black hat USA 2023

verichains

TSSHOCK

Breaking MPC Wallets and Digital Custodians for $BILLION$ Profit

verichains.io/tsshock

Duy Hieu Nguyen

Anh Khoa Nguyen

Huu Giap Nguyen

Thanh Nguyen

Nguyen Anh Quynh

info@verichains.io

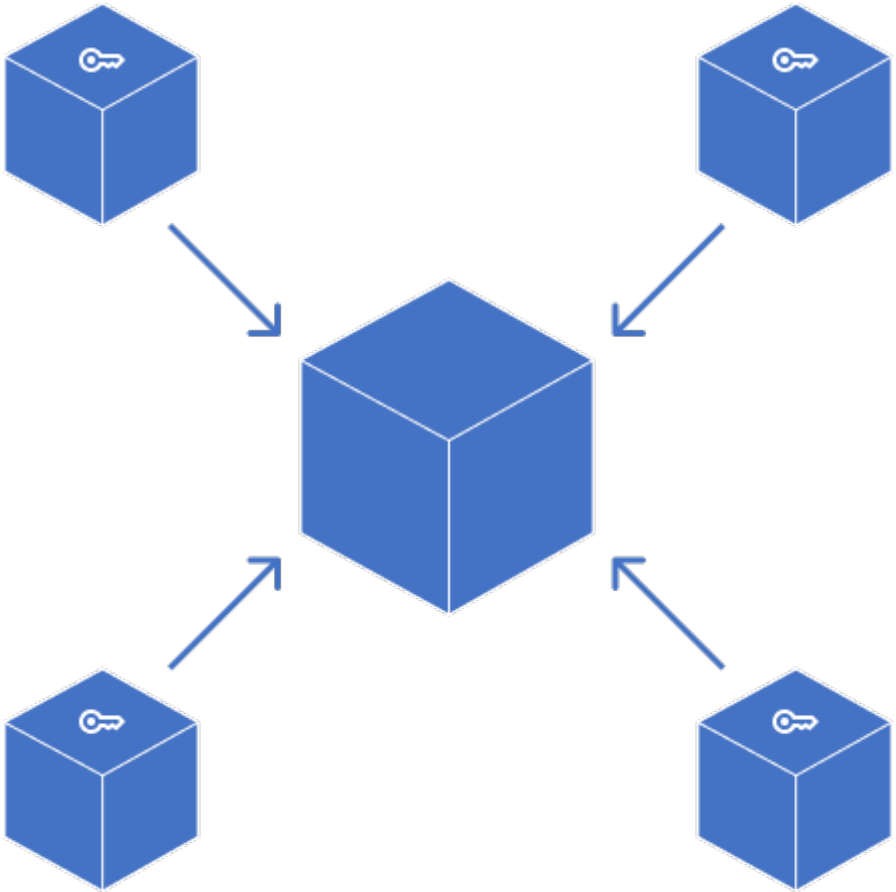
What is this talk about?

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

verichains

Multi Party Computation

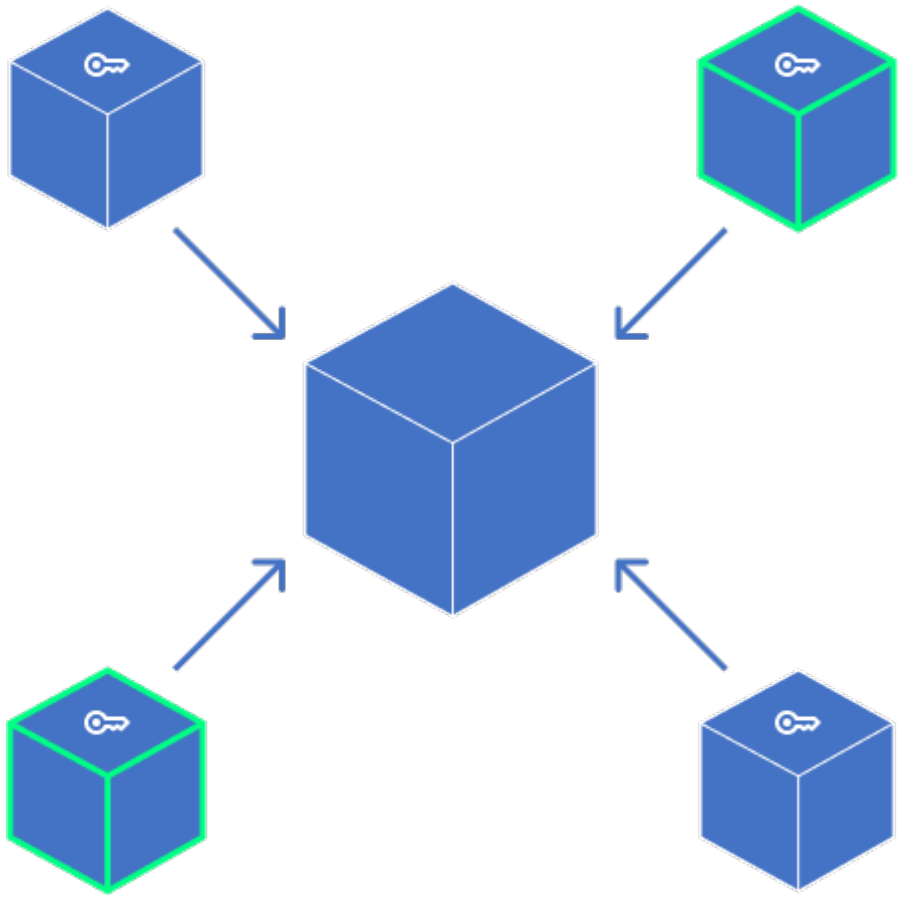
- Joint Computation with **Private inputs**
- Private inputs are **never** revealed/computed
- **Accurately** calculate the result



