Recent Developments on Multi-Party Schnorr Signatures

Elizabeth Crites
University of Edinburgh

Nov. 24, 2022

Why Schnorr?

- RSA, ECDSA, EdDSA (Aug.’22) standardized through NIST
- EdDSA is deterministic version of Schnorr
- RSA signatures are large (~6x ECDSA/EdDSA)
- ECDSA requires nonce inversion and other complexities
    - no security reduction like Schnorr -> DL + ROM
- BLS requires bilinear pairings (no NIST standardization)

Why now?

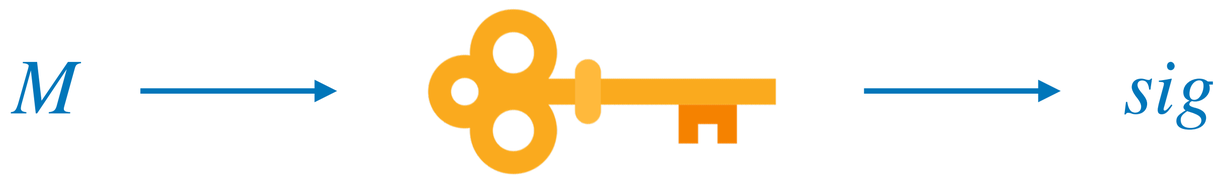
- Bitcoin moved from ECDSA to Schnorr (Nov.'21)
- MuSig2 [NRS21] / FROST [KG20] proposed to secure cryptocurrency wallets
- FROST IETF draft [CKGW22], 9+ implementations
- NIST call for threshold EdDSA/Schnorr threshold signatures [BD22]
    - EdDSA not verifiably deterministic so Schnorr can be used instead

Roadmap

- Background on threshold cryptography
- Security of Schnorr signatures
- Classic solutions for multi-party Schnorr
- Recent broken solutions
- Recent secure solutions
- Future directions

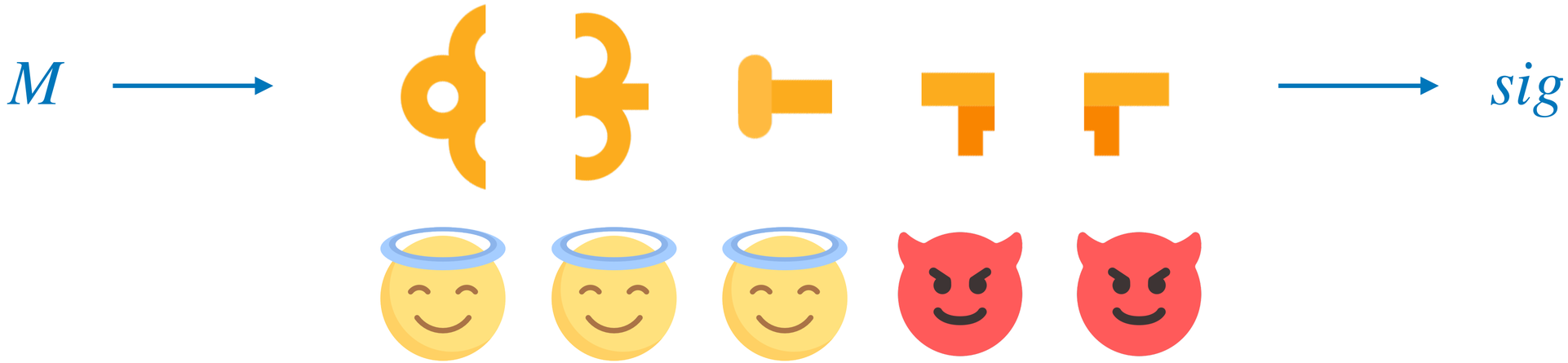
Threshold Cryptography

- introduced by [D87, DF89]
- secret key enables signatures, decryption, etc.
    - single point of failure
- idea: distribute the secret key among several parties
    - some fraction may be corrupt



Threshold Cryptography

- distribute the secret key via:
  - trusted key generation (Shamir secret sharing)
  - distributed key generation (DKG)



Shamir Secret Sharing [Sha79]

- To share a secret s, the dealer samples random $a_i \xleftarrow{\$} \mathbb{Z}_p$ and sets:

