IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 July 2024, accepted 5 August 2024, date of publication 8 August 2024, date of current version 25 September 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3440370*

RESEARCH ARTICLE

# Cloning Hardware Wallet Without Valid Credentials Through Side-Channel Analysis of Hash Function

**DONGJUN PARK[1], JOONSUP KIM[1], HEESEOK KIM[2], (Member, IEEE), AND SEOKHIE HONG[1], (Member, IEEE)**

[1]Institute of Cyber Security and Privacy (ICSP), Korea University, Seoul 02841, South Korea
[2]Department of AI Cyber Security, College of Science and Technology, Korea University, Sejong-si 30019, South Korea

Corresponding author: HeeSeok Kim (80khs@korea.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by Korean Government (MSIT) (Development of Physical Channel Vulnerability-Based Attacks and its Countermeasures for Reliable On-Device Deep Learning Accelerator Design) under Grant 2021-0-00903.

**ABSTRACT** Hardware wallets, specialized devices designed to securely manage users' credentials, play a crucial role in securing cryptocurrencies, ensuring credentials remain under user control without reliance on third-party entities. However, despite extensive research on Side-Channel Analysis (SCA) attacks, studies specifically addressing their implications for hardware wallets remain relatively limited. While previous work has demonstrated various SCA attacks on hardware wallets, most of these attacks require sophisticated environmental controls or detailed knowledge of target device. In addition, some attacks assume unrealistic scenarios that require valid credentials to conduct the attacks. This paper introduces a novel SCA attack on hardware wallets to extract master seeds—a foundational component in the security of hardware wallets. Our proposed attack leverages power traces obtained during the processing of the Keyed-Hash Message Authentication Code (HMAC), or more precisely, the Secure Hash Algorithm 2 (SHA-2) inside the HMAC. Notably, our attack is non-invasive, ensuring the integrity of the target device, thereby making it difficult for the wallet owners to detect the attack. Furthermore, our attack can be conducted without a profiling phase, excluding the excessive capabilities required for the attack.

**INDEX TERMS** Cryptocurrency, hardware security, side-channel analysis.

## I. INTRODUCTION

Recent advances in cryptography have given rise to blockchain technology enabling a distributed ledger accessible to all around the world without a central authority. One of the prominent applications of blockchain is a cryptocurrency, a digital payment system where all transactions are transparently recorded in a blockchain-based ledger [1].

In cryptocurrencies, cryptographic hash functions are essential as they uphold the immutability of that ledger and the integrity of recorded transactions. Furthermore, hardware wallets, dedicated devices for safely managing cryptocurrencies without entrusting credentials to a third party, also employ the Keyed-Hash based Message Authentication Code (HMAC) to generate the most critical secret information in hardware wallets, namely a master seed [2].

Even if the cryptographic hash function is considered secure, however, the potential for adversaries to gain physical access to a cryptographic device is a commonly overlooked consideration in the security analysis of cryptosystems [3]. In particular, Side-Channel Analysis (SCA) attacks, one of the physical attacks that inject intentional faults into the device or passively observe physical leakages from the device, have been highlighted as a real threat to cryptographic devices. Since the advent of SCA attacks, not only traditional cryptographic algorithms but also the latest technology,

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

post-quantum cryptography, have been studied. However, there is a lack of research on SCA attacks on cryptocurrency hardware wallets and countermeasures against such attacks.

As the cryptocurrency market keeps expanding, the demand for countermeasures against SCA attacks is also growing. This paper aims for the side-channel resistant design of hardware wallets, focusing on the secure implementation of cryptographic algorithms. As part of this effort, we propose a novel SCA attack to show that master seeds can be extracted in practice. The proposed attack targets the open-source cryptographic library, namely Trezorlib [2], because it has had a major impact on the various hardware wallets. In this study, we heavily referenced the Bitcoin Improvement Proposal (BIP) documents that serve as de facto standards in cryptocurrency [4], [5], [6], [7], [8].

### A. RELATED WORKS

Within the literature, we could identify two primary categories of SCA attacks that exploit inherent physical leakages. The first is invasive attacks, which disrupt the normal behavior of the device to bypass security mechanisms or to extract sensitive data by analyzing the faulty output. One can induce such faults by manipulating the device's clock frequency [9] or voltage supply [10]. Faults can also be provoked by exposing the device to intense laser [11] or electromagnetic (EM) pulses [12]. Fortunately, invasive attacks are easily detectable because they cause conspicuous malfunctions or modifications [13].

A recent line of papers has shown that invasive SCA attacks can expose the secret values stored in hardware wallets, potentially leading to wallet cloning and cryptocurrency thefts. In the study by Nedospasov et al., [14], it was presented that voltage glitches could downgrade the readout protection level of devices, granting access to the device's Static Random-Access Memory (SRAM). Subsequently, they could extract the recovery seed by interrupting the firmware update process before the SRAM is cleared, as hardware wallets typically back up recovery seeds to SRAM before being updated. However, it is worth noting that the wallet owner may detect such attacks, as the attack involves the removal of the package to use debugging tools.

