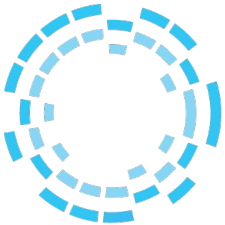


Blockstream

Shrunken Schnorr Shortcuts

x-only pubkeys, security reductions, MuSig shortcuts and Wagner's attack

2019-10-20 Jonas Nick @n1ckler

GPG: 36C7 1A37 C9D9 88BD E825 08D9 B1A7 0E4F 8DCD 0366

This talk is about Bitcoin Improvement **Proposals**

- BIP-schnorr: Schnorr Signatures
- BIP-taproot: SegWit version 1 output spending rules, uses BIP-Schnorr

This talk is about Bitcoin Improvement **Proposals**

- BIP-schnorr: Schnorr Signatures
- BIP-taproot: SegWit version 1 output spending rules, uses BIP-Schnorr

There’s no guarantee that a BIP-taproot softfork activates in its current form or at all. This depends on community consensus.

Compressed public keys

```
02 + <32 byte array>
```

or

```
03 + <32 byte array>
```

Compressed public keys

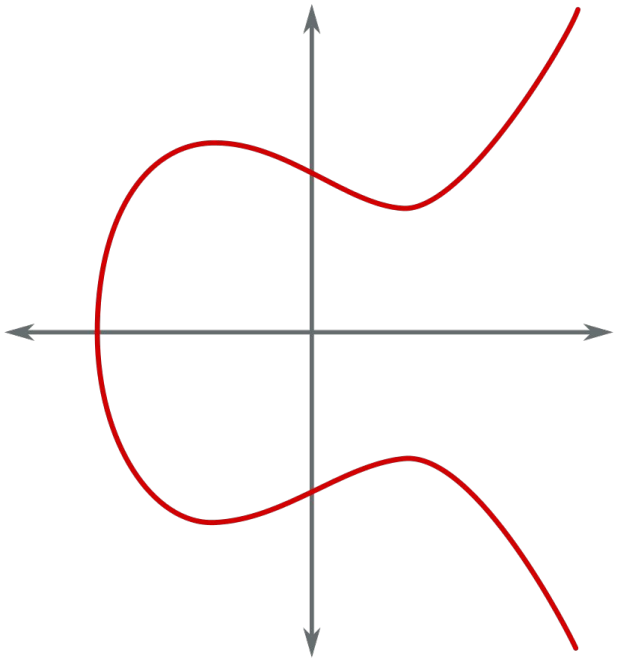
```
02 + <32 byte array>
```

or

```
03 + <32 byte array>
```

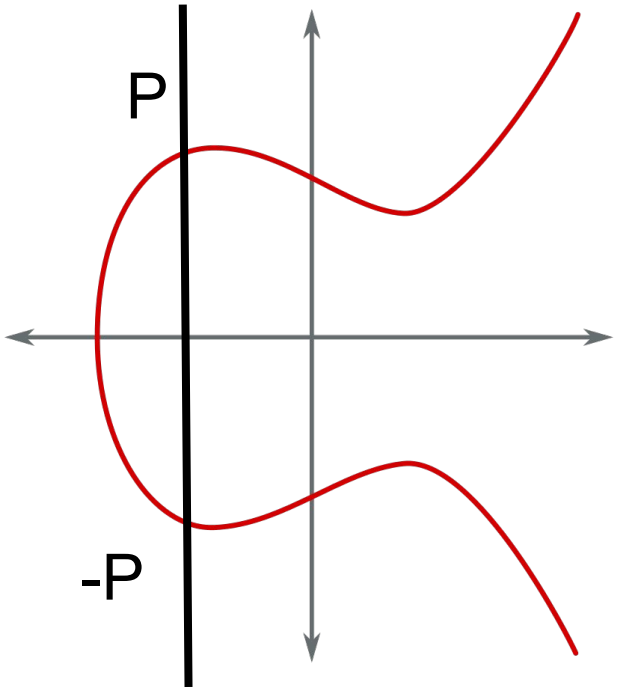
“tie breaker” X-coordinate

Purpose of tie breaker



Purpose of tie breaker

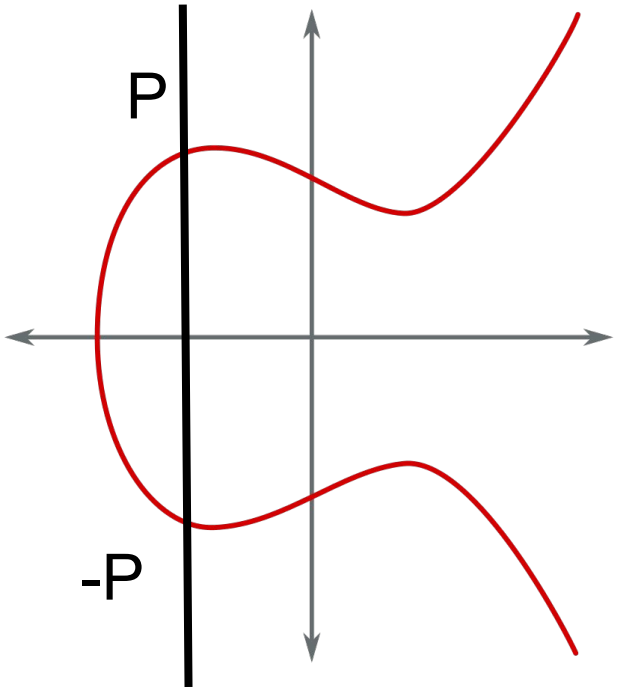
Given X-coordinate



Purpose of tie breaker

Given X-coordinate

Determines whether encoded point is P or -P



x-only pubkeys in BIP-Schnorr

Implicitly assume 0-th byte is `02` (*)

