Practical Key-Extraction Attacks in Leading MPC Wallets

MPTS 2023: NIST Workshop on Multi-Party Threshold Schemes
September 27, 2023

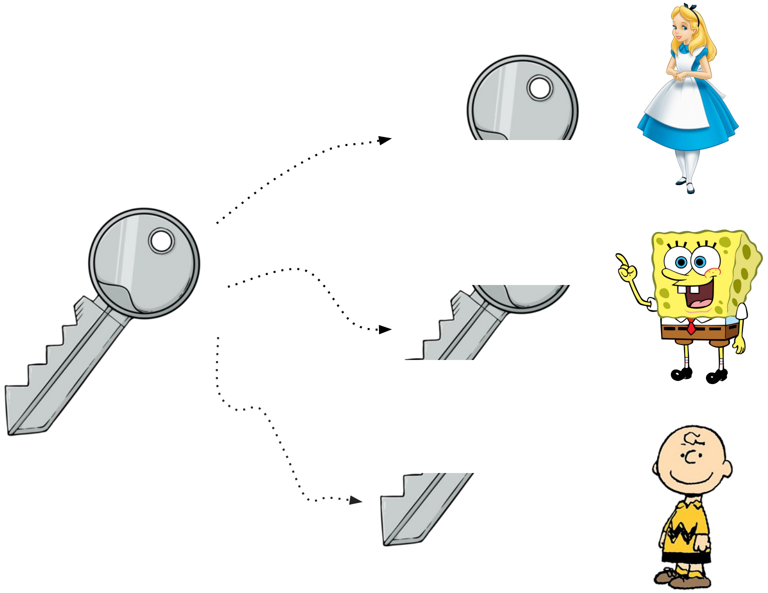
Nikolaos Makriyannis & Oren Yomtov

Fireblocks

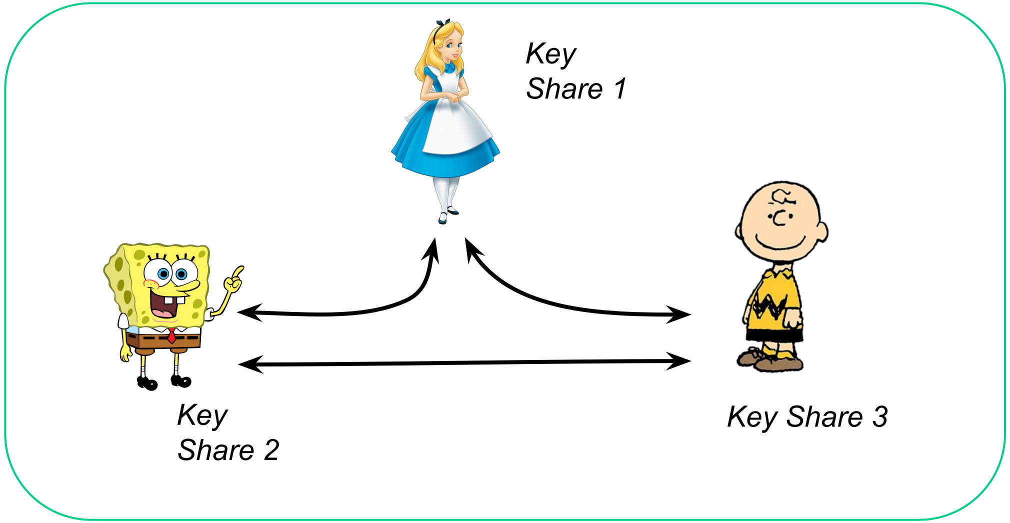
Crypto Wallets in 30”

Cryptocurrency Wallets 101

Enter MPC
(through the lense of threshold signing)



Enter MPC
(through the lense of threshold signing)