The study by O'Flynn [15] demonstrated that EM fault injection into a Universal Serial Bus (USB) could bypass a protection mechanism involved in request message handling. When a USB request message for accessing the flash memory containing the recovery seed is sent, the request handler will generally reject such a request. However, it was possible to skip a comparison instruction responsible for checking whether the received request accesses the flash memory. Even though this attack does not physically damage the wallet, realizing this attack requires a thorough fine-tuning of fault injection parameters such as location, duration, intensity, and delay to trigger an effective skipping.

The second category is non-invasive attacks, which exploit power fluctuations of the device originating from sensitive data and operations being processed. It has been shown that power consumption analysis makes it possible to reverse-engineer those data and operations [16]. Power consumption traces can be obtained by measuring the voltage across a shunt register or power supply line itself [17]. EM radiation emitted from the device is also exploitable under the fundamental laws of electromagnetism [18]. Non-invasive SCA attacks, especially those utilizing EM waves that can be measured over relatively long distances, are considered more covert and thus threatening compared to invasive ones. While non-invasive attacks on HMAC [19], [20], [21], [22], [23], [24] and general defences against them [25], [26], [27] have been well studied, there is a lack of research when the goal of the attack is recovering the input message – the master seed – of HMAC as in this paper.

Non-invasive SCA attacks also pose a threat to the security of hardware wallets. In 2019, the wallet manufacturer Ledger conducted two profiling attacks on the Trezor wallet, showing vulnerabilities in Personal Identification Number (PIN) and private keys [28]. The first attack aimed at extracting a four-digit PIN by observing differences in power patterns when the input PIN coincides with the stored one compared to when they do not. Before the attack, they built 40 unique power templates from 0 to 9 for each digit using a device with known PINs. Then, they executed the PIN verification with a target device of the same model, with the correct PIN unknown, and collected corresponding power traces. Finally, they compared the power traces with the templates to deduce the correct PIN digit-by-digit. Remarkably, their matching rate was 100%, indicating that an attacker could have reconstructed the correct PIN within 10 queries. This vulnerability has been mitigated by modifying the operations inside the PIN verification so that the entire PIN can only be recovered after more attempts. In addition, the wallet will be erased if the PIN is incorrect 16 times in a row. These countermeasures prevent the sufficient acquisition of power traces, providing robustness against SCA attacks on PINs.

The second attack discussed in the same paper by the Ledger research team focused on extracting a private key from the Elliptic Curve Scalar Multiplication (ECSM) algorithm [28]. The ECSM implementation of the Trezor consists of 64 iterations of point addition and conditional negation, processing a 256-bit secret scalar in increments of four bits. Point addition involves the eight precomputed operands, thereby providing three bits of partial information about the scalar for each iteration. The remaining bit is associated with conditional negation, which adjusts the sign of the accumulated result based on a specific condition. The extraction of the former three bits was achieved through template matching, while the remaining bit was extracted by exploiting timing differences depending on the condition being true or false. However, as with their first attack, this second attack also involves a profiling stage that requires the attacker to have strong capabilities. Moreover, the conditional negation algorithm has been enhanced with a constant-time implementation to mitigate the vulnerability.

As a follow-up study to their second attack, Park et al. proposed an enhanced attack on the ECSM that does not require a profiling stage [29]. They could extract three bits by guessing intermediate values the processor currently processing and correlating them with acquired traces. In addition, the remaining one bit was successfully extracted by observing the power pattern of the conditional negation, which is significantly different depending on whether the condition is true or false. Their attack was done with a single EM trace obtained from a real Trezor hardware wallet. However, due to the low signal-to-noise ratio of the real device, the trace was averaged over the multiple queries of the same scalar. To be a successful attack with one query, it would have to be accompanied by state-of-the-art signal processing techniques.

The most relevant work to this paper is the technical report by Benadjila et al. [30]. In their ongoing project about a secure USB storage called WooKey, the integrity of the Encrypted Platform Key (EPK) is checked by treating it as an input message to the HMAC. The input key to the HMAC is called KPK and is intended to be correct only if the correct PIN is provided. They conducted a non-profiling SCA attack on the HMAC to extract the EPK. However, the EPK obtained by such an attack cannot solely be threatening without a valid PIN. That is why they presented a complicated scenario, which includes modifying the firmware, returning the device to its original owner, and finally making later a secondary robbery to retrieve all the confidential data. They also pointed out that to target the real WooKey device, the attacker would have to replace the touchscreen with another input interface for automated PIN entry.

Notably, they assumed the HMAC message to be the adversary's goal, and that the HMAC key could be arbitrarily manipulated. This assumption may not be appropriate for typical scenarios where HMAC is utilized for integrity checking of public messages. Except for their work, almost all known SCAs of HMAC are limited to profiling attacks to forge HMAC output without extracting confidential data [19], [20], [23], [24]. However, this assumption aligns well with the context of hardware wallets, where the message is confidential and adversaries can manipulate the key.

