

  
**black hat**<sup>®</sup>  
USA 2023  
**TSSHOCK**



verichains

Breaking MPC Wallets and  
Digital Custodians for \$BILLION\$ Profit  
[verichains.io/tsshock](https://verichains.io/tsshock)

Duy Hieu Nguyen

Anh Khoa Nguyen

Huu Giap Nguyen

Thanh Nguyen

Nguyen Anh Quynh

[info@verichains.io](mailto:info@verichains.io)



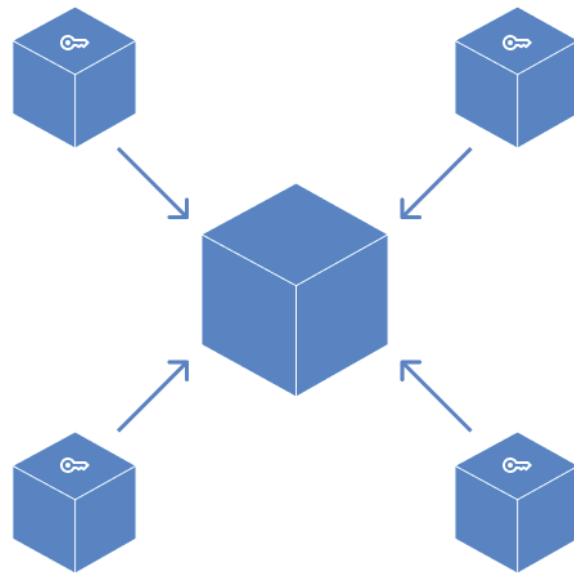
# What is this talk about?

---

Private-key extraction attacks in popular GG18/GG20 based Threshold Signature Scheme (TSS) implementations, a Multi-Party Computation (MPC) protocol.

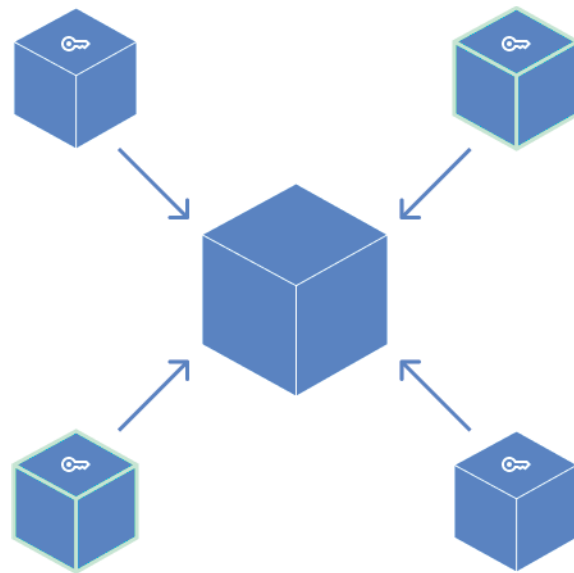
# Multi Party Computation

- Joint Computation with **Private inputs**
- Private inputs are **never** revealed/computed
- **Accurately** calculate the result



# Threshold Signature Scheme

- Joint Computation of **Digital Signatures** with **Private shares**
- Private key is split into multiple key shares
- Private key is never reconstructed by any party
- Threshold  $t/n$  party to produce the signature

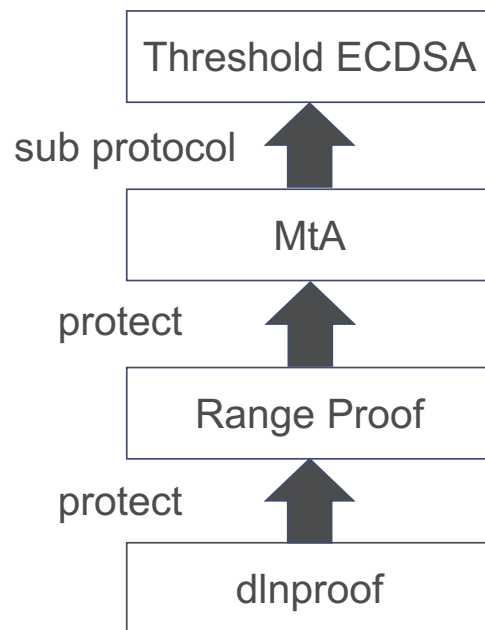


# GG18/GG20 protocols

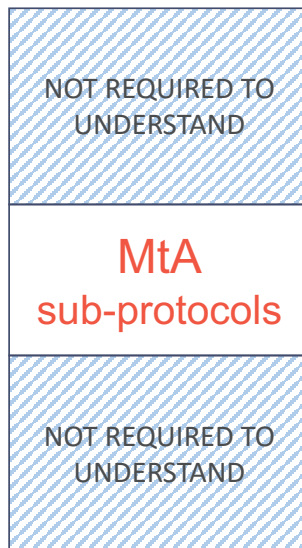
---

- TSS for ECDSA
- Well known in the industry
- Multiple revisions
- Open-sourced implementations
- Being widely used in production