verichains

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

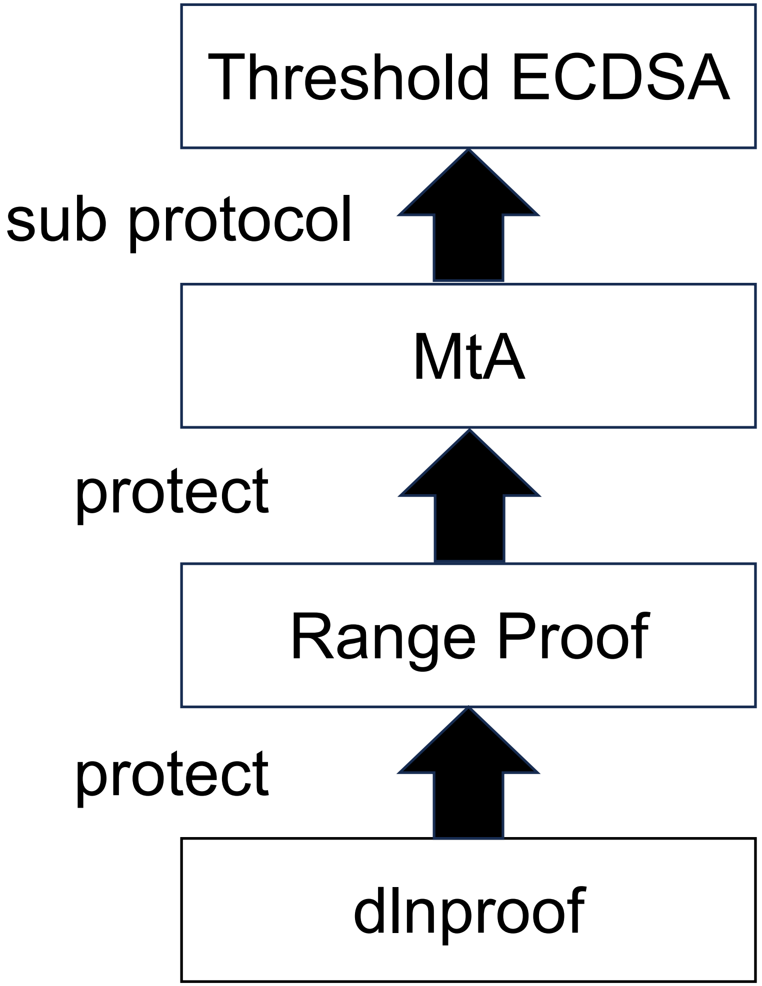


GG18/GG20 protocols

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

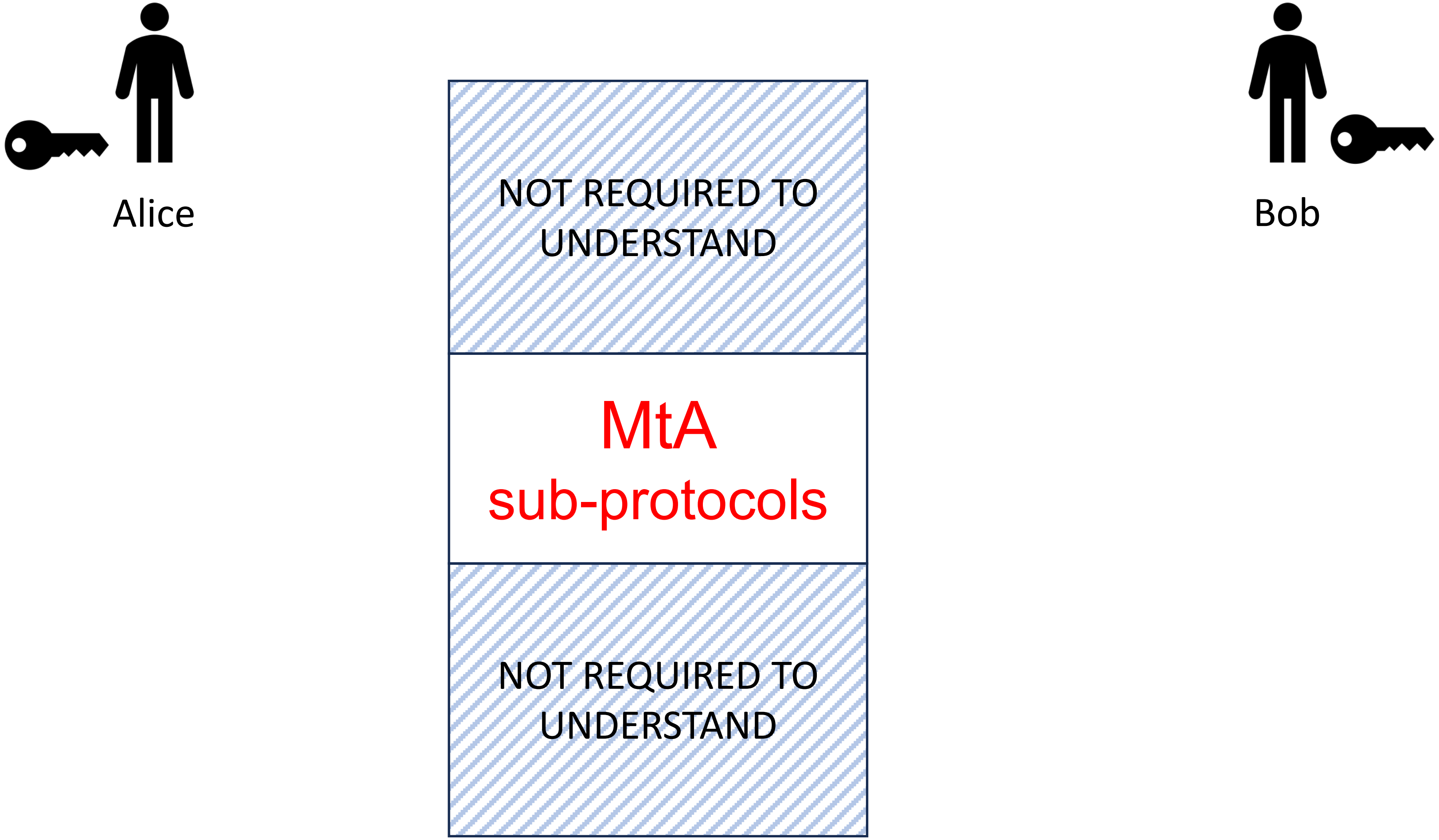
verichains

GG18/GG20 protocols

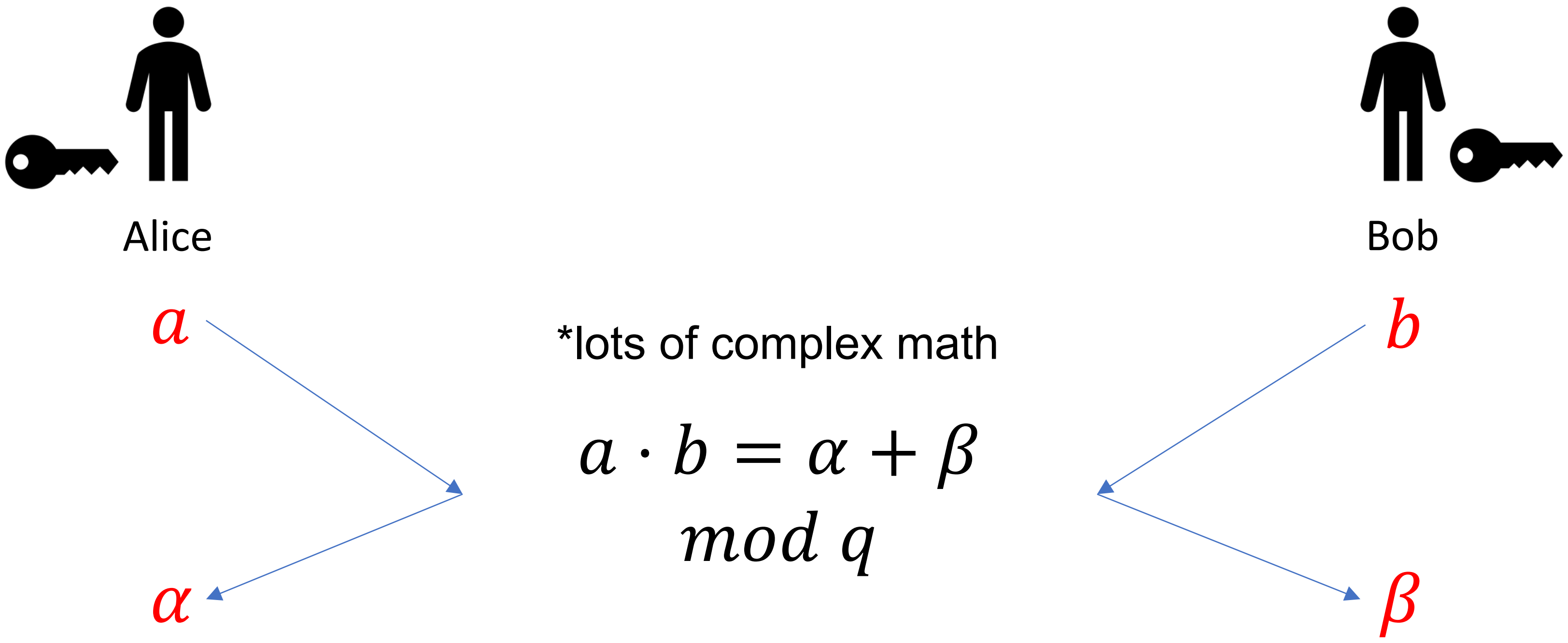


verichains

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

Alice

a

Bob

b

During Key generation phase

Alice → Bob: $\widetilde{N}_{\mathrm{A}}, h_{1\mathrm{A}}, h_{2\mathrm{A}}$

Bob → Alice: $\widetilde{N}_{\mathrm{B}}, h_{1\mathrm{B}}, h_{2\mathrm{B}}$