~~`02` +~~ `<32 byte array>`

or

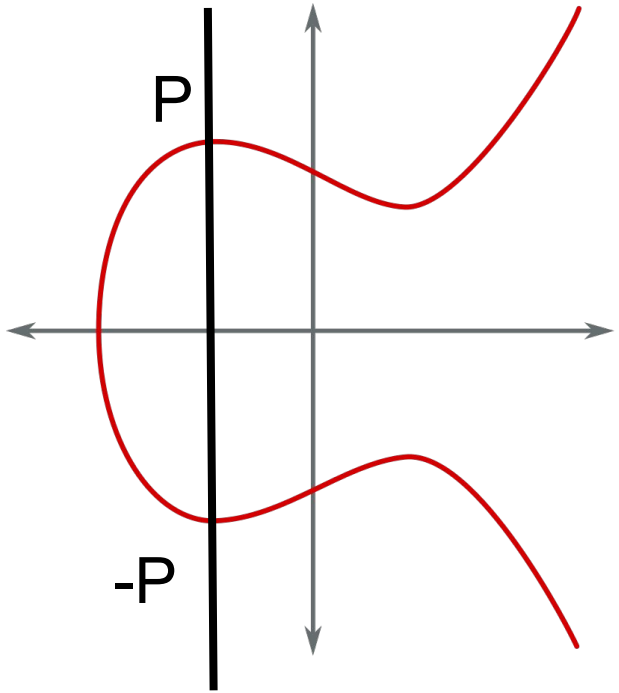
~~`03` +~~ `<32 byte array>`

~~"tie breaker"~~ X-coordinate

(*) In the BIP it's a different tie breaker actually

Why does this work?

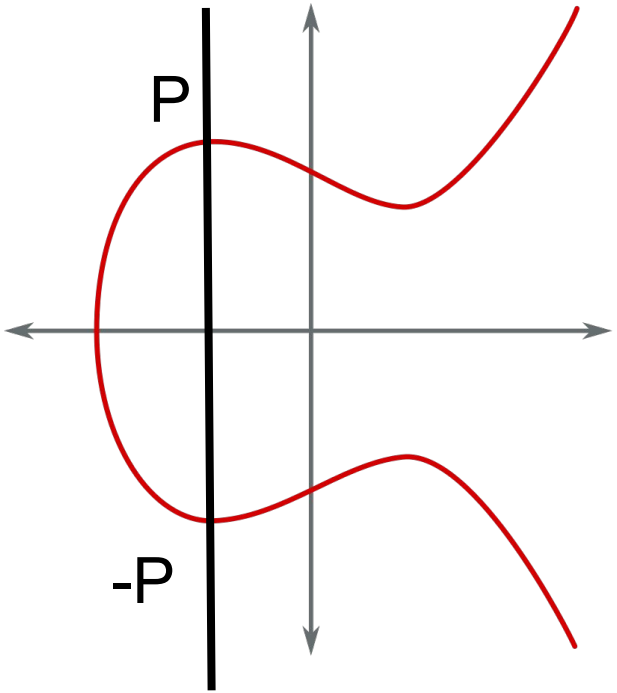
- P and -P are still different points!



Why does this work?

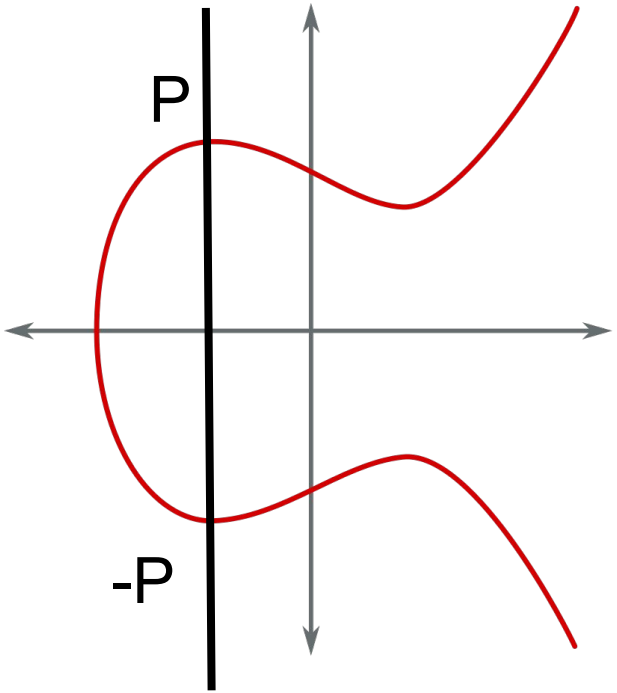
- P and -P are still different points!
- But public key $P = x \cdot G$

  and $-P = -x \cdot G$



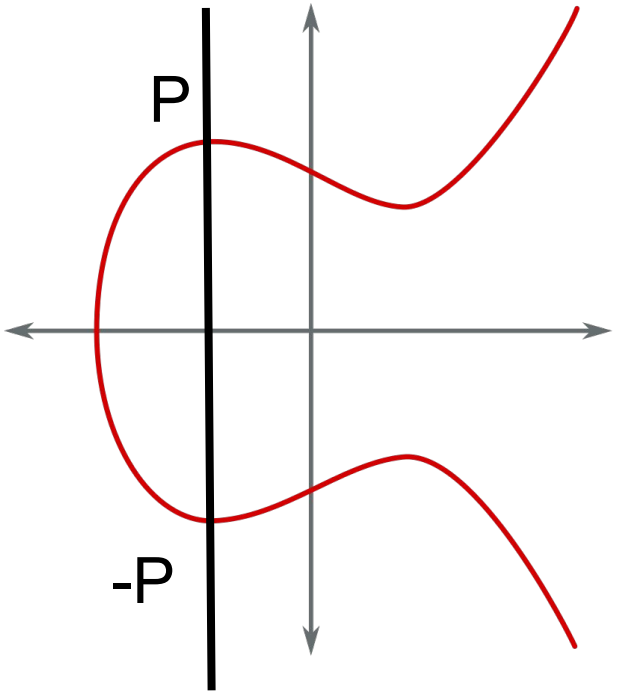
Why does this work?

- P and -P are still different points!
- But public key $P = x \cdot G$
  and $-P = -x \cdot G$
- Therefore, just negate secret key before signing if necessary



Why does this work?

- P and -P are still different points!
- But public key $P = x \cdot G$
  and $-P = -x \cdot G$
- Therefore, just negate secret key before signing if necessary
- No action required from wallet devs, handled by crypto library. BIP32 derivation unaffected



Why x-only?

1. saves about 0.7% weight units (WU) in average block

Why x-only?