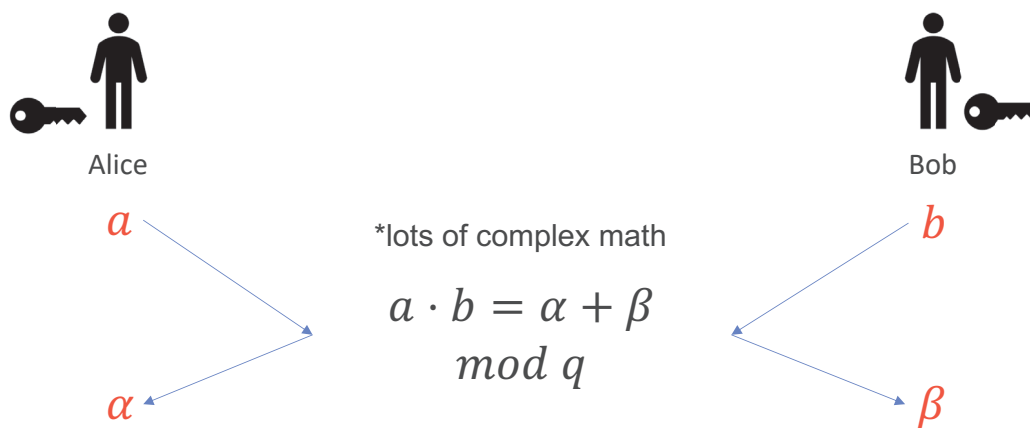
# GG18/GG20 protocols



# Signing ceremony



# MtA sub-protocol

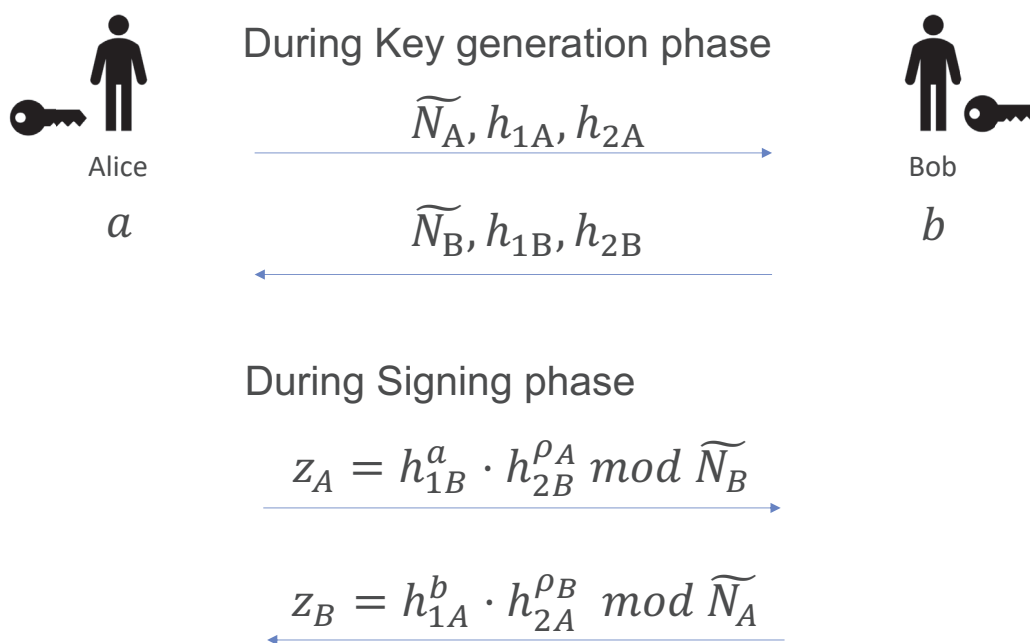


All inputs and outputs are **sensitive** data

This protocol requires a **range proof**!