### B. CONTRIBUTIONS

This paper proposes a practical SCA attack targeting the HMAC implementation, specifically focusing on the Secure Hash Algorithm 2 (SHA-2) within the HMAC. Our attack, the Master Seed Recovery (MSR) attack, has four distinguishing features.

First, the MSR attack attempts to extract master seeds, the most critical value in hardware wallets. Compared to the attacks on ECSM, which allow the theft of only one transaction associated with one scalar, the MSR attack can grant access to all past, present, and future transactions. Compared to the attacks on PIN, which allow transactions to be stolen only when the attack currently possesses the wallet, the MSR attack can steal transactions even after the wallet is returned to the original owner.

Second, the MSR attack eliminates the necessity for a profiling stage. The scenario where an adversary can build power templates of the hardware wallet under attack is unlikely to be realistic because building power templates often requires a valid credential or full control over the device. Despite an option for building power templates from another device of the same model, subtle variations in semiconductor microstructures may thwart effective profiling.

Third, the MSR attack can be carried out non-invasively, that is, the attacker does not need to alter the software or hardware, nor physically damage the wallet to conduct the attack. This simplifies the attack setup, making it easier to reproduce compared to invasive attacks, and avoids the risk of data loss within the wallet during compromise. Additionally, even if the wallet should be returned to its original owner after the attack, it is challenging to be aware of the attack.

Lastly, the MSR attack leverages multiple power traces acquired during the computation of HMAC, which generates root nodes for the hierarchical deterministic wallets. Unlike the previous attack that targets ECSM, the MSR attack can be carried out even in a noisy environment, as the power traces can be acquired without limitations since generating a root node does not require valid credentials in some real-world scenarios. Moreover, the MSR attack is practical for recovering master seeds even from lost or stolen wallets since no credentials are required to conduct the attack.

We present a comparison of existing studies and ours in Table 1. The first [14] and second studies [15] achieve master seed recovery but are invasive attacks that cause physical damage to the device. The third study [28] requires a profiling stage to extract the PIN. The fourth [28] and fifth studies [29] achieve private key recovery, but only one transaction can be stolen with that private key. Finally, the sixth study [30], while not originally analyzing the hardware wallet, can be utilized for achieving master seed recovery. However, it requires a large number of power traces, which may not be possible due to the settings of the hardware wallet. In contrast, we have been able to dramatically reduce the required traces by utilizing leakages from multiple points of interest in a single trace.

**TABLE 1. Comparison with existing studies.**

| Study | Attack Target | | Attacker Capability | | |
|---|---|---|---|---|---|
| | Algorithm | Goal | Invasive | Profiling | Attempts |
| [14] | Firmware update | Master seed | Yes | No | Many |
| [15] | Request handler | Master seed | Yes | No | Many |
| [28] | PIN verification | PIN | No | Yes | 10 |
| [28] | ECSM | Private key | No | Yes | 1 |
| [29] | ECSM | Private key | No | No | 1 |
| [30] | HMAC | Master seed | No | No | 1000 |
| This paper | HMAC | Master seed | No | No | 4 |

The rest of this paper is structured as follows. Section II introduces background knowledge on hardware wallets and SCA attacks. Section III explains how our attack extracts

the master seed from hardware wallets, with a detailed description of the target algorithms. This section also discusses its exploitability in real-world scenarios. Section IV validates our attack with the experimental results. Finally, Section V concludes this paper, discussing countermeasures against SCA attacks.

## II. PRELIMINARIES

In this section, we first present the definition and notation that will be used throughout this paper. The rest of this section introduces the standard structure of hardware wallets in Subsection II-B, and the non-invasive power analysis attack, one of the statistical techniques that will form the basis of our attack, in Subsection II-C.

### A. GLOSSARY

In this subsection, we provide brief definitions for highly technical terms. Power trace is time series data representing the power consumed by an electronic device during its operation. Intermediate value is a value that is calculated while the electronic device is performing some operation, and typically the power consumption of the device at any given time is dependent on the intermediate values it is processing at that time. Point of Interest (PoI) is the point in time at which the operation targeted by the attacker or analyst is performed, which is also used in this paper to mean the point in time that is meaningful for recovering the secret information. Peak is the local maximum in a correlation coefficient graph between the actual measured power trace and the hypothetical power model of the intermediate value, which is an important piece of information that indicates at what point the guessed value was actually processed. Finally, the meanings of the symbols used throughout this paper are shown in Table 2.

**TABLE 2. Meaning of symbols used in this paper.**

| Symbol | Meaning |
|---|---|
| $\|\|$ | Concatenate two expressions on either side |
| $\leftarrow$ | Assign the expression on the right side to the left |
| $\neg$ | Bitwise negation operator |
| $\gg$ | Bitwise right shift operator |
| $\ggg$ | Bitwise right rotation operator |
| $\wedge$ | Bitwise and operator |
| $\oplus$ | Bitwise exclusive or operator |
| $\boxplus$ | Addition modulo $2^{64}$ |
| $\mathbb{E}$ | Expected value of random variable |
| 0b | Prefix to denote binary representation |
| 0x | Prefix to denote hexadecimal representation |

### B. HIERARCHICAL DETERMINISTIC WALLETS

In cryptocurrency software, the generation of private keys often employs the Password-Based Key Derivation Function 2 (PBKDF2), incorporating random numbers as entropy. However, generating a new random number for each private key can be inefficient, particularly for small devices like hardware wallets that may lack built-in random number generators. To address this, the Hierarchical Deterministic (HD) wallet has been proposed in BIP-0032 [4]. This tree structure entails multiple keys derived from a master node, as depicted in Fig. 1. As specified in BIP-0043 [5], BIP-0044 [6], BIP-0049 [7], and BIP-0084 [8], Level 1 is for the purpose, level 2 is for the coin type, and level 3 is for the account, and so on.

