' sharing transcripts before making its own contribution, causing the DKG output to be biasable [61].

Insight 4. In DKG protocols using VSS the adversary can potentially bias the output by manipulating the set of contributing parties, while in the ones based on PVSS it can do so by observing the contributions and speaking then. This is due to the use of private channels in former and public channels in the latter.

Insight 5. The recent result by Katz [79] shows it is feasible to design a fully secure one-round DKG with pseudorandom output. While using a PVSS with information-theoretic hiding [45] seems inevitable to protect against rushing adversary, the potential hurdle is in deriving the public keys.

A subtle challenge in designing DKG protocols using PVSS is how they handle the NIZK over encrypted shares [35], [61], [70]. This often is realized by encrypting the shares in the exponent, either in subgroups where solving the discrete logarithm problem is hard (*e.g.*, a cyclic group of prime order) or feasible (*e.g.*, a non-cyclic group of composite order). For the former, Groth [70] deployed a technique to sidestep the harness of DL computation for obtaining the original share $s_i \in \mathbb{Z}_p$ by splitting it into chunks and encrypting each one in the exponent of an ElGamal ciphertext to let the receiver obtain them using efficient brute-force. While suffering from efficiency issues, the benefit of this approach is allowing the DKG secret key sk to be a field element that is needed for many of the well-used threshold cryptosystems like (threshold) BLS signature [22]. A few recent recent works look into designing PVSS schemes in class groups of unknown order to have an efficient NIZK over encrypted shares [35], [77].

A well-used PVSS scheme in existing DKG protocols is the one due to Schoenmakers [96] and its recent variants [33], [34], [36]. An important benefit of Schoenmakers' PVSS and its variants is their efficient NIZK proof for validity of the sharing at the cost of weakening the functionality of PVSS that instead of sharing a field element $s \in \mathbb{Z}_p$, it shares a function of it which is essentially a (elliptic curve) group element, *e.g.*, $S = h^s \in \mathbb{G}$. Its transcript consists of Feldman commitments to sharing polynomial together with the encryption of the shares. An interesting property of this transcript is its *aggregatablity*, which allows aggregating two PVSS transcripts $\mathsf{pvss1}$ and $\mathsf{pvss2}$ for polynomials f_1 and f_2 into a single transcript for their sum f_1+f_2. Gurkan *et al.* [72] utilized this property to design a DKG with a gossiping communication pattern and minimal use of broadcast channel, where each party aggregates a bunch of PVSS sharing transcripts and forward it to the others. An issue, however, is how to handle the event of having some party's contribution get counted multiple times in final transcript due to the nature of gossiping. To tackle the issue,

9. That is, the complaint is effective right after a party raises it against the dealer.

each (aggregated) DKG transcripts contain a list of weights $(w_1, \ldots, w_n) \in \mathbb{Z}^n$ showing how many times each party has contributed to the current transcript. A party broadcasts the full transcript upon having $t+1$ non-zero weights. This is necessary to ensure at least one honest contributions is involved to guarantee the secret of the DKG output. In case multiple different full transcripts are broadcast, one of the candidates is considered by parties in a deterministic manner. It turns out the DKG protocol of [72] has a probability of failure with a trade-off between security and robustness such that it tolerates only up to $\log n$ Byzantine parties.[10]

Insight 6. To benefit from the aggregatablity of PVSS transcripts, there should be a way to verify the correctness of aggregation. Existing PVSS schemes enable this efficiently only in pairing-friendly groups [18], [72].

Rather than deploying a gossiping style pattern, Bacho *et al.* [10] utilizes the aggregatablity of PVSS transcript in a *recursive style* pattern allowing parties to end up with an aggregated transcript with at least one honest contribution. At a high level, they partition the set of parties into two disjoint subsets Q_1, Q_2 and run the protocol recursively within each, resulting in two aggregated transcripts of apvss_1 and apvss_2. Emulating a single-sender broadcast, the protocol allows the parties in each set to obtain the aggregated PVSS transcript of the other set, which results in the final aggregated transcript $\mathsf{apvss} := \mathsf{apvss}_1 + \mathsf{apvss}_2$ and concludes by having each part P_i compute their secret key share sk_i and public key pk from apvss.

Insight 7. The recursive design in [10] resembles the structure of a binary tree, where every pair of adjacent leaves at a given layer creates an aggregated PVSS transcript and sends it to their parent leaf to end up with a root which is the final transcript.

Given an honest majority for the parties, it turns out at least one half has an honest majority and thus can emulate an honest single-sender broadcast protocol, ensuring a successful termination [85]. While using a PVSS scheme for group elements each party can only share a secret of form $S = h^s$, it is still feasible to design a DKG with its secret key sk being field element. The authors in [37] observed that any $t+1$ of these secrets could result in a DKG public key of form $\mathsf{pk} = \prod_{j \in I} h^{s_j}$. The underlying difficulty is, however, in letting each party P_i obtain their partial secret key $\mathsf{sk}_i = \Sigma s_{i,j}$, where $s_{i,j}$ is the share of party P_i from the correctly shared secret of P_j. The idea is to encrypt each party's secret share $s_{i,j}$ using their corresponding group share $h^{s_{i,j}}$ that they get after decrypting PVSS encrypted share, *i.e.*, $s_{i,j} \oplus H(h^{s_{i,j}})$. Due to the form of encryption, this approach suffers from the need for a complaint round despite using PVSS.

10. Setting this threshold is with the goal of minimizing the number of total broadcasts while having a high probability of termination in a logarithmic number of rounds. We refer the reader to [72] for more details.

Gap 3. To date, no efficient aggregatable PVSS exists that shares a field element. Having such an aggregatable PVSS, one could obtain a nearly-quadratic field element DKG following the techniques in [10].

Insight 8. Deploying PVSS in the sharing phase of a DKG protocol does not inherently make it non-interactive. Although the agreement phase can be done non-interactively due to the public verifiability of the transcripts, some form of agreement—via Byzantine broadcast or public ledger—is still required to ensure a sufficient number of parties receive consistent shares.

Complexity. Using PVSS in the sharing phase of a DKG reduces the overall latency as less communication rounds is required but increases the computation/verification cost due to the non-trivial cryptography involved. Due to the linear sized PVSS transcript, majority of the publicly verifiable DKGs such as [37], [61], [70] have a cubic communication complexity of $O(\lambda n^3)$ using an optimal broadcast channel. Scrape [33] introduced a technique to reduce the verification cost of each party for a PVSS transcript from $O(n^2)$ [61] to $O(n)$ via an observation from coding theory regarding the equivalence of Shamir secret sharing and the Reed-Solomon code [94]. While PVSS transcript verification for n parties naively could lead to $O(n^2)$ computational cost, using an aggregatable PVSS with the gossiping approach of [72] allows reducing the cost of verifying the DKG transcript down to $O(n \log n)$ and its size to $O(n)$. Utilizing the aggregatablity of PVSS transcript further allows breaking the cubic communication barrier for the DKG protocol, leading to a subcubic communication $O(\lambda n^2 \log n)$ [10]. An interesting trick to get better *amortized* communication cost is to deploy packed secret sharing [20] to build a packed DKG protocol. It allows a dealer to share a vector of secrets of size l using a single polynomial of degree $t+l-1$ as in the standard scheme, but at the cost of lowering the resilience threshold to ensure robustness.