# Range proof used in MtA sub-protocol



# Range proof used in MtA sub-protocol



$\tilde{N}, h_1, h_2$

Alice receives  $z = h_1^x \cdot h_2^\rho \bmod \tilde{N}$

$x$  is private value of the other party

$\rho$  is random nonce of the other party

$x$  should not be revealed

# Range proof used in MtA sub-protocol



$\tilde{N}, h_1, h_2$

Alice receives  $z = h_1^x \cdot h_2^\rho \bmod \tilde{N}$

$h_1$  is in the multiplicative subgroup generated by  $h_2$

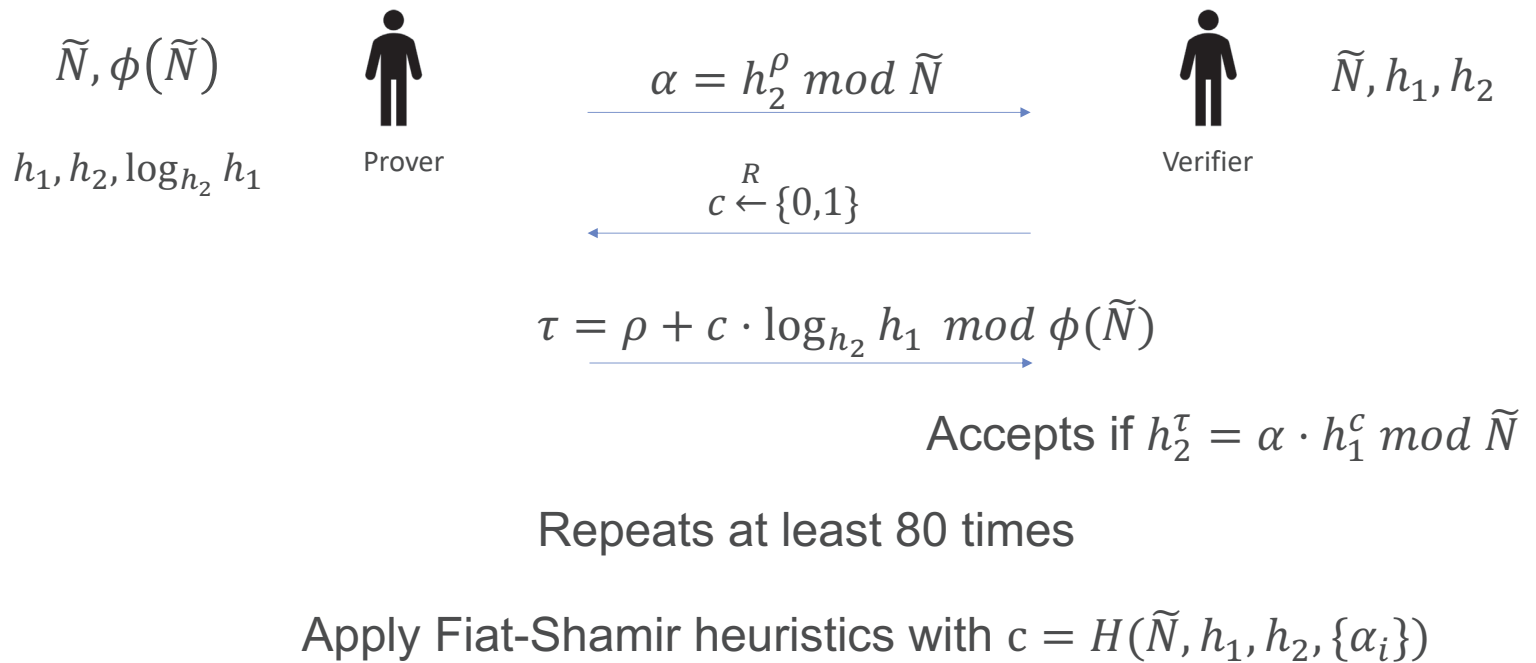
$$h_1 = h_2^e \Rightarrow z = h_2^{ex+\rho} \bmod \tilde{N}$$

$x$  is not revealed when  $\rho$  is big enough

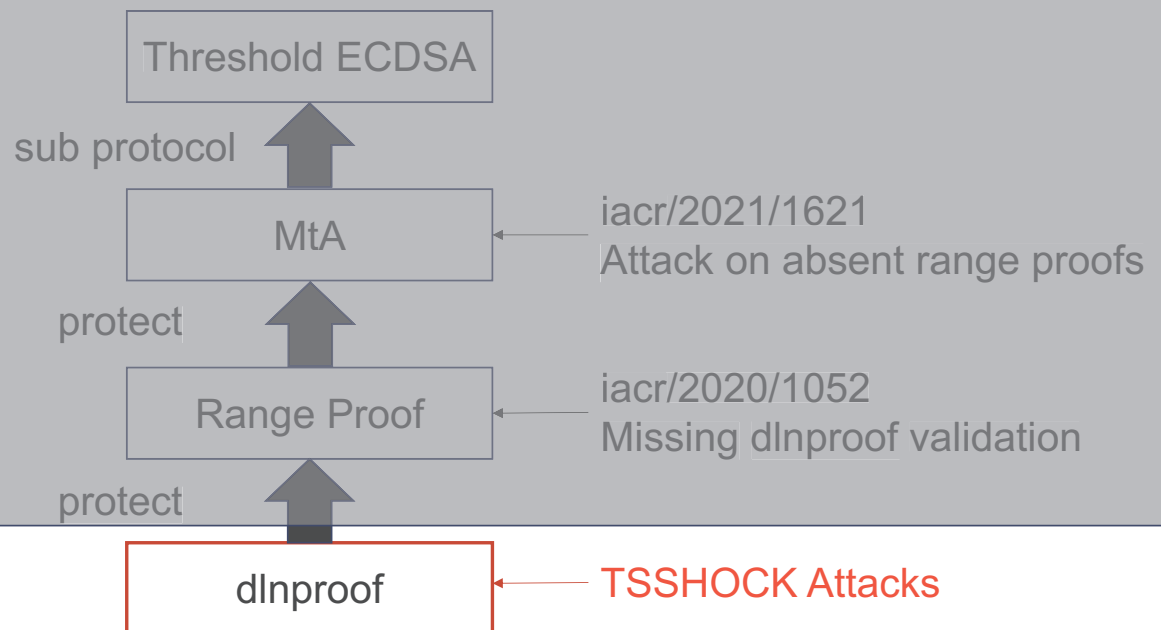
Requires a proof of knowledge of  $\log_{h_2} h_1 \bmod \tilde{N}$

