TSSHOCK: Tấn công phá vỡ tính an toàn của một số cài đặt lược đồ chữ ký số ngưỡng trong ví MPC

Một trong những tham luận thu hút sự quan tâm lớn của giới bảo mật tại Hội nghị bảo mật hàng đầu thế giới Black Hat USA 2023 là tấn công TSSHOCK của nhóm nghiên cứu mật mã đến từ công ty Verichains (Việt Nam). Đáng lưu ý, tấn công này cho phép một node ác ý có thể đánh cắp on-chain tài sản mã hoá giá trị hàng triệu đến hàng tỉ USD trên các dịch vụ này.

TỔNG QUAN

Cốt lõi của ví tính toán đa bên (Multi-Party Computation - MPC) hiện đại và giải pháp lưu ký tài sản số của các chuỗi khối chính là một giao thức mật mã có tên là lược đồ chữ ký số ngưỡng (Threshold Signature Scheme - TSS).

Ngày nay, nhiều tổ chức bao gồm ngân hàng, sàn giao dịch và ví đều dựa vào TSS để cho phép các bên ủy quyền giao dịch bằng cách tạo chữ ký mà không tiết lộ khóa bí mật cá nhân của mỗi bên. Do đó, tính an toàn của TSS là vô cùng quan trọng đối với nhiều hệ sinh thái tài chính tài sản số.

TSS là một lược đồ chữ ký số mà cho phép nhiều bên (tức người ký) có thể thiết lập các nhóm sao cho chỉ một tập hợp con nào đó của nhóm mới có thể tạo chữ ký thay mặt cho nhóm. Cụ thể hơn, một lược đồ chữ ký ngưỡng (t, n) sử dụng để ký một thông điệp là một lược đồ chữ ký số trong đó bất kỳ t (ngưỡng) (hoặc nhiều hơn) người ký nào trong một nhóm gồm n người ký có thể tạo chữ ký thay mặt cho nhóm. Vì mỗi bên chỉ giữ một phần khóa bí mật trong việc sinh khóa bí mật TSS, nên chữ ký ngưỡng không tiết lộ các thành viên nhóm thực sự đã hợp tác để sinh khóa và tạo chữ ký số trên thông điệp.

Trong chuỗi khối, lược đồ chữ ký số đường cong elliptic (Elliptic Curve Digital Signature Algorithm - ECDSA) thường được sử dụng để ký và xác minh giao dịch. Một giao thức cho TSS sử dụng ECDSA mà dựa trên mã hóa đồng cấu (homomorphic) và chứng minh không tiết lộ tri thức (zero-knowledge proof), đã được đề xuất bởi Gennaro & Goldfeder vào năm 2018 3 và được cập nhật trong 4, 5. Kể từ lần công bố đầu tiên, nhiều cài đặt của giao thức đã được phát triển và hiện đang được nhiều sản phẩm sử dụng để mang đến sự tin tưởng giữa các thành viên. Trong bài viết này, thuật ngữ t-ECDSA đề cập đến lược đồ chữ ký ngưỡng ECDSA do Gennaro và Goldfeder đề xuất.

Thật không may, mặc dù đã trải qua nhiều kiểm định an toàn, nhưng các cài đặt này, bao gồm các khung TSS mã nguồn mở thực tế trong Golang và Rust vẫn dễ bị tấn công bởi 3 tấn công trích xuất khóa mới mà nhóm nghiên cứu tại công ty Verichains đã phát hiện ra. Chúng được đặt tên là TSSHOCK.

TSSHOCK là tấn công lên giao thức chữ ký ngưỡng (Threshold Signature Scheme) được sử dụng rộng rãi trong các ví tiền điện tử dùng MPC (Multi-Party Computation), cross-chain bridge và các giải pháp quản lý tài sản kỹ thuật số.

