Monerokon Madness: Schnorr Schnadness

Andrew Poelstra
Director of Research, Blockstream

June 20, 2019

Schnorr Signatures

- Schnorr signatures are an alternate signature scheme to ECDSA
- Proposed for Bitcoin as part of Taproot
- Used in Monero since the Cryptonote days, courtesy of ed25519

Schnorr signatures have an especially simple formulation for single signers:

$$
\begin{aligned}
k &\leftarrow \$ \\
e &\leftarrow H(\ldots) \\
s &\leftarrow k - xe
\end{aligned}
$$

Schnorr signatures have an especially simple formulation for multi-signers:

$$
\begin{aligned}
k_i &\leftarrow \$ \\
e &\leftarrow H(\ldots) \\
s_i &\leftarrow k_i - x_i e
\end{aligned}
$$

Nonce Bias

- Like ECDSA, Schnorr signatures require an uniformly random nonce
- Any bias is deadly
- Publicly verifying unbiasedness is hard

Nonce Bias

- Idea: use RFC6979 to deterministically generate nonces.
- Great idea. But totally unverifiable.

Nonce Bias

- Idea: use sign to contract to mix randomness into an untrusted device's nonce
- $R \rightarrow R + H(R \| \$)$

Nonce Bias

- But naively combining RFC6979 with s2c will lead to trivial secret key extraction
- (We all know "never reuse nonces". But also, never use related nonces.)
- (Even on the same message.)

- Schnorr multisignatures are easy!
- $s_i = k_i + x_i e$
- 1. Add the nonces. 2. Add the signatures.

Multisignatures

- Rogue-key attacks require you randomize the keys and signatures
- Wagner's algorithm requires you mix randomness from every key into every key
- It also requires precommitting to nonces before adding them (MuSig)

Multisignatures

- Again, mixing RFC6979 and multisignatures will lead to key extraction
- Naive or not. No way to do it
- Heh, well, maybe with sufficiently powerful ZKPs

- Need fresh randomness for every signature. No RFC6979.
- Can we at least share nonces before choosing the message?
- No. Wagner again. (Jonas Nick, 2 days ago)

Threshold Signatures

- Schnorr threshold signatures are easy!
- Secret-share the keys. Replace keys with sums of shares.
- 1. Add the nonces. 2. Add the signatures.

- First, all of the above problems apply.
- Then, make sure you have a new nonce for every signature, even for the same sig with same (combined) key

Threshold Signatures

- If you need k honest participants, have k honest participants, but also have some dishonest ones, can you recover? (Looks like it. But no.)
- Can you at least determine who was dishonest? (Not easy.)
- What if "dishonest" just means timing out? (Still not easy. Harder actually.)

- Unrelatedly, provable security is much harder (public key biasing)

Thank You

Andrew Poelstra
monerokon@wpsoftware.net