./ WAC6

Practical Key-Extraction Attacks in Leading MPC Wallets

Nikolaos Makriyannis & Oren Yomtov

Fireblocks

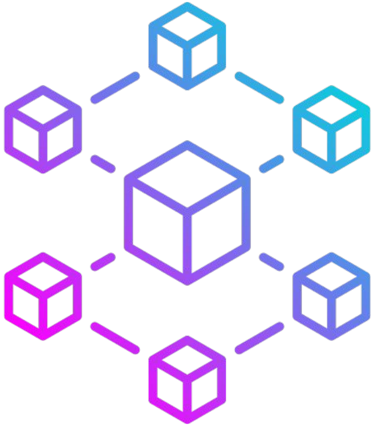
./ WAC6

Intro to crypto wallets

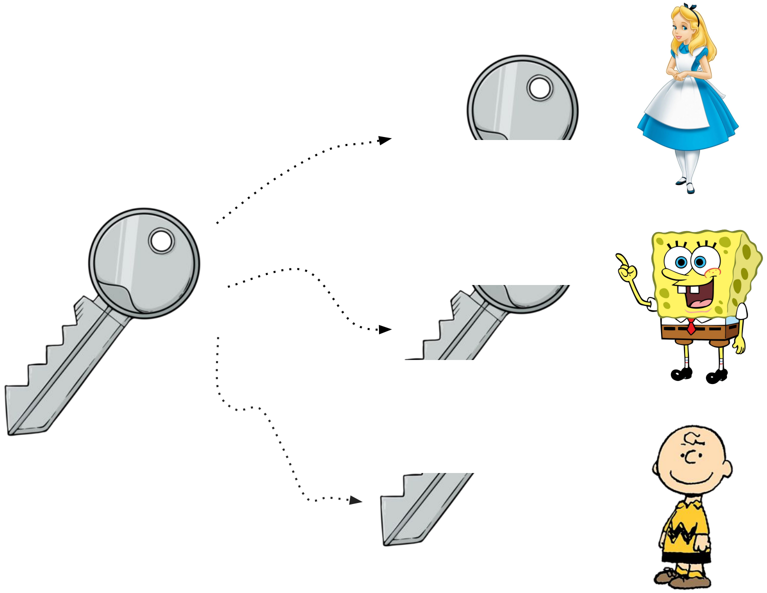
Cryptocurrency Wallets 101

Crypto Wallet Holding a Private Key

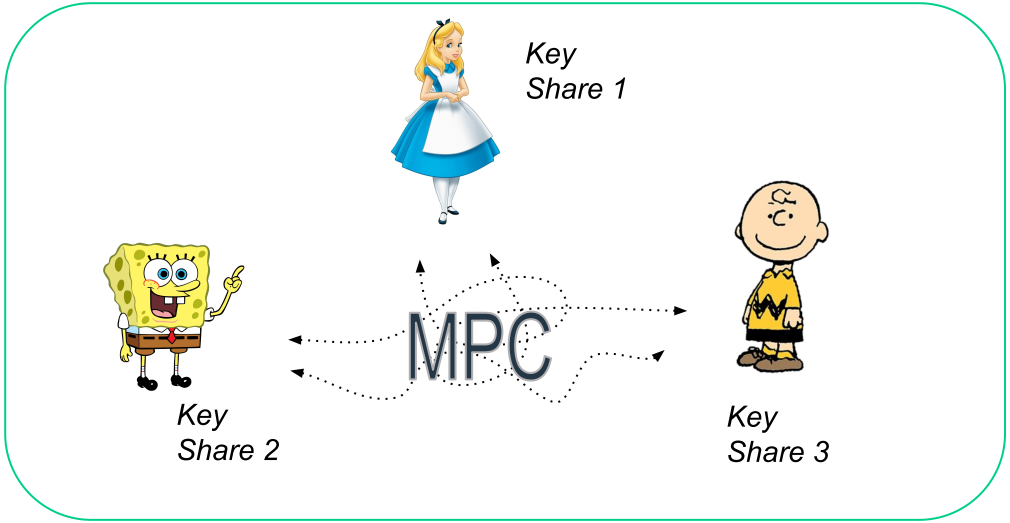
Sign Transaction



MPC
(through the lense of threshold signing)



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 is much bigger than threshold signatures

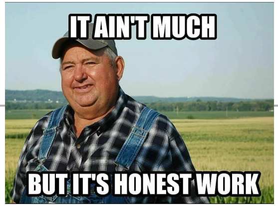
MPC (Multi-Party Computation) is the crown jewel of modern cryptography