Verichains đã phát hiện ra tấn công TSSHOCK sau khi tiến hành kiểm tra diện rộng một loạt các cài đặt t-ECDSA mã nguồn mở. Hầu hết các cài đặt có thể bị tấn công chỉ với một thành viên để khôi phục khóa bí mật ECDSA. TSSHOCK phá vỡ hoàn toàn tính an toàn của TSS, với việc khai thác bằng chứng về khái niệm thể hiện toàn bộ quá trình trích xuất khóa bí mật bởi một kẻ tấn công sau 1-2 chữ ký trên nhiều ví phổ biến, cơ sở hạ tầng khóa không lưu ký (non-custodial, tức cho phép người dùng có toàn quyền kiểm soát tài sản của mình bằng khóa bí mật được liên kết với ví) và giao thức quản lý tài sản chuỗi chéo (cross-chain).

Kẻ tấn công có thể khai thác TSSHOCK để đánh cắp tiền số có giá trị lên đến hàng tỉ USD từ cả người dùng cá nhân và các tổ chức/doanh nghiệp, đồng thời không để lại dấu vết và có vẻ không liên quan đối với các bên tham gia khác.

Phép biến đổi Fiat-Shamir

Phép biến đổi Fiat-Shamir là một kỹ thuật nổi tiếng nhằm loại bỏ tính tương tác khỏi các hệ chứng minh tương tác. Các hệ chứng minh tương tác thường có cấu trúc 3 bước như sau:

Bước 1: Người chứng minh sinh ra một giá trị cam kết và gửi tới người xác minh.

Bước 2: Người xác minh sinh ngẫu nhiên đều một giá trị thách thức và gửi tới người chứng minh.

Bước 3: Người chứng minh tính toán bằng chứng dựa trên cả giá trị cam kết và thách thức.

Ý tưởng đằng sau phép biến đổi Fiat-Shamir là thay vì yêu cầu người xác minh gửi một giá trị thách thức ngẫu nhiên cho người chứng minh, người chứng minh có thể tự tính toán giá trị này bằng cách sử dụng hàm ngẫu nhiên, chẳng hạn như hàm băm mật mã. Điều này giúp hệ chứng minh trở thành không tương tác.

dlnproof

dlnproof được sử dụng để xác minh, theo cách chứng minh không tiết lộ tri thức rằng người chứng minh biết $\log_g h \pmod N$, trong đó N là hợp số. Tại bước sinh khóa, mỗi bên được yêu cầu sinh và truyền phát bộ ba giá trị $(\tilde{N}, h_1, h_2)$, sẽ được sử dụng trong bước ký tiếp theo, cùng với một dlnproof chứng minh rằng bên tham gia biết $\log_{h_2} h_1 \pmod{\tilde{N}}$. Một số cài đặt cũng yêu cầu dlnproof bổ sung cho $\log_{h_1} h_2 \pmod{\tilde{N}}$. Phiên bản tương tác của dlnproof như sau:

Bước 1: Peggy (bên chứng minh) cam kết một giá trị ngẫu nhiên $\rho \in \mathbb{Z}_{\phi(N)}$ (chỉ Peggy biết $\phi(N)$) bằng cách gửi $\alpha = g^{\rho} \bmod N$ tới Victor (bên xác minh).

Bước 2: Victor chọn và gửi cho Peggy một bit thách thức ngẫu nhiên $c \in \{0,1\}$.

Bước 3: Peggy tính toán và gửi lại $\tau = \rho + c \log_g h \pmod{\phi(N)}$.

Bước 4: Victor chấp nhận khi và chỉ khi $g^{\tau} = \alpha h^{c} \bmod N$.

Hệ chứng minh tương tác ở trên được chuyển đổi thành dlnproof không tương tác bằng cách áp dụng phép biến đổi Fiat-Shamir được mô tả trước đó. Việc chứng minh cũng được lặp lại λ lần (thường là $\lambda \geq 80$) để giảm *lỗi an toàn* (*soundness error*) từ $\frac{1}{2}$ (xác suất đoán đúng c ở đầu giao thức) xuống $\frac{1}{2^{\lambda}}$.

Trong đó, *lỗi an toàn (soundness error)* là lỗi an toàn của một giao thức/lược đồ là xác suất mà một người chứng minh ác ý thuyết phục một người xác minh trung thực rằng một tuyên bố sai là đúng.