Generate public key and calculate signatures via an **interactive protocol**

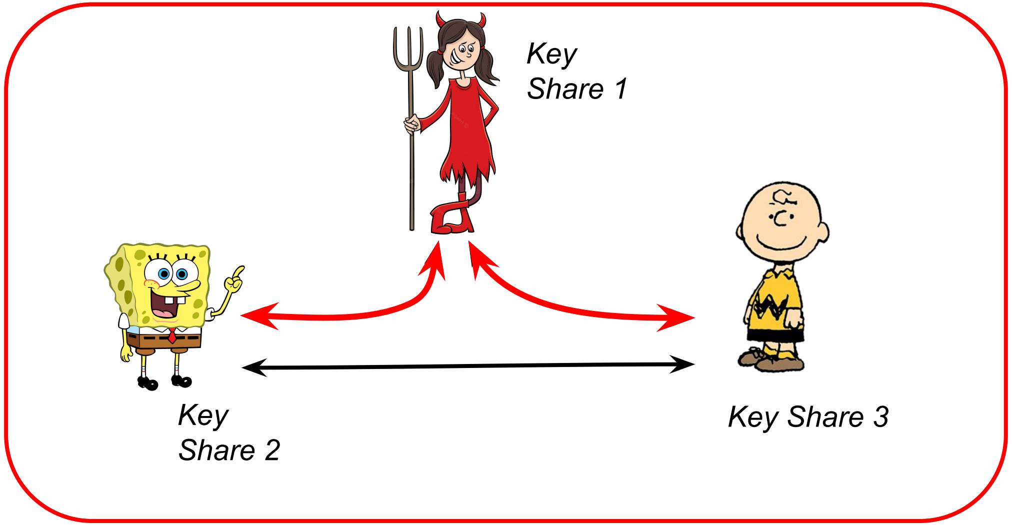
The private key is ***NEVER*** *assembled in one place*

MPC Wallet Attack Outcomes

- Denial of Service
- Signature Forgery
- Private Key Exfiltration

Today's Talk

MPC Threat model



Malicious Alice wants to exfiltrate her counterparties' shares

Our Findings

Our Findings

- Discovered 4 **novel attacks**
- Affecting **16** vendors / libraries
- Releasing 3 **PoC exploits**
- Exfiltrated keys from 2 vendor **production environments**
- Most of our attacks are **not** implementation specific

Only 3 mentioned in the talk today

Our Attacks

1. The most popular 2PC signing implementations: Lindell17 (**256-sig attack**)
2. The most popular MPC signing protocols: GG18&20 (**16-sig attack**)
3. A DIY protocol used by a crypto custodian: BitGo TSS (**1-sig attack**)

Relevance to MPTS23

1. We identify critical flaws in popular protocols/implementations of t-ECDSA
2. Protocol designers/implementers should be aware of these pitfalls
3. We propose fixes from the literature that align with the standardization effort

NIST

Cryptographic Exploit Development

Math/Notation

- **No** elliptic curves (or even abstract groups)
- The **modulo** operator

$$x \,\%\, N$$

Remainder of
x divided by N

Paillier Encryption

Paillier Encryption is **linear** homomorphic

$$\mathrm{Enc}(42) \rightarrow$$

$$\leftarrow \mathrm{Enc}(2 \cdot 42 + 100)$$

$$N = p \cdot q$$

$$\mathrm{Dec}(\ldots) = 184$$

ECDSA Signature Generation

Ephemeral key

$$k = \mathrm{random}()$$

$$s = \mathrm{sig}(\mathrm{msg}, k, x, \ell)$$

Private key

ECDSA constant

ECDSA signing with 2 parties

Keys

x

k

Key Shares

x_1, x_2

k_1, k_2

```
Threshold ECDSA Protocols:
Lindell17
GG18
HLNR18
DKLs18
DKLs19
CMP20
GG20
CGGMP21
...
```

Compromising Lindell17 Implementations

Broken Record Attack

Exfiltrate the key in **256** signatures

Fireblocks

Lindell17 Key Generation

Choose a random key share

x_1

Choose a random key share

x_2

Lindell17 Key Generation

$$\mathrm{Enc}(x_2), N$$

(only bob can decrypt,
but alice can operate on it)

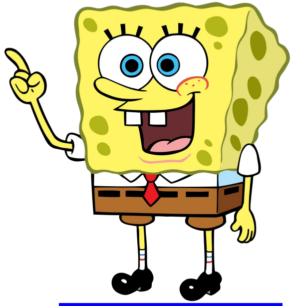
Encrypts x_2 using Paillier pk N

Key-Extraction Attacks in Leading Wallets

Lindell17 Signing (Step 1/2)

Alice sends a encrypted partial signature

$$\mathrm{Enc}\left(\left(k_1^{-1} \,\%\, \ell\right) \cdot \left(\mathrm{msg} + x_1 \cdot x_2\right)\right)$$