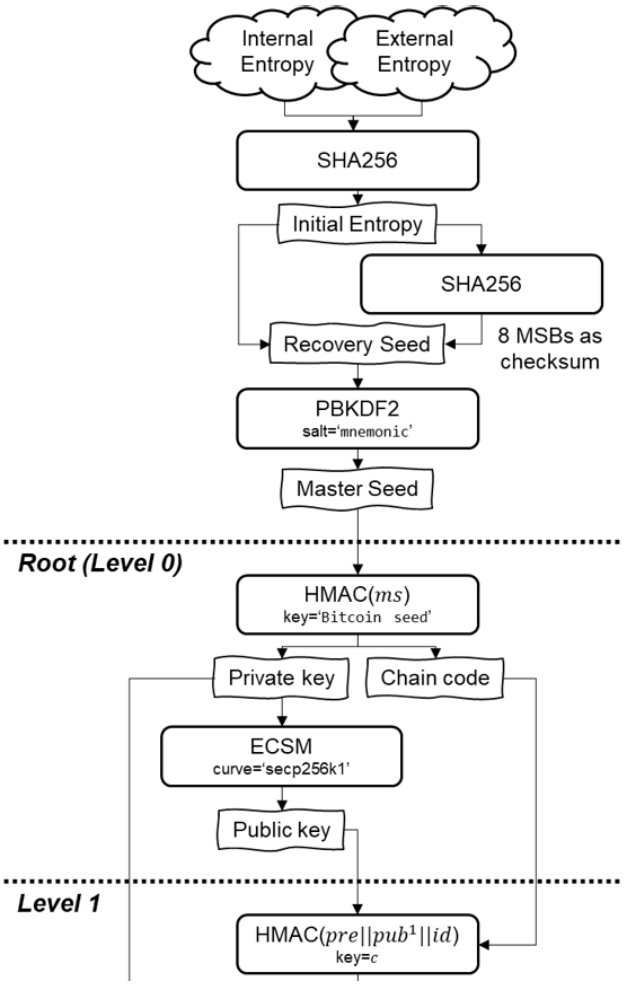



**FIGURE 1. Structure of hierarchical deterministic wallets for Bitcoin where *ms* stands for master seed. Usually, the private and public keys at level 5 are directly involved in the transaction.**

The creation of a root node is as follows. A 256-bit initial entropy, concatenated with its 8-bit checksum, is fed into PBKDF2 with the string `mnemonic` as the salt, yielding a 512-bit master seed. Subsequently, the master seed undergoes HMAC computation with the string `Bitcoin seed` as the HMAC key in the case of Bitcoin. Optionally, the wallet owner can append a secret passphrase for additional security, although this is not mandatory. Finally, the upper and lower 256 bits of the HMAC output serve as the private key and chain code, respectively. The process of deriving child keys of arbitrary depth relies solely on a root node.

### C. CORRELATION POWER ANALYSIS

Correlation Power Analysis (CPA) is a statistical technique that exploits the correlation between measured power traces

and hypothetical power consumption [16]. Consider a set $T = \{t_1, t_2, \cdots t_n\}$ comprising $n$ power traces acquired during a cryptographic operation. If each trace $t_i$ encompasses $m$ points in time, denote the $j$-th point as $t_{i,j}$ for $1 \leq j \leq m$. It is essential to align all traces so that the power consumption of identical instructions corresponds to the same point.

For $i$-th query, one can compute intermediate values $v_i$ from known values $p_i$ (e.g., ciphertext) by guessing an unknown value $k$ (e.g., secret key). Let us assume that the hypothetical power consumption of the device $l_i$ follows the Hamming weight model, in other words, $l_i \propto h_i$ where $h_i$ refers to the number of the bit 1 in the binary representation of $v_i$. Analyzing the correlation between the actual power traces $T$ and the power model $L = \{l_1, l_2, \cdots l_n\}$ at PoIs enables the determination of the correct value $k^\star$ most closely associated with the observed power consumption.