5. Asynchronous DKG

In this section, we discuss asynchronous DKG protocols for which we use the abbreviation "ADKG".

5.1. Dual-Threshold Security

An important notion that is sometimes considered in ADKG protocols and distinguishes it from its synchronous counterparts is that of (t, f)-*dual-threshold* security. This notion defines two thresholds of t and $f \in [t, n-t)$: one for robustness and the other for enhanced secrecy, respectively. As standard, robustness means that protocol successfully completes in the presence of an adversary corrupting up to t parties. On the other hand, enhanced secrecy means that the shared secret remains hidden from an adversary corrupting up to t parties, even if up to $f - t$ additional shares of

the secret are given.[11] In the synchronous setting, a single threshold t is used for both robustness and secrecy (*i.e.*, there is no enhanced secrecy). If $f = t$ for an asynchronous protocol, then the term "dual-threshold" is just dropped.

Why dual-threshold in asynchrony? The natural question that arises is: Why does dual-threshold property only matter for asynchronous protocols? The reason for this comes from the lowered robustness threshold. Concretely, known lower bounds show that asynchronous consensus protocols inherently require $t < n/3$ to terminate successfully.[12] This, however, implies that there are at least $n - t \geq 2t + 1$ honest parties. As such, it is possible to set the reconstruction/secrecy threshold up to $n - t - 1$ (*i.e.*, $f \in [t, n - t)$).

By having two different thresholds, ADKG protocols allow for a higher tolerance of corrupt parties (after termination of the protocol) without compromising the security of the generated keys. This notion of enhanced security is particularly important in many real-world applications, such as in the context of consensus and blockchain systems [13], [65], [102] where signatures from a Byzantine quorum of size $n - t \geq 2t + 1$ are needed.[13] In Table 2, we provide a summary of the differences in security for synchronous and asynchronous protocols. We note that some synchronous DKG protocols can tolerate up to $t < n$ corrupt parties. The reason is that synchronous consensus protocols are available that tolerate up to $t < n$ corrupt parties. Concretely, under a public-key infrastructure (PKI), broadcast protocols exist for $t < n$ corruption threshold.

Insight 9. ADKG protocols provide dual-threshold security. While they can tolerate only up to $t < n/3$ corrupt parties, secrecy of the shared secret remains even if up to $f < n - t$ shares are leaked. In contrast, synchronous protocols have only one threshold t.

5.2. General Paradigm

Conceptually, all ADKG protocols follow the same structure consisting of three phases: a sharing phase, an agreement phase, and a final key derivation phase. Although the first ADKG protocol was proposed by Kokoris-Kogias, Malkhi and Spiegelman [82], a protocol following the above structure could be designed with building blocks that were already existing before. In the following, we briefly explain the common structure in more detail.

In the first phase, each party P_i samples a uniformly random element $s_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and secret shares it with the other parties via an AVSS scheme, which can have any reconstruction threshold f. This gives the following guarantee: If an honest party successfully completes some AVSS (of another party), then all honest parties will eventually also complete it and receive their shares. However, since the network is asynchronous, sharings of corrupt dealers may never terminate at any honest party, and sharings of honest dealers may terminate for different parties at different times. Therefore, parties aim to reach agreement on a set of at least $t + 1$ parties whose AVSS sharing has either successfully completed or will eventually terminate.

In the second phase, parties run some consensus protocol such as ACS or MVBA. A folklore ACS was introduced by Ben-Or *et al.* [16] and works by running n parallel instances of binary Byzantine agreement (BA). The j-th BA instance decides whether or not P_j's shared secret should be included for the secret key, which will be a sum of the agreed-upon shared secrets from individual parties. Concretely, parties input 1 to the j-th BA instance when they see that party P_j has successfully completed its AVSS sharing. Conversely, once a party sees at least $t + 1$ BA instances terminate with output 1, it inputs 0 to all the remaining BA instances for which it has not yet input anything. The latter ensures termination of all BA instances. Let $T \subseteq [n]$ be the set of indices for which the respective BA instances terminated with 1.

In the final phase, parties derive their share sk_i of the secret key $\mathsf{sk} := \sum_{j \in T} s_j$ by summing up its received shares $s_{j,i} := f_j(i)$ from the AVSS sharings $j \in T$. Further, parties also reconstruct the public key pk and public key shares $\{\mathsf{pk}_i\}_{i \in [n]}$. Depending on the design of the AVSS, the latter can either be done locally by each party or requires some interaction between parties. We do not specify this, as it is not relevant here. We further emphasize that the final secret key sk is shared among all parties and has a reconstruction threshold f given by the AVSS.

Insight 10. All existing asynchronous DKG protocols internally use some form of (weak) shared randomness. A natural question to ask is if there exists an efficient (*i.e.*, non-exponential) asynchronous DKG protocol that terminates without any form of shared randomness.

Asynchronous consensus. Finally, we briefly explain how typical Byzantine agreement protocols achieve consensus without assuming any threshold key setup. Typically, asynchronous BA protocols follow the framework established by Feldman-Micali [55], which reduces BA to the design of a *weak common coin*[14] protocol. A weak common coin protocol allows parties to agree on a common randomness with some probability $\delta > 0$, where agreement is oblivious, *i.e.*, parties are not aware whether or not agreement has been achieved. This probability δ inversely determines the BA protocol's running time and is often some constant. To elaborate more, many of these BA protocols proceed through a series of iterations. In each iteration, parties begin by exchanging suggestions based on the messages of previous iterations. This is followed by an exchange of proposals, after which a weak common coin mechanism retroactively selects one of the proposals. Parties then check whether the

11. Sometimes f is also called the *reconstruction threshold* because the polynomial interpolating the secret key has degree f.

12. The authors in [9] show that the same is true for ADKG protocols.

13. Observe that in such settings an enhanced security is crucial as any attacks on consensus are more likely to leave public traces than on secrecy.

14. Sometimes also called *oblivious* common coin in the literature. Further, the related notion of *weak/oblivious leader election* is essentially a weak/oblivious common coin with range $[n]$.

TABLE 2: Summary of distinctive DKG protocol features.

Property	Synchronous DKG	Asynchronous DKG
Threshold property	single threshold t	dual thresholds t and f
Robustness threshold	up to t corrupt parties	up to $t < n/3$ corrupt parties
Secrecy threshold	up to t leaked shares	up to $f < n - t$ leaked shares
Network assumption	bounded message delays	unbounded message delays

selected proposal can safely be output by the protocol. If that is the case, they do so; otherwise, they proceed to the next iteration. Crucially, when the weak common coin elects an honest leader, then all parties can immediately output its proposal and terminate.

5.3. Design and Complexity

In this section, we present three main approaches that form the backbone of existing asynchronous DKG protocols. We emphasize that the difficulty in the design of efficient asynchronous protocols necessitates a rather different categorization compared to the synchronous ones. At a high level, the three main approaches differ in how to achieve consensus on a set of at least $t + 1$ parties whose shared secrets are used for the final secret key. As such, we will mostly focus on their agreement phases.

