Distributed Key Generation (DKG) in the Discrete-Logarithm Setting

Jonathan Katz
Chief Scientist, Dfns

Sept. 26, 2023
MPTS 2023: NIST Workshop on Multi-Party Threshold Schemes 2023

Thanks to Anna Kaplan and Chelsea Komlo for helpful discussions

Overview of the talk

Part 1

Standardizing DKG protocols as independent primitives

Defining DKG security via a simulation-based approach

Overview of the talk

Part 1

Standardizing DKG protocols as independent primitives

Defining DKG security via a simulation-based approach

Part 2

A (round-optimal) robust DKG protocol in the honest-majority setting

DKG in the dlog setting

Notation

- n is the total number of parties
- t is an upper bound on the number of corrupted parties
- $\mathbb{G}$ is a cyclic group of prime order q, with generator g

DKG in the dlog setting

Notation

- n is the total number of parties
- t is an upper bound on the number of corrupted parties
- $\mathbb{G}$ is a cyclic group of prime order q, with generator g

Goal

Distributed protocol for n parties to generate

DKG in the dlog setting

Notation

- n is the total number of parties
- t is an upper bound on the number of corrupted parties
- $\mathbb{G}$ is a cyclic group of prime order q, with generator g

Goal

Distributed protocol for n parties to generate

- (Common) public key $y = g^x$

DKG in the dlog setting

Notation

- n is the total number of parties
- t is an upper bound on the number of corrupted parties
- $\mathbb{G}$ is a cyclic group of prime order q, with generator g

Goal

Distributed protocol for n parties to generate

- (Common) public key $y = g^x$
- $(t+1)$-out-of-n secret sharing[a] $\{\sigma_i\}_{i=1}^n$ of the private key x
- (Optional) common commitments $\{g^{\sigma_i}\}_{i=1}^n$ to the parties' shares

[a] Assume Shamir secret sharing here, but it could also be n-out-of-n additive sharing.

Applications

A DKG protocol as described could be used for, e.g.,

- Threshold ECDSA, EdDSA/Schnorr, or BLS signing
- Threshold ElGamal decryption

Designing threshold schemes

At a high level, there are two approaches to designing and proving secure a threshold cryptosystem (here taken to be signing for concreteness):

Designing threshold schemes

At a high level, there are two approaches to designing and proving secure a threshold cryptosystem (here taken to be signing for concreteness):

- (Monolithic approach:) Design DKG protocol + signing protocol jointly, and prove security of the combination

Designing threshold schemes

At a high level, there are two approaches to designing and proving secure a threshold cryptosystem (here taken to be signing for concreteness):

- (Monolithic approach:) Design DKG protocol + signing protocol jointly, and prove security of the combination
- (Modular approach:) Design a signing protocol, and prove security when used with any DKG protocol satisfying certain properties

Advantages of a modular approach

Advantages of a modular approach

- Can streamline/simplify security proofs and analysis

Advantages of a modular approach

- Can streamline/simplify security proofs and analysis
- Can use one DKG for multiple threshold protocols

Advantages of a modular approach

- Can streamline/simplify security proofs and analysis
- Can use one DKG for multiple threshold protocols
- Can replace one DKG protocol with another satisfying the same requirements

Simulation-based security

High-level idea

Simulation-based security

High-level idea

Specify the *real-world execution* of some protocol Π

Simulation-based security

High-level idea

Specify the *real-world execution* of some protocol Π

Define an *ideal-world execution* in which honest parties and the adversary interact with some ideal functionality $\mathcal{F}$

Simulation-based security

High-level idea

Specify the *real-world execution* of some protocol Π

Define an *ideal-world execution* in which honest parties and the adversary interact with some ideal functionality $\mathcal{F}$

Π *t-securely realizes* $\mathcal{F}$ if the actions of any adversary corrupting $\leq t$ parties in the real world can be *simulated* by a corresponding adversary in the ideal world

Ideal functionalities for (dlog-based) DKG

There are multiple functionalities one could consider for DKG

We illustrate several possibilities here

Strongest(?) ideal functionality

$$\mathcal{F}_{\mathsf{DKG}}^{t,n}$$

1. Choose $x \leftarrow \mathbb{Z}_q$ and compute $y := g^x$.
2. Compute $\{\sigma_i\}_{i=0}^{n} \leftarrow \mathsf{SS}_t(x)$. For $i \in [n]$, set $y_i := g^{\sigma_i}$; let $Y := (y_1, \ldots, y_n)$.
3. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Strongest(?) ideal functionality