The Pearson correlation coefficient evaluates the linear relationship between two sets of data $L$ and $T$, expressed as

$$\rho = \frac{\mathbb{E}[TL] - \mathbb{E}[T]\mathbb{E}[L]}{\sqrt{\mathbb{E}[T^2] - \mathbb{E}^2[T]}\sqrt{\mathbb{E}[L^2] - \mathbb{E}^2[L]}}. \quad (1)$$

Assuming additive white Gaussian noise,[1] the sample Pearson correlation coefficient between $n$ sampled power consumption $t_{i,j} \in T$ at point $j$ and $n$ sampled power model $l_i \in L$ is computed as

$$r_j = \left| \frac{n\sum t_{i,j} l_i - \sum t_{i,j} \sum l_i}{\sqrt{n\sum t_{i,j}^2 - (\sum t_{i,j})^2}\sqrt{n\sum l_i^2 - (\sum l_i)^2}} \right| \quad (2)$$

for the guessed value $k$. Taking the absolute value of the correlation coefficient accounts for potential negative correlation due to reversed power probe setups. The correct value $k^\star$ is determined by the value $k$ that maximizes the correlation between the measured and hypothetical consumption, in other words, $k^\star = \arg\max_k(r_j(k))$.

If an adversary does not know the exact location of the PoIs in the trace, a basic resolution to this issue involves computing $r_j(k)$ for every time point $1 \leq j \leq m$, like a brute-force attack. For instance, in Fig. 2, the correlation coefficient is depicted between the power model with 256 guessed round keys and 1,000 power traces obtained during the first round of the Advanced Encryption Standard (AES) algorithm [31]. In the AES case, the power model is the Hamming weight of $S[p_i \oplus k]$, where $S$ denotes the AES substitution table and $p_i$ refers to 1,000 plaintexts of one byte. The peak correlation coefficient for the power model with $k = 0\text{xF2}$ at $j = 2.31\text{ms}$ indicates that the actual round key is 0xF2 and that the AES substitution operation occurs at that time.

## III. MASTER SEED RECOVERY ATTACK

This section introduces the Master Seed Recovery (MSR) attack on HMAC-SHA-2, extracting the master seed from hardware wallets. First, Subsection III-A covers the basics of our attack in a non-profiling, non-invasive environment.

[1]By definition, it converges to zero with a sufficient number of traces.

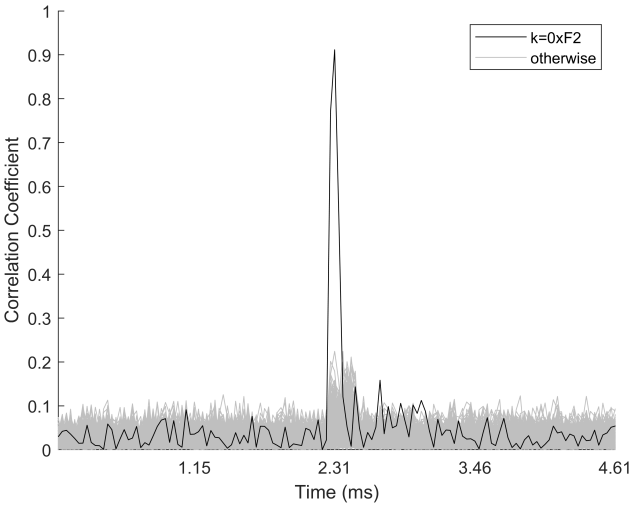



**FIGURE 2. Pearson correlation coefficients between the power models and measured traces. The model with guessed key 0xF2 shows the highest correlation at near 2.31ms.**

Next, Subsection III-B proposes two ways to enhance our attack. Finally, considering that our attack necessitates the condition of acquiring multiple traces without valid credentials, Subsection III-C discusses the exploitability of our attack in real-world scenarios.

### A. BASIC ATTACK

The HMAC algorithm serves to create root nodes of the HD wallets. The MSR attack targets the inner hash $H((K \oplus ipad)\,||\,M)$ of HMAC [32], where $H$ is the 512-bit version of SHA-2 as mentioned in BIP-0032 [4]. The SHA-2 algorithm operates in two stages: preprocessing and hash computation [33]. Preprocessing involves padding an input and parsing the padded input into 1024-bit blocks. Let us assume the length of the input is $l$ bits. The padding ensures the padded input length is a multiple of 1024 bits by appending the bit 0b1 to the end of the input, followed by $k$ zero bits, where $k$ is the smallest non-negative solution to the equation $l + 1 + k \equiv 896 \mod 1024$, and appending the 128-bit value that is equal to the number $l$ expressed using a binary representation.

In hardware wallets, the message $M$ is a secret 512-bit master seed, while the key $K$ is a variable string specific to each coin type, subject to manipulation by potential adversaries. Although FIPS 198-1 specifies that $K$ can have an arbitrary length and needs to be processed differently based on its length so that it is 1024 bits long, we only consider the case without loss of generality where the bit length of $K$ is not greater than 1024 for better readability. In this case, $K$ is treated as a 1024-bit value by appending as many zero bits as necessary after the original value. As a result, the padded input for the inner hash should be $(K \oplus ipad)\,||\,(M\,||\,0 \times 8000 \cdots 0600)$ where $ipad = 0 \times 3636 \cdots 3636$ is the 1024-bit constant. The preceding and

following terms are parsed into two 1024-bit blocks, namely $X^{(0)}$ and $X^{(1)}$, as shown in Fig. 3.