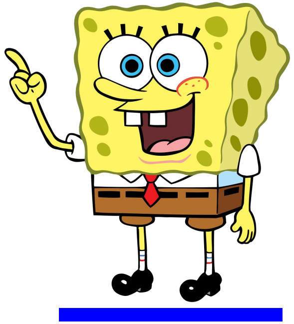
Lindell17 Signing (Step 2/2)

Bob finalizes the signature

$$\text{Decrypt}(\ldots)$$

$$\downarrow$$

$$s = k_2^{-1} \cdot (k_1^{-1} \,\%\, \ell) \cdot (\text{msg} + x_1 \cdot x_2) \,\%\, \ell$$

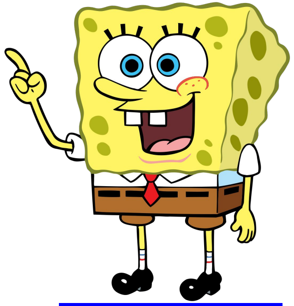


Bob then verifies the signature is valid

What if alice deviates from the protocol?

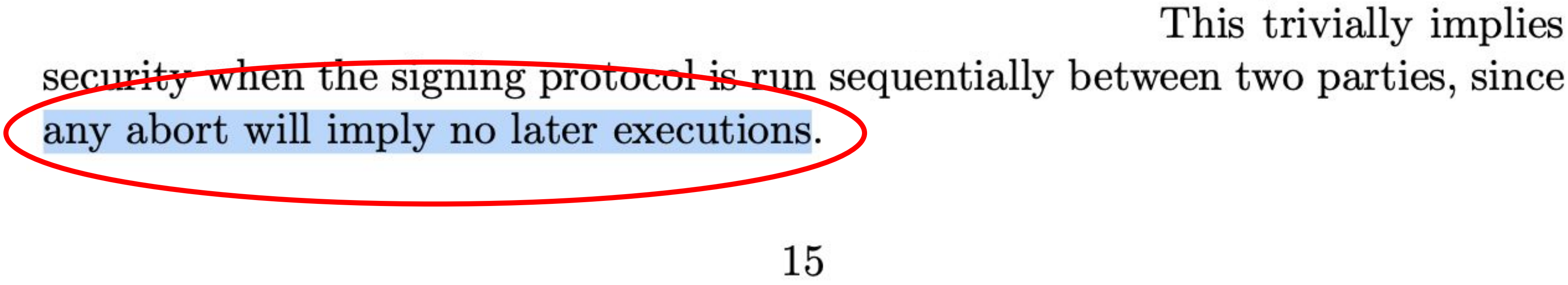
Hey! the signature is invalid

$$\mathrm{Enc}\left(\left(\cancel{k_1^{-1}\,\%\,\ell}\right)\cdot(\mathrm{msg}+x_1\cdot x_2)\right)$$



Bob fails to verify the resulting signature!

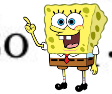
What does the paper say about that?


This trivially implies security when the signing protocol is run sequentially between two parties, since any abort will imply no later executions.

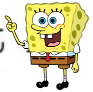
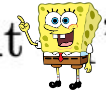
15

Denial-of-Service Attack

Back to the drawing board

The only problem that remains is that  may send an incorrect s' value to .

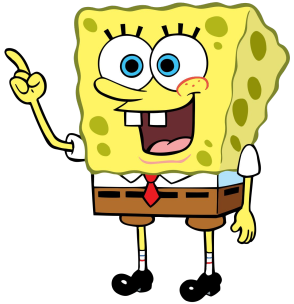
...

In such a case, the mere fact that  aborts or not can leak a single bit about 's private share of the key.