1. saves about 0.7% weight units (WU) in average block
2. same cost for sender as P2WSH

	P2WPKH	P2WSH	taproot (x-only)
scriptPubKey	88 WU	136 WU	136 WU

Why x-only?

1. saves about 0.7% weight units (WU) in average block
2. same cost for sender as P2WSH

	P2WPKH	P2WSH	taproot (x-only)
scriptPubKey	88 WU	136 WU	136 WU
with witness	196 WU	-	201 WU

Why x-only?

1. saves about 0.7% weight units (WU) in average block
2. same cost for sender as P2WSH

	P2WPKH	P2WSH	taproot (x-only)
scriptPubKey	88 WU	136 WU	136 WU
with witness	196 WU	-	201 WU

3. same security level

x-only security

We know (in Random Oracle Model):

Discrete Logarithm Problem is hard $\Rightarrow$ Schnorr sig is secure

x-only security

To prove:

Schnorr sig secure $\Rightarrow$ x-only Schnorr sig secure

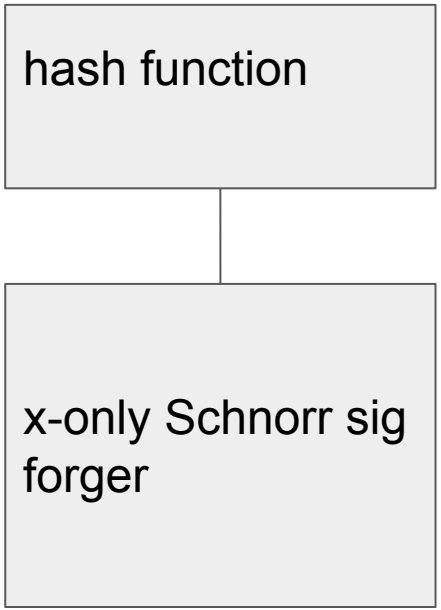
or equivalently

x-only Schnorr sig insecure $\Rightarrow$ Schnorr sig insecure

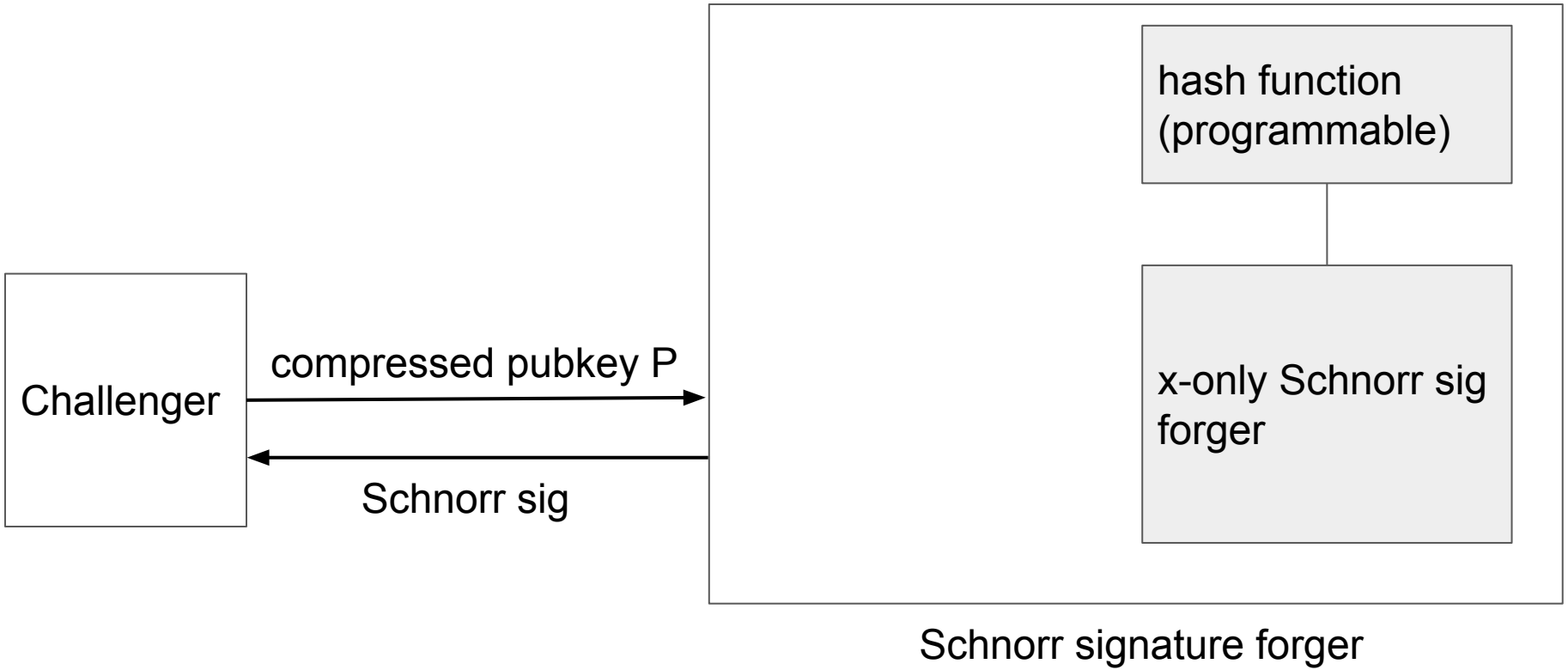
Proof sketch

Schnorr signature on message m with public key $P = xG$:

