Asynchronous Distributed Key Generation: A State-of-the-Art Survey

Vocdoni

Work in progress v0.01 - February 2026

Abstract

Distributed Key Generation (DKG) enables a group of participants to jointly generate cryptographic keys without relying on a trusted third party. While classical DKG protocols assume synchronous communication, real-world deployments, particularly in blockchain systems, operate under asynchronous conditions where message delays are unpredictable. This report surveys Asynchronous Distributed Key Generation (ADKG), examining its motivation, core technical components, and recent advances. We trace the evolution from Pedersen's seminal 1991 protocol through modern constructions that achieve optimal resilience and communication complexity in fully asynchronous networks.

Contents

1 Introduction **3**

2 Why Asynchrony Matters **3**
2.1 The Synchrony Spectrum . . . 3
2.2 The FLP Impossibility . . . 4
2.3 Why Blockchains Need Asynchrony . . . 4
2.4 Example of Synchrony Mismatch . . . 5

3 Background: Classical DKG and Its Limitations **5**
3.1 Secret Sharing Foundations . . . 5
3.2 Verifiable Secret Sharing . . . 5
3.3 From VSS to DKG: Pedersen's Protocol . . . 6
3.4 The GJKR Critique and Repair . . . 6
3.5 Synchrony: The Hidden Assumption . . . 6
3.6 Partially Synchronous DKG . . . 7
3.7 Towards ADKG . . . 7

4 ADKG: Design Goals and Core Components **7**
4.1 Security Definitions . . . 7
4.2 Reliable Broadcast (RBC) . . . 7
4.3 Asynchronous Binary Agreement (ABA) . . . 8
4.4 Asynchronous Verifiable Secret Sharing (AVSS) . . . 8
4.5 Asynchronous Complete Secret Sharing (ACSS) . . . 8
4.6 Putting It Together: The ADKG Structure . . . 9
4.7 The Complexity of Asynchronous Composition . . . 9
4.8 Practical Threshold Considerations . . . 10

5 State of the Art **10**
5.1 The First ADKG: Kokoris-Kogias et al. (2020) . . . 10
5.2 Improved Asymptotics: Abraham et al. (PODC 2021) . . . 11
5.3 First Practical Implementation: Das et al. (S&P 2022) . . . 11
5.4 High-Threshold Efficiency: Das et al. (USENIX Security 2023) . . . 11
5.5 Standard Model Security: Zhang et al. (DSN 2023) . . . 12
5.6 Adaptive Security: Bingo (CRYPTO 2023) . . . 12
5.7 Optimal Amortized Complexity: Haven++ . . . 12
5.8 Breaking the Cubic Barrier: Abraham et al. (2025) . . . 12
5.9 Quadratic Communication with Silent Setup: Feng & Tang (CRYPTO 2025) . . . 13
5.10 Experimental Comparison . . . 13
5.11 On Protocol Complexity . . . 13
5.12 Open Problems . . . 14

6 Alternative Communication Models: Blockchain-Assisted DKG **15**
6.1 The Public Bulletin Board Model . . . 15
6.2 Smart-Contract-Based DKG . . . 15
6.3 The Synchrony Question . . . 16
6.4 Production Deployment: Shutter Network . . . 17
6.5 Tradeoffs and Protocol Selection . . . 17
6.6 Open Questions . . . 18

7 Conclusion **18**

1 Introduction

Modern distributed systems increasingly rely on threshold cryptography to eliminate single points of failure. A (t, n)-threshold cryptosystem distributes a secret key among n participants so that any subset of at least t can perform cryptographic operations, while coalitions of fewer than t cannot. The critical question is how to generate such threshold keys without a trusted dealer.

Distributed Key Generation (DKG) solves this problem. In a DKG protocol, n parties jointly compute a public-private key pair where the private key is never reconstructed, or is reconstructed only under specific conditions. Each party obtains a share of the secret key, and the protocol guarantees correctness and secrecy even if up to $t - 1$ parties behave maliciously.

The stakes are high. Threshold signatures secure billions of dollars in cryptocurrency wallets. Randomness beacons are used for leader election in proof-of-stake blockchains. Byzantine fault-tolerant consensus protocols rely on threshold cryptography for efficiency. In each case, a compromised key generation phase can undermine all subsequent security guarantees.

Classical DKG protocols, beginning with Pedersen's 1991 construction [27], assume *synchronous* communication: messages arrive within a known bounded delay. This assumption simplifies protocol design but can fail catastrophically in practice. Real networks exhibit unbounded delays, partitions, and adversarial scheduling. A synchronous protocol that times out waiting for a message may exclude honest participants or proceed with an inconsistent state.

Asynchronous Distributed Key Generation (ADKG) removes the synchrony assumption entirely. An ADKG protocol must terminate and produce correct output regardless of message scheduling, provided that fewer than one-third of participants are Byzantine. This stronger model matches the conditions under which blockchain consensus operates and makes ADKG the natural primitive for decentralized applications.

This report surveys the ADKG landscape. Section 2 discusses why asynchrony matters, with emphasis on blockchain deployments. Section 3 traces the evolution from synchronous DKG to the asynchronous setting. Section 4 examines the building blocks of ADKG protocols. Section 5 reviews recent constructions and their performance characteristics. Section 7 identifies open problems and future directions.

2 Why Asynchrony Matters

The choice of network model fundamentally shapes what a distributed protocol can achieve. This section examines the synchrony spectrum and explains why blockchain systems require asynchronous primitives.

2.1 The Synchrony Spectrum

Distributed systems literature distinguishes three network models based on timing assumptions:

Synchronous. Messages arrive within a known bound Δ. Protocols can use timeouts to detect failures: if a message does not arrive within Δ, the sender has failed. This simplifies protocol design but requires conservative timeout values. Setting Δ too low causes false positives (honest nodes mistaken for faulty), setting it too high degrades performance.

Partially Synchronous. The concept of partial synchrony was introduced by Dwork, Lynch, and Stockmeyer [17]. The bound Δ exists but is either unknown or only holds after some unknown *Global Stabilization Time* (GST). Protocols can guarantee safety always but liveness only after GST. Many practical BFT systems, such as PBFT, HotStuff and Tendermint, operate in this model.

Asynchronous. There are no timing assumptions: Messages can be delayed arbitrarily, although they must eventually arrive. An adversary controls message scheduling, subject only to eventual delivery. This is the strongest adversarial model and the hardest setting for protocol design.

Network-Agnostic. Recent work by Bacho et al. [7] explores protocols that provide meaningful guarantees under *both* synchrony and asynchrony simultaneously. In such protocols, different corruption thresholds apply in different network conditions: a protocol might tolerate t_s corruptions under synchrony but only $t_a < t_s$ under asynchrony. In other words, higher corruption tolerance is achievable in the synchronous setting, but if synchrony is not guaranteed the protocol gracefully degrades to asynchronous guarantees. Bacho et al. proved a tight bound for network-agnostic DKG: $t_a + 2t_s < n$.