$\mathcal{F}^{t,n}_{\mathsf{DKG}}$

1. Choose $x \leftarrow \mathbb{Z}_q$ and compute $y := g^x$.
2. Compute $\{\sigma_i\}_{i=0}^n \leftarrow \mathsf{SS}_t(x)$. For $i \in [n]$, set $y_i := g^{\sigma_i}$; let $Y := (y_1, \ldots, y_n)$.
3. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Notes

Adversary given (y, Y)

- Those values are public, and are revealed even to an eavesdropping adversary who corrupts no one

Strongest(?) ideal functionality

$\mathcal{F}_{\text{DKG}}^{t,n}$

1. Choose $x \leftarrow \mathbb{Z}_q$ and compute $y := g^x$.
2. Compute $\{\sigma_i\}_{i=0}^n \leftarrow \text{SS}_t(x)$. For $i \in [n]$, set $y_i := g^{\sigma_i}$; let $Y := (y_1, \ldots, y_n)$.
3. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Notes

Adversary given (y, Y)

- Those values are public, and are revealed even to an eavesdropping adversary who corrupts no one

This functionality ensures robustness (aka guaranteed output delivery)

Strongest(?) ideal functionality

$\mathcal{F}_{\text{DKG}}^{t,n}$

1. Choose $x \leftarrow \mathbb{Z}_q$ and compute $y := g^x$.
2. Compute $\{\sigma_i\}_{i=0}^n \leftarrow \mathsf{SS}_t(x)$. For $i \in [n]$, set $y_i := g^{\sigma_i}$; let $Y := (y_1, \ldots, y_n)$.
3. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Notes

Impossible to t-securely realize unless $t < n/2$

Alternate (robust) functionality I

Let adversary choose its own shares

$\mathcal{F}^{t,n}_{\text{DKG}}$

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. Receive $\{\sigma_i\}_{i \in \mathcal{C}}$ from the adversary.
2. Choose $x \leftarrow \mathbb{Z}_q$ and set $y := g^x$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$, set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Alternate (robust) functionality II

Let adversary choose its own shares, depending on y

$\mathcal{F}_{\text{DKG}}^{t,n}$

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. Choose $x \leftarrow \mathbb{Z}_q$ and set $y := g^x$. Send y to the adversary $\mathcal{S}$.
2. Receive $\{\sigma_i\}_{i \in \mathcal{C}}$ from $\mathcal{S}$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$, set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send Y to the adversary.

Non-robust functionality

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

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. Receive $\{\sigma_i\}_{i \in \mathcal{C}}$ from the adversary $\mathcal{S}$.
2. Choose $x \leftarrow \mathbb{Z}_q$ and set $y := g^x$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$ set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. Send (y, Y) to $\mathcal{S}$, who responds with either abort or continue. If abort and $|\mathcal{C}| \geq 1$ then send $\perp$ to all honest parties and stop. Otherwise, for $i \in [n]$ send (y, σ_i, Y) to P_i.

Non-robust functionality

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

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. Receive $\{\sigma_i\}_{i \in \mathcal{C}}$ from the adversary $\mathcal{S}$.
2. Choose $x \leftarrow \mathbb{Z}_q$ and set $y := g^x$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$ set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. Send (y, Y) to $\mathcal{S}$, who responds with either abort or continue. If abort and $|\mathcal{C}| \geq 1$ then send $\perp$ to all honest parties and stop. Otherwise, for $i \in [n]$ send (y, σ_i, Y) to P_i.

Note

Allows $\mathcal{S}$ to bias the public key

Fair (non-robust) functionality

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

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. $\mathcal{S}$ sends either abort or $\{\sigma_i\}_{i \in \mathcal{C}}$. If abort and $|\mathcal{C}| \geq 1$ then send $\perp$ to all honest parties and stop. Otherwise, continue.
2. Choose $x \leftarrow \mathbb{Z}_q$ and set $y := g^x$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$, set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Fair (non-robust) functionality

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

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. $\mathcal{S}$ sends either abort or $\{\sigma_i\}_{i\in\mathcal{C}}$. If abort and $|\mathcal{C}| \geq 1$ then send $\perp$ to all honest parties and stop. Otherwise, continue.
2. Choose $x \leftarrow \mathbb{Z}_q$ and set $y := g^x$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$, set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send (y, Y) to the adversary.