Using dlnproof of  $\log_{h_2} h_1 \bmod \tilde{N}$

# dlnproof



# GG18/GG20 protocols



# TSSHOCK Attacks

Implementation weaknesses found in **dlproof** allows **forging proofs**

Attacks	$\alpha$ -shuffle	All attacks can recover <b>private key</b> Most implementations <ul style="list-style-type: none"><li>- <b>Single malicious party member</b></li><li>- <b>Protocol seamlessly continues with no abort on attack</b></li></ul>
	c-split	
	c-guess	

Many implementations, including de-facto open-source TSS frameworks in Golang and Rust found to be vulnerable.

Widely used in  
MPC Wallet  
Asset Custodian  
Decentralized Bridge

## $\alpha$ -shuffle

---

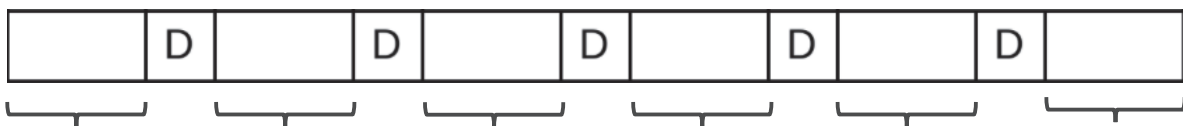
Problem: Ambiguous encoding scheme

```
for i := range in values {  
    data = append(data, values[i])  
    data = append(data, delimiter)  
}
```

# $\alpha$ -shuffle

## Problem: Ambiguous encoding scheme

```
for i := range in values {  
  data = append(data, values[i])  
  data = append(data, delimiter)  
}
```

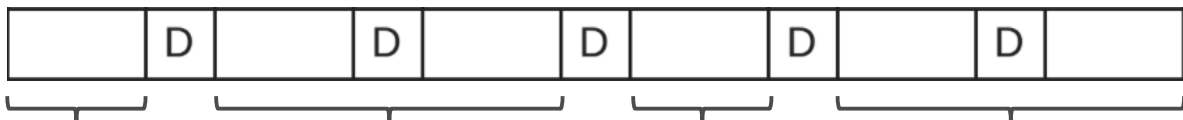




# $\alpha$ -shuffle

## Problem: Ambiguous encoding scheme

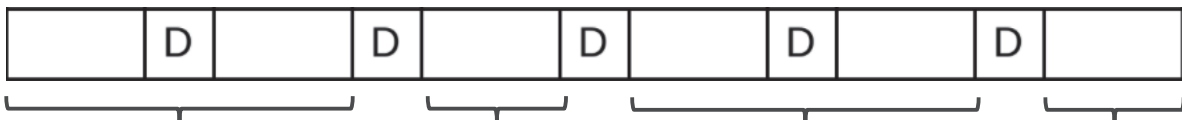
```
for i := range in values {  
  data = append(data, values[i])  
  data = append(data, delimiter)  
}
```



# $\alpha$ -shuffle

## Problem: Ambiguous encoding scheme

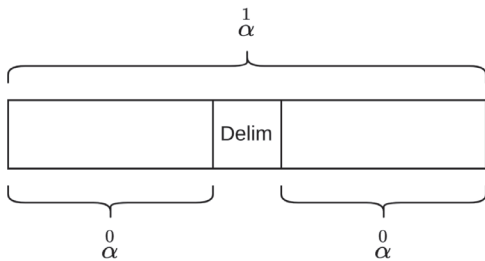
```
for i := range in values {  
  data = append(data, values[i])  
  data = append(data, delimiter)  
}
```



# $\alpha$ -shuffle

Take a guess on how many bits of 0 and 1

Prepare  $\alpha$  for  $c = 1$  and  $c = 0$




---

## Algorithm 1 $\alpha$ -shuffle $\text{dlnproof}$ forging

---

Input:  $g, N$ .