Giao thức con MtA

MtA là viết tắt của phép nhân với cộng, là một giao thức con liên quan đến 2 bên Alice và Bob lần lượt giữ các giá trị bí mật a, b. Khi kết thúc MtA, Alice thu được α và Bob thu được β sao cho $ab = \alpha + \beta \bmod q$ trong đó q là cấp của nhóm ECDSA đang sử dụng.

Tóm lại, việc rò rỉ đầu vào a hoặc b của MtA đều dẫn đến việc khôi phục khóa bí mật TSS. Trong giao thức MtA, cả hai bên cần trao đổi các giá trị được mã hóa với nhau để thực hiện các phép tính mong muốn. Tuy nhiên, vì thiếu sự tin tưởng giữa các bên nên khó có thể đảm bảo rằng các giá trị mã hóa được gửi đi là hợp lệ. Để giảm thiểu vấn đề này, giao thức yêu cầu cả hai bên cũng gửi bằng chứng phạm vi về các giá trị được mã hóa.

CÁC LỖ HỔNG BẢO MẬT

Sau khi tiến hành đánh giá tính an toàn toàn diện đối với một loạt các cài đặt TSS mã nguồn mở, Verichains đã phân loại các vấn đề thành các lỗ hổng phổ biến dưới đây.

Lỗ hổng 1: Điểm yếu trong việc sử dụng lược đồ mã code mơ hồ

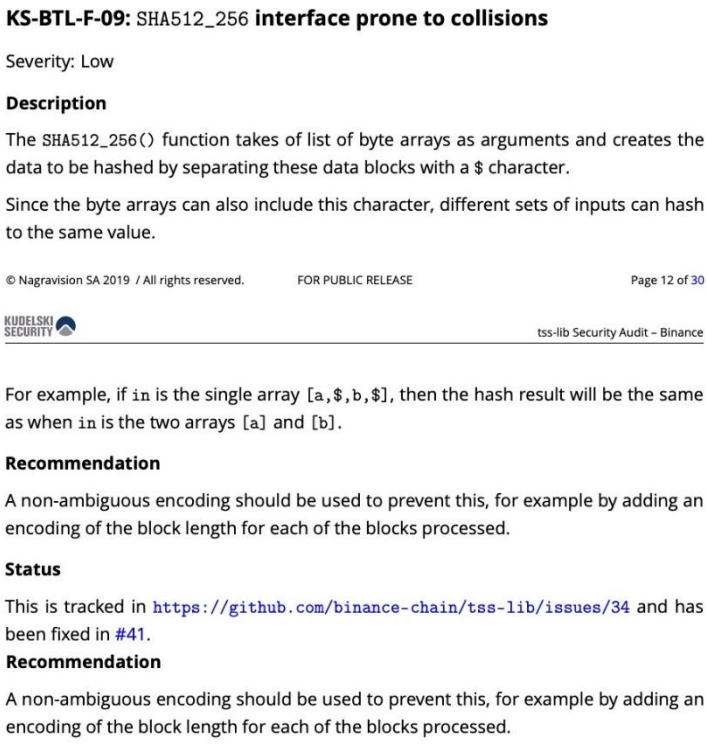
Một trong những cách cài đặt phổ biến nhất của TSS là thư viện tss-lib của BNB Chain, được viết bằng *go-lang*. Nhiều dự án khác, chẳng hạn như Multichain, THORChain, io.Finnet, Threshold Network (Keep Network) và Swingby sử dụng phiên bản sửa đổi của thư viện tss-lib.

tss-lib sử dụng lược đồ mã code không rõ ràng, trong đó, việc ghép nối các giá trị đầu vào với dấu phân cách ′$′ dẫn đến hai bộ giá trị mảng byte đầu vào ["a$","b"] *và* ["a","$b"] *có thể đưa ra kết quả cùng một giá trị băm.*

Đây được xem là vấn đề đã biết từ lâu đối với tss-lib. Vấn đề này đã được báo cáo lần đầu tiên tới Binance vào 4 năm trước (2019) bởi Kudelski Security trong báo cáo kiểm định an toàn cho thư viện tss-lib (xem Hình 1). Verichains không hiểu tại sao vấn đề (Giao diện KS-BTL-F-09: SHA512_256 dễ bị va chạm) được đánh dấu "An toàn thấp" trong báo cáo kiểm định. Điều này có thể do nhận thức của kiểm định viên rằng vấn đề này khó có thể khai thác được. Tuy nhiên, tấn công TSSHOCK đã chứng minh rằng đây thực sự là một vấn đề nghiêm trọng có thể phá vỡ hoàn toàn TSS.