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

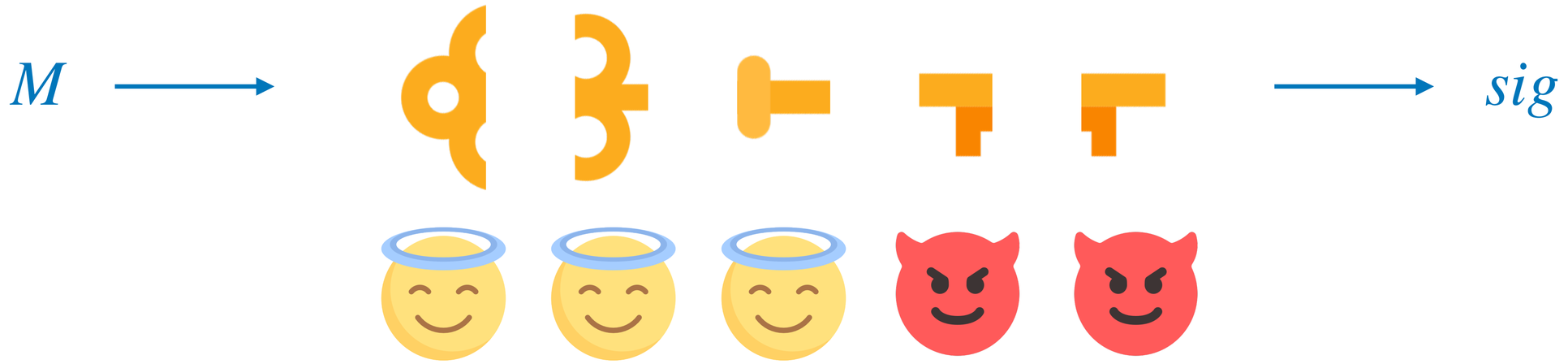
- t points uniquely define this polynomial, but $t-1$ reveal nothing about s
- The dealer privately sends share $s_i = f(i)$ to each party P_i, $i = 1, \ldots, n$
- For $|\mathcal{S}| \geq t$, it holds that:

$$s = \Sigma_{i \in \mathcal{S}} \, \lambda_i^{\mathcal{S}} \cdot s_i$$

where $\lambda_i^{\mathcal{S}} = \Pi_{j \in \mathcal{S}, j \neq i} \, \frac{j}{j - i}$

Linear Solution

- simply pick t out of n signatures (linear in n)
- what Bitcoin was doing for Multisig addresses previously
- verifier knows policy and signers
    - privacy: verifier should only see final signature under aggregate public key



Schnorr Signatures [Sch91]

- To generate a key pair (PK, sk), sample $sk \xleftarrow{\$} \mathbb{Z}_p$ and set $PK \leftarrow g^{sk}$
- To sign a message M,
  - sample nonce $r \xleftarrow{\$} \mathbb{Z}_p$ and set $R \leftarrow g^r$
  - $c \leftarrow H(PK, M, R)$
  - $z \leftarrow r + c \cdot sk$
  - output $sig = (R, z)$

$$R \cdot PK^c \stackrel{?}{=} g^z$$

Unforgeability of Signatures [GMR88]