Approach I. The first asynchronous DKG protocol was proposed by Kokoris-Kogias, Malkhi and Spiegelman [82]. At a high level, the protocol follows the idea of running n parallel instances of AVSS, with each party P_i sharing a secret $s_i \in \mathbb{Z}_p$, and then establishing consensus on which of these shared secrets to use for the key. The protocol has the dual-threshold property with $f \in [t, n - t)$, which is achieved using a novel dual-threshold AVSS with secrecy threshold f presented in their paper.[15]

To establish consensus, the authors do not directly employ any (inefficient) black-box binary Byzantine agreement protocol for each AVSS sharing. Instead, parties use the shared AVSS secrets themselves as source for an "eventually perfect common coin (EPCC)" that is used for an efficient Byzantine agreement protocol (which assumes a weak common coin). Once this BA protocol is established, the protocol proceeds similar to our strawman solution by running n parallel instances of the BA to decide which AVSS instances to include for the final secret key, followed by a simple key derivation phase. Here, an EPCC is stronger than a weak common coin, and can roughly be defined as a (perfect) common coin that fails a finite number of times. In more detail, after the sharing phase, each party P_i waits for $n - t$ AVSS sharings to complete locally and stores the corresponding indices of the parties in a set T_i. That is, $|T_i| = n - t$ and $j \in T_i$ if and only if party P_j's sharing completed at P_i. Since the network is asynchronous, each party might have a different set T_i of $n-t$ locally completed sharings. Nonetheless, each party can treat its set T_i as a *local DKG setup* from which each can derive a secret share (by summing up its received shares $s_{j,i} = f_j(i)$ from the AVSS sharings $j \in T_i$). Crucially, if the sets T_i from honest parties do not match, parties will not be able to derive the same common coin.

At this point, the authors make the following observation, which they utilize to build an EPCC: By completeness of AVSS, $j \in T_i$ for an honest P_i implies that the j-th AVSS sharing will eventually terminate at all honest parties. Thus, if parties also keep track of each new one AVSS sharing that completes locally, each obtains an increasing sequence of sets $T_i =: T_i(1) \subset T_i(2) \subset \ldots \subset T_i(k)$, where $k \leq t$ (since $|T_i| = n-t$) and $|T_i(j+1)| = |T_i(j)|+1$ for $j \in [k-1]$, with the crucial property that honest parties' sets converge (*i.e.*, they are identical at some point). In particular, all honest parties will have the same set $T := T.(k)$ at some point and will get the same local DKG, implying the same common coin. As such, each party can use its local DKG $T_i(l)$ as source of randomness for the l-th iteration of the underlying BA protocol. This implies that the BA will succeed and terminate after at most $O(t)$ iterations.

> **Insight 11.** The main difficulty in the design of efficient asynchronous DKG protocols revolves mainly around the consensus phase and the generation of a common coin for it. Further, a weak common coin already allows to bootstrap full consensus efficiently.

Approach II. We observe that the overhead in efficiency comes from two different sources. First, from their AVSS with its cubic communication cost, leading to quartic overall communication. And second, from their EPCC that succeeds only after $O(n)$ iterations, leading to $O(n)$ round complexity. In order to improve upon that, Das *et al.* [46] design a new (high-threshold) AVSS scheme with $O(\lambda n^2)$ communication cost per sharing and a more sophisticated way to obtain common randomness for the Byzantine agreement protocol. Intuitively, their AVSS is the PVSS scheme from Fouque-Stern [61] combined with a reliable broadcast.[16] In particular, the resulting AVSS can directly have the high-threshold property. Their ADKG protocol follows a similar structure as before: First, each party P_i shares a secret $s_i \in \mathbb{Z}_p$ via their (high-threshold) AVSS, and then form

15. In fact, they design the *first* high-threshold AVSS with optimal resilience $t < n/3$ and answer an open question posed by Cachin *et al.* [27]

16. Note that a PVSS transcript that is reliable broadcast trivially gives an AVSS scheme, as termination of RBC gives completeness of AVSS.

the set of indices $T_i \subset [n]$ (which they call *key set*) once it sees $t+1$ completed AVSS sharings.

The authors insight is to directly use this key set as a local DKG setup for the i-th instance of BA itself. To this end, the authors add a key set proposal phase after the sharing phase, in which each party P_i reliably broadcasts its index set T_i before entering the consensus phase. Then, parties run n parallel instances of binary BA to agree on which key sets to use for the final secret, where a party P_i inputs 1 into the j-th instance only if the reliable broadcast of T_j (initiated by party P_j) terminates at P_i and the set T_j is valid (*i.e.*, all AVSS sharings $k \in T_j$ completed locally at P_i). Noteworthy, parties wait until one BA instance terminates with 1 and then input 0 to the remaining instances, instead of waiting for $t+1$ instances terminate with 1. This is because a key set already includes $t+1$ AVSS sharings and thus it is enough to agree on one such set.

Conceptually, the difference to Kokoris-Kogias *et al.*'s lies in using a different source of randomness for each BA instance (from the respective party itself), whereas Kokoris-Kogias *et al.* use the same source of randomness for all BA instances (from all parties). One subtle issue exists in the approach for Das *et al.* we described: Namely, if a corrupt party P_j does not propose any key set T_j, then there is no source of randomness for the respective BA instance. Interestingly, the authors circumvent this problem by making the crucial observation that the FLP impossibility only applies when honest parties have different inputs. In particular, when each honest party inputs 0, then no randomness is needed to terminate the BA. Thus, if a corrupt party P_j does not propose any key set, all honest parties will eventually input 0 and terminate that BA instance without any randomness.

A follow-up work by Zhang *et al.* [104] uses the same conceptual design, but gives improved concrete efficiency. Essentially, they achieve this by improving the underlying AVSS and replacing the binary BA protocol with a reproposable asynchronous binary Byzantine agreement (RABA) protocol, which allows parties to change their input once during the execution of the RABA. Another follow-up work by Das *et al.* [43] also employs the same conceptual design, but obtains a high-threshold ADKG with much better concrete efficiency. Instead of directly using a high-threshold AVSS (obtained from a PVSS) as in [43], they use a much more efficient low-threshold AVSS and combine this with a linear randomness extractor (concretely, a superinvertible matrix [73]) to obtain f random secret sharings that can be interpreted as the coefficients of a degree-f polynomial.

Approach III. A new approach was given by Abrahm *et al.* [6] in order to obtain $O(1)$ expected round complexity. Their ADKG also follows the same structure: Each party P_i shares a secret $h_i \in \mathbb{G}$ via an AVSS, and then form the set of indices $T_i \subset [n]$ (which they call *key set*) once it sees $t+1$ completed AVSS sharings. This set is then proposed to all parties via reliable broadcast. Similar to Das *et al.*'s approach, the goal is now to agree on a single key set proposal. Here, we emphasize that in contrast to the previous ADKG protocols we have described so far, their AVSS is instantiated with a PVSS that shares a group element $h_i \in \mathbb{G}$ rather than a field element.