Vấn đề này đã được theo dõi và khắc phục theo khuyến nghị của Kudelski tại vào tháng 9/2019. Tuy nhiên, trên thực tế, cách khắc phục này không đủ để giảm thiểu nguyên nhân gốc rễ của vấn đề lược đồ mã code không rõ ràng và thư viện tss-lib vẫn có thể bị khai thác.

Năm 2022, Kudelski Security đã được io.Finnet thuê để kiểm tra phiên bản sửa đổi tss-lib của BNB-Chain. Kudelski Security đã báo cáo lại cho io.Finnet vấn đề va chạm giá trị băm tương tự do việc thực hiện ghép nối các giá trị đầu vào với dấu phân cách ′$′. Vấn đề này đã được io.Finnet khắc phục theo cách khéo léo hơn và sau đó được công bố công khai với tên gọi CVE-2022-47931 vào ngày 28/3/2023 (xem Hình 2).



KS-BTL-F-09: `SHA512_256` **interface prone to collisions**

Severity: Low

Description

The `SHA512_256()` function takes of list of byte arrays as arguments and creates the data to be hashed by separating these data blocks with a $ character.

Since the byte arrays can also include this character, different sets of inputs can hash to the same value.

For example, if `in` is the single array `[a,$,b,$]`, then the hash result will be the same as when `in` is the two arrays `[a]` and `[b]`.

Recommendation

A non-ambiguous encoding should be used to prevent this, for example by adding an encoding of the block length for each of the blocks processed.

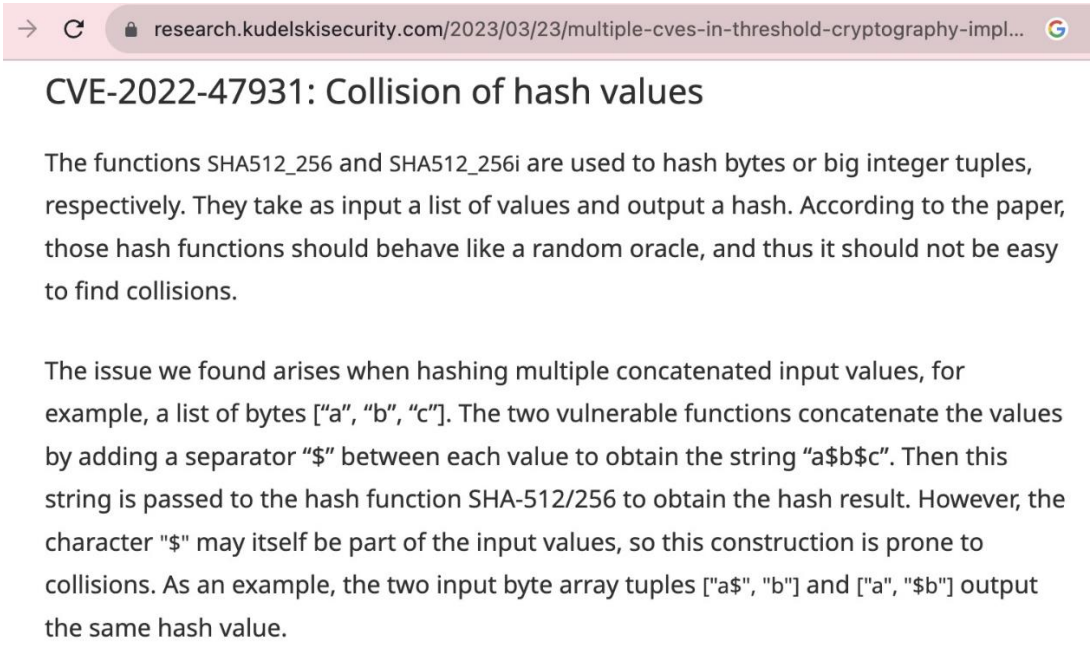
Status

This is tracked in `https://github.com/binance-chain/tss-lib/issues/34` and has been fixed in #41.