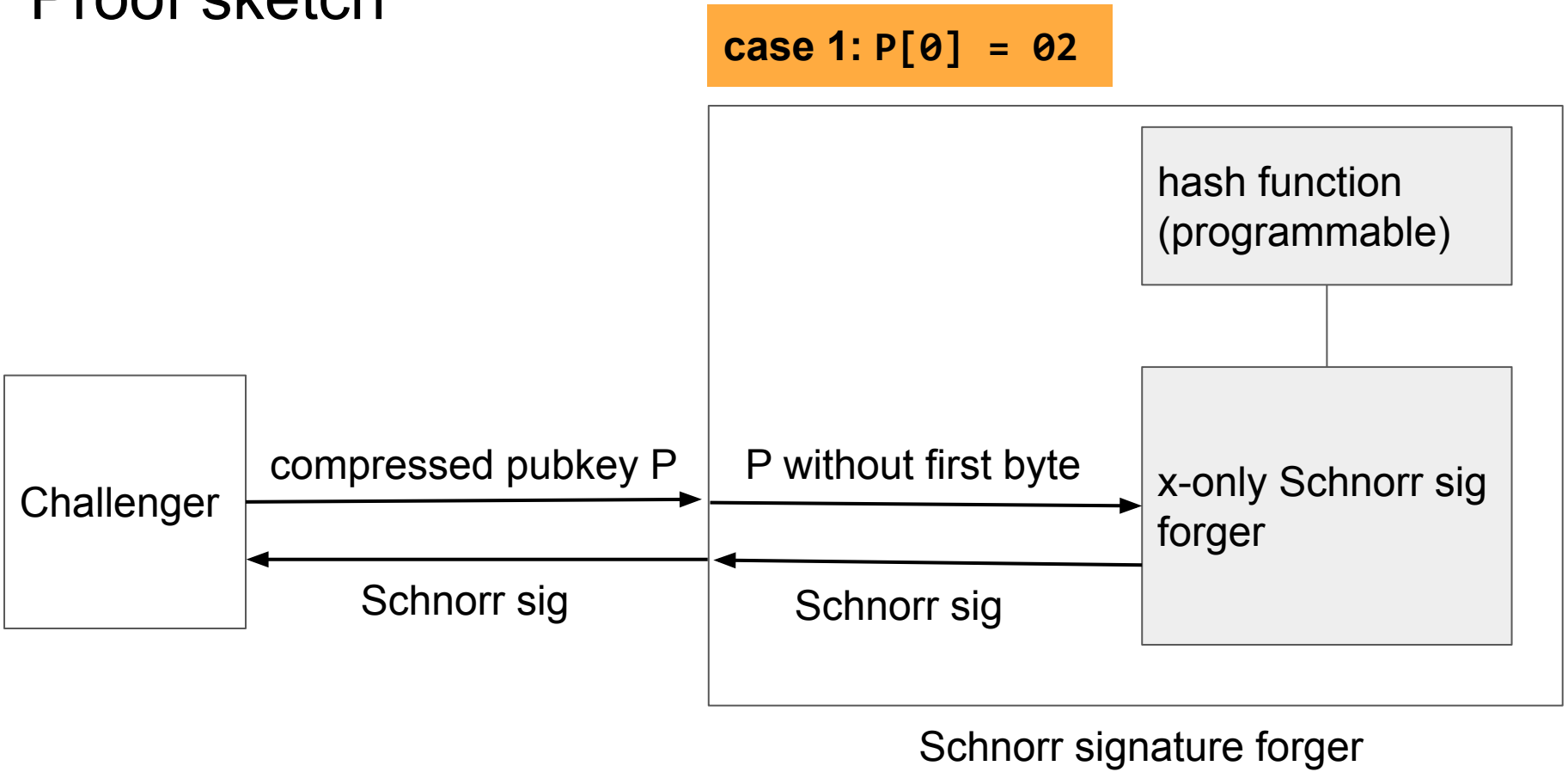
$(R = k \cdot G,\ s = k + hash(R, P, m) \cdot x)$