Signing Oracle knows sk

M_i

σ_i

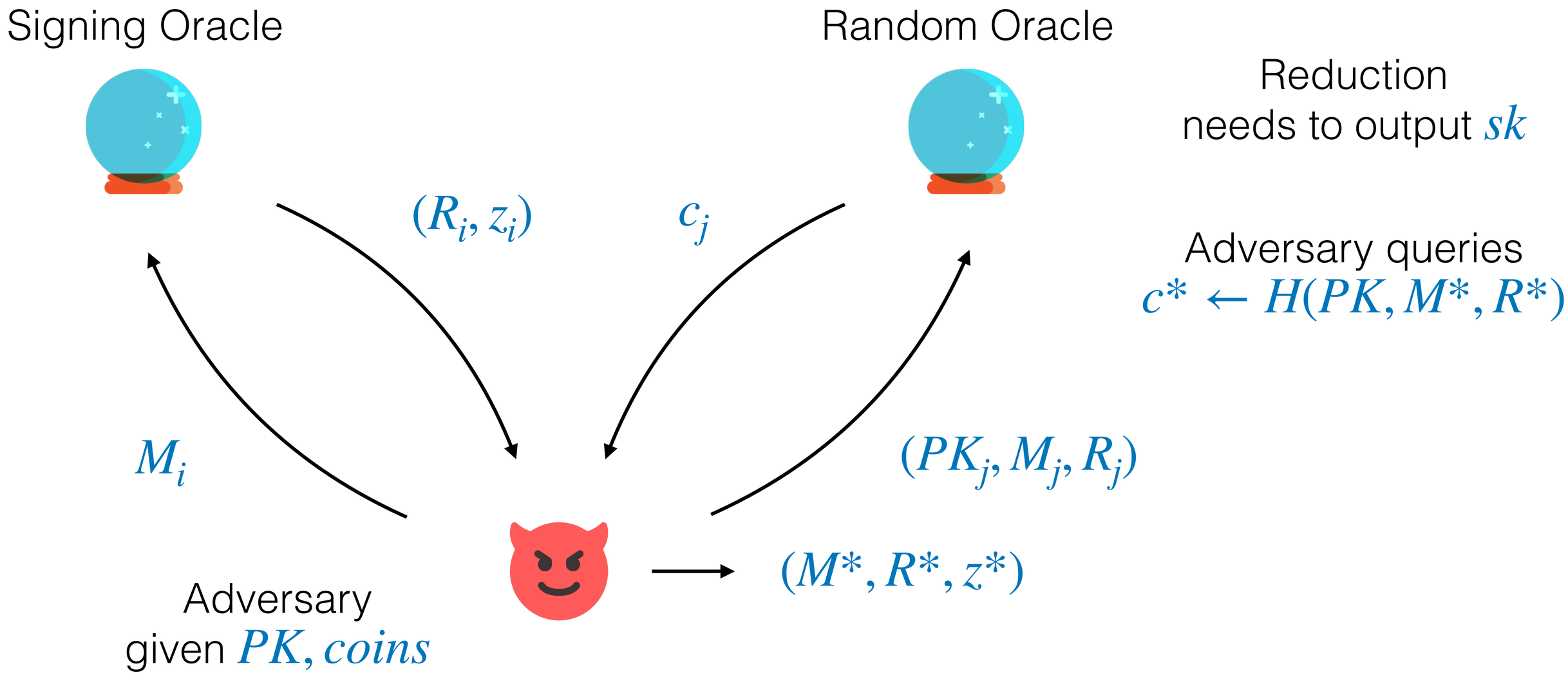
Adversary given PK

(M^*, σ^*)

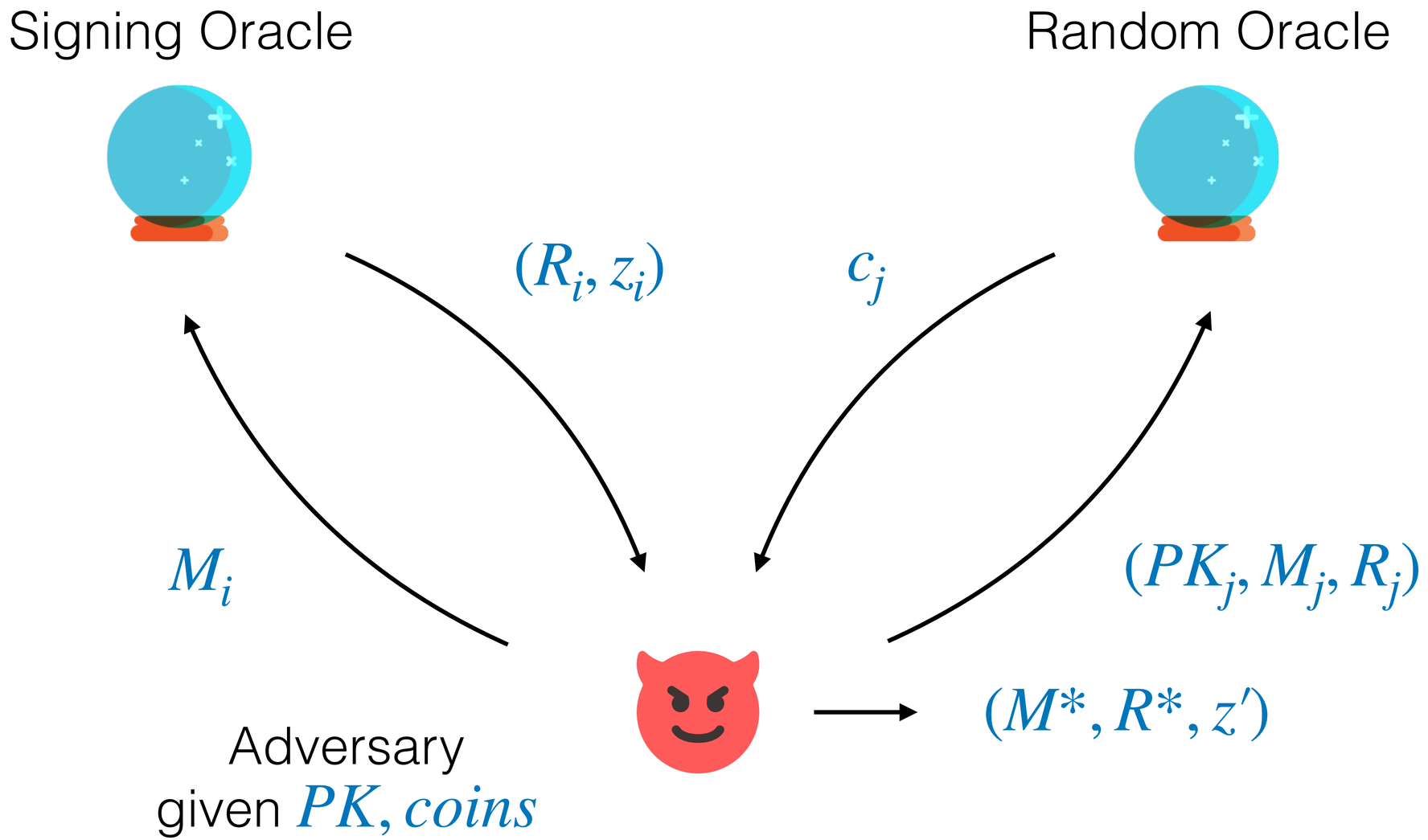
Adversary wins if:

1. $\mathsf{Verify}(PK, M^*, \sigma^*) = 1$
2. $M^* \neq M_i$ for any i

Security of Schnorr Signatures [PS00]