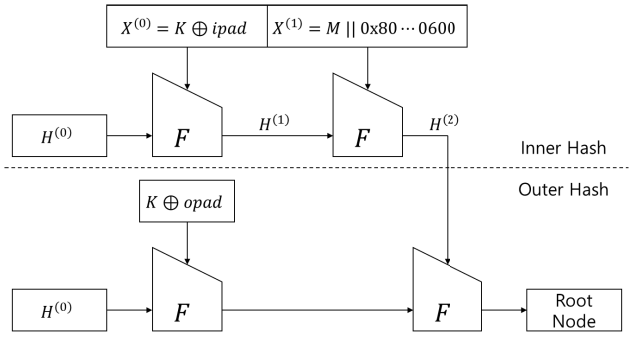



**FIGURE 3.** HMAC-SHA-2 in hardware wallets where *F* denotes the compression function.

The inner hash computation firstly constructs an input schedule $W = \{W_0, W_1, \cdots, W_{79}\}$ for each block $X^{(i)}$ and successively produces a series of hash values $H^{(i)}$. The eight 64-bit words of the hash value are labeled $H_0^{(i)}, H_1^{(i)}, \cdots, H_7^{(i)}$, which will hold the initial hash value $H^{(0)}$ (defined in FIPS 180-4 [33]), replaced by intermediate hash value $H^{(1)}$, and ending with the final hash value $H^{(2)}$. The input schedule's $t$-th term $W_t$ is the input $X_t^{(i)}$ itself for $0 \leq t \leq 15$, or a combination of the previous terms $\sigma_1(W_{t-2}) \boxplus W_{t-7} \boxplus \sigma_0(W_{t-15}) \boxplus W_{t-16}$ for $16 \leq t \leq 79$. The main body of each hash computation is an 80-round compression function consisting of four logical functions ($\Sigma_1$, $\Sigma_0$, *Ch*, and *Maj*) and modular addition ($\boxplus$), along with the eight working variables ($a, b, c, d, e, f, g$, and $h$) and the constants $K_t$ (also defined in FIPS 180-4 [33]), as illustrated in Fig. 4. The logical functions used in the hash computation stage are defined as follows:

$$\sigma_1(x) = (x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6) \quad (3)$$

$$\sigma_0(x) = (x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7) \quad (4)$$

$$\Sigma_1(x) = (x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41) \quad (5)$$

$$\Sigma_0(x) = (x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39) \quad (6)$$

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (7)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (8)$$

Algorithm 1 represents the inner hash computation in a hardware wallet, as discussed previously. First of all, please note that the intermediate hash value $H^{(1)}$ can be computed solely from known values ($H^{(0)}$ and *ipad*) and $K$ chosen by the adversary, implying that $H^{(1)}$ is known to the adversary. Also note that in the compression function, $W_t$ is summed with $H^{(1)}$-driven values and assigned to $U_{t+1}$ in Step 10. The operands $W_t (= X_t^{(1)})$ for $0 \leq t \leq 7$ exactly correspond to the master seed $M$. The adversary can compute the value of $U_{t+1}$ by guessing the 64-bit value of $W_t$. Therefore, we can infer that $M$ can be extracted through CPA where the Hamming weight of $U_{t+1}$ gives the hypothetical power consumption $L$.

Certainly, guessing $2^{64}$ possibilities is infeasible. However, adversaries can employ the divide-and-conquer strategy,

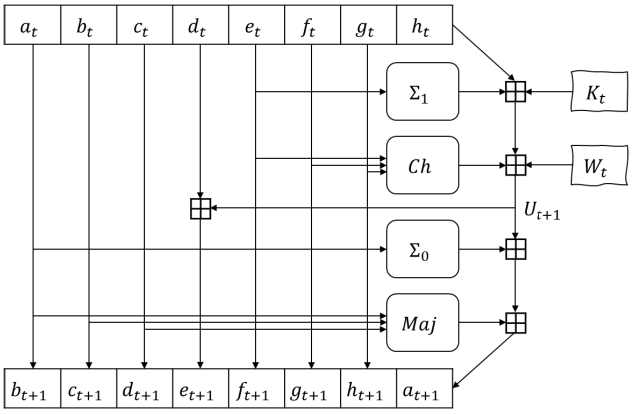



**FIGURE 4.** One round of compression function of SHA-2.

where they guess only a portion of the bits (e.g., one byte). While this strategy makes the attack feasible with $8 \times 2^8$ guesses, it diminishes the reliability of statistical analysis because it ignores the power consumption contributed by the remaining bits (e.g., seven bytes). Another issue of this strategy is that the lower bits must be correctly extracted since the carry from the lower bits affects the upper bits. We can mitigate these issues by having a sufficiently large number of traces available for the attack, for example by making more queries or by using the advanced approach described in the next subsection.

Please also note that in order to extract $W_{t+1}$ for some $t$, we must have fully restored the 64 bits of $W_t$. As shown in Eq. (5), $\Sigma_1$ is not a byte-wise operation, that is, a one-byte difference in the input changes the output by five bytes. Therefore, we cannot correctly compute $U_{t+2}$ using only a fraction of $e_{t+1}$ obtained with only a few bytes of $W_t$.