Hypothetical Attack Visualization

s' that fails to finalize if x_2's lsb = 0

Signed successfully



x_2=

0b____

____0

Hypothetical Attack Visualization

s' that fails to finalize if x_2's 2nd lsb = 0 →

← **Failed to finalize signature**

x_2=

0b____10

Hypothetical Attack Visualization

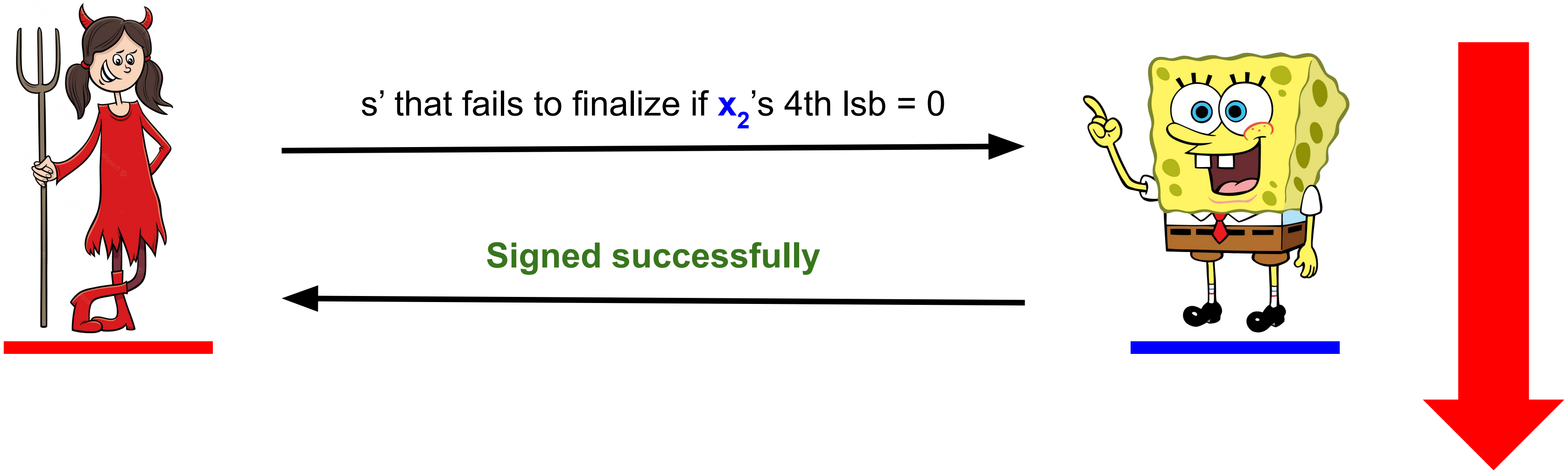
s' that fails to finalize if x_2's 3rd lsb = 0

Failed to finalize signature

x_2=

0b____110

Hypothetical Attack Visualization



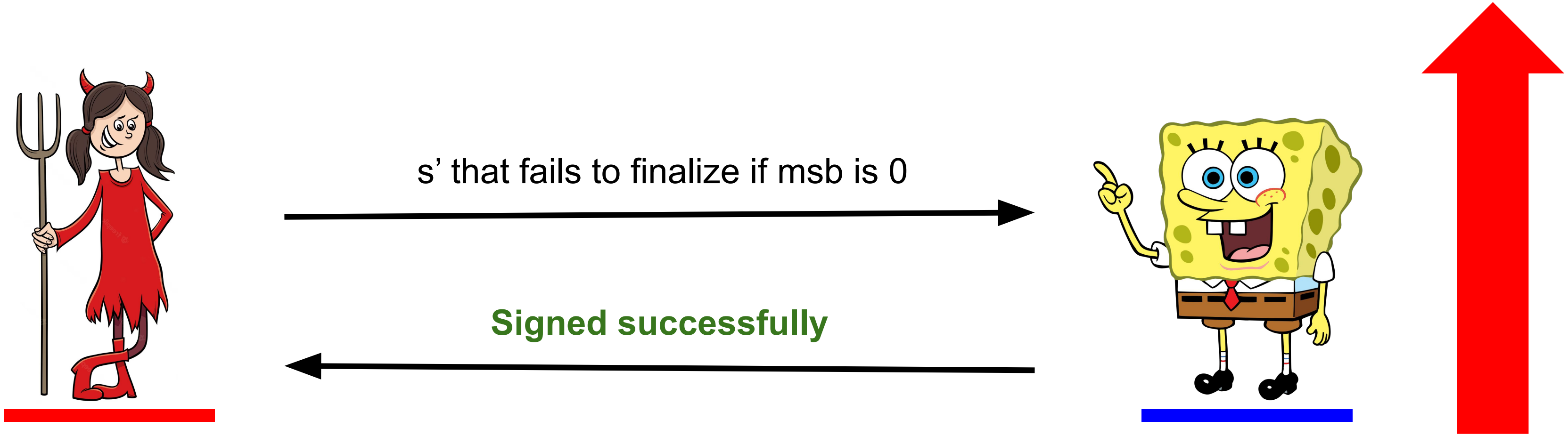
x_2=

0b____

____0110

256 signatures later...