2.2 The FLP Impossibility

The Fischer-Lynch-Paterson (FLP) impossibility result [20] established that no deterministic protocol can solve consensus in an asynchronous network even if only a single process may crash. The intuition is that a protocol cannot distinguish a crashed process from one whose messages are merely delayed. Any protocol that waits risks waiting forever, while any protocol that proceeds risks inconsistency.

FLP does not render asynchronous consensus impossible, but rules out *deterministic* solutions. There are two approaches to overcome FLP impossibility:

1. **Randomization.** Protocols terminate with probability 1, not certainty. Ben-Or's protocol [10] and its descendants use local coin flips or shared random coins to break symmetry.
2. **Partial synchrony.** Protocols guarantee safety always but liveness only when the network behaves synchronously. PBFT and its variants take this approach.

Modern asynchronous protocols, including ADKG, follow the randomization path. They use threshold cryptography to implement *common coins*, which are shared random values that all honest parties agree on, to drive agreement protocols toward termination.

2.3 Why Blockchains Need Asynchrony

Blockchain networks operate under conditions that violate synchrony assumptions:

Adversarial scheduling. Public networks face active adversaries who can delay, reorder, or selectively drop messages. A synchronous protocol's timeout becomes an attack surface: an adversary who can delay messages by $\Delta+\epsilon$ can partition the network or stall progress indefinitely.

Geographic distribution. Blockchain nodes span continents. Round-trip times between Tokyo and São Paulo can exceed 300ms under normal conditions and spike during congestion. Any fixed Δ must accommodate worst-case latency, penalizing the common case.

Heterogeneous infrastructure. Nodes operate on diverse hardware configurations and have a variety of connectivity profiles. A single Δ cannot capture this heterogeneity.

Permissionless participation. In open networks, nodes join and leave freely. The set of participants and their network characteristics change continuously. Synchronous protocols struggle to set appropriate timeouts when the participant set is unknown.

Partially synchronous protocols offer a compromise: they guarantee safety under asynchrony but require eventual synchrony for liveness. So while the adversary cannot make the system behave incorrectly, it can halt the system for as long as it can sustain network delays. This may be acceptable in many cases, but not for some high-value applications.

Fully asynchronous protocols guarantee progress regardless of network conditions. HoneyBadgerBFT [26] demonstrated that practical asynchronous consensus is achievable with competitive throughput. The natural question follows: can we perform *key generation* under the same model?

2.4 Example of Synchrony Mismatch

Consider an ABFT (Asynchronous Byzantine Fault-Tolerant) blockchain that uses threshold signatures for block finalization. Assume that the DKG system used requires synchrony. For the initial bootstrapping, this is not really a problem, as the procedure can be repeated until it is successful. However, if the protocol requires that the keys be refreshed from time to time, then the system is periodically vulnerable to network attacks. ADKG eliminates this mismatch. By generating keys under the same asynchronous model as the consensus protocol, the entire system maintains consistent security guarantees.

3 Background: Classical DKG and Its Limitations

This section traces the evolution of distributed key generation from its foundations in secret sharing through to the challenges that motivated asynchronous constructions.

3.1 Secret Sharing Foundations

Shamir's seminal 1979 scheme [30] introduced (t, n)-threshold secret sharing. A dealer encodes a secret s as the constant term of a random polynomial $f(x)$ of degree $t-1$ over a finite field:

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$$

Each party P_i receives the share $s_i = f(i)$. Any t parties can reconstruct s via Lagrange interpolation. Fewer than t parties learn nothing about s information-theoretically. Shamir's scheme assumes an honest dealer. A malicious dealer could distribute inconsistent shares, causing reconstruction to fail[1] or yield different secrets for different subsets of parties.

3.2 Verifiable Secret Sharing

Feldman [18] addressed the malicious dealer problem by adding public verification data. Working in a group $\mathbb{G}$ of prime order q with generator g where the discrete logarithm problem is hard, the dealer publishes commitments to the polynomial coefficients:

$$C_j = g^{a_j} \quad \text{for } j \in \{0, \ldots, t-1\}$$

where $a_0 = s$. Party P_i can verify its share s_i by checking:

$$g^{s_i} \stackrel{?}{=} \prod_{j=0}^{t-1} C_j^{i^j}$$

If verification fails, the party broadcasts a complaint. The dealer must then reveal the disputed share publicly. If too many complaints arise, the dealer is disqualified.

Feldman's VSS reveals g^s, the public key corresponding to secret s. For many threshold cryptosystems this is acceptable or even required. Pedersen [28] proposed an alternative using dual commitments that hides g^s, at the cost of additional complexity.

[1] The reconstruction cannot "fail" for any subset of t parties, as there will always exist a polynomial of degree $t-1$ corresponding to t shares. However, with a malicious dealer, $t+1$ (or more) parties may find out that their shares do not correspond to any polynomial of degree $t-1$.

3.3 From VSS to DKG: Pedersen's Protocol

Pedersen's 1991 DKG protocol [27] eliminates the trusted dealer by having all parties act as dealers simultaneously. Each party P_i:

1. Chooses a random polynomial $f_i(x)$ of degree $t-1$ with secret $s_i = f_i(0)$.
2. Distributes shares $f_i(j)$ to each party P_j using Feldman VSS.
3. Publishes commitments $C_{i,k} = g^{a_{i,k}}$ for verification.

After a complaint resolution phase, parties agree on a qualified set $\mathcal{Q}$ of dealers whose sharing succeeded. The final secret key is:

$$s = \sum_{i \in \mathcal{Q}} s_i$$

and each party's share is:

$$\sigma_j = \sum_{i \in \mathcal{Q}} f_i(j)$$

The public key $g^s = \prod_{i \in \mathcal{Q}} g^{s_i}$ is computable from the published commitments. No single party knows s, yet any t parties can reconstruct it, or use their shares for threshold operations without reconstruction.

3.4 The GJKR Critique and Repair

Gennaro, Jarecki, Krawczyk, and Rabin [21] identified a subtle flaw in Pedersen's protocol: an adversary can bias the distribution of the generated secret. By choosing its polynomial coefficients adaptively after seeing honest parties' commitments, a Byzantine party can influence which value s takes. While s remains unpredictable, it is not uniformly random.

For some applications this bias is harmless. Gennaro et al. showed in follow-up work [22] that Pedersen's protocol remains secure for Schnorr signatures and similar schemes where the secret's distribution need not be uniform. However, applications requiring true randomness, such as randomness beacons, need stronger guarantees.

The GJKR repair adds a second round using Pedersen commitments, which hide the committed value, before revealing Feldman commitments. This ensures that adversaries commit to their polynomials before learning anything about honest parties' choices, restoring uniformity at the cost of an additional communication round.

Recent systematization work [6] has formalized a weaker notion called *security-preserving* DKG: the generated key need only be indistinguishable from one chosen by a simulator, rather than uniformly random. This relaxation is enough for "rekeyable" cryptographic schemes, i.e. those whose security holds even when the adversary influences key generation. This includes many threshold signature and encryption schemes.