During Signing phase

Alice → Bob: $z_A = h_{1B}^{a} \cdot h_{2B}^{\rho_A} \; mod \; \widetilde{N_B}$

Bob → Alice: $z_B = h_{1A}^{b} \cdot h_{2A}^{\rho_B} \; mod \; \widetilde{N_A}$

Range proof used in MtA sub-protocol

$\tilde{N}, h_1, h_2$

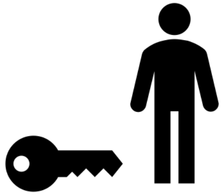
Alice receives $z = h_1^x \cdot h_2^\rho \ mod \ \tilde{N}$

x is private value of the other party

ρ is random nonce of the other party

x should not be revealed

Range proof used in MtA sub-protocol



Alice

$\tilde{N}, h_1, h_2$

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

h_1 is in the multiplicative subgroup generated by h_2

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

x is not revealed when ρ is big enough

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

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

verichains

dlnproof

$\tilde{N}, \phi(\tilde{N})$

$h_1, h_2, \log_{h_2} h_1$

Prover

$\alpha = h_2^{\rho} \bmod \tilde{N}$ →

$c \overset{R}{\leftarrow} \{0,1\}$ ←

Verifier

$\tilde{N}, h_1, h_2$

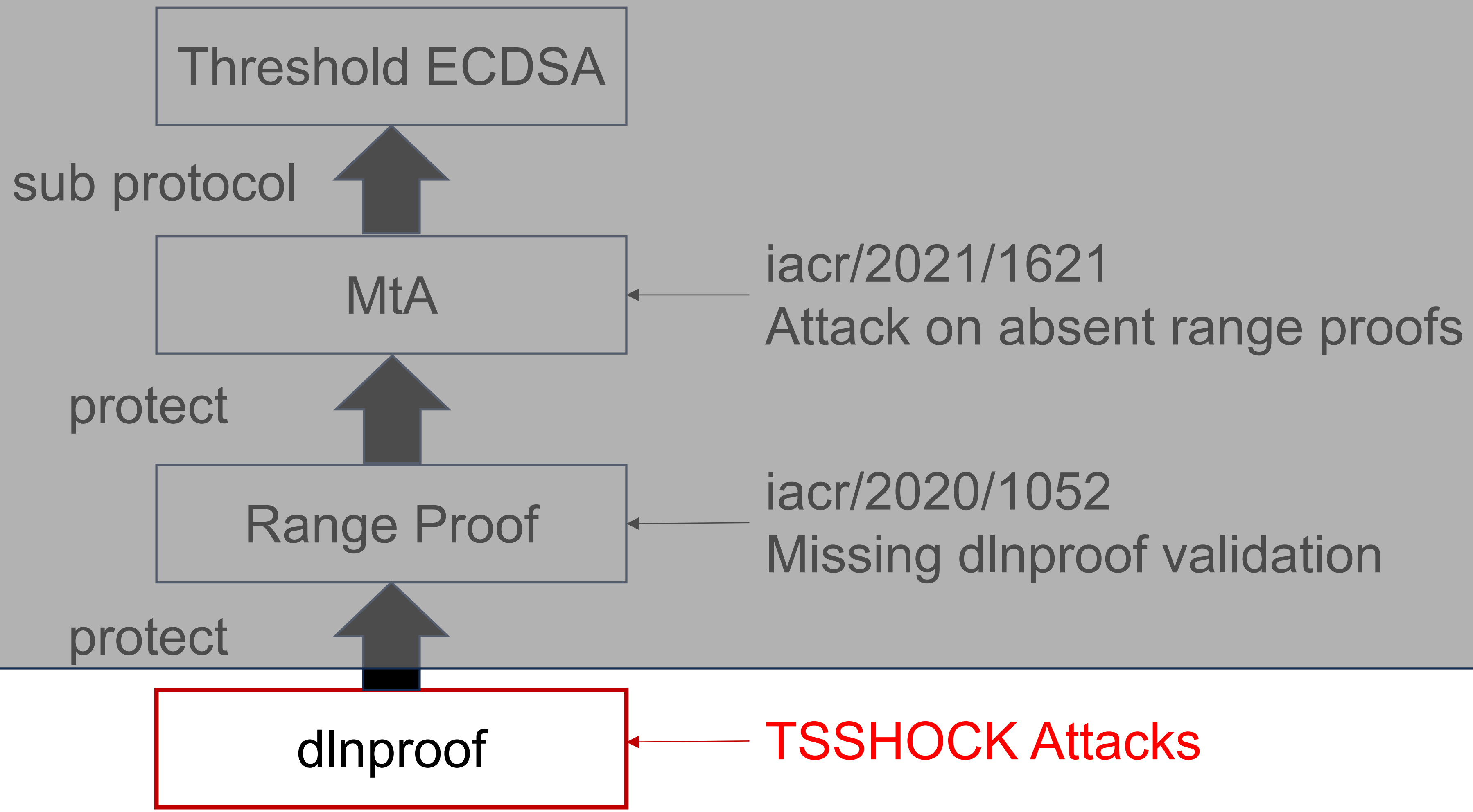
$\tau = \rho + c \cdot \log_{h_2} h_1 \bmod \phi(\tilde{N})$ →