Security of Schnorr Signatures - Rewind



Reduction
needs to output sk

Adversary queries
$c' \leftarrow H(PK, M^*, R^*)$

Reduction programs
$c' \neq c^*$

$$R^* PK^{c^*} = g^{z^*}$$
$$R^* PK^{c'} = g^{z'}$$

Multi-Party Schnorr Signatures

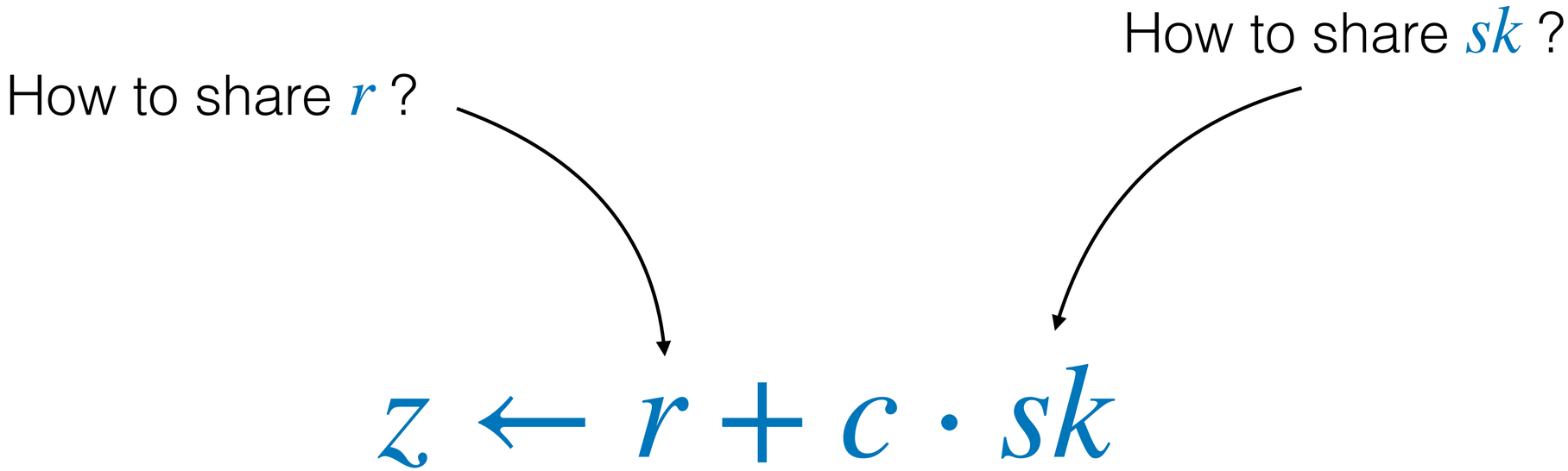
Key Generation: Multi- vs. Threshold

- rogue key attack
  - corrupted signer sets its public key to $PK_1 = g^{sk_1} \left(\Pi_2^n PK_i\right)^{-1}$
  - proofs of possession of sk_i allow $PK = PK_1 \cdots PK_n$
- multisignatures (n,n)
  - pros: non-interactive key generation & aggregation
  - special key aggregation techniques or proofs of possession
- threshold signatures (t,n)
  - trusted key generation or DKG

Multi-Party Schnorr

How to share r ?

How to share sk ?



$$z \leftarrow r + c \cdot sk$$

Idea #1 for Threshold Schnorr

- run Shamir or DKG for secret key sk
- run Shamir for nonce r
    - trusted for *every* signature
- run DKG for nonce r [SS01]
    - needs to be run for *every* message
    - most DKGs are multiple rounds

What do we want?

- fewer rounds to produce nonce r
- output standard Schnorr signature
- concurrent security
    - can open many sessions with the same signer at once

Idea #2

- PK output by DKG
- To sign a message M, party P_i
  - Round 1: samples $r_i \xleftarrow{\$} \mathbb{Z}_p$ and outputs $R_i \leftarrow g^{r_i}$
  - Round 2: computes
    - $R = \Pi_{i \in \mathcal{S}} R_i$
    - $c \leftarrow H(PK, M, R)$
    - $z_i \leftarrow r_i + c \cdot \lambda_i^{\mathcal{S}} \cdot sk_i$
    - outputs $sig_i = (R_i, z_i)$

$$z = \Sigma_{i \in \mathcal{S}} z_i$$

$$sig = (R, z)$$

$$R \cdot PK^c \stackrel{?}{=} g^z$$

Idea #2 ❌

- PK output by DKG
- To sign a message M, party P_i
  - Round 1: samples $r_i \xleftarrow{\$} \mathbb{Z}_p$ and outputs $R_i \leftarrow g^{r_i}$
  - Round 2: computes
    - $R = \Pi_{i \in \mathcal{S}} R_i$
    - $c \leftarrow H(PK, M, R)$
    - $z_i \leftarrow r_i + c \cdot \lambda_i^{\mathcal{S}} \cdot sk_i$
    - outputs $sig_i = (R_i, z_i)$

$$z = \Sigma_{i \in \mathcal{S}} z_i$$

$$sig = (R, z)$$

$$R \cdot PK^c \stackrel{?}{=} g^z$$

ROS Attacks

- ROS problem originally stated in [Sch91]
- first identified concurrency issue for 2-party Schnorr [NKDM03]
- [DEFKLNS19] show how to break unforgeability in sub-exponential time
- confirmed polynomial-time attack by [BLLOR21]
- concurrent attack:
    - adversary opens multiple signing sessions at once
    - see honest nonces first and makes its nonce a function of them
    - forges signature