3.5 Synchrony: The Hidden Assumption

All classical DKG protocols share a critical assumption: synchronous communication. The protocols proceed in rounds, and each round requires:

- Every party to broadcast or send messages.
- Every party to receive all messages from the previous step.
- A timeout mechanism to detect non-responding (presumably crashed) parties.

The complaint mechanism exemplifies this dependency. If party P_j receives an invalid share from P_i, it broadcasts a complaint. Party P_i must respond within a timeout, either by proving the

share valid or being disqualified. But what if P_i's response is merely delayed? A synchronous protocol cannot distinguish a slow honest party from a crashed or malicious one.

Setting the timeout too short risks excluding honest parties experiencing network delays. Setting it too long degrades performance and opens denial-of-service vectors: an adversary can force the protocol to wait out every timeout. In adversarial networks, there may be no "correct" timeout value.

3.6 Partially Synchronous DKG

Kate, Huang, and Goldberg [24] proposed a DKG protocol for the partially synchronous model, building on asynchronous verifiable secret sharing primitives. Their construction tolerates $f < n/3$ Byzantine faults among n parties and achieves $O(\kappa n^4)$ communication complexity, where κ is the security parameter. This represented progress toward asynchrony but retained a fundamental limitation: liveness depends on eventual synchrony.

3.7 Towards ADKG

By 2019, fully asynchronous BFT consensus protocols existed, notably HoneyBadgerBFT [26]. The following year, Kokoris-Kogias, Malkhi, and Spiegelman [25] proposed the first DKG scheme under the same asynchronous model as HoneyBadgerBFT? This launched the modern ADKG literature, which we survey in Section 5.

4 ADKG: Design Goals and Core Components

Constructing ADKG requires assembling several asynchronous primitives into a coherent protocol. This section defines the security goals and examines each building block.

4.1 Security Definitions

An ADKG protocol for $n = 3f + 1$ parties (tolerating f Byzantine faults) must satisfy:

Termination. Every honest party eventually outputs a key share and public key.
Agreement. All honest parties output shares corresponding to the same secret s and the same public key g^s.
Correctness. The shares held by honest parties lie on a polynomial of degree $t - 1$, enabling reconstruction by any t parties.
Secrecy. The adversary learns nothing about s beyond what is implied by the public key, even after corrupting up to f parties.
Uniformity. The secret s is uniformly distributed (or computationally indistinguishable from uniform), preventing adversarial bias.

The reconstruction threshold t is a protocol parameter. The "low threshold" setting uses $t = f+1$; the "high threshold" setting uses $t = 2f + 1$, which is important for applications like SBFT-style consensus [34] requiring stronger secrecy guarantees.

4.2 Reliable Broadcast (RBC)

Reliable broadcast [12] allows a designated sender to deliver a message to all parties with the following guarantees:

Validity. If the sender is honest and broadcasts m, all honest parties eventually deliver m.
Agreement. If any honest party delivers m, all honest parties eventually deliver m.

Integrity. Each honest party delivers at most one message, and if the sender is honest, it equals what was broadcast.

Bracha's classic protocol achieves RBC with $O(n^2)$ messages and $O(n \cdot |m|)$ communication for message m. The protocol proceeds in three phases (ECHO, READY, DELIVER), using quorum intersection to ensure agreement despite Byzantine behaviour.

RBC is the workhorse of asynchronous protocols. It ensures that honest parties have a consistent view of broadcast data—critical for ADKG where parties must agree on which sharing attempts succeeded.

4.3 Asynchronous Binary Agreement (ABA)

Asynchronous Binary Agreement solves consensus on a single bit:

Validity. If all honest parties input b, all honest parties output b.
Agreement. All honest parties output the same value.
Termination. All honest parties eventually output a value (with probability 1).

By FLP, ABA cannot be deterministic. Randomized solutions use a *common coin*, which is a shared random bit that all honest parties agree on with constant probability. Ben-Or's protocol [10] uses local coin flips, yielding exponential expected rounds. Protocols using threshold cryptography to implement a strong common coin achieve constant expected rounds [14].

A *strong common coin* satisfies: (1) all honest parties output the same bit b, and (2) b is uniformly random and unpredictable until queried. Threshold signatures provide an efficient instantiation: parties contribute signature shares on a nonce, combine them into a full signature, and hash the result to obtain the coin value.

The circularity is immediate: threshold signatures require distributed keys, but ADKG requires ABA, which requires threshold signatures. Breaking this cycle is a central challenge of ADKG design.

4.4 Asynchronous Verifiable Secret Sharing (AVSS)

AVSS [13] extends VSS to the asynchronous setting. A dealer shares a secret s among n parties such that:

Termination. If the dealer is honest, all honest parties eventually receive valid shares. If any honest party completes sharing, all honest parties eventually complete.
Correctness. Honest parties' shares are values of a degree-$(t-1)$ polynomial with constant term s.
Secrecy. The adversary learns nothing about s until reconstruction.

The key challenge is termination without synchrony. Cachin et al. [13] solved this using bivariate polynomials. The dealer commits to a polynomial $F(x, y)$ with $s = F(0, 0)$. Party P_i receives the univariate polynomial $F(x, i)$ and can verify consistency by cross-checking evaluations with other parties. The protocol achieves $O(\kappa n^3)$ communication, where κ is the security parameter.

4.5 Asynchronous Complete Secret Sharing (ACSS)

ACSS strengthens AVSS with an additional guarantee:

Completeness. If any honest party completes sharing, then the shared secret is uniquely determined, and all honest parties can eventually reconstruct it.

The distinction matters for ADKG. In AVSS, a Byzantine dealer might create a situation where some honest parties complete but the secret cannot be reconstructed. ACSS prevents this: completion implies reconstructibility.

Modern ACSS constructions [35] use polynomial commitments (e.g., KZG commitments) combined with reliable broadcast. The dealer commits to its polynomial, broadcasts the commitment, and sends encrypted shares to each party. Parties verify their shares against the commitment and broadcast acknowledgments. Once enough acknowledgments arrive, all honest parties can complete.

4.6 Putting It Together: The ADKG Structure

Recent systematization [6] identifies a common "share-disperse-elect-reconstruct" paradigm underlying most ADKG constructions. This framework clarifies how the building blocks combine:

1. **Share.** Each party generates a random secret and creates shares using a polynomial.
2. **Disperse.** Shares are distributed to all parties with verifiability guarantees (ACSS or AVSS).
3. **Elect.** Parties agree on a qualifying set of successful dealers via agreement primitives (ABA, ACS, or MVBA).
4. **Reconstruct.** The final key is derived by combining contributions from elected dealers.

In practice, ADKG protocols proceed in three phases:

Sharing Phase. Each party P_i acts as a dealer, running ACSS to share a random secret s_i. Parties participate in n concurrent ACSS instances.