Every distributed task can be solved trustlessly with MPC

Auctions *Games of Chance* *Nuclear Deterrence*

Voting *Threshold Signatures* *Private ML*

MPC is invented — 80s



Danish Sugar Beet Auction — 2008

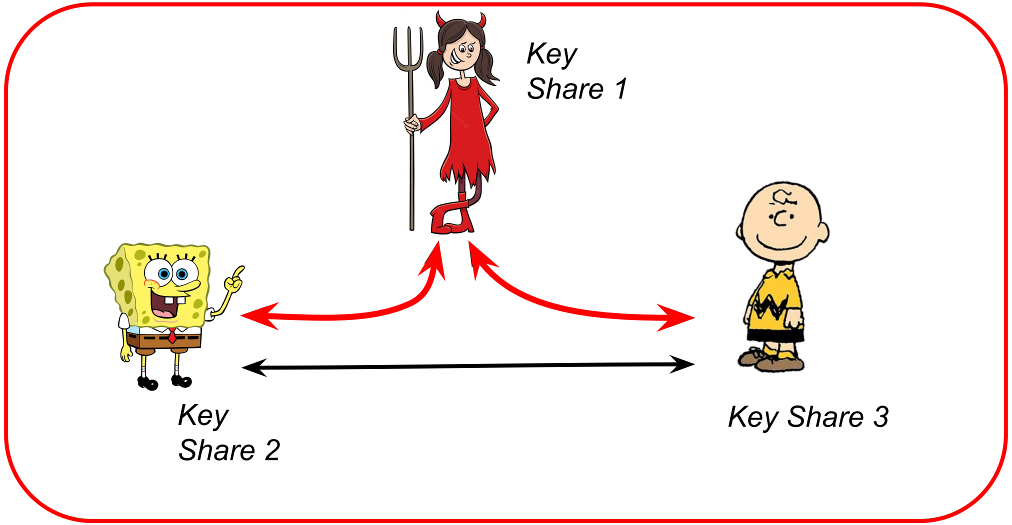
MPC Wallets — Late 10s

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

./ WAC6

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

Affected Parties

- Some of the biggest crypto wallets (e.g. Coinbase WaaS)
- A number of crypto custodians (e.g. BitGo TSS)
- The most popular consumer MPC wallet (e.g. Zengo)
- Some of the most popular open source libraries (e.g. Binance, Apache)

Our Attacks

Today's Talk

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**)

./ WAC6

Background

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

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

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

$$N = p \cdot q$$

$$\mathrm{Dec}(\ldots) = 184 \,\%\, N$$

ECDSA Signature Generation

Ephemeral key

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

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

Private key

ECDSA constant

ECDSA signing with 2 parties

Keys

x

k

Key Shares

x_1, x_2

k_1, k_2

Compromising Lindell17 Implementations

Broken Record Attack

Lindell17 Key Generation (Step 1/2)

Sample key shards

Chooses a random key share

x_1

Chooses a random key share

x_2

Lindell17 Key Generation (Step 2/2)

Saving Bob's key share under HE

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

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

Encrypts their x_2 using their HE key N

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!

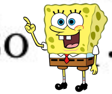
What does the paper say about that?

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

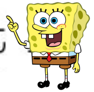
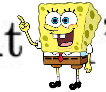
15