Note

Could also incorporate *identifiable abort*

DKG with shift

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

Let $\mathcal{C}'$ be an arbitrary set of size t with $\mathcal{C} \subseteq \mathcal{C}' \subset [n]$.

1. Choose $x' \leftarrow \mathbb{Z}_q$ and set $y' := g^{x'}$. Send y' to the adversary $\mathcal{S}$.
2. Receive $\{\sigma_i\}_{i \in \mathcal{C}}, \Delta$ from $\mathcal{S}$. Set $x := x' + \Delta$ and $y := g^x$. Choose $\sigma_i \leftarrow \mathbb{Z}_q$ for $i \in \mathcal{C}' \setminus \mathcal{C}$.
3. Let f be the polynomial of degree at most t such that $f(0) = x$ and $f(i) = \sigma_i$ for $i \in \mathcal{C}'$. Set $\sigma_i := f(i)$ for $i \in [n] \setminus \mathcal{C}'$.
4. For $i \in [n]$, set $y_i := g^{\sigma_i}$. Let $Y := (y_1, \ldots, y_n)$.
5. For $i \in [n]$, send (y, σ_i, Y) to P_i. Send Y to the adversary.

Recommendations

- Submissions of threshold (dlog) protocols should modularize the DKG
  - Define required properties of the DKG
  - Prove security of the protocol using a DKG satisfying those properties
  - (Optional) specify a DKG that satisfies those properties

Recommendations

- Submissions of threshold (dlog) protocols should modularize the DKG
  - Define required properties of the DKG
  - Prove security of the protocol using a DKG satisfying those properties
  - (Optional) specify a DKG that satisfies those properties
- Specify required properties for DKG via an ideal functionality
  - Possibly using a common template

Recommendations

- Submissions of threshold (dlog) protocols should modularize the DKG
  - Define required properties of the DKG
  - Prove security of the protocol using a DKG satisfying those properties
  - (Optional) specify a DKG that satisfies those properties
- Specify required properties for DKG via an ideal functionality
  - Possibly using a common template
- In general, encourage submissions not only of gadgets to be used by other protocols, but also of protocols relying on abstract gadgets

A robust DKG protocol

A round-optimal, robust DKG protocol in the honest-majority setting

- Assuming broadcast, synchrony
  - Note: recommend abstracting broadcast channel
- Efficient for small t, n

A robust DKG protocol

A round-optimal, robust DKG protocol in the honest-majority setting

- Assuming broadcast, synchrony
  - Note: recommend abstracting broadcast channel
- Efficient for small t, n

Round optimality

Protocol has one round of preprocessing, followed by a 2-round online phase that can be executed an unbounded number of times

Robust (unbiased) DKG is impossible in one round regardless of prior setup

Motivating robustness

- Most practical applications need robustness (in a broader sense)
- Can potentially achieve by other means, but less efficient (and possibly less secure)
- Robustness is an advantage of working in the honest-majority setting

Background: Pseudorandom secret sharing [CDI05]

Notation

Let $\mathbb{S}_{n-t,n}$ be the collection of all subsets of $[n]$ of size $n-t$

For $S \in \mathbb{S}_{n-t,n}$, let $Z_S \in \mathbb{Z}_q[X]$ be the t-degree polynomial with $Z_S(0) = 1$ and $Z_S(i) = 0$ for $i \in [n] \setminus S$

$F : \{0,1\}^\kappa \times \{0,1\}^n \to \mathbb{Z}_q$ is a pseudorandom function

Background: Pseudorandom secret sharing [CDI05]

Notation

Let $\mathbb{S}_{n-t,n}$ be the collection of all subsets of $[n]$ of size $n-t$

For $S \in \mathbb{S}_{n-t,n}$, let $Z_S \in \mathbb{Z}_q[X]$ be the t-degree polynomial with $Z_S(0) = 1$ and $Z_S(i) = 0$ for $i \in [n] \setminus S$

$F : \{0,1\}^\kappa \times \{0,1\}^n \to \mathbb{Z}_q$ is a pseudorandom function

Assume for all $S \in \mathbb{S}_{n-t,n}$ and all $i \in S$, party P_i holds $k_S \in \{0,1\}^\kappa$

Given a nonce $N \in \{0,1\}^n$, each party P_i can compute share

$$\sigma_i := \sum_{S \in \mathbb{S}_{n-t,n} : i \in S} F_{k_S}(N) \cdot Z_S(i)$$