Agreement Phase. Parties must agree on which ACSS instances completed successfully. This requires solving n instances of agreement (one per dealer) or, more efficiently, a single instance of *Multi-Valued Validated Byzantine Agreement* (MVBA) or *Asynchronous Common Subset* (ACS) agreeing on a subset of at least $n - f$ dealers whose sharings succeeded.

Key Derivation. Once parties agree on a set $\mathcal{Q}$ of successful dealers, the final secret is $s = \sum_{i \in \mathcal{Q}} s_i$, and each party's share is $\sigma_j = \sum_{i \in \mathcal{Q}} s_{i,j}$, where $s_{i,j}$ is P_j's share from P_i's ACSS.

4.7 The Complexity of Asynchronous Composition

Several issues arise when composing these primitives:

The common coin bootstrap. ABA needs a common coin, typically from threshold signatures, which need distributed keys, but we're trying to generate those keys. The first ADKG constructions [25] broke this cycle using an *Eventually Perfect Common Coin*: the secrets from the sharing phase themselves provide randomness. The coin may fail (disagree) up to f times, but eventually succeeds, yielding an *Eventually Efficient* ABA.

Adaptive security. An adaptive adversary can corrupt parties during protocol execution. If the adversary sees partial protocol state before choosing whom to corrupt, it may bias the output or learn the secret. Achieving adaptive security typically requires additional techniques like proactive refresh or carefully timed reveals.

Threshold flexibility. Low-threshold ADKG ($t = f + 1$) is easier: any $f + 1$ honest parties can reconstruct. High-threshold ADKG ($t = 2f + 1$) requires that even if the adversary corrupts f parties *after* key generation, the secret remains hidden. This demands more sophisticated ACSS constructions with dual thresholds.

Communication complexity. Naive composition yields $O(\kappa n^4)$ communication: n ACSS instances each costing $O(\kappa n^3)$. Reducing this to $O(\kappa n^3)$ using batching and amortization is a focus of recent work.

Table 1: Comparison of ADKG protocols. Communication in words (κ = security parameter). Threshold: $t+1$ (low) or $2t+1$ (high). Setup: PKI, powers-of-tau (τ), VRF, or none.

Protocol	Comm.	Rounds	Threshold	Setup	Adaptive
Kokoris-Kogias et al. [25]	$O(\kappa n^4)$	$O(f)$	low	—	no
Abraham et al. [1]	$O(\kappa n^3)$	$O(1)$	low	τ	no
Das et al. [15]	$O(\kappa n^3)$	$O(\log n)$	both	PKI	no
Das et al. [16]	$O(\kappa n^3)$	$O(\log n)$	high	—	no
Zhang et al. [36]	$O(\kappa n^3)$	$O(\log n)$	both	—	no
Bingo [2]	$O(\kappa n^3)$	$O(1)$	high	τ	**yes**
Haven++ [4]	$O(\kappa n)^*$	$O(1)$	high	τ	no
Abraham et al. [3]	$\tilde{O}(n^{2+1/k})$	$O(k)$	low	VRF	**yes**
Feng & Tang [19]	$O(\kappa n^2)$	$O(1)$	low	silent	**yes**

*Amortized per secret when batching $O(n)$ secrets.

4.8 Practical Threshold Considerations

An important subtlety arises when deploying ADKG in systems with node churn. Suppose an (n, t)-threshold ADKG completes successfully, but k of the n parties later become unavailable. The resulting scheme is effectively $(n-k, t)$: the shares still lie on a degree-$(t-1)$ polynomial, so reconstruction still requires exactly t shares, but only $n-k$ parties hold shares. The "headroom" for tolerating additional failures shrinks: originally $n-t$ parties could fail while preserving reconstructibility; now only $(n-k)-t$ can. Several implications follow:

Liveness during key generation. ADKG requires at least $n-f = 2f+1$ participating honest parties to terminate. If more than f parties fail to participate, the protocol may not complete at all—there will be no output. The qualifying set $\mathcal{Q}$ is guaranteed to contain at least $n-f$ members, but some contributors to the final key might be among those who later disappear.

Security margin erosion. The original design assumes the adversary controls at most f of n parties. If k honest parties drop out post-protocol, the adversary's *relative* influence increases. For high-threshold settings ($t = 2f+1$), security holds as long as the adversary controls fewer than t shares. But the margin for tolerating *additional* post-protocol corruptions has shrunk from f to $f-k$ (assuming the k departed nodes were honest).

Deployment implications. Systems expecting churn have several options: (1) run ADKG with a larger n than the target operational committee size, building in headroom; (2) plan for periodic key refresh when participation drops below a comfort threshold; or (3) use proactive secret sharing to "repair" the threshold scheme by generating new shares for replacement nodes. The choice depends on the application's availability requirements, the cost of re-running ADKG, and whether the system can tolerate temporary periods of reduced fault tolerance.

5 State of the Art

This section surveys ADKG constructions from 2020 to 2024, tracing the evolution from theoretical feasibility to practical deployability. Table 1 summarizes key protocols.

5.1 The First ADKG: Kokoris-Kogias et al. (2020)

Kokoris-Kogias, Malkhi, and Spiegelman [25] presented the first ADKG protocol at CRYPTO 2020. Their construction addresses the circular dependency between ADKG and asynchronous agreement by introducing an *Eventually Perfect Common Coin* (EPCC).

The key insight is that the secrets being shared during the ADKG protocol can themselves serve

as randomness for the agreement phase. Each party runs a High-threshold AVSS (HAVSS) to share a random secret with reconstruction threshold $2f + 1$. The protocol uses a "weak DKG" to detect which secrets are available to all honest parties, then uses these secrets to instantiate a common coin. The coin may fail (output different values to different parties) up to f times, but eventually succeeds once enough secrets are agreed upon.

The resulting protocol achieves $O(\kappa n^4)$ communication complexity and $O(f)$ expected rounds. While asymptotically suboptimal, it demonstrated that ADKG is achievable without trusted setup, breaking a conceptual barrier.

5.2 Improved Asymptotics: Abraham et al. (PODC 2021)

Abraham, Jovanovic, Maller, Meiklejohn, Stern, and Tomescu [1] at PODC 2021 improved communication to $O(\kappa n^3)$ and rounds to expected $O(1)$. Their protocol uses Aggregatable Publicly Verifiable Secret Sharing (APVSS), which allows combining multiple sharing transcripts into a single compact proof.

The protocol structure follows the standard pattern: n concurrent sharing instances, agreement on a qualifying set, and local aggregation. The improvement comes from efficient agreement: instead of n separate ABA instances, they use a single Multi-Valued Validated Byzantine Agreement (MVBA) to agree on the entire qualifying set at once.

A limitation is that the secret is a group element rather than a field element, making it incompatible with some threshold cryptosystems that require field-element secrets. The protocol also requires a powers-of-tau trusted setup for the underlying polynomial commitment scheme.

5.3 First Practical Implementation: Das et al. (S&P 2022)

Das, Yurek, Xiang, Miller, Kokoris-Kogias, and Ren [15] presented the first implemented and benchmarked ADKG at IEEE S&P 2022, often called "DYX+ ADKG." Their contribution is a concretely efficient protocol with field-element secrets, compatible with standard threshold cryptosystems.

