NIST Threshold Cryptography Workshop, March 12, 2019

Multisignatures and Threshold Signatures in a Bitcoin Context

Andrew Poelstra

Director of Research, Blockstream

- Bitcoin is a cryptocurrency denominated in *unspent transaction outputs* (UTXOs) labelled by a value and (script) public key.
- Transactions destroy UTXOs and create new UTXOs with equivalent value and different public keys.
- Transactions are serialized onto a *blockchain* which defines a canonical history.

Bitcoin

- Bitcoin users generate a lot of keys; must store and recognize these.
- Loss or theft of a key is not recoverable.
- Keys are typically not uniform random; are related in detectable ways.
- Diverse hardware: PCs, tiny devices, cell phones, virtual machines. Allergic to randomness.

Schnorr Signatures

$$P = xG$$

$$\begin{aligned} R &= kG \\ e &= H(P, R, m) \\ s &= k + ex \end{aligned}$$

Notice P in the hash function.

Schnorr Signatures

- Consider "BIP32" keys P and P', where $P' = P + \gamma G$ for some non-secret γ.
- Used to make key generation and backup more tractable.

$$
\begin{aligned}
R &= kG \\
e &= H(R, m) \\
s &= k + ex \\
&\to k + ex + e\gamma
\end{aligned}
$$

Sign-to-Contract

- Consider the "sign-to-contract" construction which overloads a signature as a signature on another, auxiliary message.
- Used for timestamping, wallet audit logging, and anti-covert-sidechannel resistance.

$$P = xG$$

$$\begin{aligned} R^0 &= kG \\ R &= R^0 + H(R^0 \| c)G \\ e &= H(P, R, m) \\ s &= (k + H(R^0 \| c)) + ex \end{aligned}$$

Sign-to-Contract Replay Attack

Now suppose $k = H(x \| m)$, as in RFC6979.

$$
\begin{array}{rl}
s &= (k + H(R^0 \| c)) + ex \\
-\ s &= (k + H(R^0 \| c')) + e'x \\
\hline
\end{array}
$$

$$0 = H(R^0 \| c) - H(R^0 \| c') + (e - e')x$$

So we'd better have $k = H(x \| m \| c)$!

Interlude: Randomness

- If k deviates from uniform by any amount, given enough signatures lattice techniques can be used to extract secret keys. (In practice at least a couple bits of bias are needed.)
- A malicious manufacturer could insert such bias in a way that only the attacker could detect the deviation.
- No way to prove that deterministic randomness was used (general zkps? Hard on typical signing hardware.)

Sign-to-Contract as an Anti-Nonce-Sidechannel Measure

- If the hardware device knows c before producing R^0 it can grind k so that $(k + H(R^0 \| c))$ has detectable bias.
- If it doesn't know c how can it prevent replay attacks?
- Send hardware device $H(c)$ and receive R^0 before giving it c.
- Then $k = H(x \| m \| H(c))$.

Multisignatures

- Bitcoin people use “multisignature” in a funny way.
- Includes thresholds (or arbitrary monotone functions of individuals’ keys).
- Do *not* expect or want verifiers to see the original keys, for efficiency and privacy.

Multisignatures

- Plain public-key model.
- May be chosen (from the set of available keys) adversarially and adaptively.
- Keys controlled by inflexible offline signing hardware.
- No good place to store KOSK proofs. No keygen authorities.
- Keys may encode semantics (e.g. Taproot, pay-to-contract) where KOSK is insufficient for security!

Multisignatures

- Consider Schnorr multisignatures with combined keys of the form $P = \sum P_i$.
- Vulnerable to rogue-key attacks where one participant cancels others' keys.
- Bitcoin's *Taproot* uses keys of the form $P = P' + H(P' \| c)G$ which admits a new form of rogue-key attack.
- KOSK cannot protect against the latter!

Multisignatures

- Derandomization of the form $k = H(x \| c)$ no longer works.
- In a multi-round protocol need to consider replay attacks, parallel attacks, VM forking, etc.
- General ZKPs can save us here. More R&D needed.

Threshold Signatures and Accountability

- Accountability: ability to prove which specific set of signers contributed to a threshold signature.
- Constant-size non-accountable signatures. Log-sized accountable signatures.
- Can we close this gap?

Thank you.

Andrew Poelstra
apoelstra@blockstream.com