Accepts if $h_2^{\tau} = \alpha \cdot h_1^{c} \bmod \tilde{N}$

Repeats at least 80 times

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

GG18/GG20 protocols



TSSHOCK Attacks

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

Attacks: α-shuffle, c-split, c-guess

All attacks can recover **private key**

Most implementations
- **Single malicious party member**
- **Protocol seamlessly continues with no abort on attack**

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

verichains

α-shuffle

Problem: Ambiguous encoding scheme

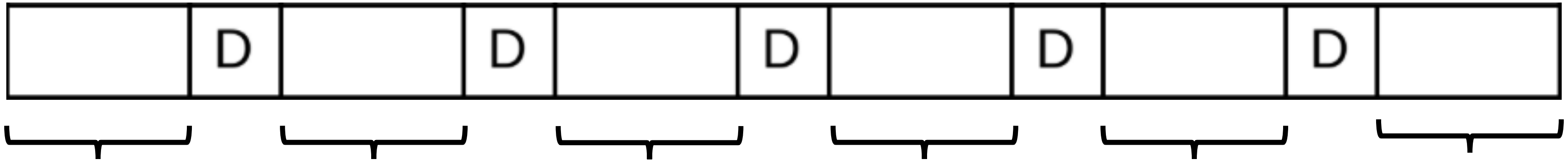
```
for i := range in values {
  data = append(data, values[i])
  data = append(data, delimiter)
}
```

verichains

α-shuffle

Problem: Ambiguous encoding scheme

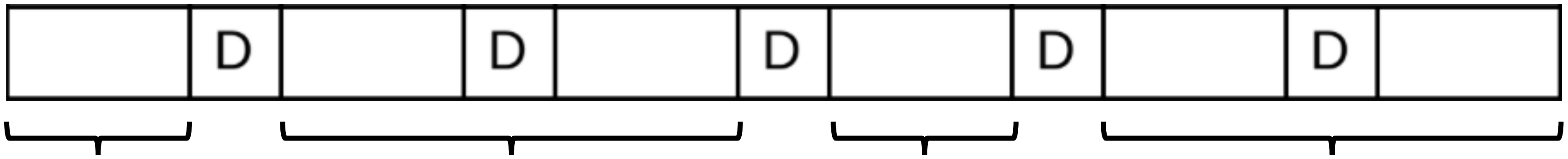
```
for i := range in values {
  data = append(data, values[i])
  data = append(data, delimiter)
}
```



α-shuffle

Problem: Ambiguous encoding scheme

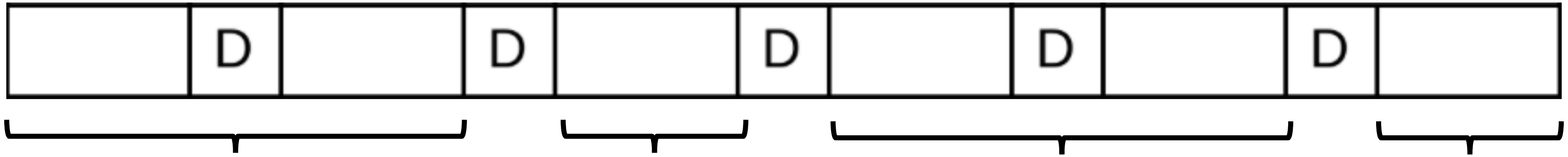
```
for i := range in values {
  data = append(data, values[i])
  data = append(data, delimiter)
}
```