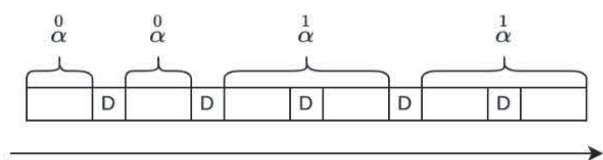
Output:  $h, \text{dlnproof}$  for  $\log_g h \bmod N$ .

1. Let  $\tau = \text{rand}(\mathbb{Z}_{\text{ord}(g)})$ . Let  $\alpha = g^\tau \bmod N$ . Set all  $\tau_i = \tau$ .
  2. Let  $a = \text{bytes}(\alpha)$ . Let  $\beta = \text{int}(a|D|a)$ .
  3. Set  $h = \frac{\alpha}{\beta} \bmod N$  (so that  $\beta = \frac{g^\tau}{h} \bmod N$ ).
  4. For  $l$  in  $\{0, 1, 2, \dots, \lambda\}$ :
    - (a) Temporarily set  $\alpha_i = \begin{cases} \alpha & (1 \leq i \leq l) \\ \beta & (l+1 \leq i \leq \lambda) \end{cases}$  (assign  $\alpha$  to  $l$  first  $\alpha_i$  and  $\beta$  to the remaining).
    - (b) Let  $c_1, c_2, \dots, c_\lambda = H(g, h, N, \alpha_1, \alpha_2, \dots, \alpha_\lambda)$ .
    - (c) If  $\sum c_i = \lambda - l$  (there are  $l$  challenge bits equal to 0), set  $\alpha_i = \begin{cases} \alpha & (c_i = 0) \\ \beta & (c_i = 1) \end{cases}$  and return.
  5. Go back to step 1.
-

# $\alpha$ -shuffle

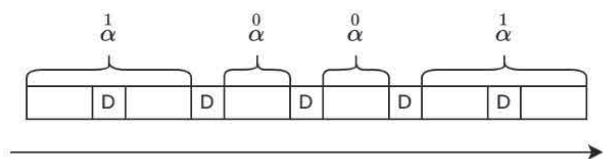


Prover



Verifier

1,0,0,1



1,0,0,1

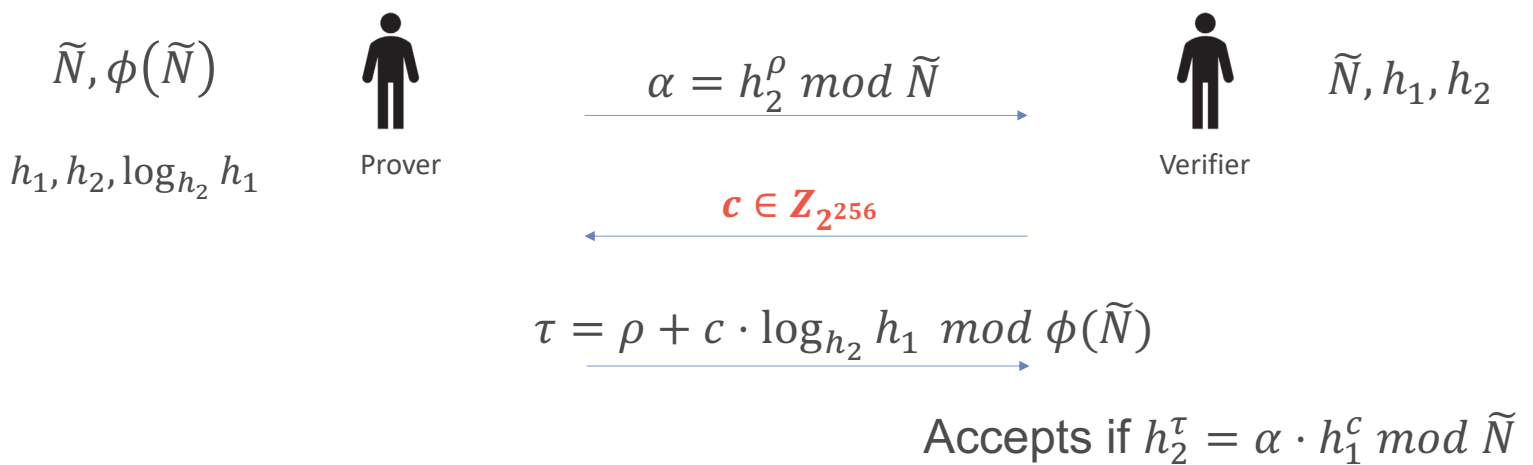


## c-split

---

Problem: Use a larger challenge space with  
no dlnproof iteration

# c-split



No iteration

Apply Fiat-Shamir heuristics with  $c = H(\tilde{N}, h_1, h_2, \{\alpha_i\})$

## c-split

$$\tau = \rho + c \cdot \log_{h_2} h_1 \bmod \phi(\tilde{N})$$

If  $\log_{h_2} h_1 = \frac{1}{2}$  and  $c$  is an even number,  $\tau$  exists, proof exists

