Threshold Schnorr with Stateless Deterministic Signing

François Garillot, **Yash**vanth Kondi, Payman Mohassel, Valeria Nikolaenko
Novi/Facebook | Northeastern University | Facebook | Novi/Facebook

Schnorr: Practical Issues

$\text{SchnorrSign}(\mathsf{sk}, m):$

$$r \leftarrow \mathbb{Z}_q$$

$$R = r \cdot G$$

$$e = H(R \| m)$$

$$s = r - \mathsf{sk} \cdot e$$

$$\sigma = (s, e)$$

$$\text{output } \sigma$$

Schnorr: Practical Issues

$$
\begin{aligned}
&\mathrm{SchnorrSign}(\mathsf{sk}, m):\\
&\qquad \boxed{r \leftarrow \mathbb{Z}_q}\\
&\qquad R = r \cdot G\\
&\qquad e = H(R \| m)\\
&\qquad s = r - \mathsf{sk} \cdot e\\
&\qquad \sigma = (s, e)\\
&\qquad \text{output } \sigma
\end{aligned}
$$

Fresh randomness needed to sign on **every invocation**

Schnorr: Practical Issues

$$
\begin{aligned}
&\mathrm{SchnorrSign}(\mathsf{sk}, m):\\
&\quad \boxed{r \leftarrow \mathbb{Z}_q}\\
&\quad R = r \cdot G\\
&\quad e = H(R \| m)\\
&\quad s = r - \mathsf{sk} \cdot e\\
&\quad \sigma = (s, e)\\
&\quad \text{output } \sigma
\end{aligned}
$$

Fresh randomness needed to sign on **every invocation**

Even a tiny amount of bias can completely wreck security

Schnorr: Practical Issues

$$
\begin{aligned}
&\mathrm{SchnorrSign}(\mathsf{sk}, m):\\
&\qquad \boxed{r \leftarrow \mathbb{Z}_q}\\
&\qquad R = r \cdot G\\
&\qquad e = H(R \| m)\\
&\qquad s = r - \mathsf{sk} \cdot e\\
&\qquad \sigma = (s, e)\\
&\quad \text{output } \sigma
\end{aligned}
$$

Fresh randomness needed to sign on **every invocation**

Even a tiny amount of bias can completely wreck security

In practice: bad PRGs, software bugs, etc. Reliable entropy is scarce!

Schnorr: Practical Issues

$\mathrm{SchnorrSign}(\mathsf{sk}, m):$

$$\boxed{r \leftarrow \mathbb{Z}_q}$$
$$R = r \cdot G$$
$$e = H(R \| m)$$
$$s = r - \mathsf{sk} \cdot e$$
$$\sigma = (s, e)$$
$$\text{output } \sigma$$

Fresh randomness needed to sign on **every invocation**

Even a tiny amount of bias can completely wreck security

In practice: bad PRGs, software bugs, etc. Reliable entropy is scarce!

Solution: de-randomize *r*

Naive Derandomization

- Canonical solution is via a Pseudorandom Generator (PRG) - invoke for each new nonce
- However the state of the PRG must be updated reliably—security is very sensitive to this
- This creates a new practical hurdle, eg. state is usually backed up on secure storage where frequent reliable updates may not be possible
- We therefore require derandomization to be **stateless**

Deterministic Signing

$$
\begin{aligned}
&\mathrm{DetSign}(\mathsf{sk}, k, m):\\
&\qquad r = \mathsf{F}_k(m)\\
&\qquad R = r \cdot G\\
&\qquad e = H(R \| m)\\
&\qquad s = r - \mathsf{sk} \cdot e\\
&\qquad \sigma = (s, e)\\
&\quad \text{output } \sigma
\end{aligned}
$$

Deterministic Signing

Sampled during key generation

$$\mathrm{DetSign}(\mathsf{sk}, \boxed{k}, m):$$

$$r = \mathsf{F}_k(m)$$

$$R = r \cdot G$$

$$e = H(R \| m)$$

$$s = r - \mathsf{sk} \cdot e$$

$$\sigma = (s, e)$$

$$\text{output } \sigma$$

Deterministic Signing

$\mathrm{DetSign}(\mathsf{sk}, \boxed{k}, m):$

$\boxed{r = \mathsf{F}_k(m)}$

$R = r \cdot G$

$e = H(R \| m)$

$s = r - \mathsf{sk} \cdot e$

$\sigma = (s, e)$

output σ

Sampled during key generation

F is a pseudorandom function
eg. AES, or SHA as in EdDSA

The problem we asked was:

*How can we build a **threshold** signing protocol for Schnorr that is **deterministic** and **stateless**?*

The problem we asked was:

*How can we build a **threshold** signing protocol for Schnorr that is **deterministic** and **stateless**?*

i.e. after a one-time distributed key generation phase, parties interactively sign messages without sampling new randomness or updating their state

The problem we asked was:

How can we build a ***threshold*** *signing protocol for Schnorr that is* ***deterministic*** *and* ***stateless****?*

i.e. after a one-time distributed key generation phase, parties interactively sign messages without sampling new randomness or updating their state

Implicit: deterministic nonce derivation

Challenge

- “Naive” derandomization of threshold Schnorr: direct application of single party derandomization. Works for semi-honest adversaries
- Naive scheme completely broken by an adversary that deviates from the protocol (‘rewinding’ attack)
- Malicious setting: commit to *k*, **prove** correct nonce derivation (applying PRF(*k*,m))

Towards a solution

Two very different settings:

- **Honest majority**: simple protocol with replicated secret sharing (small number of parties)
- **Dishonest majority**: “throw zero-knowledge proofs at it” [Goldreich-Micali-Wigderson 87]

Dishonest Majority

- Non-linear signing equation: reminiscent of Threshold ECDSA
- Unlike ECDSA, this problem is *trivial* with semi-honest adversaries
- Before “fully malicious”, we ask: can we interpolate a meaningful intermediate between semi-honest and malicious?

Covert Model

- Introduced by Aumann and Lindell (TCC ’07, JoC ’10)
- Sits between semi-honest and fully malicious security
- Quantified over arbitrarily cheating adversaries, but a cheating adversary can statistically evade detection with noticeable probability (eg. 10%)
- Reasonable in many scenarios (eg. business-to-business, among parties that know each other)

Covert 2P Signing

- Protocol intuition: “watchlist” technique. Alice derives nonce as a linear combination of *n* PRFs, Bob obliviously checks *n*-1 of them.
- Even for 90% deterrence, only marginally slower than semi-honest
- One extra curve point transmitted compared to SH, rounds unchanged (i.e. two)
- Likely usable in any setting where SH is feasible

Malicious nP Signing

- We adapt Zero-knowledge from Garbled Circuits [Jawurek-Kerschbaum-Orlandi 13] to prove these statements
- GCs are lightweight, efficient for small Boolean circuits like AES
- Novel techniques for:
  - GC labels -> Elliptic curve point translation (almost for free)
  - Preprocessing *Committed* Oblivious Transfer (only PRF evaluations online)

In Summary

- We study Schnorr with **stateless deterministic** threshold signing
- Alternatively, EdDSA where nonce derivation is by adding PRF outputs
- Landscape (relative to semi-honest, which is trivial):
  - **Honest majority**: ≈ SH for few parties
  - **Covert two-party**: ≈ SH for reasonable deterrence (90%)
  - **All-but-one malicious**: within order of magnitude of OT-based threshold ECDSA (100s of KB, estd. milliseconds/low tens of ms for 256-bit curve)

Thanks!

(paper coming soon)