Hypothetical Attack Visualization



x_2=

0b0110010111010001011001111111010010101001101010000011001110111001100101101010000010101111100100000101000000000111001001000110000001
0100010110111010001100111000110110101000110010110010001011000010110110010010100111001000100010110001001000001001111011001001100110

Crafting a malicious partial signature

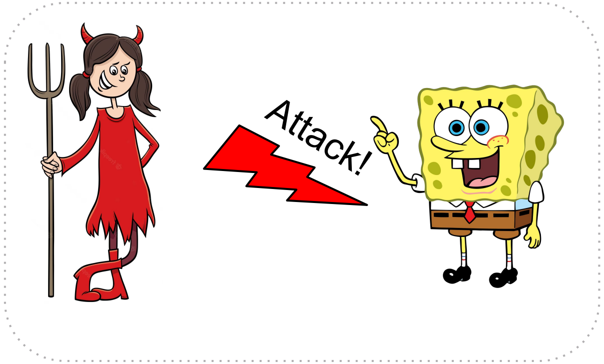
$$(k_1^{-1} \,\%\, \ell) \cdot (\mathrm{msg} + x_1 \cdot x_2)$$

After  decrypts, $=$ iff $x_2 \,\%\, k_1 = 0$

$$(k_1^{-1} \,\cancel{\%\, \ell}) \cdot (\mathrm{msg} + x_1 \cdot x_2)$$

$$\%\, N$$

Obtaining leakage on x2



Signature is valid

$$x_2 \,\%\, k_1 = 0$$

Signature is invalid

$$x_2 \,\%\, k_1 \neq 0$$

Exfiltrating the first bit

$$k_1 = 2$$

Leakage: $x_2 \% 2 = 0$

Exfiltrating the next bit

$$k_1 = 4$$

Leakage: $x_2 \% 4 = 0$

Wanted: $(x_2 - 1) \% 4 = 0$

Offsetting previous leaked bits

$$(k_1^{-1} \,\%\, N) \cdot (\mathrm{msg} + x_1 \cdot x_2)$$

$$+$$

$$(k_1^{-1} \,\%\, \ell - k_1^{-1} \,\%\, N) \cdot (\mathrm{msg} + x_1 \cdot \text{known})$$