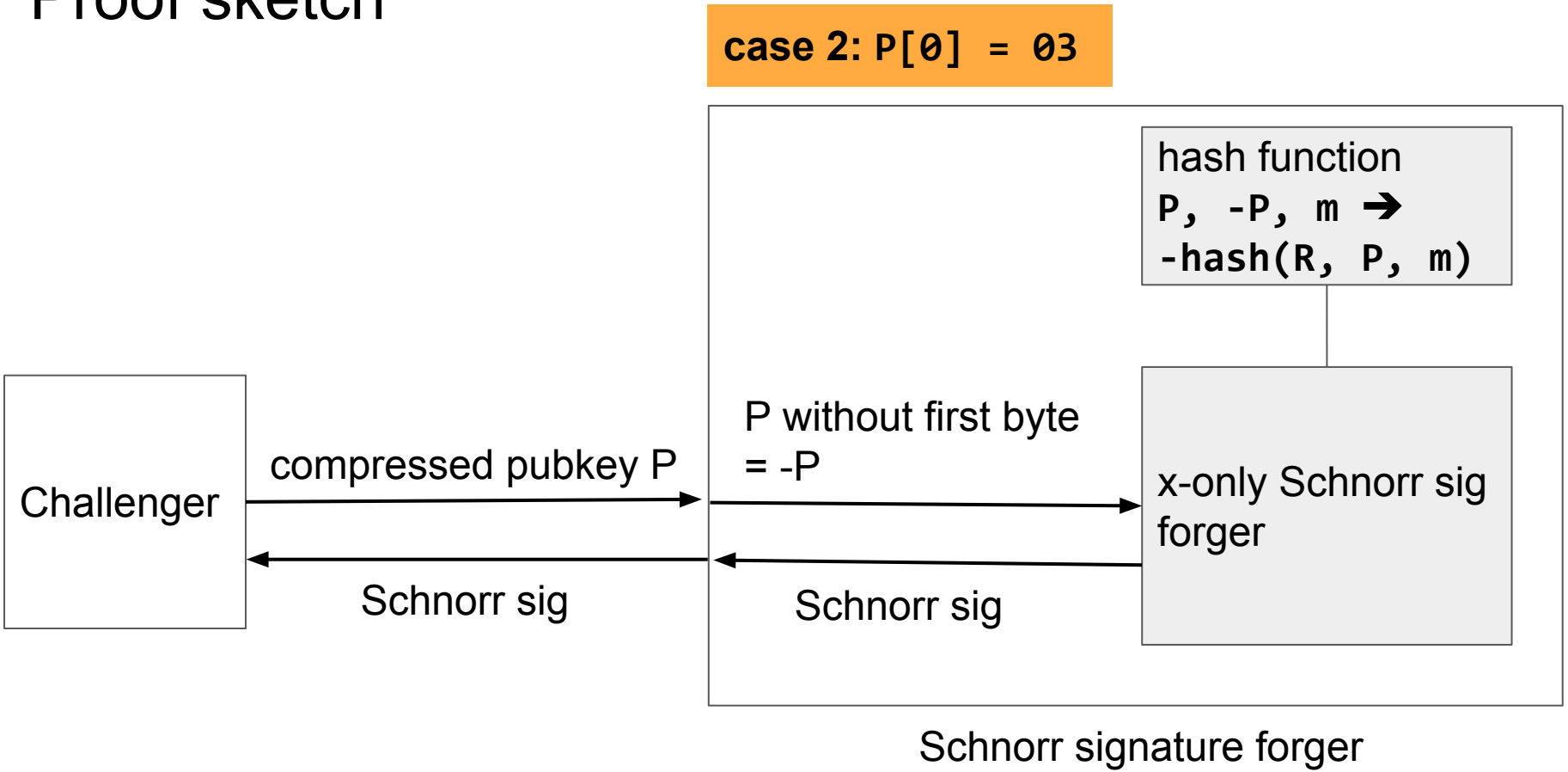
Proof sketch



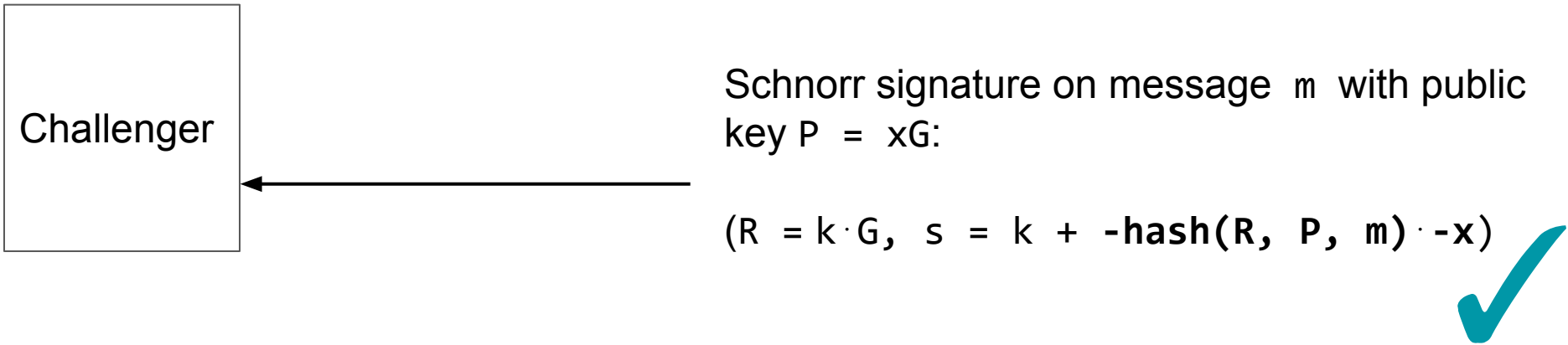
Proof sketch



Proof sketch



Proof sketch



x-only security

To prove:

Schnorr sig secure $\Rightarrow$ x-only Schnorr sig secure

or equivalently

x-only Schnorr sig insecure $\Rightarrow$ Schnorr sig insecure

MuSig shortcuts

MuSig

- Allows key aggregation on BIP-schnorr

MuSig

- Allows key aggregation on BIP-schnorr

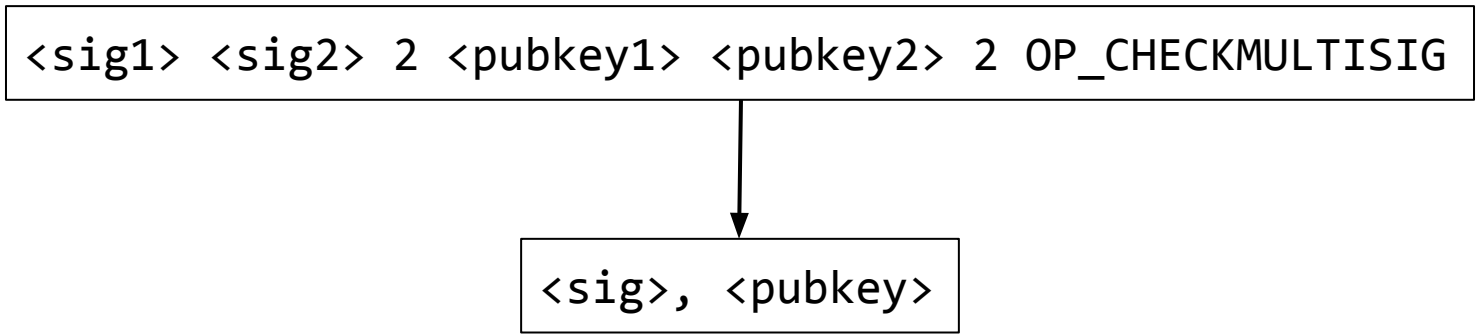
Lightning funding script with cooperative close:

```
<sig1> <sig2> 2 <pubkey1> <pubkey2> 2 OP_CHECKMULTISIG
```

MuSig

- Allows key aggregation on BIP-schnorr

Lightning funding script with cooperative close:



MuSig rounds

public key P, message m

1. Exchange nonce commitments

MuSig rounds

public key P, message m

1. Exchange nonce commitments

2. Exchange nonces R_i

   $R = \sum R_i$

MuSig rounds

public key P, message m

1. Exchange nonce commitments

2. Exchange nonces R_i

   $R = \sum R_i$

3. Exchange partial signatures.

   $s_i = k_i + hash(R, P, m) \cdot x_i$

MuSig rounds (pre-shared nonces)