ROS Attacks

- multisignatures: [STVWJGGKF16, MPSW18a]
- threshold signatures: [GJKR07, KG20a]
- blind signatures: [PS00, Sch01, AO00, ZGP17, GPZZ19]
- most did not claim concurrent security

ROS Attack Toy Example

P_1 ... P_1

$R_1^{(1)}$ $R_1^{(k)}$

$M^{(1)}, R_2^{(1)}$ $M^{(k)}, R_2^{(k)}$

$c^{(1)} = H(PK, M^{(1)}, R_1^{(1)} R_2^{(1)})$ $c^{(k)} = H(PK, M^{(k)}, R_1^{(k)} R_2^{(k)})$

$z_1^{(1)} = r_1^{(1)} + c^{(1)} \cdot sk_1$ $z_1^{(k)} = r_1^{(k)} + c^{(k)} \cdot sk_1$

Forgery

Adversary sets $R^* = R_1^{(1)} \cdots R_1^{(k)}$

and uses Wagner's algorithm [Wag02] to find $R_2^{(1)}, \ldots, R_2^{(k)}, M^{(1)}, \ldots, M^{(k)}, M^*$ such that:

$$\overbrace{H(PK, M^{(1)}, R_1^{(1)} R_2^{(1)})}^{c^{(1)}} + \cdots + \overbrace{H(PK, M^{(k)}, R_1^{(k)} R_2^{(k)})}^{c^{(k)}}$$

$$= \underbrace{H(PK, M^*, R^*)}_{c^*}$$

$k+1$-sum problem (generalized birthday attack)

Forgery

$$R^* = R_1^{(1)} \cdots R_1^{(k)}$$

$$\begin{aligned} z_1^* &= z_1^{(1)} + \cdots + z_1^{(k)} \\ &= \underbrace{(r_1^{(1)} + \cdots + r_1^{(k)})}_{r^*} + \underbrace{(c^{(1)} + \cdots + c^{(k)})}_{c^*} \cdot sk_1 \end{aligned}$$

$$z^* = z_1^* + c^* \cdot sk_2 = r^* + c^* \cdot (sk_1 + sk_2)$$

Note: can be extended to M^* of your choosing

which is a valid forgery (M^*, R^*, z^*) under $PK = PK_1 \cdot PK_2$

Idea #3: 3 Rounds (SimpleTSig / SimpleMuSig)

- To sign a message M, party P_i
  - Round 1: samples $r_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $R_i \leftarrow g^{r_i}$, outputs $cm_i \leftarrow H'(R_i)$

$M, \mathcal{S} \rightarrow$
  - Round 2: outputs R_i
  - Round 3: computes
    - $R = \Pi_{i \in \mathcal{S}} R_i$
    - $c \leftarrow H(PK, M, R)$
    - $z_i \leftarrow r_i + c \cdot \lambda_i^{\mathcal{S}} \cdot sk_i$
    - outputs $sig_i = (R_i, z_i)$

$z = \Sigma_{i \in \mathcal{S}} z_i$

$sig = (R, z)$

$R \cdot PK^c \stackrel{?}{=} g^z$

Secure Against ROS Attacks

P_1

$H'(R_2^{(1)})$ ⟶

$R_1^{(1)}$

$M^{(1)}, R_2^{(1)}$

$c^{(1)} = H(PK, M^{(1)}, R_1^{(1)} R_2^{(1)})$

$z_1^{(1)} = r_1^{(1)} + c^{(1)} \cdot sk_1$

…

P_1

$H'(R_2^{(k)})$ ⟶

$R_1^{(k)}$

$M^{(k)}, R_2^{(k)}$

$c^{(k)} = H(PK, M^{(k)}, R_1^{(k)} R_2^{(k)})$

$z_1^{(k)} = r_1^{(k)} + c^{(k)} \cdot sk_1$

“How To Prove Schnorr Assuming Schnorr”

Elizabeth Crites, Chelsea Komlo, Mary Maller [CKM21]

- 3-round multisignature **SimpleMuSig** (with proof of possession of key)
    - first proof of MSDL-PoP [BDN18]
- 3-round threshold signature **SimpleTSig**

Idea #4: 2 Nonces (FROST)

- To sign a message M, party P_i
  - Round 1: samples $r_i, s_i \xleftarrow{\$} \mathbb{Z}_p$, sets $R_i \leftarrow g^{r_i}$, $S_i \leftarrow g^{s_i}$, and outputs R_i, S_i