### B. ADVANCED ATTACK

In the previous subsection, we described how to extract the master seed $X_t^{(1)}$ (for $t = 0, 1, \cdots, 7$) by letting $U_{t+1}$ be the intermediate value for CPA. However, the basic attack faces two challenges: ghost peaks and query limitation. First, Ghost peaks stand for incorrect guesses that have a higher correlation coefficient than the correct guess. This happens when targeting linear operations that have fewer diffusion effects (e.g., arithmetic addition and exclusive-or) because a one-bit difference in the guessed value causes only a small change in the Hamming weight of the intermediate value. Therefore, we need to set the intermediate value after more operations to expect a larger diffusion effect. The candidates are two working variables, namely $e_{t+1}$ (in Step 15) and $a_{t+1}$ (in Step 19) in the same round of the compression function, that are relevant to $W_t$ for some $t$.

Second, if the attack requires too many queries, it not only increases the time complexity of the attack but also makes the attack impossible due to query limitation. These limitations can be attributed to the number of coin types supported by a hardware wallet, or the number of accounts (root nodes)

**Algorithm 1** Inner Hash Computation in Hardware Wallet

**Require:** Two 1024-bit input blocks $X^{(0)} = (K \oplus ipad)$ and $X^{(1)} = (M \,||\, 0x8000 \cdots 0600)$.
**Ensure:** A 512-bit digest $H_0^{(2)} \,||\, H_1^{(2)} \,||\, \cdots \,||\, H_7^{(2)}$.

**for** $i \leftarrow 0$ **to** 1 **do**
  **for** $t \leftarrow 0$ **to** 15 **do**
    $W_t \leftarrow X_t^{(i)}$
  **end for**
  **for** $t \leftarrow 16$ **to** 79 **do**
    $W_t \leftarrow \sigma_1(W_{t-2}) \boxplus W_{t-7} \boxplus \sigma_0(W_{t-15}) \boxplus W_{t-16}$
  **end for**
  $(a_0, b_0, \cdots, h_0) \leftarrow \left(H_0^{(i)}, H_1^{(i)}, \cdots, H_7^{(i)}\right)$
  **for** $t \leftarrow 0$ **to** 79 **do**
    $U_{t+1} \leftarrow h_t \boxplus \Sigma_1(e_t) \boxplus Ch(e_t, f_t, g_t) \boxplus K_t \boxplus W_t$
    $V_{t+1} \leftarrow \Sigma_0(a_t) \boxplus Maj(a_t, b_t, c_t)$
    $h_{t+1} \leftarrow g_t$
    $g_{t+1} \leftarrow f_t$
    $f_{t+1} \leftarrow e_t$
    $e_{t+1} \leftarrow d_t \boxplus U_{t+1}$
    $d_{t+1} \leftarrow c_t$
    $c_{t+1} \leftarrow b_t$
    $b_{t+1} \leftarrow a_t$
    $a_{t+1} \leftarrow U_{t+1} \boxplus V_{t+1}$
  **end for**
  $H_0^{(i+1)} \leftarrow H_0^{(i)} \boxplus a_{80}$
  $H_1^{(i+1)} \leftarrow H_1^{(i)} \boxplus b_{80}$
  $H_2^{(i+1)} \leftarrow H_2^{(i)} \boxplus c_{80}$
  $H_3^{(i+1)} \leftarrow H_3^{(i)} \boxplus d_{80}$
  $H_4^{(i+1)} \leftarrow H_4^{(i)} \boxplus e_{80}$
  $H_5^{(i+1)} \leftarrow H_5^{(i)} \boxplus f_{80}$
  $H_6^{(i+1)} \leftarrow H_6^{(i)} \boxplus g_{80}$
  $H_7^{(i+1)} \leftarrow H_7^{(i)} \boxplus h_{80}$
**end for**

that can be created. However, for the Pearson correlation coefficient to better represent the linearity between two populations, the number of samples needs to be larger. Therefore, the rest of this subsection discusses one approach to get more samples thereby reducing the number of queries needed for the attack.

One possible approach to augment the sample count is selecting multiple PoIs from a single trace, which we call the horizontal approach. In the basic MSR attack, we only exploit the power samples at one PoI by setting $U_{t+1}$ (alternatively $e_{t+1}$ or $a_{t+1}$) as the intermediate value. In the following three rounds, $e_{t+1}$ and $a_{t+1}$ are assigned to the working variables without any additional computation, in other words, $e_{t+1} = f_{t+2} = g_{t+3} = h_{t+4}$ and $a_{t+1} = b_{t+2} = c_{t+3} = d_{t+4}$ for some $t$. As a result, we can obtain a total of nine or more PoIs related to $X_t^{(1)}$ from one trace, as listed in Table 3. In other words, the number of queries required for the attack is reduced by a factor of $x$, where $x$ denotes the number of PoIs.