If  $\log_{h_2} h_1 = \frac{1}{e}$  and  $c$  divides  $e$ ,  $\tau$  exists, proof exists

It should be noted that  $\frac{1}{e}$  is non-existent in group  $\phi(\tilde{N})$  if  $\gcd(e, \phi(\tilde{N})) \neq 1$

But if  $c$  divides  $e$ , the proof can be calculated

Probability for a random  $c$  divides  $e$  is  $\frac{1}{e} \rightarrow$  Brute force  $c$

# c-split

---

Using brute force so  $e$  should be small 30-50 bits (computing power)

Private inputs extracted are in  $\text{mod } e$

$e$  is small so the value cannot be fully extracted from 1 signature

Acquire more signature(s) and use lattice attack\* to recover full value

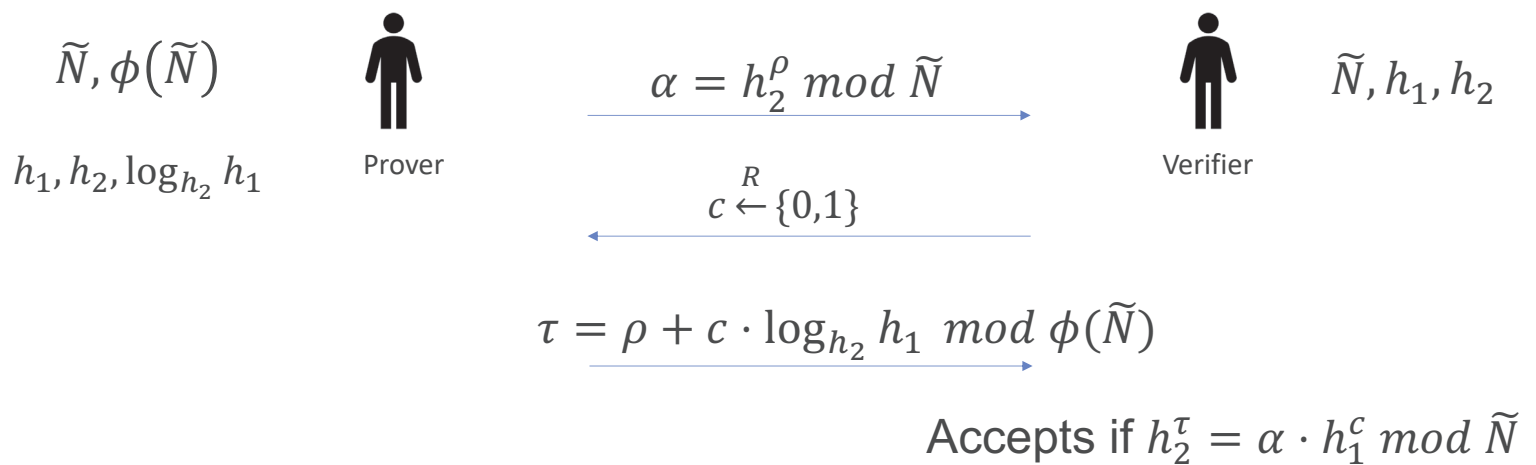
\*Similar to nonce leakage attack on ECDSA