The previously leaked bits

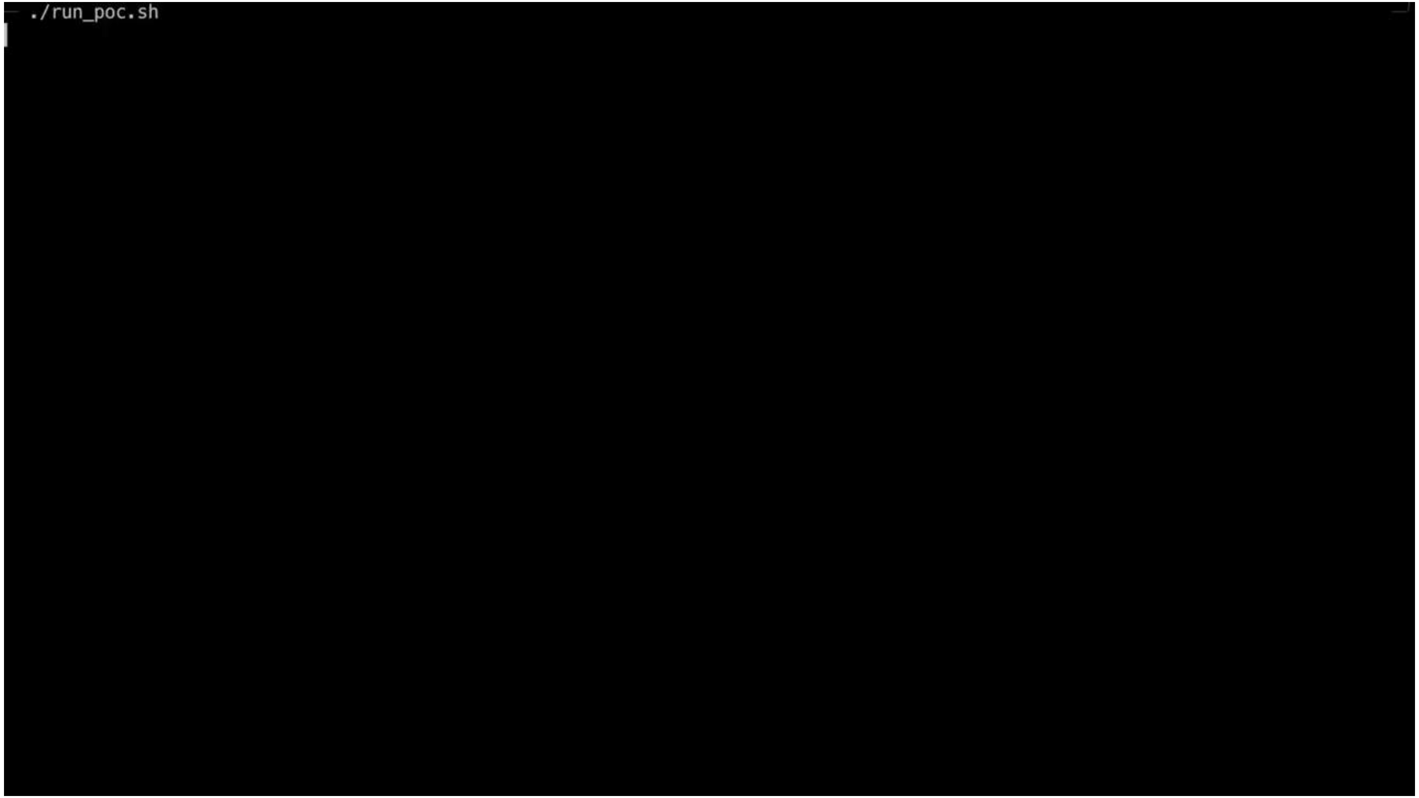
Exfiltrating the i-th bit

$$k_1 = 2^i$$

Offset: $(k_1^{-1} \,\%\, \ell - k_1^{-1} \,\%\, N) \cdot (\mathrm{msg} + x_1 \cdot \text{known})$

Leakage: i-th bit

Key-Extraction Attacks in Leading Wallets



github.com/ZenGo-X/multi-party-ecdsa Star 848

How to mitigate the Attack

Follow the paper’s instructions (e.g. don’t sign again after failure)

```
+       if abort == "true" {
+           panic!("Tainted user");
+       }
```

...or use a ZK Range Proof

NIST

A Glimpse at the Other Attacks

6ix1een Attack

Exfiltrate the key in **16** signatures

Zero Proof Attack

Exfiltrate the key in <1 signature!

Compromising GG18/20

- Pallier moduli are not checked for biprimality or small factors (via ZKP)
- Choose $N = p_1 \cdot p_2 \cdot \ldots \cdot p_{16} \cdot q$
- Choose your ephemeral share $k = N/p_i$
- Cheat in the ZKP during signing
- Extract $x \% p_i$

(do this 16 times)

6ix1een Attack

Compromising BitGo TSS

- No ZKP anywhere in the protocol
- Choose $N = p_1 q_1 \cdot p_2 q_2 \cdot \ldots \cdot p_{16} q_{16}$ where $q_i = 2p_i + 1$
- Choose encrypted ephemeral share $\text{"Enc}(k)\text{"} = 4$
- Extract x

(*one signature* suffices)

Zero Proof Attack

Concluding Remarks

Threshold-ECDSA, Paillier & standardization

1. Paillier Encryption is a popular primitive in t-ECDSA (and MPC in general)
2. There is a need to standardize the associated ZKPs
    a. Paillier Well-Formedness & Range Proofs
    b. What about sigma protocols in general? (Proofs of group homomorphism)
3. Regarding t-ECDSA, ***in my opinion,***
   there is enough overlap to standardize a single t-ECDSA framework

Thank you

Paper available on eprint

- eprint.iacr.org/2023/1234

Practical Key-Extraction Attacks in Leading MPC Wallets

Nikolaos Makriyannis* Oren Yomtov*

August 15, 2023