$M, \mathcal{S} \rightarrow$
  - Round 2: computes
    - $a_i \leftarrow H'(i, PK, M, \{R_i, S_i\}_{i \in \mathcal{S}})$
    - $R = \Pi_{i \in \mathcal{S}} R_i \cdot S_i^{a_i}$
    - $c \leftarrow H(PK, M, R)$
    - $z_i \leftarrow r_i + a_i \cdot s_i + c \cdot \lambda_i^{\mathcal{S}} \cdot sk_i$
    - outputs $sig_i = (\tilde{R}_i, z_i)$

$z = \Sigma_{i \in \mathcal{S}} z_i$

$sig = (R, z)$

$R \cdot PK^c \stackrel{?}{=} g^z$

Secure Against ROS Attacks

$$H(PK, \overbrace{M^{(1)}, R_1^{(1)}(S_1^{(1)})^{a_1^{(1)}} \cdot R_2^{(1)}(S_2^{(1)})^{a_2^{(1)}}}^{c^{(1)}}) + \cdots + H(PK, \overbrace{M^{(k)}, R_1^{(k)}(S_1)^{a_1^{(k)}} R_2^{(k)}(S_2)^{a_2^{(k)}}}^{c^{(k)}})$$

$$= \underbrace{H(PK, M^*, R^*)}_{c^*}$$

$$a_1^{(1)} = H'(1, PK, M^{(1)}, R_1^{(1)}, S_1^{(1)}, R_2^{(1)}, S_2^{(1)})$$

$$R^* = R_1^{(1)}(S_1^{(1)})^{a_1^{(1)}} \cdots R_1^{(k)}(S_1^{(k)})^{a_1^{(k)}}$$

Note: single nonce $(R_1^{(1)})^{a_1^{(1)}}$ fails, can cancel out $a_1^{(1)}$

Idea #4: 2 Nonces (FROST2 / SpeedyMuSig)

- To sign a message M, party P_i
  - Round 1: samples $r_i, s_i \xleftarrow{\$} \mathbb{Z}_p$, sets $R_i \leftarrow g^{r_i}$, $S_i \leftarrow g^{s_i}$, and outputs R_i, S_i

$M, \mathcal{S} \rightarrow$
  - Round 2: computes
    - $a \leftarrow H'(PK, M, \{R_i, S_i\}_{i \in \mathcal{S}})$
    - $R = \Pi_{i \in \mathcal{S}} R_i \cdot S_i^a$
    - $c \leftarrow H(PK, M, R)$
    - $z_i \leftarrow r_i + a \cdot s_i + c \cdot \lambda_i^{\mathcal{S}} \cdot sk_i$
    - outputs $sig_i = (\tilde{R}_i, z_i)$

$z = \Sigma_{i \in \mathcal{S}} z_i$

$sig = (R, z)$

$R \cdot PK^c \stackrel{?}{=} g^z$

Idea #4: 2 Nonces (MuSig2)

- $PK = PK_1^{d_1} \cdots PK_n^{d_n}$ where $d_i = H''(PK_i, \{PK_1, \ldots, PK_n\})$
  - Round 1: samples $r_i, s_i \xleftarrow{\$} \mathbb{Z}_p$, sets $R_i \leftarrow g^{r_i}$, $S_i \leftarrow g^{s_i}$, and outputs R_i, S_i

$M \rightarrow$
  - Round 2: computes
    - $a \leftarrow H'(PK, M, R_1 \cdots R_n, S_1 \cdots S_n)$
    - $R = \Pi_1^n \, R_i \cdot S_i^a$
    - $c \leftarrow H(PK, M, R)$
    - $z_i \leftarrow r_i + a \cdot s_i + c \cdot sk_i$
    - outputs $sig_i = (\tilde{R}_i, z_i)$

$z = \Sigma_1^n \, z_i$

$sig = (R, z)$

$R \cdot PK^c \stackrel{?}{=} g^z$

“How To Prove Schnorr Assuming Schnorr”

Elizabeth Crites, Chelsea Komlo, Mary Maller [CKM21]