**c-guess**

Problem: Reduction of dlnproof iterations

## c-guess



Repeat fewer times

Apply Fiat-Shamir heuristics with  $c = H(\tilde{N}, h_1, h_2, \{\alpha_i\})$

# c-guess

---

Predictable challenge is insecure

Low rounds number can be brute force for all challenge bits

Probability for a successful guess is  $\frac{1}{2^{\text{iterations}}}$

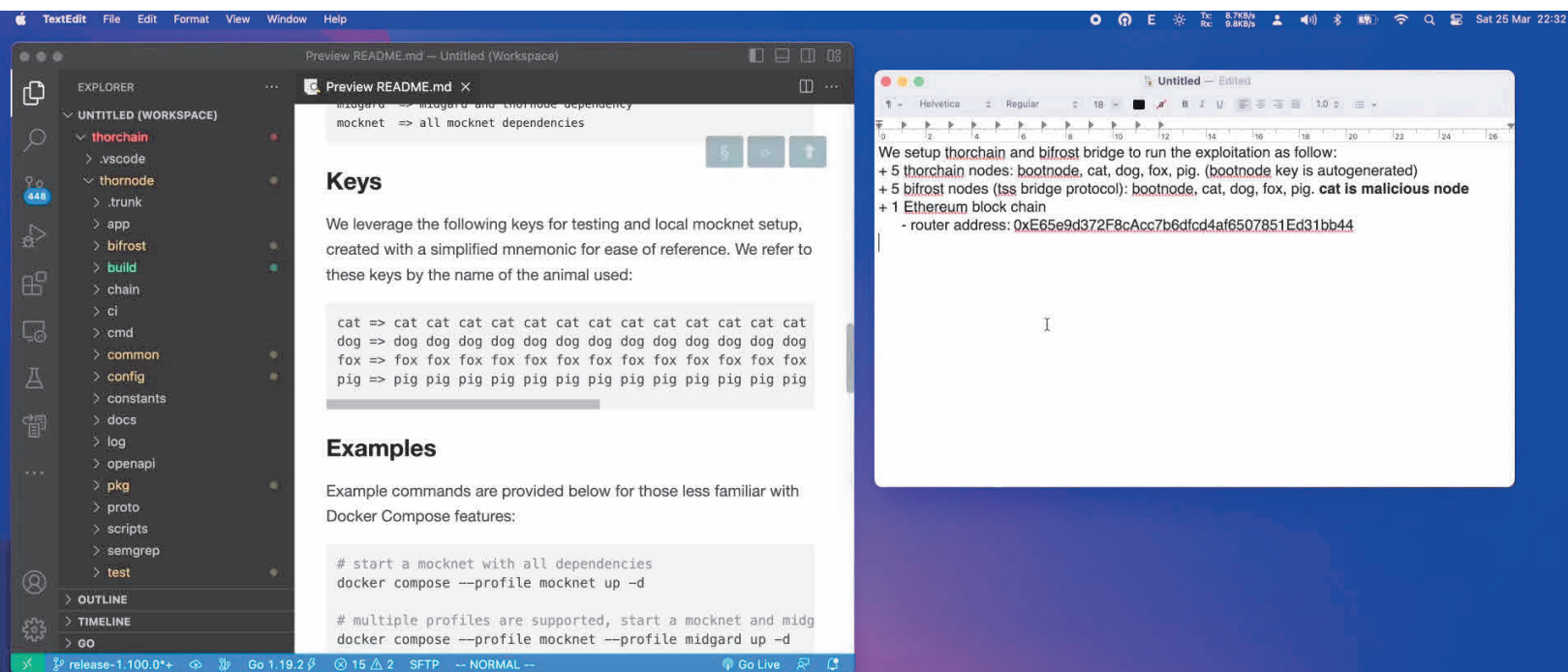
# Recap attacks

Bug	Attack	Why?
Ambiguous encoding scheme	$\alpha$ -shuffle	Same encoding for different integer lists
Reduction of dlnproof iterations	c-guess	Easily guess challenge bits for a small number of iterations
Use a larger challenge space with no dlnproof iterations	c-split	Optimize the scheme without proving its <b>soundness error is negligible</b>

# Affected Vendors/Libraries

Implementations	Attack Technique	PoC	Required number of		
			Malicious parties	(Re)sharing ceremonies	Signing ceremonies
Axelar (tofn)	c-split	YES	1	1	2
Binance/BNBChain (tss-lib)	$\alpha$ -shuffle	YES	1	1	1
ING Bank (threshold-signatures)	c-split	YES	1	1	2
Keep Network/Threshold Network	$\alpha$ -shuffle	YES	1	1	1
Multichain (fastMPC)	$\alpha$ -shuffle	YES	1	1	1
	c-guess	YES	1	1	1
Swingby (tss-lib)	$\alpha$ -shuffle	YES	1	1	1
Taurus (multi-party-sig)	$\alpha$ -shuffle	YES	1	1.5526	1
Thorchain (tss-lib)	$\alpha$ -shuffle	YES	1	1	1
ZenGo X (multi-party-ecdsa)	c-split	YES	2	1	1

# DEMO



THORChain mainnet halted globally after our report.

Our PoC exploit could steal all assets from THORChain's vaults (US\$180M TVL) via a single malicious node.