This is a $(t+1)$-out-of-n Shamir secret sharing of

$$x_N = \sum_{S \in \mathbb{S}_{n-t,n}} F_{k_S}(N) \cdot Z_S(0) = \sum_{S \in \mathbb{S}_{n-t,n}} F_{k_S}(N)$$

Background: Pseudorandom secret sharing (PRSS)

Notes

PRSS is not DKG (still need to interact to compute $y = g^{x_N}$)

PRSS typically assumes a trusted dealer; without a trusted dealer, it is not clear how to ensure correctness

A robust DKG protocol

Preprocessing: For $S \in \mathbb{S}_{n-t,n}$, a designated party in S chooses $k_S \leftarrow \mathbb{Z}_q$ and sends it to $\{P_i\}_{i \in S}$. Each P_i lets $k_{i,S}$ be the value received

A robust DKG protocol

Preprocessing: For $S \in \mathbb{S}_{n-t,n}$, a designated party in S chooses $k_S \leftarrow \mathbb{Z}_q$ and sends it to $\{P_i\}_{i \in S}$. Each P_i lets $k_{i,S}$ be the value received

Key generation: Given nonce N, each party P_i does:

- Round 1: For all $S \in \mathbb{S}_{n-t,n}$ with $i \in S$: compute $\hat{y}_{i,S} := g^{F_{k_{i,S}}(N)}$ and $h_{i,S} := H(\hat{y}_{i,S})$; then broadcast $h_{i,S}$

A robust DKG protocol

Preprocessing: For $S \in \mathbb{S}_{n-t,n}$, a designated party in S chooses $k_S \leftarrow \mathbb{Z}_q$ and sends it to $\{P_i\}_{i \in S}$. Each P_i lets $k_{i,S}$ be the value received

Key generation: Given nonce N, each party P_i does:

- Round 1: For all $S \in \mathbb{S}_{n-t,n}$ with $i \in S$: compute $\hat{y}_{i,S} := g^{F_{k_{i,S}}(N)}$ and $h_{i,S} := H(\hat{y}_{i,S})$; then broadcast $h_{i,S}$
- Round 2: Initialize $\mathcal{I} := \emptyset$. For each $S \in \mathbb{S}_{n-t,n}$, do:

  If there is a value h_S s.t. $h_{j,S} = h_S$ for all $j \in S$, add S to $\mathcal{I}$.

  Broadcast $\{\hat{y}_{i,S}\}_{S \in \mathcal{I} : i \in S}$

A robust DKG protocol

Preprocessing: For $S \in \mathbb{S}_{n-t,n}$, a designated party in S chooses $k_S \leftarrow \mathbb{Z}_q$ and sends it to $\{P_i\}_{i \in S}$. Each P_i lets $k_{i,S}$ be the value received

Key generation: Given nonce N, each party P_i does:

- Round 1: For all $S \in \mathbb{S}_{n-t,n}$ with $i \in S$: compute $\hat{y}_{i,S} := g^{F_{k_{i,S}}(N)}$ and $h_{i,S} := H(\hat{y}_{i,S})$; then broadcast $h_{i,S}$
- Round 2: Initialize $\mathcal{I} := \emptyset$. For each $S \in \mathbb{S}_{n-t,n}$, do:

  If there is a value h_S s.t. $h_{j,S} = h_S$ for all $j \in S$, add S to $\mathcal{I}$.

  Broadcast $\{\hat{y}_{i,S}\}_{S \in \mathcal{I} \,:\, i \in S}$
- Output determination: For $S \in \mathcal{I}$, if any P_j broadcasted $\hat{y}_{j,S}$ with $H(\hat{y}_{j,S}) = h_S$, set $\hat{y}_S := \hat{y}_{j,S}$. Then:
  1. Set $\sigma_i := \sum_{S \in \mathcal{I} \,:\, i \in S} F_{k_{i,S}}(N) \cdot Z_S(i)$
  2. Set $y := \prod_{S \in \mathcal{I}} \hat{y}_S$, and for $j \in [n]$ set $y_j := \prod_{S \in \mathcal{I} \,:\, j \in S} \hat{y}_S^{Z_S(j)}$

A robust DKG protocol

Theorem

Let F be a pseudorandom function, and model H as a random oracle. Then for $t < n/2$ this protocol t-securely realizes $\mathcal{F}^{t,n}_{\mathsf{DKG}}$

Easy to modify to achieve adaptive security as well

Paper available at `https://eprint.iacr.org/2023/1094`

We would be interested in collaborating on a submission to NIST

Thank you!