Recommendation

A non-ambiguous encoding should be used to prevent this, for example by adding an encoding of the block length for each of the blocks processed.

Hình 1. Báo cáo kiểm định an toàn đối với thư viện tss-lib, 2019



Hình 2. Tư vấn bảo mật của Kudelski Security/io.Finnet, 2023

Verichains phân loại vấn đề này là lược đồ mã code không rõ ràng và khai thác nó để khôi phục khóa bí mật, không phải va chạm hàm băm. Khi thực hiện phép biến đổi Fiat-shamir, một lược đồ mã code được sử dụng để tuần tự hóa bản ghi sắp được băm theo byte. Do đó, nếu lược đồ mã code không rõ ràng sẽ dẫn đến thực hiện sai.

Vào tháng 11/2022, Verichains đã phát triển một tấn công trích xuất khóa mới, được gọi là *α-shuffle*, đã khai thác thành công vấn đề về lược đồ ghi mã rõ ràng để phá vỡ TSS. Verichains đã xây dựng trích xuất khóa chứng minh khái niệm nhằm khai thác đối với tss-lib (và biến thể) từ các dự án Multichain, THORChain, Threshold Network (tBTC), Swingby. Lưu ý rằng, mỗi dự án có các sửa đổi mã code riêng cho các trường hợp sử dụng khác nhau nên việc khai thác thực tế khá phức tạp và khác nhau đối với mỗi cài đặt.

Multi-party-sig của Taurus, một cài đặt t-ECDSA của CGGMP-21 cũng dễ bị tổn thương trước tấn công *α-shuffle*.

Lỗ hổng 2: Tối ưu hóa TSS không an toàn

Do độ phức tạp theo hàm mũ của MPC/TSS, nhiều dự án đã cố gắng tối ưu hóa lược đồ để cải thiện hiệu suất và/hoặc mở rộng quy mô các sản phẩm MPC mà không chứng minh được *lỗi an toàn* của lược đồ. Các bẫy lỗi phổ biến gồm:

Bẫy lỗi 1: Giảm số lần lặp gói dlnproof

Việc giảm số lần lặp gói dlnproof có thể làm cho MPC chạy nhanh hơn nhưng lại giúp kẻ tấn công dễ dàng đoán được các bit thách thức đối với một số lần lặp nhỏ.

Trong cài đặt fastMPC của Multichain, tham số lần lặp đã giảm từ 128 xuống 1, do đó khiến nó dễ dàng bị giả mạo bằng phương pháp thử và sai. Verichains đã phát triển tấn công trích xuất khóa thứ hai, được gọi là *c-guess*, có thể khai thác thành công vấn đề này để khôi phục khóa bí mật từ một nút độc hại.

Bẫy lỗi 2: Sử dụng không gian thách thức lớn hơn (tương tự như giao thức Schnorr - Chứng minh tri thức về logarit rời rạc) và loại bỏ việc lặp lại dlnproof

Tối ưu hóa lược đồ mà không chứng minh được lỗi an toàn của lược đồ là không đáng kể. Một số thư viện TSS đã tối ưu hóa thuật toán cho ***dlnproof*** bằng cách chọn thách thức là 256 bit và chỉ chạy một lần. Điều này tương tự như bằng chứng logarit rời rạc đối với một nhóm có cấp nguyên tố. Nhưng ở Galoa Group, cấp nhóm $\phi(n)$ không phải là số nguyên tố mà là hợp số. Do đó, việc thực hiện tấn công trích xuất khóa thứ ba, *c-split*, được đề xuất bởi Verichains có thể thực hiện được.

Các dự án như tofn của Axelar, chữ ký ngưỡng của Ngân hàng ING, multi-party-ecdsa của ZenGo X có thể bị tấn công *c-split*.