Different from the previous two approaches, the authors design a novel MVBA protocol that is based upon a previous constant-round MVBA which assumes threshold signatures [7]. The threshold signatures are used for a (perfect) leader election. The main idea is to replace the leader election by a weak leader election, which is obtained using similar ideas from Canetti-Rabin [31]. At a high level, each party samples a secret for every other party and sends its PVSS transcript to it. Now, each party P_i waits for $n-t$ such contributions from other parties and aggregates them into one transcript R_i (thought as a local DKG). The encoded secret will later be used to assign a rank to this party. As such, each party is assigned a rank that it does not know until reconstruction. In the next step, parties run a so called *Gather protocol* on R_i. In a Gather protocol, each party P_i has an input, and each party outputs a set S_i of received values (with corresponding party indices) with the crucial property that there is a core set S of size at least $n-t$ such that all honest parties include S in their output set but might differ otherwise. After Gather, parties start reconstructing all secrets encoded in the transcripts $\{R_j\}_{j\in[n]}$, and each party P_i elects the party with maximal rank among its set S_i for the leader election in the MVBA protocol. Intuitively, a party with maximal rank among all parties is in the core set S with probability $(n-t)/n \geq 2/3$. Thus, an honest party having maximal rank and being in the core set happens with probability $1/3$. Ultimately, this gives a weak leader election with probability $1/3$, and thus an MVBA with $O(1)$ expected rounds and $O(\lambda n^3)$ communication. Having this consensus protocol, parties can agree on a key set proposal of a single party and enter the key derivation phase.

Insight 12. The Gather framework by Abraham *et al.* [6] allows to achieve asynchronous consensus in constant expected rounds without assuming threshold signatures or (strong) trusted setup in the first place.

Gao *et al.* [63] manage to get rid of the $\log n$ factor in the communication cost by making some minor changes in the sharing and agreement phases. However, conceptually, they still follow the same framework of Abraham *et al.* Another follow-up work by Abraham *et al.* [5] uses the same approach, but achieves to replace the (aggregatable) PVSS for weak leader election by an AVSS that shares a field element. To this end, they design a novel packed AVSS with amortized linear communication cost per shared secret. They do so because in their leader election, each party needs to share n secrets, one for each other party. Thus, using their packed AVSS, this multi-secret sharing can be done via a single AVSS sharing. Further, a subsequent work has used the Gather framework to design a hash-based ACS protocol [41] (without the use of public-key cryptography). They achieve this by employing a hash-based weak AVSS, which can guarantee only that $t+1$ honest parties obtained their shares and not all $n-t > t+1$, and adding a *binding cover* property to the gather protocol to make it also work

TABLE 3: Comparison of existing asynchronous DKG protocols.

Protocol	Year	Corr	Dual	Communication	Rounds	Field	Sharing	Secrecy	Miscellaneous
Kokoris *et al.* [82]	*2019*	✓	✓	$O(\lambda n^4)$	$O(n)$	✓	AVSS	✗	eventually perfect common coin
Abraham *et al.* [6]	*2021*	✓	✓	$O(\lambda n^3 \log n)$	$O(1)$	✗	APVSS	✗	Gather for constant-round MVBA
Gao *et al.* [63]	*2021*	✓	✓	$O(\lambda n^3)$	$O(1)$	✗	APVSS	✗	subtle modifications to [6]
Das *et al.* [46]	*2021*	✓	✗	$O(\lambda n^3)$	$O(\log n)$	✓	AVSS	✓	no pairings, $\log n$-round MVBA
Das *et al.* [46]	*2021*	✓	✓	$O(\lambda n^3)$	$O(\log n)$	✓	PVSS	✓	heavy NIZKs, DCR
Das *et al.* [43]	*2022*	✓	✓	$O(\lambda n^3)$	$O(\log n)$	✓	AVSS	✓	superinvertible matrices
Zhang *et al.* [104]	*2022*	✓	✓	$O(\lambda n^3)$	$O(1)$	✓	PVSS	✓	subtle modifications to [46]
Abraham *et al.* [5]	*2022*	✓	✓	$O(\lambda n^3)$	$O(1)$	✓	AVSS	✓	PoT setup, optimal packed AVSS
Feng *et al.* [56], [59]	*2024*	✓	✓	$O(\lambda \kappa n^2)$	$O(1)$	✓	(A)PVSS	✗	silent setup [64], heavy NIZKs
Abraham *et al.* [3]†	*2025*	✓	✓	$O(\lambda \kappa n^{2+1/k})$	$O(k)$	✗	APVSS	✗	recursive design, heavy NIZKs
Groth-Shoup [71]	*2023*	✓	✗	$O(\lambda n^3)$	MVBA_n	✓	AVSS	✗	hash-based AVSS, batching
Bacho *et al.* [13]	*2024*	✓	✓	$O(\lambda n^3 \log n)$	MVBA_n	✓	AVSS	✗	no pairings, batching
Alhaddad *et al.* [8]	*2024*	✓	✓	$O(\lambda n^3 \log n)$	MVBA_n	✓	AVSS	✓	no pairings, batching

Corr indicates whether or not optimal Byzantine corruption threshold $t < n/3$ is achieved. Communication and round complexity are given in expected numbers, as all these protocols are randomized. Here, the protocols [8], [13], [71] assume an MVBA protocol and where MVBA_n denotes its round complexity among n parties. Further, these protocols generate a batch of $O(n)$ keys with the listed communication cost; thus, they have $\tilde{O}(\lambda n^2)$ amortized communication cost per key. † The generic transformation from common coins to ADKG given in [56] can also be applied here to share a field element secret key (via an optimized interactive PVSS in the initial sharing phase). Further, $k \in [\log n]$ is a selectable parameter, specifying the depth of the tree.

for a weak AVSS.

New works on subcubic ADKG. Two recent concurrent works [3], [59] present ADKG protocols with *subcubic* communication cost, using fundamentally different approaches. In [59], Feng and Tang construct a common coin protocol and use the generic transformation from [56] to obtain an ADKG. To achieve the subcubic common coin, the authors introduce a new primitive called *Asynchronous Subset Alignment (Align)*, which has weaker guarantees than the Gather primitive by Abraham *et al.* [6], but notably is still good enough for their purposes. Recall that the purpose of Gather is to allow all honest parties to agree with constant probability on a leader with the (globally) highest rank. In previous approaches following Canetti-Rabin [31], the rank is derived from n parallel instances of (variants of) AVSS, which leads to $\Omega(n^3)$ communication cost. Deviating from this conventional approach, Feng *et al.* utilize a silent-setup threshold encryption scheme [64] with certain homomorphic properties to create these ranks for all parties with subcubic communication. This silent setup, however, requires each party to register an $O(\lambda n)$ long bit-string on a public bulletin board for the PKI.

In [3], Abraham *et al.* directly construct an ADKG protocol using a recursive approach inspired by the synchronous DKGs in [10], [57]. At a high level, their protocol proceeds as follows. The set S of n parties is partitioned into $\ell = n^{1/k}$ subsets $\{S_i\}_{i\in[\ell]}$ of roughly the same size. In the first phase, parties in S collect aggregated PVSS transcripts that contain (with high probability) an unpredictable secret. They achieve this by letting each party send a PVSS to a random set of $\Theta(\kappa)$ other parties via a VRF. The (local) aggregated PVSS transcripts serve as input for each party in subsequent steps. In the second phase then, the protocol for S runs an MVBA protocol on each subset S_i, whose output is an aggregated PVSS transcript for S. The randomness for this agreement is established by recursively running the protocol on the subsets. Further, their MVBA protocol is inspired by the No-Waitin-Hotstuf MVBA [6] and utilizes new primitives such as a communication-efficient *Weighted Gather* protocol. Although their protocol shares a group element secret key, it can be combined with the interactive PVSS scheme (in an initial sharing phase) given in [56] to share a field element secret key, with no additional asymptotic cost.