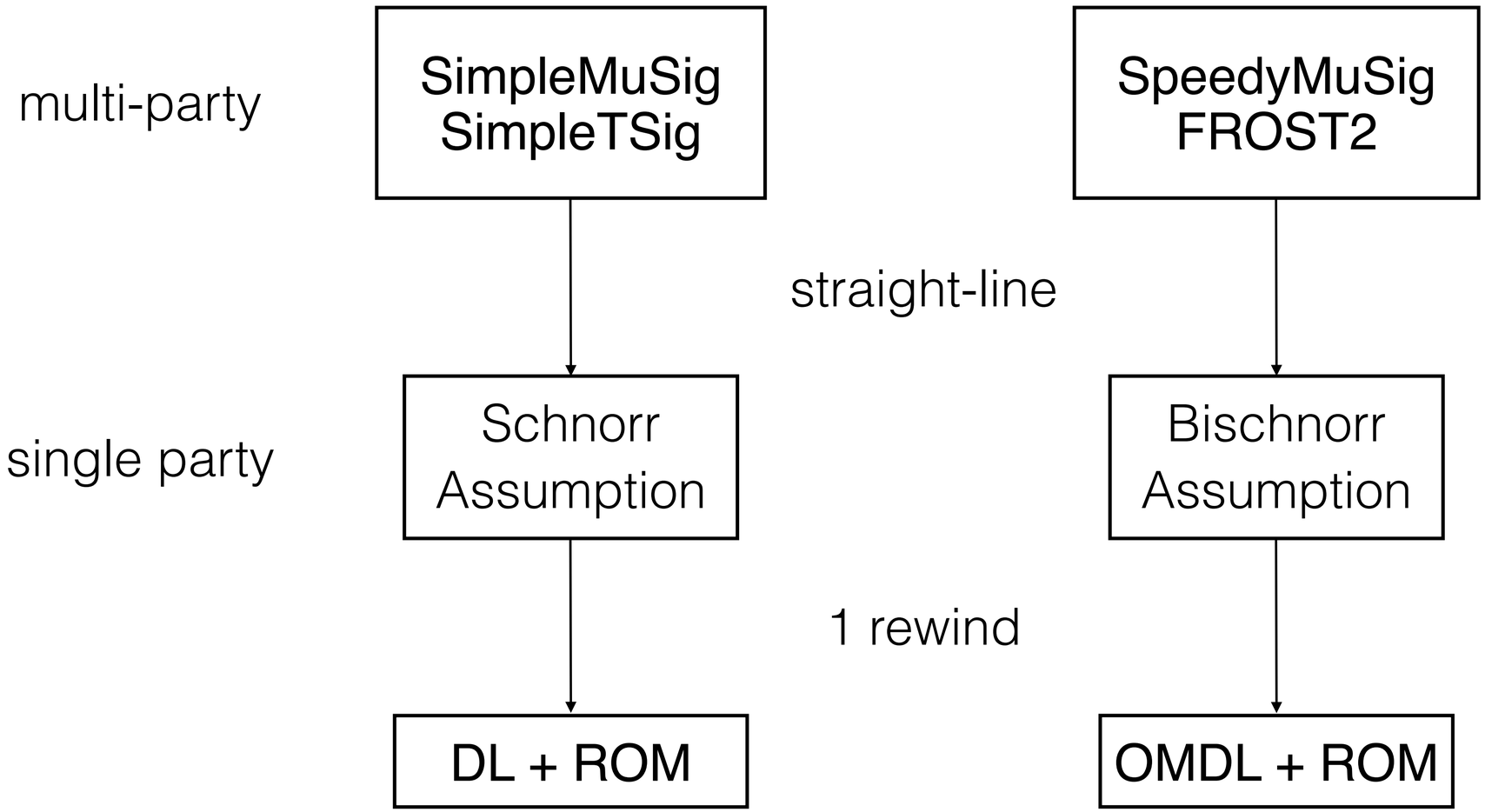
- 3-round multisignature **SimpleMuSig** (with proof of possession of key)
- 3-round threshold signature **SimpleTSig**
- 2-round multisignature **SpeedyMuSig** (with proof of possession of key)
- 2-round threshold signature **FROST2**
  - reduces number of exponentiations from t to 1
  - original proof of FROST [KG20] relied on heuristic assumptions
- new proving framework (all are concurrently secure)

Proving the Security of Multi-Party Schnorr Signatures

Proving the Security of Multi-Party Schnorr

- Security reductions for multi-party signatures have two moving parts:
    - simulating honest users interacting with the adversary
    - extracting a solution to some hard problem from the adversary's responses
- Idea: separate the two parts for a more modular reduction

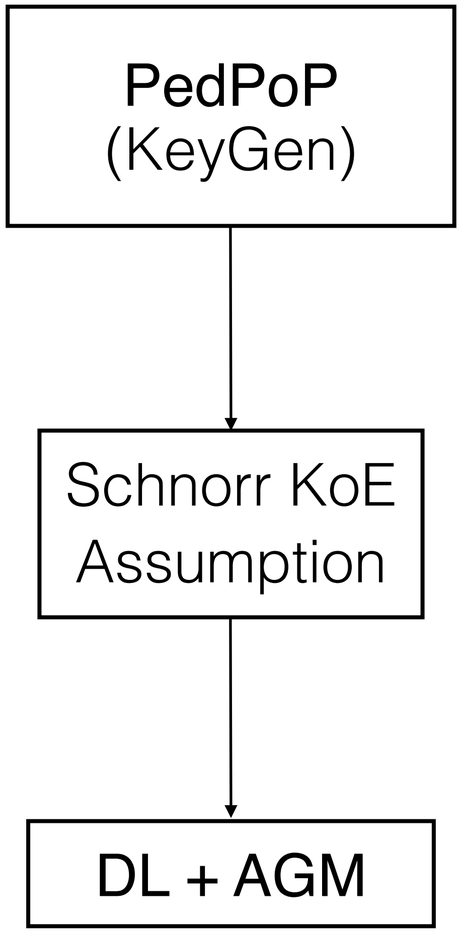
Proving the Security of Multi-Party Schnorr



PedPoP Distributed Key Generation

- proposed in [KG20]
- PedPoP = Pedersen DKG + Proofs of Possession (PoP)
- proofs of possession are themselves Schnorr signatures
- Knowledge of Exponent assumption (KoE)
    - for simplicity, instead of adding rounds
- allows up to $t - 1$ corrupt signers (no honest majority)