TẤN CÔNG TSSHOCK

Verichains đã phát triển 3 tấn công mới như là hai phần của TSSHOCK có thể phát sinh trong ngữ cảnh của lược đồ chữ ký số ngưỡng t-ECDSA (và các điều kiện kích hoạt của chúng) bao gồm: *α-shuffle*, *c-split* và *c-guess*; và được gọi dưới tên chung là tấn công *TSSHOCK*. Như đã đề cập trước đó, các phương pháp tấn công này khai thác cả các điểm yếu cài đặt đã biết và các lỗ hổng mới được xác định duy nhất đối với cài đặt t-ECDSA.

Các tấn công này có thể phá vỡ hoàn toàn tính an toàn của TSS, với việc khai thác chứng minh khái niệm cho thấy một bên độc hại có thể trích xuất toàn bộ khóa bí mật sau 1-2 chữ ký trên các thư viện và sản phẩm phổ biến khác nhau.

Các tấn công của Verichains yêu cầu không hủy bỏ (no abort) để giao thức tiếp tục liền mạch, không để lại dấu vết và tỏ ra vô hại đối với các bên khác.

Tấn công *α-shuffle*

Tấn công *α-shuffle* khai thác lỗ hổng của lược đồ ghi mã không rõ ràng. Trong tấn công *α-shuffle*, ghi mã không rõ ràng của các giá trị α trong mỗi lần lặp của **dlnproof** cho phép nhiều hơn 1 danh sách giá trị khác nhau được ghi mã như nhau. Ý tưởng tấn công là liên kết chuỗi các giá trị α sao cho nó có thể được biểu diễn theo nhiều hơn 2 cách, ví dụ: $\alpha \parallel \alpha \parallel \alpha$ có thể được hiểu là $\alpha \parallel (\alpha \parallel \alpha)$ hoặc $(\alpha \parallel \alpha) \parallel \alpha$. Đặt α tương ứng với bit thách thức 0 và $\alpha \parallel \alpha$ tương ứng với bit thách thức 1. Tấn công *α-shuffle* trước tiên đoán có bao nhiêu bit thách thức là 1

(hoặc 0) và sinh các giá trị α tạm thời. Sau khi áp dụng phép biến đổi Fiat-Shamir, các bit thách thức sẽ được tiết lộ, nếu số lượng bit 1 (hoặc 0) được đoán chính xác thì các giá trị α sẽ được sắp xếp lại sao cho chúng được sắp thứ tự đúng theo từng bit thách thức. Nếu đoán sai thì tấn công sẽ thử lại.

THUẬT TOÁN 1: *α-shuffle* giả mạo dlnproof 1

Đầu vào: g, N

Đầu ra: h, dlnproof *đối với* $\log_g h \mod N$

1 Đặt $\tau = \text{rand}(\mathbb{Z}_{\text{ord(g)}})$. Đặt $\alpha = g^{\tau} \bmod N$. Đặt tất cả $\tau_i = \tau$.

2 Đặt $a = \text{bytes}(\alpha)$. Đặt $\beta = \text{int}(a|D|a)$.

3 Đặt $h = \alpha/\beta \bmod N$ (sao cho $\beta = g^{\tau}/h \bmod N$).

4 Với l thuộc $\{0, 1, 2, \ldots, \lambda\}$:

(a) Đặt tạm thời $\alpha_i = \begin{cases} \alpha, & (1 \le i \le l) \\ \beta, & (l+1 \le i \le \lambda) \end{cases}$

(b) Đặt $c_1, c_2, \ldots, c_\lambda = H(g, h, N, \alpha_1, \alpha_2, \ldots, \alpha_\lambda)$.
Nếu $\sum c_i = \lambda - l$ (có l bit thách thức bằng 0), thì đặt

(c) $\alpha_i = \begin{cases} \alpha, & (c_i = 0) \\ \beta, & (c_i = 1) \end{cases}$ và quay lại