Gap 4. While the two recent works of [3], [59] show how to construct ADKG protocols with nearly quadratic communication cost, they both rely on secure erasures for adaptive security, which is undesirable in practice.

Complexity. We provide the communication and round complexities in expected terms, as all the protocols here are randomized. The ADKG protocol by Kokoris-Kogias *et al.* [82] has $O(\lambda n^4)$ bits communication cost and $O(n)$ rounds. Further, their novel (dual-threshold) AVSS has a communication cost of $O(\lambda n^3)$ bits. The ADKG protocol by Das *et al.* [46] has $O(\lambda n^3)$ bits communication cost and $O(\log n)$ rounds. They achieve this partially due to their (dual-threshold) AVSS scheme with $O(\lambda n^2)$ communication cost per sharing. The protocol by Abraham *et al.* [6] has $O(\lambda n^3 \log n)$ communication cost and $O(1)$ rounds, but outputs a group element as secret key. Gao *et al.* [63] achieve to get rid of the $\log n$ factor in the communication cost, but it still outputs a group element as secret key. A further improvement was made by Abraham *et al.* [5], where they design an ADKG protocol with $O(\lambda n^3)$ communication cost and $O(1)$ rounds that outputs a field element as secret key. Finally, the two recent works [3], [59] obtain near-quadratic ADKG protocols. Concretely, the one by Feng *et al.* [59] has a communication cost of $O(\lambda \kappa n^2)$ and $O(1)$ rounds (relying on a silent setup phase). The one by Abraham *et al.* [3] has a communication cost of $O(\lambda \kappa n^{2+1/k})$ and $O(k)$ rounds, where $k \in [\log n]$ is a parameter that can be chosen, specifying the depth of the tree in their recursive design.

We further remark that there are some works on improving the efficiency of ACS or MVBA protocols [50], [51], [86], but these already assume a strong common coin. In this context, it is important to note that the recent hash-based ACS by Das *et al.* [41] also directly implies a hash-based common coin protocol with $O(\lambda n^3)$ communication cost and $O(1)$ rounds. This remains the best hash-based common coin protocol to date, which is in contrast to the recent near-quadratic ADKG protocols [3], [59].

Gap 5. Is it possible to design a hash-based common coin protocol with subcubic communication cost (without private setup) or is there an inherent barrier for this?

Batched ADKG. We recall that the main difficulty in designing efficient ADKG protocols comes from the consensus phase, and how to generate the internal randomness needed for it. But assuming such a consensus protocol (*e.g.*, ACS or MVBA), an ADKG can be build generically as discussed in Section 5.2. As such, recent works [8], [13], [17], [71] have focused on designing batched ADKG protocols that generate many keys at once without increasing asymptotic communication cost, given an asynchronous consensus protocol. All these works do this with the motivation of high-throughput robust (asynchronous) threshold Schnorr signatures, where the generated keys from the batched ADKG serve as nonces for the threshold Schnorr signature. This idea of generating the nonce via a DKG protocol for robust threshold Schnorr signatures goes back to Gennaro *et al.* [66], but in the synchronous setting and without batching.

Insight 13. Using linear randomness extractors, it is possible to obtain more shared keys with much better efficiency than just running many instances of ADKG in parallel. In particular, amortized linear communication cost per shared key is possible.

At a high level, these protocols achieve this by using linear randomness extractors. Concretely, after the usual sharing phase, parties agree on a set of $n - t > t + 1$ dealers whose AVSS sharings completed successfully using a consensus protocol. Then parties locally apply a linear randomness extractor to obtain $k := n - 2t$ new shares. The special property of the randomness extractor is: if at least k input secrets (or shares thereof) are independent and uniformly random, then the k output secrets (or shares) are also guaranteed to be independent and uniformly random. Since there are at most t corrupt parties, at least $(n-t)-t = k$ of the agreed-upon dealers are honest and provided uniformly random and independent secrets. And linearity of the randomness extractor is used to translate shares of the input secrets to shares of the new secrets. In this way, these works manage to achieve batched DKG with amortized $O(\lambda n)$ communication cost for low-threshold keys and amortized $O(\lambda n^2)$ communication cost for high-threshold keys (still with total $O(\lambda n^3)$ communication cost per instance).

6. Further Discussion

In this section, we discuss several concepts and properties relevant to DKG protocols that are vital to highlight in addition to their underlying design and complexity.

Other DKGs. There are some few additional DKG protocols that do not fit into our categorization, thus we elaborate on them here. In [9], the authors design a network-agnostic DKG protocol that provides security guarantees with different threshold for the synchronous and asynchronous network simultaneously. Concretely, their DKG works in a synchronous network with corruption threshold t_s and in an asynchronous network with corruption threshold t_a where $2t_s + t_a < n$. Further, the authors show that this bound is optimal for network-agnostic DKG protocols. In [76], Kate *et al.* extended Pedersen's DKG to the partially synchronous network, which does not suffer from the FLP impossibility [60] (that introduces the main difficulty in the design of asynchronous DKG protocols). The protocol has $O(\lambda n^4)$ total communication cost, Byzantine corruption threshold $t < n/3$, and reconstruction threshold t.

The deployment of DKG protocols in practical applications demand new design considerations. Flamingo [88] uses a DKG to realize a secure aggregation in federated learning setting where there is a server that coordinates the communication among parties (who is supposed to run DKG). Due to the *star topology* for the communication, the system cannot use normal DKGs in synchrony as the server might delay the messages between parties. To get around this challenge the authors modify the Gennaro *et al.* DKG [67] to handle a *non-responsive* server that results in reducing the fault tolerance of the protocol from $1/2$ to $1/3$.

Gap 6. One direction to explore is designing efficient leader-based DKG in the partially synchronous network setting for use in applications with a central coordinator such as federated learning [88].

Adaptive security. The literature on DKG usually considers two distinguishing types of adversaries: (i) a *static adversary* who chooses the set of corrupt parties before the protocol execution, and (ii) an *adaptive adversary* who dynamically corrupts parties over time by observing the flow of the protocol execution. Assuming an adaptive adversary is much closer to the reality, where malicious entities may observe a system before targeting specific parties, *e.g.*, in the blockchain setting.

Most of the DKG protocols in the literature are proven only statically secure. Typically, the challenge in dealing with adaptive corruptions is simulating internal states consistent with messages that the adversary has already seen, *e.g.*, commitments and encryptions. Canetti *et al.* [30] proposed the first DKG protocol with adaptive security that follows the same blueprint design of Gennaro *et al.* [66]. They achieve adaptive security by introducing the *single-inconsistent player (SIP)* technique and using non-committing encryption with secure erasures. In the SIP technique, there is only a single party whose internal state cannot

be consistently revealed to the adversary. Since this single-inconsistent party is chosen at random at the beginning of the protocol simulation, it is corrupted by the adversary with probability at most $t/n \leq 1/2$ (assuming $t < n/2$). Jarecki and Lysyanskaya [75] then extended the SIP technique to remove the need for secure erasures. But a significant downside of their approach is that it incurs a large amount of interaction between parties for every secret signing key operation (*i.e.*, for subsequent applications of the DKG such as threshold signing or threshold decryption). Recently, Bacho and Loss [11] introduced a new security notion of *oracle-aided algebraic simulatability* for DKG protocols and show that this notion is sufficient to get adaptively secure applications of DKG such as threshold BLS signing. Further works [5], [13], [58] adopted this security notion for their DKG protocols, also for threshold Schnorr signatures.