Denial-of-Service Attack

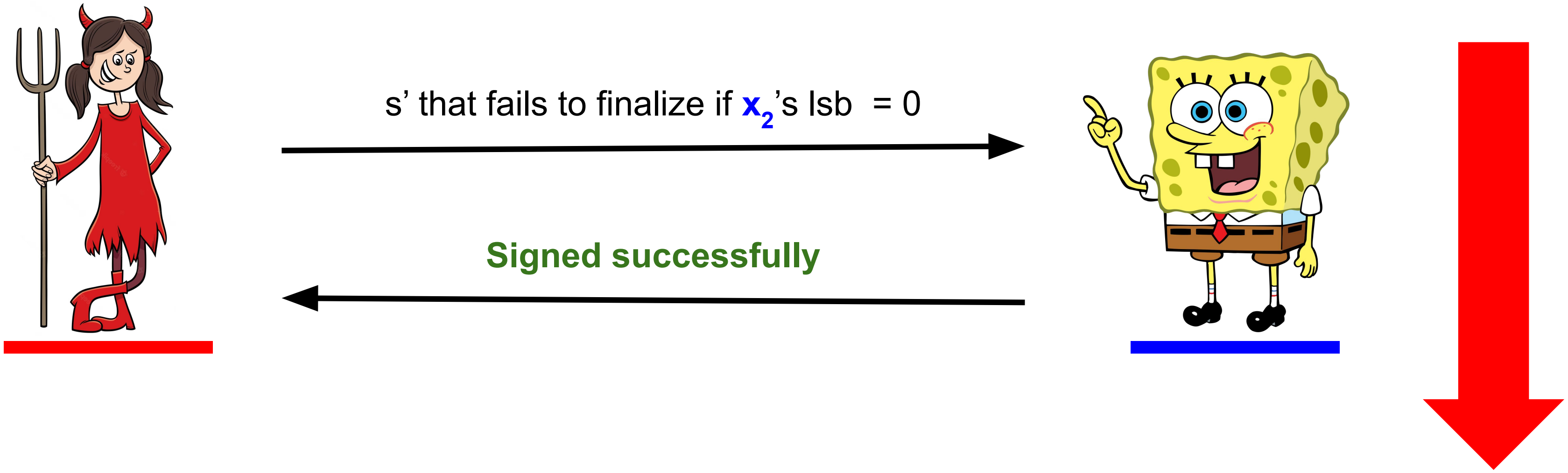
Back to the drawing board

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



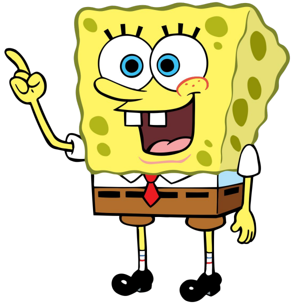
x_2=

0b____
____0

Hypothetical Attack Visualization

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

← **Failed to finalize signature**



x_2=

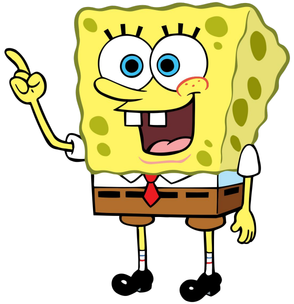
0b____

____10

Hypothetical Attack Visualization

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

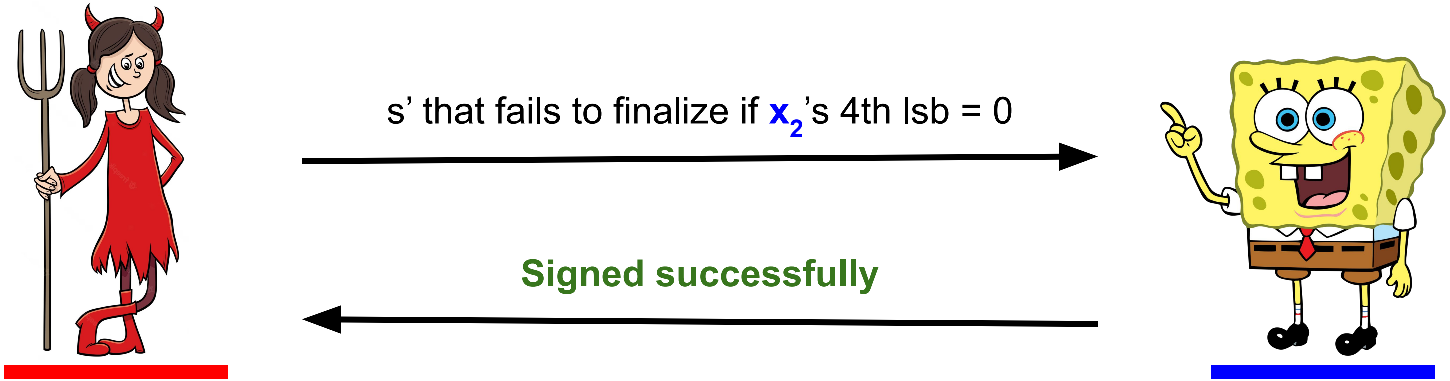
← **Failed to finalize signature**



x_2=

0b____110

Hypothetical Attack Visualization



x_2=

0b____

____0110

256 signatures later...

./ WAC6

Hypothetical Attack Visualization

s' that fails to finalize if msb is 0

Signed successfully

x_2=

0b011100101110100010110011111110100101010011010100000110011101110011001011010100000101011111001000001010000000011100100100011000001
01000101101111010001100111000110110101000110010110010001011000010110110010010100111001000100010110001001000001001111011001001100110