public key P, message m

Prepare

1. Exchange nonce commitments

2. Exchange nonces R_i

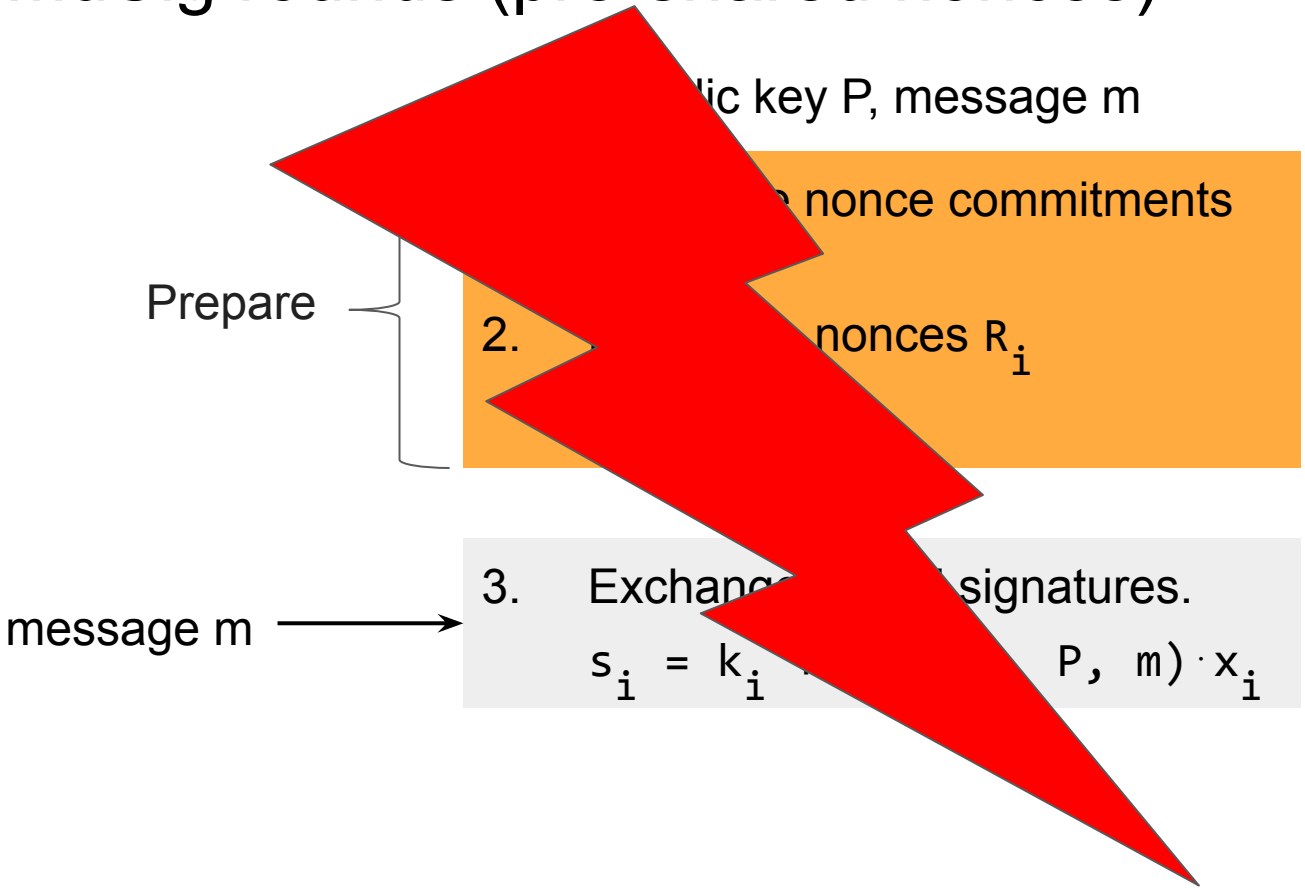
   $R = \sum R_i$

message m →

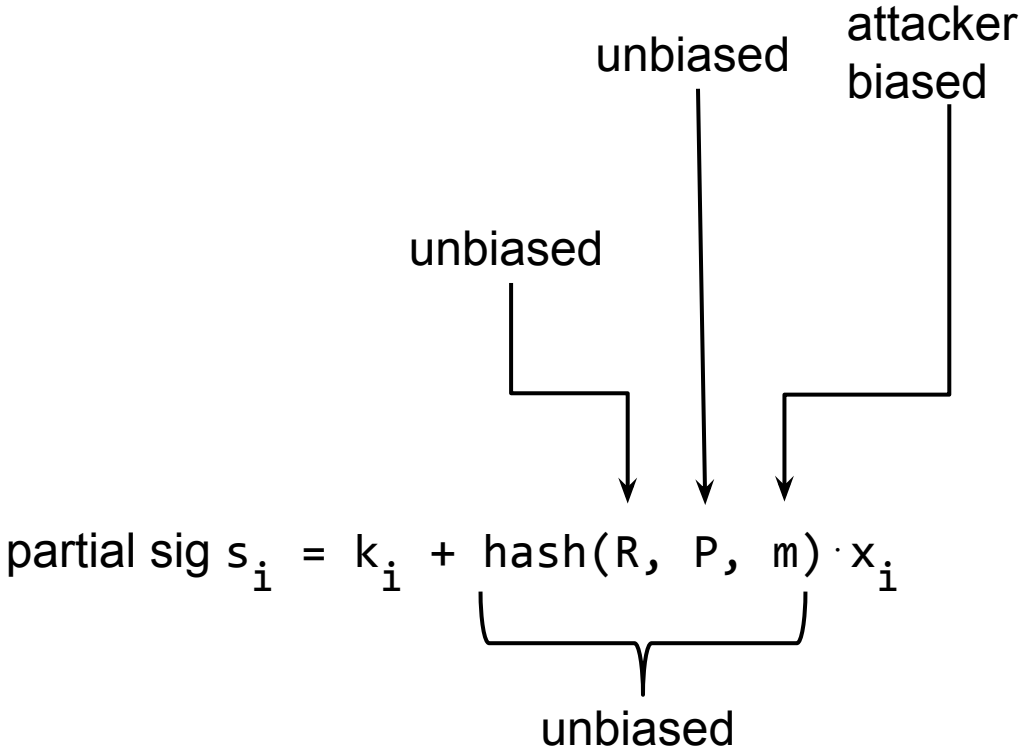
3. Exchange partial signatures.