MAR 28, 2023

## THORChain mainnet halted amid new vulnerability reports

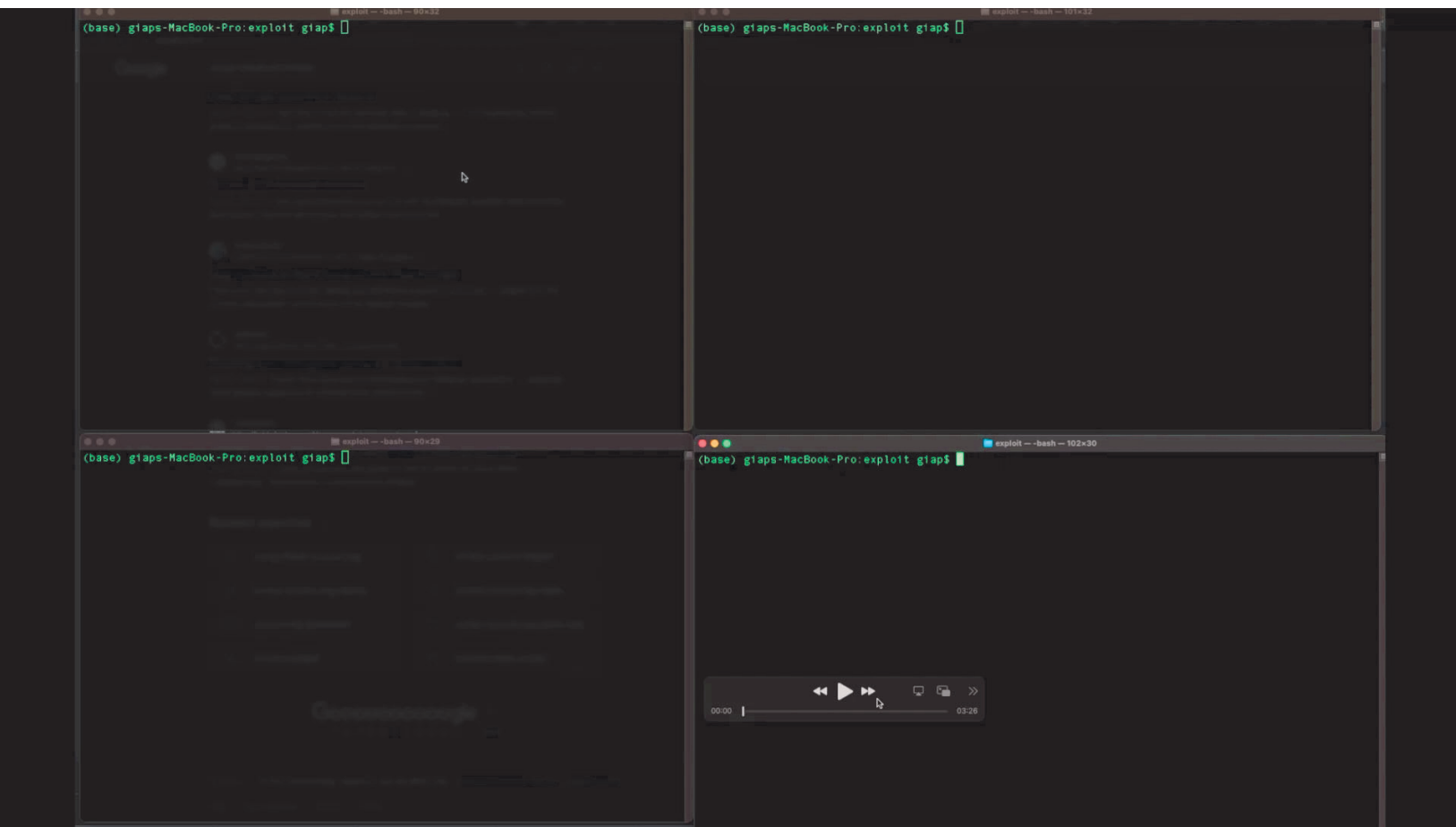
THORChain has once again halted its network, taking action as a precautionary measure while verifying reports on a potential network vulnerability.

Cross-chain liquidity protocol THORChain has paused its network due to new claims of a potential network vulnerability.

THORChain took to Twitter on March 28 to announce it has halted all trading amid reports of a potential vulnerability with a THORChain dependency that may affect the network. The decision was taken as a precautionary measure while the reports are verified, THORChain said.

The announcement came soon after social media reports indicated THORChain's liquidity platform Nine Realms and the dedicated security team THORSec received "credible reports" of a potential vulnerability affecting THORChain. The THORChain network has reportedly been subsequently halted globally.





# Conclusions

---

- Implement new and complex cryptography protocols can be extremely **challenging** and **dangerous**.
- Optimizations for cryptographic protocols can be also extremely **challenging** and **dangerous**.
- New cryptographic protocols must undergo a rigorous security evaluation before widely used in production.
- MPC/TSS is pretty new and has not been standardized yet.
  - prone to new vulnerabilities.
  - gradually more secure by being battle-tested and challenged by new attacks

  
**black hat**<sup>®</sup>  
USA 2023  
**TSSHOCK**



verichains

Breaking MPC Wallets and  
Digital Custodians for \$BILLION\$ Profit  
[verichains.io/tsshock](https://verichains.io/tsshock)

Duy Hieu Nguyen

Anh Khoa Nguyen

Huu Giap Nguyen

Thanh Nguyen

Nguyen Anh Quynh

Q&A

[info@verichains.io](mailto:info@verichains.io)



verichains