The protocol supports both low threshold $(t + 1)$ and high threshold $(2t + 1)$ reconstruction. For low threshold, they use an efficient ACSS construction based on Feldman commitments. For high threshold, they introduce High-threshold ACSS (HACSS) using the Decisional Composite Residuosity (DCR) assumption for verifiable encryption of shares.

Experiments on Amazon EC2 with up to 49 nodes showed practical performance: with $n = 49$ and low threshold, the protocol completes in under 10 seconds with approximately 3 MB bandwidth per node. High-threshold mode is significantly slower due to the expensive DCR-based cryptography, taking over 100 seconds with failure threshold 16.

The reliance on DCR (more expensive than elliptic-curve operations) and the Random Oracle Model motivated subsequent work on efficiency improvements.

5.4 High-Threshold Efficiency: Das et al. (USENIX Security 2023)

Das, Xiang, Kokoris-Kogias, and Ren [16] at USENIX Security 2023 significantly improved high-threshold ADKG efficiency. The key observation is that high-threshold ACSS is the bottleneck: with $n = 128$ nodes, running n parallel high-threshold ACSS instances takes 504 seconds, versus 0.19 seconds for low-threshold ACSS.

Their solution is *distributed polynomial sampling*: instead of each party sharing a high-degree polynomial directly, parties share low-degree polynomials and collaboratively compute shares of a high-degree polynomial. Each party shares two random secrets using low-threshold ACSS, and the protocol derives a degree-$2t$ polynomial whose coefficients are random linear combinations

of the shared secrets.

The protocol achieves 90% reduction in running time and 80% reduction in bandwidth compared to DYX+ ADKG for high-threshold settings. With 64 nodes, the protocol completes in approximately 5 seconds. The construction requires only the Discrete Logarithm assumption, avoiding DCR entirely.

5.5 Standard Model Security: Zhang et al. (DSN 2023)

Zhang, Duan, and collaborators [36] at DSN 2023 addressed the Random Oracle Model (ROM) dependency. Previous protocols either required ROM for efficiency or achieved standard-model security with worse performance.

Their protocol achieves $O(\kappa n^3)$ communication in the standard model (without ROM) using only the Discrete Logarithm assumption. The key technique is a more direct reduction from ADKG to underlying building blocks, reducing both computational overhead and communication rounds in the normal case. Benchmarks show 2–4.6× improvement in latency over DYX+ ADKG.

5.6 Adaptive Security: Bingo (CRYPTO 2023)

All prior ADKG protocols assume a *static* adversary that chooses which parties to corrupt before protocol execution. In practice, an *adaptive* adversary can corrupt parties during execution based on observed protocol messages—a significantly stronger attack model.

Abraham, Jovanovic, Maller, Meiklejohn, and Stern [2] presented Bingo at CRYPTO 2023, the first adaptively secure ADKG matching the asymptotic complexity of static protocols. The core is a Packed Asynchronous Verifiable Secret Sharing (PAVSS) protocol allowing a dealer to share $f + 1$ secrets with $O(\kappa n^2)$ total communication.

In the security proof, the simulator must be able to explain corrupted parties' views without knowing the secret in advance. Bingo achieves this through a bivariate polynomial construction where the dealer's commitment can be "opened" to any consistent sharing after the fact.

Using Bingo, they construct an adaptively secure VABA with $O(\kappa n^3)$ expected communication, which yields an adaptively secure high-threshold ADKG with the same complexity. The protocol requires a powers-of-tau setup.

5.7 Optimal Amortized Complexity: Haven++

Alhaddad, Varia, and Zhang [4] pushed communication complexity further with Haven++. While single-secret ADKG has an $\Omega(\kappa n^2)$ lower bound, amortization over multiple secrets can achieve better per-secret complexity.

Haven++ is a dual-threshold ACSS protocol using packing and batching. Packing allows multiple secrets per sharing; batching amortizes proof generation across invocations. When sharing $O(n)$ secrets, Haven++ achieves $O(\kappa n)$ amortized communication per secret—optimal for the setting.

Applied to ADKG, Haven++ achieves optimal worst-case amortized communication complexity of $O(\kappa n)$ per key when generating multiple keys simultaneously. This is particularly relevant for applications like key refresh, where parties periodically generate new keys.

5.8 Breaking the Cubic Barrier: Abraham et al. (2025)

Abraham, Bacho, Loss, and Stern [3] presented the first ADKG with sub-cubic communication complexity, breaking the $O(n^3)$ barrier that had persisted since the first ADKG constructions. For any parameter $1 \leq k \leq \log n$, their protocol achieves $\tilde{O}(n^{2+1/k})$ message complexity in $O(k)$ expected rounds. Setting $k = \log n$ yields nearly quadratic communication: $\tilde{O}(n^2)$ messages with

$O(\log n)$ rounds.

The protocol requires a VRF (Verifiable Random Function) setup and assumes secure erasures (parties can delete intermediate state). It tolerates a strongly adaptive adversary corrupting up to $f < n/3$ parties, making it both communication-efficient and security-strong. There is a tradeoff between communication ($n^{2+1/k}$) and rounds (k), so that the protocol can be tuned for specific use cases, depending on latency/bandwidth constraints.

5.9 Quadratic Communication with Silent Setup: Feng & Tang (CRYPTO 2025)

Feng and Tang [19] at CRYPTO 2025 achieved optimal $O(\kappa n^2)$ communication with only a *silent* public setup, which is a weaker assumption than powers-of-tau or VRF setup. Their construction introduces an Asynchronous Common Coin protocol with quadratic communication, which immediately yields quadratic-communication ADKG.

The key primitive is *asynchronous subset alignment*, enabling parties to agree on overlapping subsets efficiently. Combined with enhanced silent-setup threshold encryption, the protocol achieves $O(1)$ expected rounds while maintaining adaptive security against $f < n/3$ corruptions.

This result demonstrates that optimal communication complexity is achievable for ADKG without complex trusted setup ceremonies, significantly lowering deployment barriers.

5.10 Experimental Comparison

Across implementations, several patterns emerge:

Scalability. Protocols have been tested with up to 128 nodes. Communication grows as $O(n^3)$ in the worst case, meaning bandwidth per node grows as $O(n^2)$. With $n = 128$ and typical parameters, each node sends tens of megabytes.

Latency. Low-threshold protocols complete in seconds; high-threshold protocols in tens of seconds. The gap has narrowed from 100× to approximately 5–10× through algorithmic improvements.

Assumptions. Early protocols required expensive assumptions (DCR) or trusted setup (powers-of-tau). Recent work achieves competitive performance with only the Discrete Logarithm assumption and no trusted setup.

Threshold flexibility. Supporting arbitrary thresholds $t \leq \ell \leq n - t$ remains challenging. Most protocols optimize for either $\ell = t + 1$ or $\ell = 2t + 1$.