   $s_i = k_i + hash(R, P, m) \cdot x_i$

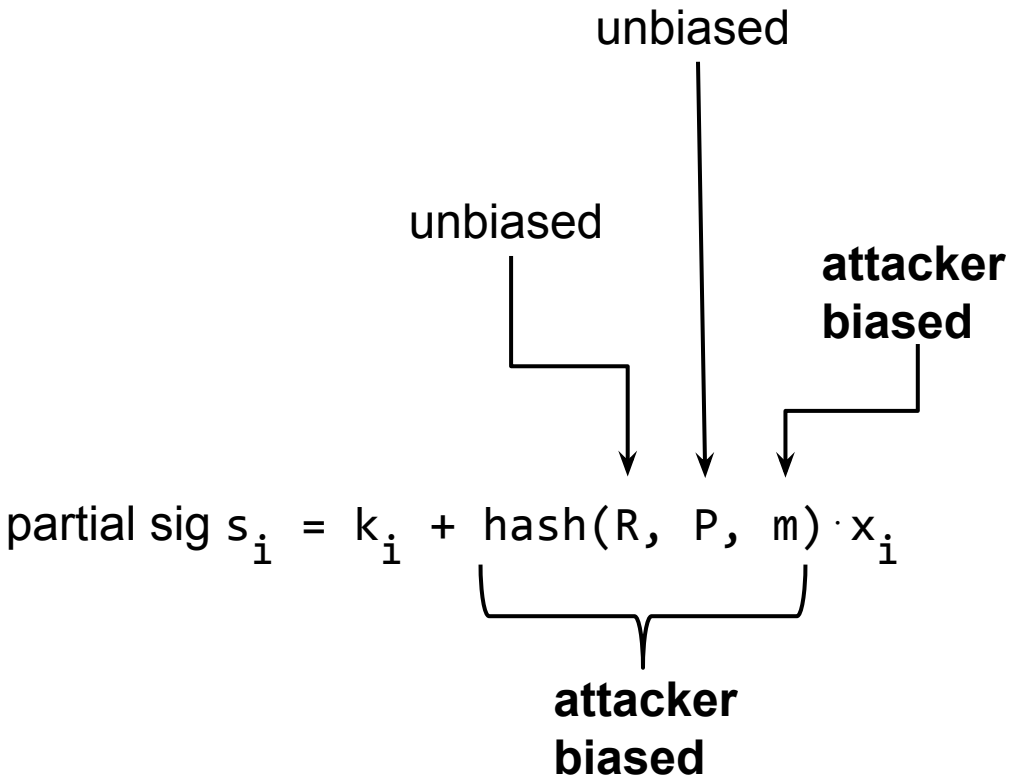
MuSig rounds (pre-shared nonces)



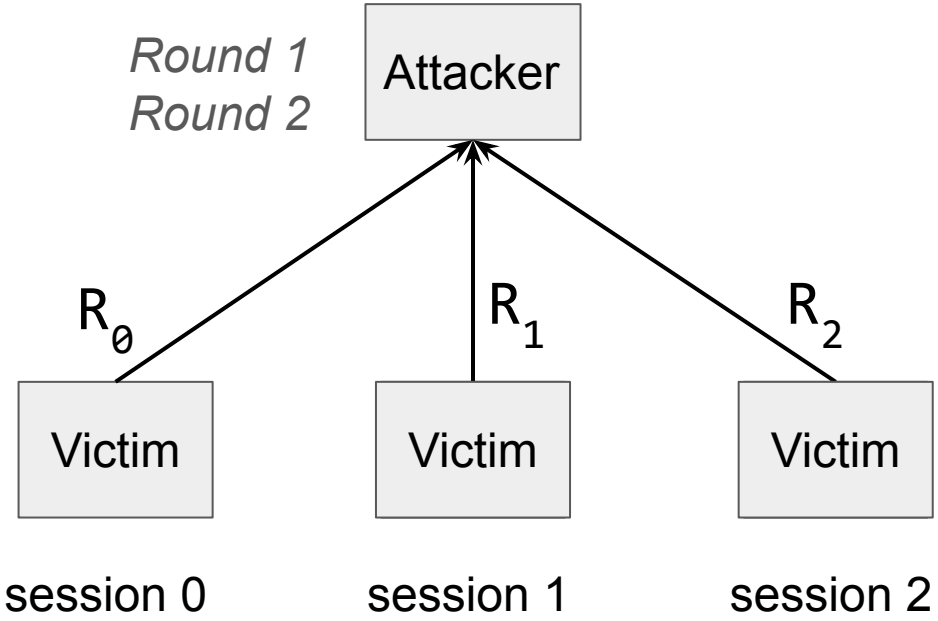
What's the difference?



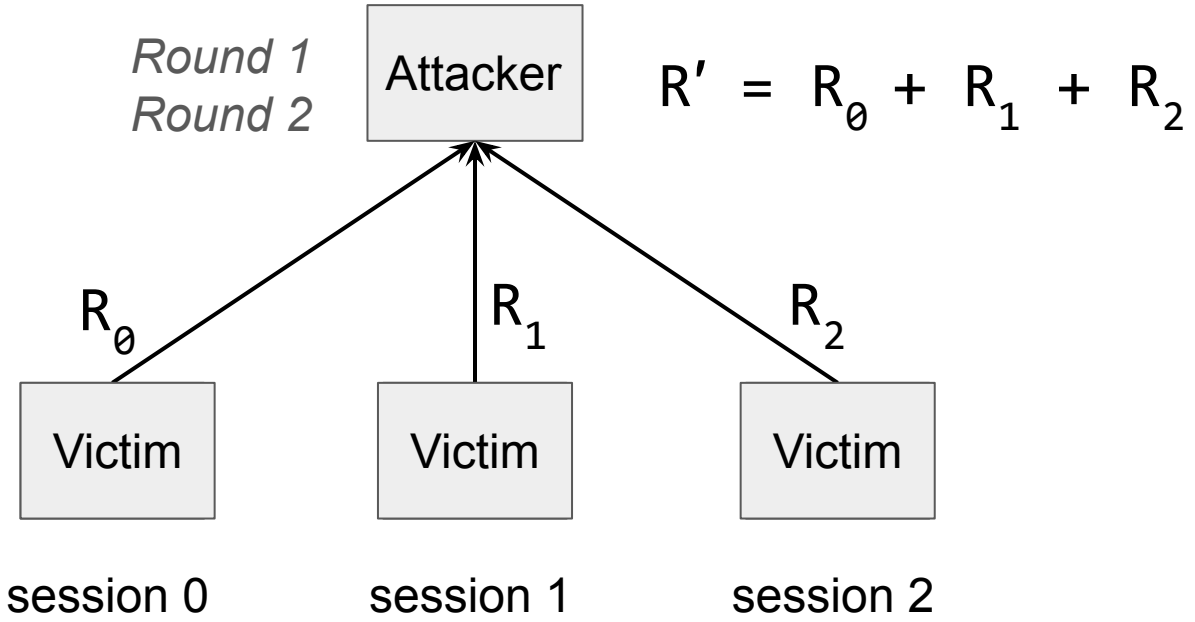
What's the difference?