5 Quay lại bước 1.

Thuật toán 1 cho phép người chứng minh giả mạo một chứng minh hợp lệ cho g, N tùy ý chọn. Thuật toán đưa ra h và dlnproof cho $\log_g h \mod N$ gồm λ cặp (α_i, τ_i). Ta có thể xác minh $g^{\tau_i} = \alpha_i h^{c_i}$ với mọi i khi thuật toán trả về, do đó đầu ra dlnproof là đúng. Vì danh sách α_i $(\alpha, \alpha, \ldots, \alpha, \beta, \beta, \ldots, \beta)$ được sắp xếp lại dựa trên các bit c_i, nên Verichains đặt tên cho kỹ thuật này là *α-shuffle*.

Tấn công *c-split*

Trong tấn công *c-split*, thuật toán tối ưu cho dlnproof chọn thách thức là 256 bit và chạy một lần. Điều này tương tự như bằng chứng logarit rời rạc đối với một nhóm có cấp nguyên tố. Nhưng trong Galoa Group, cấp nhóm $\phi(N)$, không phải là số nguyên tố mà là hợp số. Ý tưởng tấn công *c-split* như sau:

Theo công thức $\tau = \rho + c \cdot \log_{h_2} h_1 \mod \phi(N)$, gọi e là ước nhỏ của $ord(h_1)$ (lưu ý, $h_2 = h_1^e$). Không tồn tại $\log_{h_2} h_1 = \frac{1}{e}$ trong nhóm có cấp $\phi(N)$ nếu $\gcd(e, \phi(N)) \neq 1$. Khi xây dựng một dlnproof cho $\log_{h_2} h_1$, ta có thể giữ ρ cho đến khi c chia hết cho e. Nếu $\log_{h_2} h_1 = \frac{1}{e}$ và $e|c$, thì τ tồn tại, tức là $(\alpha, \tau) = (h_2^{\rho} \bmod N, \rho + \frac{c}{e})$ là một bằng chứng dlnproof được tạo hợp lệ mặc dù không tồn

tại $\log_{h_2} h_1$. Thực hiện vét cạn số ngẫu nhiên c đối với số e nhỏ đã chọn sao cho c chia hết cho e có thể được thực hiện với xác suất thành công $\frac{1}{e}$ và không có rủi ro vì c được tính theo phép biến đổi Fiat-Shamir. Vì các đầu vào bí mật được trích xuất theo **mod** e và e nhỏ nên không thể khôi phục hoàn toàn các giá trị bí mật từ 1 chữ ký. Để trích xuất đầy đủ giá trị bí mật, (các) giai đoạn ký số khác có thể được thực hiện để thu thập các phần chưa hoàn chỉnh khác của cùng một giá trị bí mật và kết hợp bằng cách sử dụng tấn công lưới (tương tự như tấn công khôi phục rò rỉ nonce trong ECDSA).

c-split *2*

Người chứng minh		***Người xác minh***
$N, \phi(N), h_1, h_2, \log_{h_2} h_1$	$\alpha = h_2^{\rho} \; mod \; N$ →	N, h_1, h_2
Áp dụng Fiat-Shamir với $c = H(N, h_1, h_2, \{\alpha_i\})$	← $c \in \mathbb{Z}_{2^{256}}$	
	$\tau = \rho + c \cdot log_{h_2} h_1 \mod \phi(N)$ →	
		Chấp nhận nếu $h_2^{\tau} = \alpha \cdot h_1^{c} \bmod N$

Vì cần điều kiện $e \mid c$ để giả mạo chứng minh nên Verichains đặt tên cho kỹ thuật tấn công này là *c-split*.