Insight 14. Full secrecy/simulatability of a DKG protocol is not required to show adaptive security in applications such as threshold signing or threshold decryption. Often, weaker notions of (adaptive) security for DKG are sufficient for (adaptively) secure applications.

Roughly, this weakens the standard simulatability notion for DKG by giving the simulator additional access to a discrete logarithm oracle. Intuitively, this allows the simulator by calling the discrete logarithm oracle to obtain the secret key shares sk_i from simulated public key shares $\mathsf{pk}_i = g^{\mathsf{sk}_i}$ upon adaptive corruptions. In the actual reduction, the discrete logarithm oracle is then instantiated with the one-more discrete logarithm assumption. While there are many adaptively secure AVSS schemes that share field elements, there is currently no known adaptively secure PVSS scheme that shares field elements. As such, this sets a current barrier in obtaining adaptively secure DKG protocols that are built upon PVSS schemes for field elements.

Gap 7. Currently, there is no known adaptively secure PVSS scheme for field elements. Is it possible to design such a PVSS scheme?

Noteworthy, a recent work [12] shows adaptive security of several existing (aggregatable) PVSS schemes for group elements under a relaxed notion of secrecy called *(aggregated) unpredictability*. Further, the authors show that this is sufficient for secure applications such as DRB.

Cryptographic assumptions and setup. The common underlying cryptographic hardness assumption for DKG protocols is the decisional Diffie-Hellman (DDH) assumption. However, recent works have also used the one-more discrete logarithm (OMDL) assumption to prove a weaker notion of adaptive security in the algebraic group model (AGM) [62], as discussed earlier. Further, some of the works also assume the strong Diffie-Hellman (SDH) assumption due to the use of the powers-of-tau setup to improve efficiency (*e.g.*, by employing the constant-sized KZG polynomial commitment scheme [78] or constant-sized bilinear accumulators [90]).

Many DKG protocols do not require any setup beyond a common random string (*i.e.*, public parameters). However, a few protocols assume a powers-of-tau setup to improve efficiency [78]. Further, some DKG protocols also explicitly assume a public-key infrastructure (PKI), either due to the use of digital signatures or PVSS schemes that employ public-key encryption. That said, all DKG protocols assume authenticated channels, which are typically implemented via PKI. Moreover, most synchronous DKG protocols assume a Byzantine broadcast channel, which also necessitates a PKI whenever $t \geq n/3$ by known impossibility results [91].

Using DKG for distributed randomness beacons. Distributed randomness beacon (DRB) protocols play a crucial role in numerous cryptographic and distributed system applications [38]. A well-known approach to generate efficient (one-round) distributed randomness goes back to the seminal work of Cachin *et al.* [29]. The idea is to use a threshold signature scheme with unique signatures (per message m and public key pk) and a non-interactive signing procedure, such as threshold BLS [1], [21]. Thus, after an initial DKG setup, one obtains a simple and efficient randomness beacon protocol. An intriguing property of this type of DRB is their compatibility with both synchronous and asynchronous networks (after the DKG setup). However, a limiting factor of this approach is that it is not well-suited for dynamic settings, where parties frequently join and leave. For instance, if a single party leaves the system, the DKG needs to be executed again. Further, recent works [10], [42], [72] show how to construct an efficient DRB using a DKG setup that outputs group element secret keys.

Weighted setting. In proof-of-stake (PoS) blockchains, the validators have different weights which are proportional to the amount of stake it holds. Usually, these systems assume that malicious validators control at most $1/3$-fraction of the total stake (weight). Because of that, some works [42], [58] have designed DKG protocols for the weighted setting. Feng *et al.* [58] give a generic mechanism that allows to efficiently apply conventional distributed protocols in the weighted setting, and thus also applies to DKG protocols. On the other hand, Das *et al.* [42] design a concrete DKG protocol for the weighted setting. The simplistic approach of treating a validator as a DKG party undermines security, as it allows an adversary to corrupt a large number of low-weight parties and thus dominate the system.

7. Conclusion

In this work, we presented a systematization of knowledge on the DKG literature as an established yet thriving field of research. We believe our study represents the first comprehensive investigation of DKG protocols to date. Our work offers a structured overview of the literature, classifying DKG protocols according the way they handle secret sharing and agreement as their two main building blocks. Through a critical examination of the underlying techniques and cryptographic tools, we suggested key insights and identified research gaps that could guide future advancements, either by improving existing solutions or by addressing unexplored challenges.

References

[1] Team drand, drand project website. https://drand.love, 2020.

[2] Masayuki Abe and Serge Fehr. Adaptively secure feldman vss and applications to universally-composable threshold cryptography. In *Annual International Cryptology Conference*, pages 317–334. Springer, 2004.

[3] Ittai Abraham, Renas Bacho, Julian Loss, and Gilad Stern. Nearly quadratic asynchronous distributed key generation. Cryptology ePrint Archive, Paper 2025/006, 2025.

[4] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected o (1) rounds, expected communication, and optimal resilience. In *International Conference on Financial Cryptography and Data Security*, pages 320–334. Springer, 2019.

[5] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In *Annual International Cryptology Conference*, pages 39–70. Springer, 2023.

[6] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 363–373, 2021.

[7] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.

[8] Nicolas Alhaddad, Mayank Varia, and Ziling Yang. Haven++: Batched and packed dual-threshold asynchronous complete secret sharing with applications. *IACR Communications in Cryptology*, 1(4), 2025.

[9] Renas Bacho, Daniel Collins, Chen-Da Liu-Zhang, and Julian Loss. Network-agnostic security comes (almost) for free in dkg and mpc. In *Annual International Cryptology Conference*, pages 71–106. Springer, 2023.

[10] Renas Bacho, Christoph Lenzen, Julian Loss, Simon Ochsenreither, and Dimitrios Papachristoudis. Grandline: adaptively secure dkg and randomness beacon with (log-) quadratic communication complexity. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 941–955, 2024.

[11] Renas Bacho and Julian Loss. On the adaptive security of the threshold bls signature scheme. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 193–207, 2022.

[12] Renas Bacho and Julian Loss. Adaptively secure (aggregatable) pvss and application to distributed randomness beacons. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1791–1804, 2023.

[13] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. Harts: High-threshold, adaptively secure, and robust threshold schnorr signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 104–140. Springer, 2024.

[14] Akhil Bandarupalli, Adithya Bhat, Saurabh Bagchi, Aniket Kate, and Michael K Reiter. Hashrand: Efficient asynchronous random beacon without threshold cryptographic setup. *IACR Cryptol. ePrint Arch.*, 2023:1755, 2023.

[15] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 52–61, New York, NY, USA, 1993. Association for Computing Machinery.

[16] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '94, page 183–192, New York, NY, USA, 1994. Association for Computing Machinery.

[17] Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. Sprint: High-throughput robust distributed schnorr signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 62–91. Springer, 2024.

[18] Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. Optrand: Optimistically responsive reconfigurable distributed randomness. In *Network and Distributed System Security Symposium (NDSS)*, 2023.

[19] Adithya Bhat, Nibesh Shrestha, Zhongtang Luo, Aniket Kate, and Kartik Nayak. Randpiper–reconfiguration-friendly random beacons with quadratic communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3502–3524, 2021.