5.11 On Protocol Complexity

A natural question arises: have the advances surveyed above come at the cost of ever-increasing protocol complexity? The answer is nuanced.

Sources of increased complexity. Certain capabilities genuinely require sophisticated machinery. Adaptive security demands careful simulation arguments; Bingo's bivariate polynomial construction enables post-hoc "opening" to any consistent sharing, a feature absent from simpler static protocols. Sub-cubic communication [3, 19] introduces new primitives that add conceptual overhead: VRF setup, secure erasures, asynchronous subset alignment. The cryptographic toolkit has expanded: KZG commitments, packed secret sharing, and aggregatable PVSS are now standard components.

Sources of reduced complexity. On the other hand, several developments have simplified the landscape. The agreement phase improved dramatically: Abraham et al. [1] replaced n

parallel ABA instances with a single MVBA, a genuine simplification that also improved efficiency. For high-threshold ADKG, Das et al. [16] eliminated DCR-based verifiable encryption entirely; their distributed polynomial sampling approach is both conceptually cleaner and more efficient than the machinery it replaced. Cryptographic assumptions have weakened progressively, from DCR plus Random Oracle to Discrete Logarithm only to silent setup. This simplifies security proofs.

Paradigm alignment. Perhaps most importantly, the field has converged on a common structure. The "share-disperse-elect-reconstruct" paradigm [6] means that new protocols are variations on a theme rather than novel architectures. Building blocks such as RBC, ACSS or MVBA have well-understood properties. Complexity is encapsulated within modules rather than spread throughout a monolithic design.

Assessment. The total complexity of state-of-the-art protocols exceeds that of 2020 constructions, but the complexity per unit of capability has arguably decreased. A 2025 protocol achieving sub-cubic communication, adaptive security, and minimal setup is more complex than 2020's $O(\kappa n^4)$ static protocol, but is likely simpler than what would result from naively augmenting the 2020 design with those features. New simplifying abstractions have been introduced: Eventually Perfect Common Coin, distributed polynomial sampling, packed AVSS. Whether this trend continues as the field tackles remaining challenges, such as post-quantum security and dynamic participation, remains to be seen.

5.12 Open Problems

A recent systematization of knowledge by Bacho and Kavousi [6] provides comprehensive analysis of DKG protocols across network models. Drawing on their analysis and the state of the art, several challenges remain:

Truly quadratic without setup. Abraham et al. [3] achieved $\tilde{O}(n^2)$ with VRF setup; Feng & Tang [19] achieved $O(\kappa n^2)$ with silent setup. Whether truly quadratic ADKG is achievable with *no* setup remains open. GRandLine [8] achieves (log-)quadratic for synchronous DKG, suggesting further async improvements may be possible.

Adaptive security without setup. Bingo achieves adaptive security but requires powers-of-tau. The 2025 constructions [3, 19] achieve adaptive security with weaker setup (VRF or silent), but removing setup entirely while preserving adaptive security and efficiency is open.

Network-agnostic DKG. Bacho et al. [7] showed that protocols tolerating t_a async corruptions and t_s sync corruptions must satisfy $t_a+2t_s < n$. Achieving optimal network-agnostic ADKG with practical efficiency remains an active area.

Dynamic participation. Current protocols assume a fixed set of n parties. Allowing parties to join or leave during key generation would enable truly permissionless ADKG.

Post-quantum security. All discrete-log ADKG protocols are vulnerable to quantum computers. Yang et al. [33] recently proposed LADKG, the first lattice-based ADKG. LADKG combines Asynchronous Verifiable Short Secret Sharing (AV3S) with an Approximate Asynchronous Common Subset (AACS) protocol, achieving practical performance: with $n = 121$ nodes on geo-distributed AWS clusters, key generation completes in 45 seconds under optimistic conditions—comparable to classical ADKG schemes. The growing ecosystem of lattice-based threshold cryptography (including Olingo for threshold signatures [23] and module-lattice DKG [9]) suggests post-quantum ADKG is maturing, though further efficiency improvements and security analysis remain active areas.

Proactive refresh. Generating a key is insufficient; long-running systems need periodic key refresh to limit adversarial advantage. Asynchronous proactive secret sharing exists [13], but efficient integration with modern ADKG is ongoing work.

Security notions. Gurkan et al. introduced "security-preserving" DKG—a weaker notion where the DKG preserves security of threshold primitives built atop it. Under this notion, even biased protocols like Pedersen DKG suffice for "rekeyable" schemes. Understanding which applications require full uniformity versus security-preserving guarantees could simplify protocol design.

6 Alternative Communication Models: Blockchain-Assisted DKG

The preceding sections surveyed ADKG in the standard point-to-point model, where parties communicate via pairwise authenticated channels and implement broadcast through protocols like Bracha's RBC. This section examines an alternative architectural approach: leveraging existing blockchain infrastructure as a communication and coordination layer. These protocols are typically *not* fully asynchronous—most assume synchrony or partial synchrony provided by the underlying chain—but they represent an important practical path for applications already deployed on blockchain platforms. We include this survey for completeness and because the tradeoffs involved inform protocol selection for real-world deployments.

6.1 The Public Bulletin Board Model

In the *public bulletin board* (PBB) model, broadcast communication is implemented via an external append-only ledger, typically a blockchain. Parties post messages to the bulletin board; all parties can read all posted messages. The bulletin board provides ordering and persistence guarantees that would otherwise require explicit agreement protocols.

This model fundamentally changes what is achievable. In point-to-point ADKG, per-party communication ranges from $O(n^2)$ to $O(n^3)$ bits. The PBB model can reduce this dramatically—in some constructions to $O(1)$ bits per party—by offloading agreement to the bulletin board.

Constant per-party communication. Applebaum and Pinkas [5] asked whether DKG is possible with per-party communication that does not grow with n. They answered affirmatively: in the PBB model, each party sends and receives only $O(1)$ bits (depending on the security parameter and field size, but independent of n).

Their protocol operates in a "near-threshold" setting with privacy parameter τ_p and correctness parameter τ_c satisfying $0 < \tau_p < \tau_c < 1$. Intuitively, $\tau_p n$ bounds the number of corruptions for which secrecy holds, while $\tau_c n$ bounds the number for which correctness holds; the gap $\tau_c - \tau_p$ is the "near-threshold" regime where security degrades gracefully. The construction uses Low-Density Parity-Check (LDPC) codes to build secret-sharing schemes with special robustness properties, combined with non-interactive zero-knowledge proofs and commitments.

Silent setup. A related model is the *silent setup* of Feng and Tang [19], where each party posts their public key once to a bulletin board (or sends it to a trusted party). This generalizes conventional PKI while remaining a "public setup" that disallows distributing secrets to individual parties. Under silent setup, they achieve $O(\kappa n^2)$ total communication with $O(1)$ rounds and adaptive security—the best known result for adaptive ADKG without trusted setup. Notably, Feng and Tang's protocol *is* fully asynchronous; the silent setup serves as a weak form of bulletin board for initial key distribution only.