Tấn công *c-guess*

Tấn công *c-guess* được thực hiện bằng cách đoán xem các bit thách thức là gì khi tính toán dlnproof bằng cách chọn c_i ngẫu nhiên và áp dụng hàm băm *H* để kiểm tra xem các bit thách thức đầu ra thực tế có giống với các bit đã đoán. Nếu số lần lặp λ (số bit) đủ nhỏ thì điều này có thể được đoán một cách hiệu quả. Xác suất để đoán thành công là $\frac{1}{2^{\lambda}}$.

c-guess *2*

Người chứng minh		***Người xác minh***
$N, \phi(N), h_1, h_2, \log_{h_2} h_1$	$\alpha = h_2^{\rho} \; mod \; N$ →	N, h_1, h_2
Áp dụng Fiat-Shamir với $c = H(N, h_1, h_2, \{\alpha_i\})$	← $c \leftarrow_R \{0,1\}$	
	$\tau = \rho + c \cdot log_{h_2} h_1 \mod \phi(N)$ →	
		Chấp nhận nếu $h_2^{\tau} = \alpha \cdot h_1^{c} \bmod N$

Lặp lại ít lần hơn

Hình 3 dưới đây cung cấp thông tin về ảnh hưởng của tấn công TSSHOCK lên một số nhà cung cấp (hoặc thư viện) trên thế giới.

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

Hình 3. Thống kê ảnh hưởng của tấn công TSSHOCK lên một số nhà cung cấp/thư viện (nguồn slide [2])

Biện pháp phòng tránh tấn công TSSHOCK

***Đối với tấn công α-shuffle*:** để giảm thiểu tấn công *α-shuffle*, cần áp dụng lược đồ ghi mã rõ ràng để xây dựng hàm băm H. Không nên tồn tại hai danh sách khác nhau được ghi mã thành cùng một chuỗi byte. Nguyên tắc chung là nếu không có cách xác định để giải mã chuỗi byte (trở lại danh sách số nguyên ban đầu), thì có thể đã làm sai. Cách khắc phục đơn giản là luôn tính đến độ dài của mỗi số nguyên.

***Đối với tấn công c-split*:** Trong một hệ chứng minh, yếu tố quan trọng nhất để xác định số lần lặp lại chứng minh là *lỗi an toàn* của hệ. Quyết định không lặp lại một chứng minh mà không chỉ ra chứng minh đó có lỗi an toàn không đáng kể cũng giống như việc chấp nhận một nguy cơ rủi ro. Do đó, cần tuân thủ các thông số kỹ thuật khi thực hiện các giao thức. Nên tránh tối ưu hóa cài đặt mà không có sự hiểu biết đúng đắn về tính an toàn, vì điều này có thể tạo ra các lỗ hổng có thể bị kẻ tấn công khai thác.

***Đối với tấn công c-guess*:** dlnproof yêu cầu số lần lặp tối thiểu để đạt được mức an toàn nhất định. CG GMP21 khuyến nghị số lần lặp ít nhất là 80. Số lần lặp này phải được thực thi trong tất cả các cài đặt.

Kết luận

Tấn công TSSHOCK khai thác các lỗ hổng trong cài đặt, qua đó, cho phép một bên độc hại đánh cắp tài sản trị giá hàng triệu USD mà không để lại dấu vết. Việc phát triển và tối ưu hóa các giao thức mật mã phức tạp có thể cực kỳ khó

khăn và nguy hiểm. TSSHOCK nhấn mạnh tầm quan trọng của việc kiểm tra kỹ lưỡng, đánh giá nghiêm ngặt và cải tiến liên tục các biện pháp bảo mật.

Tài liệu tham khảo

1. Duy Hieu Nguyen, Anh Khoa Nguyen, Huu Giap Nguyen, Thanh Nguyen, Anh Quynh Nguyen. "*New Key Extraction Attacks on Threshold ECDSA Implementations*". Blackhat USA August 5-10, 2023.
2. Verichains. "*TSSHOCK: Breaking MPC Wallets and Digital Custodians for $BILLION$ Profit*". Availaible at https://www.verichains.io/tsshock/#tsshock-poc.
3. Rosario Gennaro and Steven Goldfeder. *Fast Multiparty Threshold ECDSA with Fast Trustless Setup*. Cryptology ePrint Archive, Paper 2019/114, 2019.
4. Rosario Gennaro and Steven Goldfeder. *One Round Threshold ECDSA with Identifiable Abort*. Cryptology ePrint Archive, Paper 2020/540, 2020.
5. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. *UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts*. Cryptology ePrint Archive, Paper 2021/060, 2021