α-shuffle

Problem: Ambiguous encoding scheme

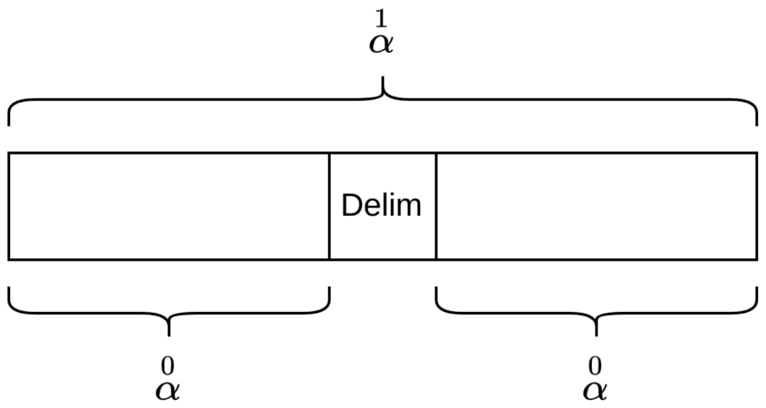
```
for i := range in values {
  data = append(data, values[i])
  data = append(data, delimiter)
}
```



α-shuffle

Take a guess on how many bits of 0 and 1

Prepare α for $c = 1$ and $c = 0$



Algorithm 1 α-shuffle `dlnproof` forging

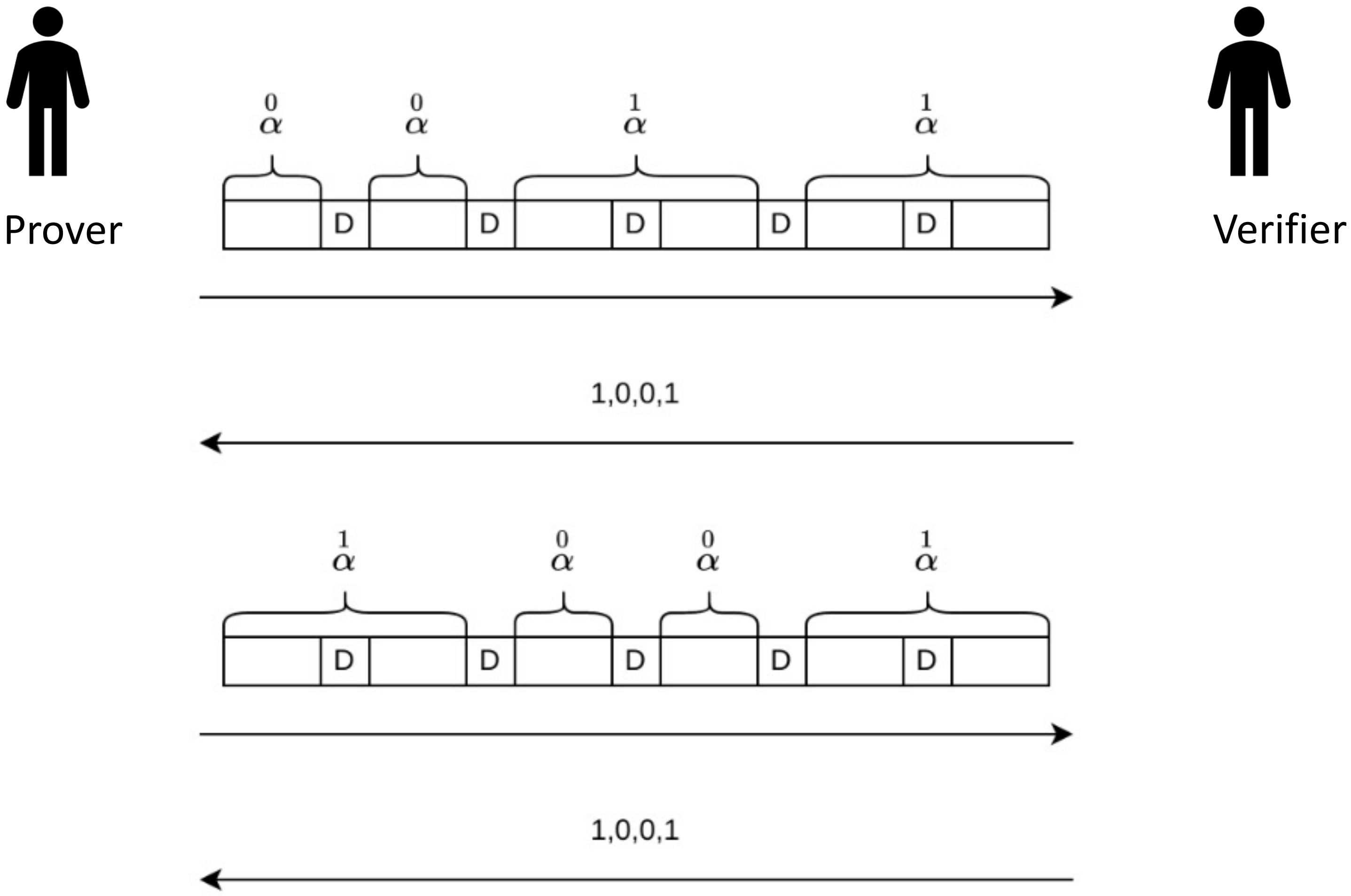
Input: g, N.