6.2 Smart-Contract-Based DKG

A practical instantiation of the PBB model uses smart contracts on existing blockchains. Table 2 summarizes the key approaches.

ETHDKG. Schindler et al. [29] introduced ETHDKG, implementing Joint-Feldman DKG on Ethereum with smart contracts coordinating the protocol phases. The blockchain provides

Table 2: Comparison of blockchain-assisted DKG protocols. All assume synchrony except where noted. "Rounds" counts on-chain phases, not network round-trips.

Protocol	Cryptographic Approach	Rounds	Key Feature
Schindler et al. [29]	Joint-Feldman	4+	First EVM implementation
Sober et al. [31]	Feldman + zk-SNARKs	4+	Cheaper dispute resolution
Zhang et al. [37]	CP-ABE	1	$O(n)$ reconstruction
Applebaum & Pinkas [5]	LDPC codes	1	$O(1)$ per-party comm.

ordering and availability; the smart contract enforces timing constraints and aggregates commitments. The protocol proceeds through registration, share distribution, dispute, and key derivation phases, with timeouts (measured in block numbers) governing transitions between phases.

However, ETHDKG requires interactive dispute resolution: if a party receives an invalid share, it posts a complaint on-chain, and the dealer must respond within a timeout by revealing the disputed share. This interaction increases gas costs and extends the protocol duration. The original implementation supported up to 256 participants within Ethereum's block gas limits.

zk-SNARK-enhanced disputes. Sober et al. [31] enhanced this approach using zk-SNARKs for dispute resolution and key derivation, receiving Best Paper in Distributed Systems at ACM SAC 2023. Their protocol retains a dispute phase, but when a party issues a complaint, the accused dealer proves correct behaviour by submitting a zk-SNARK rather than revealing secret shares on-chain. The key insight is *off-chain computation with on-chain verification*: parties perform expensive cryptographic operations locally and post succinct proofs to the contract. This reduces gas costs for dispute resolution while maintaining public verifiability.

The tradeoff is the trusted setup requirement for zk-SNARKs (typically Groth16), which necessitates an MPC ceremony to generate proving and verification keys. For applications already using zk-SNARKs elsewhere, this marginal cost may be acceptable.

Alternative cryptographic approaches. Zhang et al. [37] explored using decentralized Ciphertext-Policy Attribute-Based Encryption (CP-ABE) instead of PVSS or Feldman commitments. Their protocol achieves 1-round sharing with $O(n)$ reconstruction complexity—improving upon the $O(n^2)$ reconstruction of PVSS-based schemes—and provides adaptive security. The CP-ABE structure enables "external decryption," allowing any party to decrypt shares without storing ciphertext after the sharing phase. While this protocol assumes synchrony (1-round sharing is impossible asynchronously), it demonstrates that the bulletin board model admits diverse cryptographic instantiations beyond the Feldman/Pedersen and PVSS families.

6.3 The Synchrony Question

A critical distinction separates the protocols in this section from the ADKG constructions surveyed earlier: **most blockchain-assisted DKG protocols assume synchrony**, not asynchrony.

The synchrony assumption manifests in several ways:

- **Timeout-based phases.** ETHDKG and Sober et al. use block-number deadlines to transition between protocol phases. A party that misses a deadline (e.g., fails to post shares or respond to a complaint) is excluded or penalized. This mirrors synchronous protocol design, where timeouts distinguish crashed parties from slow ones.
- **Bulletin board access.** Applebaum and Pinkas assume synchronous access to the bulletin board: all parties can read messages posted in round r before round $r + 1$ begins. This is reasonable if the bulletin board is a blockchain with predictable block times, but it is not fully asynchronous.

- **Inherited timing assumptions.** Smart-contract DKG inherits the timing model of the underlying blockchain. Ethereum and similar chains operate under partial synchrony (Tendermint, Casper FFG) or optimistic synchrony assumptions. The DKG protocol cannot be "more asynchronous" than its substrate.

This is not necessarily a limitation. For applications deployed on a specific blockchain, matching the chain's timing model is natural. A DKG protocol need not tolerate network conditions that would also prevent the underlying chain from making progress. The synchrony assumption becomes problematic only when the DKG must bootstrap a system (where no chain exists yet) or operate across chains with different timing properties.

6.4 Production Deployment: Shutter Network

Shutter Network [32] provides a real-world example of smart-contract DKG in production. Deployed on Gnosis Chain since October 2022, Shutter implements threshold encryption for transaction privacy and front-running protection.

The system uses a relatively simple DKG design: Boneh-Franklin IBE [11] for threshold encryption, Shamir secret sharing, and Feldman VSS for verifiability. The DKG smart contract coordinates a committee of "keypers" who jointly generate decryption keys for future time slots.

Shutter's deployment illustrates both the viability and the imperfections of production DKG:

- **Viability.** The system has operated continuously for over three years, generating keys for millions of encrypted transactions.
- **Known limitations.** The protocol uses simple (not state-of-the-art) cryptography, has $O(n^2)$ communication complexity, requires synchrony, assumes an honest majority, and employs optimistic verification (invalid contributions are detected but not always prevented). A malleability bug was discovered and patched post-deployment.
- **Pragmatic tradeoffs.** Despite these limitations, Shutter demonstrates that "imperfect but deployed" can provide value. The theoretical advances surveyed in Section 5 have yet to see comparable production adoption.

6.5 Tradeoffs and Protocol Selection

The choice between point-to-point ADKG and blockchain-assisted DKG depends on several factors:

When to use blockchain-assisted DKG.

- The application is already deployed on a blockchain, and the chain's timing assumptions are acceptable.
- Crypto-economic incentives (staking, slashing) are desired to encourage honest participation.
- Public verifiability of the key generation process is required for transparency or auditability.
- The participant set is dynamic, with parties joining via on-chain registration.
- Simplicity and auditability outweigh asymptotic efficiency concerns (e.g., $n \leq 100$).

When to use point-to-point ADKG.

- Bootstrapping a new system where no trusted bulletin board exists.
- Full asynchrony is required (no timing assumptions tolerable).
- Adaptive security against strong adversaries is necessary.
- Optimal resilience ($f < n/3$) under Byzantine faults is required.
- Communication efficiency at scale ($n > 100$) is critical.

Hybrid approaches. Some deployments may benefit from hybrid designs: using point-to-point ADKG for initial key generation (or periodic refresh), then leveraging on-chain infrastructure for key management, access control, and application integration. The theoretical and practical lines of work surveyed in this report are complementary, not competing.

6.6 Open Questions

Several questions remain open for blockchain-assisted DKG:

Asynchronous bulletin boards. Can PBB-based DKG achieve full asynchrony if the bulletin board itself is asynchronous? This requires careful treatment of when messages become "visible" and whether the $O(1)$ communication results of Applebaum and Pinkas extend to this setting.

Cross-chain DKG. Generating keys usable across multiple blockchains introduces new challenges: different timing assumptions, finality properties, and trust models must be reconciled.