[20] George Robert Blakley and Catherine Meadows. Security of ramp schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84 4*, pages 242–268. Springer, 1985.

[21] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 31–46, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[22] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[23] Joseph Bonneau, Benedikt Bünz, Miranda Christ, and Yuval Efron. Good things come to those who wait: Dishonest-majority coin-flipping requires delay functions. *Cryptology ePrint Archive*, 2024.

[24] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[25] Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. In *Annual International Cryptology Conference*, pages 613–645. Springer, 2023.

[26] Jakob Burkhardt, Ivan Damgård, Tore Kasper Frederiksen, Satrajit Ghosh, and Claudio Orlandi. Improved distributed rsa key generation using the miller-rabin test. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2501–2515, 2023.

[27] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. *Proceedings of the ACM Conference on Computer and Communications Security*, 05 2003.

[28] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 524–541, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[29] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 123–132, 2000.

[30] Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 98–116. Springer, 1999.

[31] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 42–51, New York, NY, USA, 1993. Association for Computing Machinery.

[32] John Canny and Stephen Sorkin. Practical large-scale distributed key generation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 138–152. Springer, 2004.

[33] Ignacio Cascudo and Bernardo David. Scrape: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017.

[34] Ignacio Cascudo and Bernardo David. Albatross: publicly attestable batched randomness based on secret sharing. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*, pages 311–341. Springer, 2020.

[35] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to dkg and yoso. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 216–248. Springer, 2024.

[36] Ignacio Cascudo, Bernardo David, Lydia Garms, and Anders Konring. Yolo yoso: fast and simple encryption and secret sharing in the yoso model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 651–680. Springer, 2022.

[37] Ignacio Cascudo, Bernardo David, Omer Shlomovits, and Denis Varlakov. Mt. random: Multi-tiered randomness beacons. In *International Conference on Applied Cryptography and Network Security*, pages 645–674. Springer, 2023.

[38] Kevin Choi, Aathira Manoj, and Joseph Bonneau. Sok: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 75–92. IEEE, 2023.

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

[40] Ran Cohen, Sandro Coretti, Juan Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III 36*, pages 240–269. Springer, 2016.

[41] Sourav Das, Sisi Duan, Shengqi Liu, Atsuki Momose, Ling Ren, and Victor Shoup. Asynchronous consensus without trusted setup or public-key cryptography. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3242–3256, 2024.

[42] Sourav Das, Benny Pinkas, Alin Tomescu, and Zhuolun Xiang. Distributed randomness using weighted VUFs. Cryptology ePrint Archive, Paper 2024/198, 2024.

[43] Sourav Das, Zhuolun Xiang, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous high-threshold distributed key generation and distributed polynomial sampling. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5359–5376, 2023.

[44] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2721, 2021.

[45] Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. Verifiable secret sharing simplified. *Cryptology ePrint Archive*, 2023.

[46] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2518–2534. IEEE, 2022.

[47] Michael Davidson. State machine replication and consensus with byzantine adversaries. *NIST IR*, 8460, 2023.

[48] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985.

[49] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[50] Sisi Duan, Michael K. Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 2028–2041, New York, NY, USA, 2018. Association for Computing Machinery.

[51] Sisi Duan, Xin Wang, and Haibin Zhang. Fin: Practical signature-free asynchronous common subset in constant time. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 815–829, New York, NY, USA, 2023. Association for Computing Machinery.

[52] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988.

[53] Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In *Annual International Cryptology Conference*, pages 425–458. Springer, 2024.

[54] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.

[55] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 148–161, 1988.

[56] Hanwen Feng, Yingzi Gao, Yuan Lu, Qiang Tang, and Jing Xu. Practical asynchronous distributed key reconfiguration and its applications. Cryptology ePrint Archive, Paper 2025/149, 2025.

[57] Hanwen Feng, Zhenliang Lu, and Qiang Tang. Dragon: Decentralization at the cost of representation after arbitrary grouping and its applications to sub-cubic dkg and interactive consistency. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 469–479, 2024.

[58] Hanwen Feng, Tiancheng Mai, and Qiang Tang. Scalable and adaptively secure any-trust distributed key generation and all-hands checkpointing. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2636–2650, 2024.

[59] Hanwen Feng and Qiang Tang. Asymptotically optimal adaptive asynchronous common coin and DKG with silent setup. Cryptology ePrint Archive, Paper 2024/2098, 2024.

[60] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[61] Pierre-Alain Fouque and Jacques Stern. One round threshold discrete-log key generation without private channels. In *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4*, pages 300–316. Springer, 2001.

[62] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 33–62. Springer, 2018.

[63] Yingzi Gao, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Efficient asynchronous byzantine agreement without private setups. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 246–257, 2022.

[64] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In *Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part VII*, page 352–386, Berlin, Heidelberg, 2024. Springer-Verlag.

[65] Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. In *International conference on financial cryptography and data security*, pages 296–315. Springer, 2022.

[66] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 295–310. Springer, 1999.

[67] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 2007.

[68] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. Yoso: You only speak once: Secure mpc with stateless ephemeral roles. In *Annual International Cryptology Conference*, pages 64–93. Springer, 2021.

[69] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.

[70] Jens Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.

[71] Jens Groth and Victor Shoup. Fast batched asynchronous distributed key generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 370–400. Springer, 2024.

[72] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 147–176. Springer, 2021.

[73] Martin Hirt and Jesper Buus Nielsen. Robust multiparty computation with linear communication complexity. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, CRYPTO'06, page 463–482, Berlin, Heidelberg, 2006. Springer-Verlag.

[74] Ingemar Ingemarsson and Gustavus J Simmons. A protocol to set up shared secret schemes without the assistance of a mutually trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 266–282. Springer, 1990.

[75] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 221–242. Springer, 2000.

[76] Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed key generation in the wild. *Cryptology ePrint Archive*, 2012.

[77] Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive vss using class groups and application to dkg. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4286–4300, 2024.

[78] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.

[79] Jonathan Katz. Round-optimal, fully secure distributed key generation. In *Annual International Cryptology Conference*, pages 285–316. Springer, 2024.

[80] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Annual International Cryptology Conference*, pages 445–462. Springer, 2006.

[81] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. Sok: Public randomness. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 216–234. IEEE, 2024.

[82] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1767, 2020.

[83] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[84] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[85] Christoph Lenzen and Sahar Sheikholeslami. A recursive early-stopping phase king protocol. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 60–69, 2022.

[86] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, page 129–138, New York, NY, USA, 2020. Association for Computing Machinery.

[87] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[88] Yiping Ma, Jess Woods, Sebastian Angel, Antigoni Polychroniadou, and Tal Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 477–496. IEEE, 2023.

[89] Wafa Neji, Kaouther Blibech, and Narjes Ben Rajeb. Distributed key generation protocol with a new complaint management strategy. *Security and communication networks*, 9(17):4585–4595, 2016.

[90] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 275–292, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[91] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[93] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*, pages 522–526. Springer, 1991.

[94] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[95] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. Ethdkg: Distributed key generation with ethereum smart contracts. *Cryptology ePrint Archive*, 2019.

[96] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.

[97] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[98] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 1–16, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[99] Nibesh Shrestha, Adithya Bhat, Aniket Kate, and Kartik Nayak. Synchronous distributed key generation without broadcasts. *IACR Communications in Cryptology*, 1(2), 2024.