Signature forgery on message m′



Signature forgery on message m′



Generalized Birthday Problem

Find m_0, m_1, m_2 such that

$$\text{hash}(R',P,m') = \text{hash}(..m_0) + \text{hash}(..m_1) + \text{hash}(..m_2)$$

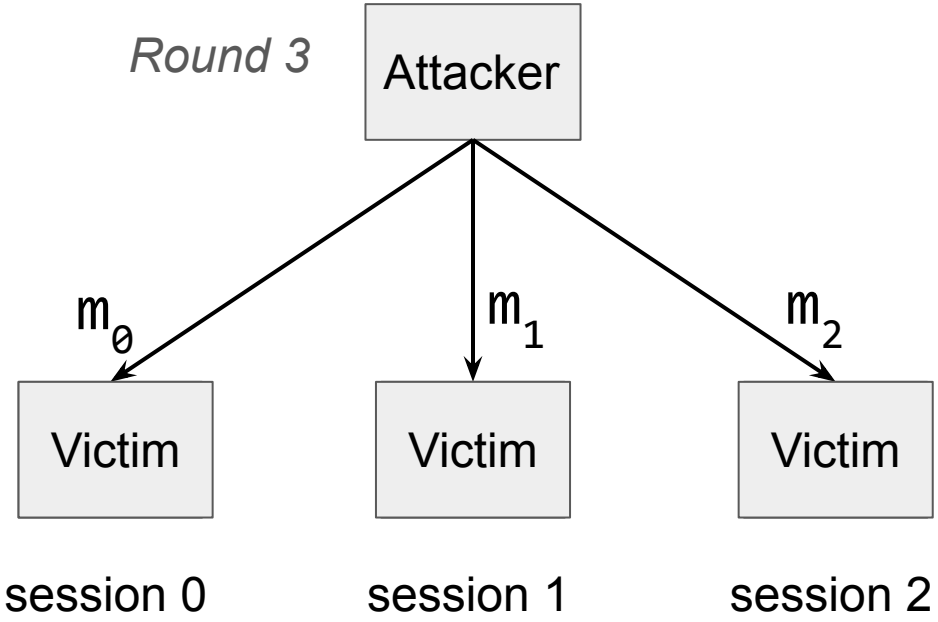
Generalized Birthday Problem

Find m_0, m_1, m_2 such that

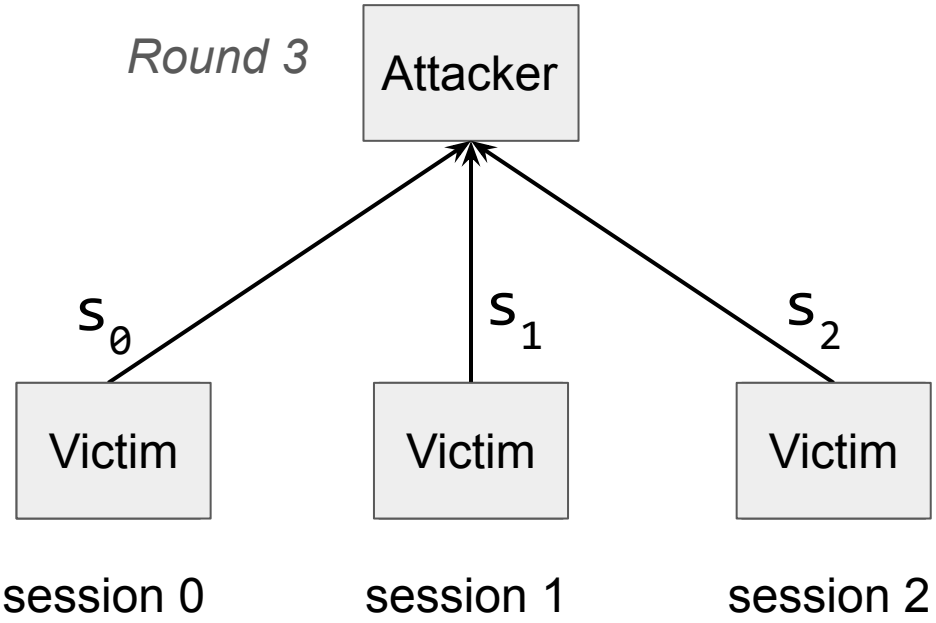
$$\text{hash}(R',P,m') = \text{hash}(..m_0) + \text{hash}(..m_1) + \text{hash}(..m_2)$$

- can be solved efficiently with *Wagner's algorithm*
- More parallel sessions: solvable with on the order of 2^{32} operations

Signature forgery on message m′



Signature forgery on message m′



$(\sum R_i, \sum s_i)$ is a Schnorr signature on m′

MuSig rounds (pre-shared nonce commitments)

public key P, message m

Prepare

1. Exchange nonce commitments

message m →

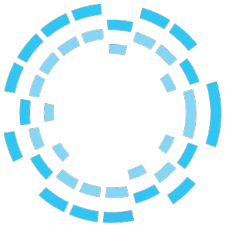
2. Exchange nonces R_i

   $R = \sum R_i$

3. Exchange partial signatures.

   $s_i = k_i + hash(R, P, m) \cdot x_i$

Conclusion



Blockstream

- BIP-taproot uses x-only pubkeys now
  - wallet devs don't need to do anything in particular
- Security can be reduced to Schnorr signatures with compressed keys
- Some shortcuts in MuSig are insecure due to Wagner's algorithm
  - ElementsProject MuSig implementation can not be misused in that way
- BIP-schnorr and BIP-taproot slowly mature from draft status. Looking for feedback.
- Slides at https://nickler.ninja/slides/2019-tlc.pdf

2019-10-20 Jonas Nick jonasd.nick@gmail.com https://nickler.ninja @n1ckler

GPG: 36C7 1A37 C9D9 88BD E825 08D9 B1A7 0E4F 8DCD 0366