Gas efficiency at scale. Current smart-contract DKG implementations are limited to $n \approx 256$ by gas costs. Techniques like recursive SNARKs or off-chain aggregation could extend this, but practical constructions remain to be developed.

Circular dependencies. If a blockchain requires threshold cryptography (e.g., for a randomness beacon or finality gadget), and the threshold keys are generated via smart-contract DKG on that same chain, a bootstrapping problem arises. Breaking this circularity cleanly remains an open design challenge.

7 Conclusion

This report traced the development of Asynchronous Distributed Key Generation from theoretical foundations to practical protocols. The journey spans over three decades: from Pedersen's synchronous DKG (1991), through the recognition that synchrony assumptions are untenable for Internet-scale systems, to the first ADKG constructions in 2020 and their rapid refinement through 2025.

The core challenge of ADKG is compositional: it requires assembling asynchronous primitives (reliable broadcast, binary agreement, and verifiable secret sharing) into a coherent protocol, while simultaneously solving the circular dependency between agreement (which needs randomness) and key generation (which provides randomness). The Eventually Perfect Common Coin abstraction broke this cycle, enabling the first ADKG protocols.

Progress since 2020 has been remarkable. Communication complexity dropped from $O(\kappa n^4)$ to $O(\kappa n^3)$, with 2025 breakthroughs achieving nearly-quadratic $\tilde{O}(n^2)$ communication. High-threshold protocols, essential for applications like SBFT-style consensus, improved from over 100 seconds to under 10 seconds for practical network sizes. Adaptive security—previously thought to require expensive setup assumptions—is now achievable with only silent setup.

The frontier continues to advance. The cubic barrier has fallen; the remaining gap to truly quadratic communication without any setup is the next target. Post-quantum ADKG has emerged: LADKG demonstrates that lattice-based constructions can match classical performance, completing in 45 seconds with 121 nodes. Dynamic participation, where parties join and leave during key generation, remains largely unexplored but would enable truly permissionless systems.

For applications already deployed on blockchain platforms, an alternative path exists. Section 6 surveyed smart-contract-based DKG protocols that leverage existing infrastructure for coordination and communication. While these protocols typically assume synchrony rather than full asynchrony, they offer practical advantages: crypto-economic incentives encourage honest

behaviour, public verifiability enables auditability, and deployment complexity is reduced by building atop existing chains. Production systems like Shutter Network demonstrate that even theoretically suboptimal protocols can provide real-world value. The choice between point-to-point ADKG and blockchain-assisted DKG depends on deployment context: bootstrapping a new system favours the former; building atop an existing chain may favour the latter.

For practitioners, the landscape is increasingly favourable. Multiple ADKG implementations exist with open-source code, benchmarked performance, and documented security proofs. Smart-contract DKG provides a simpler on-ramp for blockchain-native applications. The remaining challenge is integration: embedding these primitives into broader systems—blockchains, threshold wallets, randomness beacons, and privacy-preserving applications. That integration work, as much engineering as cryptography, is the next frontier.

References

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

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

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

[4] Nicolas Alhaddad, Mayank Varia, and Haibin Zhang. Haven++: Batched and packed dual-threshold asynchronous complete secret sharing with applications. IACR Communications in Cryptology, 2024.

[5] Benny Applebaum and Benny Pinkas. Distributing keys and random secrets with constant complexity. In *Theory of Cryptography Conference (TCC 2024)*, volume 15366 of *LNCS*, pages 478–510. Springer, 2024.

[6] Renas Bacho and Alireza Kavousi. SoK: Dlog-based distributed key generation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 614–632, 2025.

[7] Renas Bacho, Daniel Collins, Chen-Da Liu-Zhang, and Julian Loss. Network-agnostic security comes (almost) for free in DKG and MPC. In *Advances in Cryptology – CRYPTO 2023*, pages 71–106. Springer, 2023.

[8] Renas Bacho, Christoph Lenzen, Julian Loss, Simon Ochsenreither, and Dimitrios Papachristoudis. GRandLine: Adaptively secure DKG and randomness beacon with (almost) quadratic communication complexity. IACR ePrint 2023/1887, 2023.

[9] Yongqiang Bai, Debiao He, Zhenguo Yang, Min Luo, and Cong Peng. MDKG: Module-lattice-based distributed key generation. In *Information and Communications Security (ICICS 2025)*, volume 16217 of *LNCS*, pages 217–236. Springer, 2026.

[10] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, pages 27–30, 1983.

[11] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology—CRYPTO 2001*, pages 213–229. Springer, 2001.

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

[13] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

[14] 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*, pages 42–51, 1993.

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

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

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

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

[19] Hao Feng and Qiang Tang. Asymptotically optimal adaptive asynchronous common coin and DKG with silent setup. In *Advances in Cryptology – CRYPTO 2025*, pages 647–680. Springer, 2025.

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

[21] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT'99*, pages 295–310. Springer, 1999.

[22] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure applications of Pedersen's distributed key generation protocol. In *Topics in Cryptology—CT-RSA 2003*, pages 373–390. Springer, 2003.

[23] Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. Olingo: Threshold lattice signatures with DKG and identifiable abort. IACR ePrint 2025/1789, 2025.

[24] Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed key generation in the wild. In *International Conference on Security and Cryptography for Networks*, pages 295–312. Springer, 2012.

[25] 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.

[26] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42, 2016.

[27] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT'91*, pages 522–526. Springer, 1991.

[28] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91*, pages 129–140. Springer, 1991.

[29] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. ETHDKG: Distributed key generation with Ethereum smart contracts. Cryptology ePrint Archive, Report 2019/985, 2019.

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

[31] Michael Sober, Max Kobelt, Giulia Scaffino, Dominik Kaaser, and Stefan Schulte. Distributed key generation with smart contracts using zk-SNARKs. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC)*, pages 229–238. ACM, 2023. doi: 10.1145/3555776.3577677. Best Paper Award, Distributed Systems Track.

[32] Shutter Network Team. Shutter: Distributed key generation with applications to front-running protection. Cryptology ePrint Archive, Report 2024/1981, 2024. Deployed on Gnosis Chain since October 2022.

[33] Linghe Yang, Jian Liu, Jingyi Cui, et al. LADKG: Robust and scalable lattice-based distributed key generation for asynchronous networks. IACR ePrint 2025/1946, 2025.

[34] 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.

[35] Thomas Yurek, Licheng Luo, Jaiden Faez, Aniket Kate, and Andrew Miller. hbACSS: How to robustly share many secrets. In *Network and Distributed System Security Symposium (NDSS)*, 2022.

[36] Haibin Zhang, Sisi Duan, et al. Practical asynchronous distributed key generation: Improved efficiency, weaker assumption, and standard model. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 568–581, 2023.

[37] Liang Zhang, Feiyang Qiu, Feng Hao, and Haibin Kan. 1-round distributed key generation with efficient reconstruction using decentralized CP-ABE. *IEEE Transactions on Information Forensics and Security*, 17:894–907, 2022. doi: 10.1109/TIFS.2022.3152356.