[100] Markus Stadler. Publicly verifiable secret sharing. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 190–199. Springer, 1996.

[101] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Scalable anonymous group communication in the anytrust model. In *European Workshop on System Security (EuroSec)*, volume 4, 2012.

[102] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.

[103] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. hbacss: How to robustly share many secrets. In *Proceedings of the Network and Distributed System Security Symposium (NDSS) 2022*, 2022.

[104] Haibin Zhang, Sisi Duan, Chao Liu, Boxin Zhao, Xuanji Meng, Shengli Liu, Yong Yu, Fangguo Zhang, and Liehuang Zhu. Practical asynchronous distributed key generation: Improved efficiency, weaker assumption, and standard model. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 568–581. IEEE, 2023.

Appendix A. Additional Definitions

Joint-Feldman DKG. This is the original Pedersen DKG protocol [93] called *Joint-Feldman DKG*, where parties run instances of Feldman VSS in parallel to share their randomly chosen secrets. The protocol works as follows.

1) Each party P_i, as a dealer, chooses a random polynomial $f_i \in \mathbb{Z}_p[X]$ of degree t:
$$f_i(X) := a_{i0} + a_{i1}X + \ldots + a_{it}X^t.$$
Then, P_i broadcasts commitments $(A_{i0}, A_{i1}, \ldots, A_{i,t})$ where $A_{ij} := g^{a_{ij}}$ for all $j \in [0,t]$. Further, it sends the polynomial share $s_{ij} := f_i(j)$ privately to party P_j for all $j \in [n]$. Also, let $z_i := a_{i0}$ and $y_i := A_{i0}$.
2) Each party P_j verifies the share it received from another party P_i by checking that:
$$g^{s_{ij}} \stackrel{?}{=} \prod_{k=0}^{t} (A_{ik})^{j^k}.$$
If the check fails for a party index $i \in [n]$ (or it does not receive any share from P_i at all), then P_j broadcasts a complaint against P_i. A party receiving more than t complaints is directly disqualified.
3) If P_i receives a complaint from P_j, then P_i reveals (via the broadcast channel) the corresponding share s_{ij}. If any of the revealed shares fails the above verification equation, P_i is disqualified. Let *Qual* be the set of non-disqualified parties at this point.
4) The public key pk is computed as $\mathsf{pk} := \prod_{i \in Qual} y_i$, the public key shares are computed as $\mathsf{pk}_j := \prod_{i \in Qual} A_{ij}$ for $j \in [t]$ (remaining shares are obtained via Lagrange interpolation in the exponent). Further, each party P_j sets its secret key share as $\mathsf{sk}_j := \sum_{i \in Qual} s_{ij}$. The secret key sk itself is not computed by any party, but it is equal to $\mathsf{sk} = \sum_{i \in Qual} z_i$.

Definition 3 (Key-Expressability [72])**.** A Distributed Key Generation (DKG) protocol Π is *key-expressable* if for every static PPT adversary $\mathcal{A}$ corrupting up to t parties, there exists a PPT simulator $\mathcal{S}$ with the following property: On input a uniformly random group element $\mathsf{pk}' \in \mathbb{G}$, it outputs $\alpha \in \mathbb{Z}_p, \mathsf{sk}_1 \in \mathbb{Z}_p$, $\mathsf{pk}_1 = g^{\mathsf{sk}_1} \in \mathbb{G}$, and a view which is indistinguishable from $\mathcal{A}$'s view from a run of the DKG protocol that ends with $\mathsf{pk} = \mathsf{pk}'^{\alpha} \cdot \mathsf{pk}_1$.

Definition 4 (Oracle-aided Algebraic Simulatability [11])**.** A DKG protocol Π has (t,k)-*oracle-aided algebraic simulatability* if for every PPT adversary $\mathcal{A}$ corrupting up to t parties, there exists an algebraic PPT simulator $\mathcal{S}$ that makes $k-1$ queries to an accessible discrete logarithm oracle DL_g, and satisfies the following properties:

- On input $\xi = (g^{z_1}, \ldots, g^{z_k}) \in \mathbb{G}^k$, $\mathcal{S}$ simulates the role of the honest parties in an execution of Π. At the end of the simulation, $\mathcal{S}$ outputs the public key $\mathsf{pk} = g^x$. If corruptions are static, $\mathcal{S}$ gets a set of corrupted parties $\mathcal{C} \subseteq [n]$ of size at most t as an additional input.
- On input $\xi = (g^{z_1}, \ldots, g^{z_k}) \in \mathbb{G}^k$ and for $i \in [k-1]$, let $g_i \in \mathbb{G}$ denote the i-th query to DL_g. Let $(\hat{a}_i, a_{i,1}, \ldots, a_{i,k})$ denote the corresponding algebraic coefficients, i.e., $g_i = g^{\hat{a}_i} \cdot (g^{z_1})^{a_{i,1}} \cdot \ldots \cdot (g^{z_k})^{a_{i,k}}$, and set $(\hat{a}, a_{0,1}, \ldots, a_{0,k})$ as the algebraic coefficients corresponding to pk. Then, the following square matrix over $\mathbb{Z}_p$ is invertible:
$$L := \begin{pmatrix} a_{0,1} & a_{0,2} & \cdots & a_{0,k} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ \vdots & \vdots & & \vdots \\ a_{k-1,1} & a_{k-1,2} & \cdots & a_{k-1,k} \end{pmatrix}.$$
- Denote by $\mathsf{view}_{\mathcal{A},y,\Pi}$ the view of $\mathcal{A}$ in an execution of Π conditioned on all honest parties outputting $\mathsf{pk} = y$. Similarly, denote by $\mathsf{view}_{\mathcal{A},\xi,y,\mathcal{S}}$ the view of $\mathcal{A}$ when interacting with $\mathcal{S}$ on input ξ, conditioned on $\mathcal{S}$ outputting $\mathsf{pk} = y$. (For convenience, $\mathcal{S}$'s final output pk is omitted from $\mathsf{view}_{\mathcal{A},\xi,y,\mathcal{S}}$.) Here, we define $\mathsf{view}_{\mathcal{A},\xi,y,\mathcal{S}} := \perp$ in case $\mathsf{pk} \neq y$ for any run of $\mathcal{S}$ on input ξ. Then, for all y and all ξ, the views $\mathsf{view}_{\mathcal{A},\xi,y,\mathcal{S}}$ and $\mathsf{view}_{\mathcal{A},y,\Pi}$ are identically distributed.

Appendix B.
Meta-Review

B.1. Summary of Paper

This paper systematizes protocols for distributed key generation, designed for both synchronous and asynchronous network models.

B.2. Scientific Contributions

- 8. Other

B.3. Reasons for Acceptance

1) This paper provides an excellent overview, introduction and reference work for the study of DKGs, which are an important topic of growing importance especially in the cryptocurrency space.
2) The paper provides a detailed asymptotic efficiency comparison between schemes
3) The paper provides a good overview of some open problems and gaps, which will be useful to spur future research.

B.4. Noteworthy Concerns

1) The paper's scope is limited to only DKGs for DL-based secret keys. There are other types in the literature which are not covered.
2) The paper only provides asymptotic analysis of efficiency, which may cover up some important concrete cost considerations.