Output: h, `dlnproof` for $\log_g h \bmod N$.

1. Let $\tau = \texttt{rand}(\mathbb{Z}_{\text{ord}(g)})$. Let $\alpha = g^\tau \bmod N$. Set all $\tau_i = \tau$.
2. Let $a = \texttt{bytes}(\alpha)$. Let $\beta = \texttt{int}(a|D|a)$.
3. Set $h = \frac{\alpha}{\beta} \bmod N$ (so that $\beta = \frac{g^\tau}{h} \bmod N$).
4. For l in $\{0, 1, 2, ..., \lambda\}$:
   (a) Temporarily set $\alpha_i = \begin{cases} \alpha & (1 \le i \le l) \\ \beta & (l+1 \le i \le \lambda) \end{cases}$ (assign α to l first α_i and β to the remaining).
   (b) Let $c_1, c_2, ..., c_\lambda = H(g, h, N, \alpha_1, \alpha_2, ..., \alpha_\lambda)$.
   (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.

verichains

α-shuffle



c-split

Problem: Use a larger challenge space with no dlnproof iteration

verichains

c-split

$\tilde{N}, \phi(\tilde{N})$

$h_1, h_2, \log_{h_2} h_1$

Prover

$\alpha = h_2^{\rho} \; mod \; \tilde{N}$

Verifier

$\tilde{N}, h_1, h_2$

$\boldsymbol{c \in Z_{2^{256}}}$

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

Accepts if $h_2^{\tau} = \alpha \cdot h_1^{c} \; mod \; \tilde{N}$

No iteration

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

c-split

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

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

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

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

But if c divides e , the proof can be calculated

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

verichains

c-split

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

Private inputs extracted are in $mod\ e$

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

$\tilde{N}, \phi(\tilde{N})$

$h_1, h_2, \log_{h_2} h_1$

Prover

$\alpha = h_2^{\rho} \ mod \ \tilde{N}$ →

$c \xleftarrow{R} \{0,1\}$ ←

Verifier

$\tilde{N}, h_1, h_2$

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

Accepts if $h_2^{\tau} = \alpha \cdot h_1^{c} \ mod \ \tilde{N}$

Repeat fewer times

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

verichains

c-guess

Predictable challenge is insecure

Low rounds number can be brute force for all challenge bits

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

Recap attacks

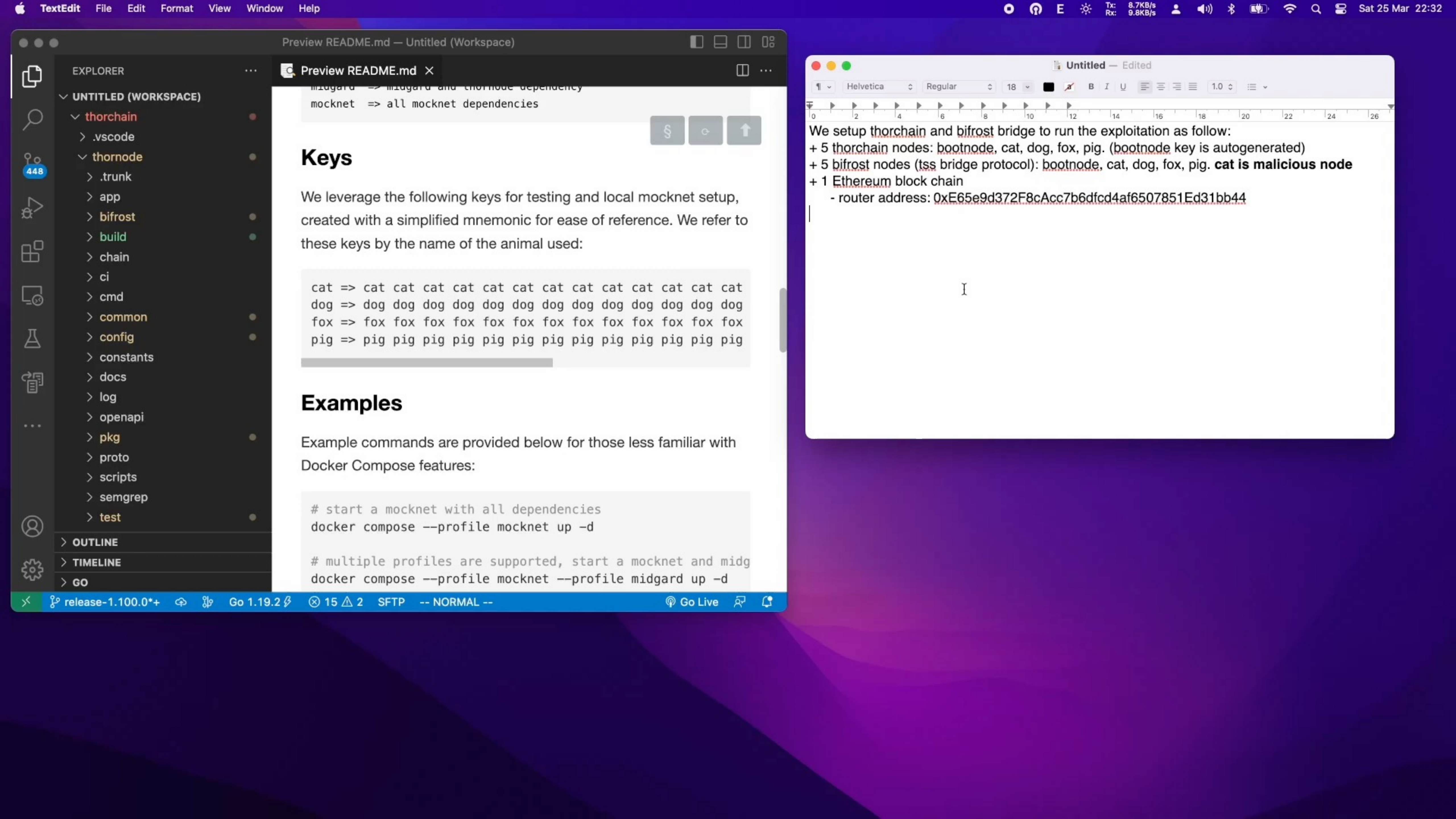
Bug	Attack	Why?
Ambiguous encoding scheme	α-shuffle	Same encoding for different integer lists
Reduction of dlnproof iterations	c-guess	Easily guess challenge bits for a small number of iterations
Use a larger challenge space with no dlnproof iterations	c-split	Optimize the scheme without proving its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