Crafting a malicious partial signature

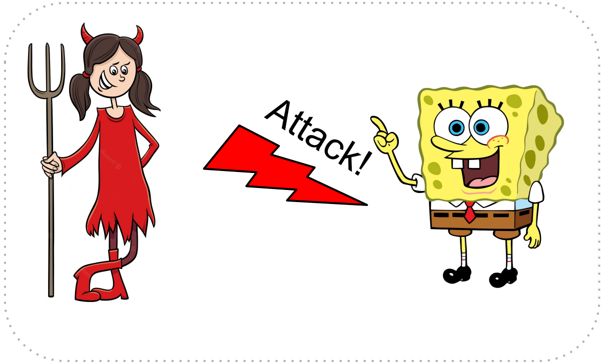
$$(k_1^{-1} \% \ell) \cdot (\text{msg} + x_1 \cdot x_2)$$

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

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

$$\% N$$

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

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

$$+$$

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

The previously leaked bits

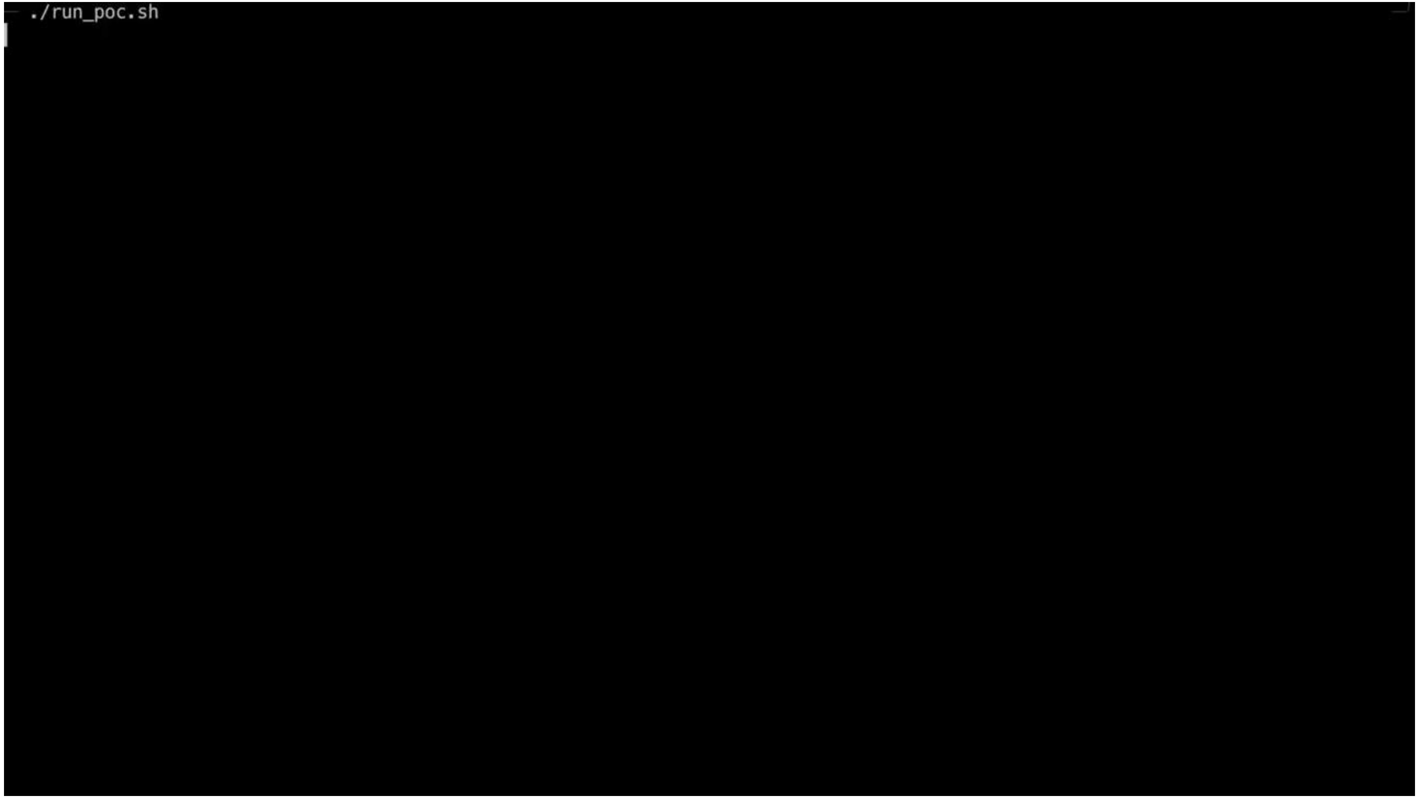
Exfiltrating the i-th bit

$$k_1 = 2^i$$

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

Leakage: i-th bit

./ WAC6



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");
+     }
```

./ WAC6

A Glimpse at the Other Attacks

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

./ WAC6

Concluding Remarks

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