Proving the Security of Multi-Party Schnorr + DKG



Unforgeability of Threshold Signatures

- standard notion: forgery (M^*, σ^*) is trivial if even one party signs M^*
- hierarchy of security notions for (partially) non-interactive schemes [BTZ22]
    - fully non-interactive, or
    - *partially* non-interactive:1 pre-processing round + 1 signing round
- covers BLS but not ECDSA (multiple pre-processing rounds)
- multiple proofs of FROST1/2 [BCKMTZ22]
- separation in security achieved by FROST1 vs. FROST2
- FROST1/2 achieve some of the highest notions of security

Future Directions

- unlinkability
- robustness
    - ROAST: Robust Asynchronous Schnorr Threshold signatures [RRJSS22]
- NIST threshold EdDSA/Schnorr requirements:
    - strong unforgeability (shown for FROST1/2 [BTZ22])
    - adaptive security

Adaptive Security

- almost all threshold signatures in the literature are proven under *static* corruption
    - adversary has to pick corrupt parties before protocol begins
- stronger *adaptive* adversary can choose to corrupt parties at any time
    - upon corruption, must reveal secret key and state
        - easy to simulate key generation when you know corrupt parties a priori
        - hard to simulate key generation and signing when you do not

Thank you!

References

[AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. CRYPTO 2000.

[BCKMTZ22] M. Bellare, E. Crites, C. Komlo, M. Maller, S. Tessaro, and C. Zhu. Better than advertised security for non-interactive threshold signatures. CRYPTO 2022.

[BD22] L. Brandao and M. Davidson. Notes on Threshold EdDSA/Schnorr Signatures. https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8214B.ipd.pdf. 2022.

[BDN18] D. Boneh, M. Drijvers, G. Neven. Compact Multi-signatures for Smaller Blockchains. ASIACRYPT 2018.

[BLLOR21] F. Benhamouda, T. Lepoint, J. Loss, M. Orru, and M. Raykova. On the (in)security of ROS. EUROCRYPT 2021.

[BTZ22] M. Bellare, S. Tessaro, C. Zhu. Stronger Security for Non-Interactive Threshold Signatures: BLS and FROST. ePrint 2022/833

[CKGW22] D. Connolly, C. Komlo, I. Goldberg, and C. Wood. Two-Round Threshold Schnorr Signatures with FROST. 2022. url: https://datatracker.ietf.org/doc/draft- irtf-cfrg-frost/.

[CKM21] E. Crites, C. Komlo, and M. Maller. How to prove Schnorr assuming Schnorr: Security of multi- and threshold signatures. ePrint: https://eprint.iacr.org/2021/1375.

[D87] Y. Desmedt. Society and group oriented cryptography: A new concept. CRYPTO 1987.

References

[DEFKLS19] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, I. Stepanovs. On the Security of Two-Round Multi-Signatures. SP 2019.

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. CRYPTO 1989.

[GJKR07] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. JoC 2007.

[GMR88] S. Goldwasser, S. Micali, R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 1988.

[GPZZ19] P. Grontas, A. Pagourtzis, A. Zacharakis, B. Zhang. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. FC 2018 Workshops.

[KG20a] C. Komlo, I. Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. https://crysp.uwaterloo.ca/software/frost/frost-extabs.pdf Jan. 7, 2020.

[KG20] C. Komlo and I. Goldberg. Frost: flexible round-optimized schnorr threshold signatures. SAC 2020.

[MPSW18a] G. Maxwell, A. Poelstra, Y. Seurin, P. Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. DESI 2019.

[NKDM03] A. Nicolosi, M. N. Krohn, Y. Dodis, D. Mazières. Proactive Two-Party Signatures for User Authentication. NDSS 2003.

References

[NRS21] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple Two-Round Schnorr Multi- signatures. CRYPTO 2021.

[PS00] D. Pointcheval, J. Stern. Security Arguments for Digital Signatures and Blind Signatures. JoC 2000.

[RRJSS22] T. Ruffing, V. Ronge, E. Jin, J. Schneider-Bensch, D. Schröder. ROAST: Robust Asynchronous Schnorr Threshold Signatures. CCS 2022.

[Sch91] C.-P. Schnorr. Efficient Signature Generation by Smart Cards. JoC 1991.

[Sch01] C.-P. Schnorr. Security of blind discrete log signatures against interactive attacks. ICICS 2001.

[Sha79] A. Shamir. How to Share a Secret. Commun. ACM 1979.

[SS01] D. R. Stinson and R. Strobl. Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates. ACISP 2001.

[STVWJGGKF16] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, B. Ford. Keeping Authorities “Honest or Bust” with Decentralized Witness Cosigning. S&P 2016.

[Wag02] D. Wagner. A Generalized Birthday Problem. CRYPTO 2002.

[ZGP17] A. Zacharakis, P. Grontas, A. Pagourtzis. Conditional blind signatures. ePrint 2017/682.