verichains

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)	α-shuffle	YES	1	1	1
ING Bank (threshold-signatures)	c-split	YES	1	1	2
Keep Network/Threshold Network	α-shuffle	YES	1	1	1
Multichain (fastMPC)	α-shuffle	YES	1	1	1
	c-guess	YES	1	1	1
Swingby (tss-lib)	α-shuffle	YES	1	1	1
Taurus (multi-party-sig)	α-shuffle	YES	1	1.5526	1
Thorchain (tss-lib)	α-shuffle	YES	1	1	1
ZenGo X (multi-party-ecdsa)	c-split	YES	2	1	1

DEMO



TextEdit File Edit Format View Window Help
Sat 25 Mar 22:32
Preview README.md — Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
thorchain
.vscode
thornode
.trunk
app
bifrost
build
chain
ci
cmd
common
config
constants
docs
log
openapi
pkg
proto
scripts
semgrep
test
OUTLINE
TIMELINE
GO
Preview README.md
mocknet => all mocknet dependencies
Keys
We leverage the following keys for testing and local mocknet setup, created with a simplified mnemonic for ease of reference. We refer to these keys by the name of the animal used:
cat => cat cat cat cat cat cat cat cat cat cat cat cat cat
dog => dog dog dog dog dog dog dog dog dog dog dog dog dog
fox => fox fox fox fox fox fox fox fox fox fox fox fox fox
pig => pig pig pig pig pig pig pig pig pig pig pig pig pig
Examples
Example commands are provided below for those less familiar with Docker Compose features:
start a mocknet with all dependencies
docker compose --profile mocknet up -d
multiple profiles are supported, start a mocknet and midg
docker compose --profile mocknet --profile midgard up -d
release-1.100.0*+
Go 1.19.2
15 2
SFTP
-- NORMAL --
Go Live
Untitled — Edited
We setup thorchain and bifrost bridge to run the exploitation as follow:
+ 5 thorchain nodes: bootnode, cat, dog, fox, pig. (bootnode key is autogenerated)
+ 5 bifrost nodes (tss bridge protocol): bootnode, cat, dog, fox, pig. cat is malicious node
+ 1 Ethereum block chain
- router address: 0xE65e9d372F8cAcc7b6dfcd4af6507851Ed31bb44

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.

COINTELEGRAPH

$ | BTC $29,614 | ETH $1,861 | BNB $243 | XRP $0.64 | ADA $0.300 | USDC $1.00 | USDT

HELEN PARTZ

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.

verichains

exploit — -bash — 90×32

```
(base) giaps-MacBook-Pro:exploit giap$
```

exploit — -bash — 101×32

```
(base) giaps-MacBook-Pro:exploit giap$
```

exploit — -bash — 90×29

```
(base) giaps-MacBook-Pro:exploit giap$
```

exploit — -bash — 102×30

```
(base) giaps-MacBook-Pro:exploit giap$
```

00:00 03:26

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

verichains

black hat USA 2023

verichains

TSSHOCK

Breaking MPC Wallets and Digital Custodians for $BILLION$ Profit

verichains.io/tsshock

Duy Hieu Nguyen

Anh Khoa Nguyen

Huu Giap Nguyen

Thanh Nguyen

Nguyen Anh Quynh

Q&A

info@verichains.io