**TABLE 3. Variables relevant to the eight 64-bit words of the master seed.**

| Word | Relevant Variable |
|---|---|
| $X_0$ | $U_1, a_1, e_1, b_2, f_2, c_3, g_3, d_4, h_4$ |
| $X_1$ | $U_2, a_2, e_2, b_3, f_3, c_4, g_4, d_5, h_5$ |
| $X_2$ | $U_3, a_3, e_3, b_4, f_4, c_5, g_5, d_6, h_6$ |
| $X_3$ | $U_4, a_4, e_4, b_5, f_5, c_6, g_6, d_7, h_7$ |
| $X_4$ | $U_5, a_5, e_5, b_6, f_6, c_7, g_7, d_8, h_8$ |
| $X_5$ | $U_6, a_6, e_6, b_7, f_7, c_8, g_8, d_9, h_9$ |
| $X_6$ | $U_7, a_7, e_7, b_8, f_8, c_9, g_9, d_{10}, h_{10}$ |
| $X_7$ | $U_8, a_8, e_8, b_9, f_9, c_{10}, g_{10}, d_{11}, h_{11}$ |

### C. DISCUSSION

Since our attack requires multiple traces of HMAC-SHA-2 at level 0 of the HD wallets, we need to query multiple root node creations to realize the attack on hardware wallets in practice. Off-the-shelf hardware wallets support two useful features: hidden wallets and multiple coin types. Taking advantage of these features, this subsection presents two scenarios for making multiple queries. Both scenarios aim to extract the master seed without valid credentials. Once adversaries have the master seed, they can create the root nodes not protected by the passphrase, resulting in wallet cloning.

The first scenario involves accessing hidden wallets. At level 0 of the HD wallets, hidden wallets can be accessed by appending additional passphrases after the coin-specific string (e.g., `Bitcoin seed`), as shown in Fig. 5. It does not matter if the hidden wallet actually exists or not. All that matters is that adversaries enter a random passphrase and the device computes the HMAC using the master seed and given passphrase. While accessing hidden wallets, the device does not ask for any credentials, such as a PIN. As a result, adversaries can collect as many HMAC-SHA-2 power traces as they want with their chosen passphrase.

The second scenario involves creating accounts for other coin types. Hardware wallets typically support multiple coin types and multiple accounts at level 2 and level 3 of the HD wallets, respectively. Different coin types use different strings for HMAC keys at level 0. For Solana and Tron, `ed25519 seed` and `Tron seed` are used as HMAC keys, respectively. Notably, Cardano utilizes an arbitrary string as an HMAC key. This observation implies the potential creation of multiple root nodes within a hardware wallet, thereby enabling the acquisition of multiple power traces of HMAC-SHA-2 with different HMAC keys. While creating accounts, the device does not ask for any credentials, such as a PIN. As a result, adversaries can collect as many HMAC power traces as the number of coin types the wallet supports. If the wallet supports Cardano, adversaries can collect as many HMAC power traces as they want with their chosen passphrase.

## IV. EXPERIMENTS

This section experimentally demonstrates the MSR attack. Fig. 6 illustrates an experimental procedure to acquire the

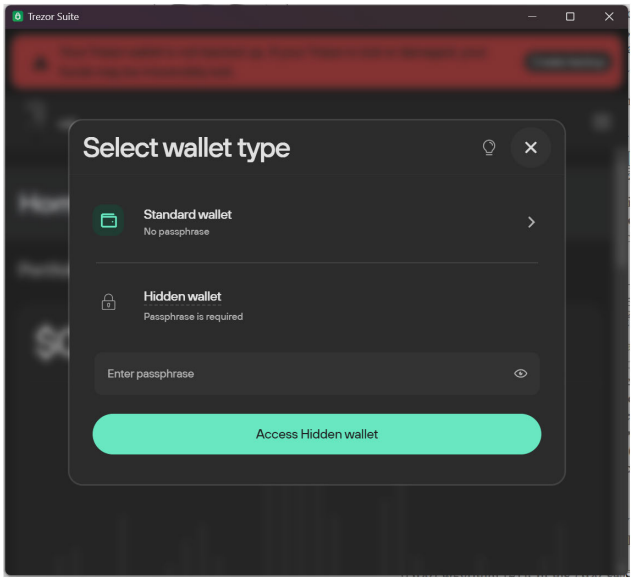



**FIGURE 5.** Example of accessing a hidden wallet by entering a passphrase.

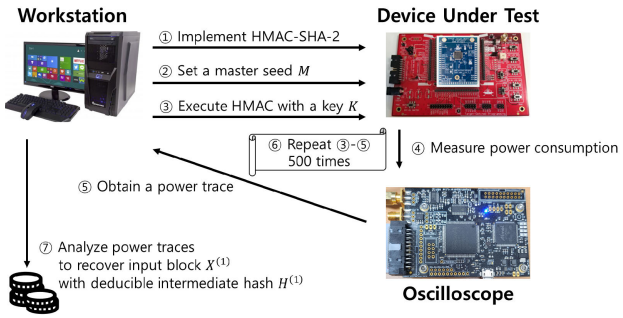



**FIGURE 6.** Power acquisition setup.

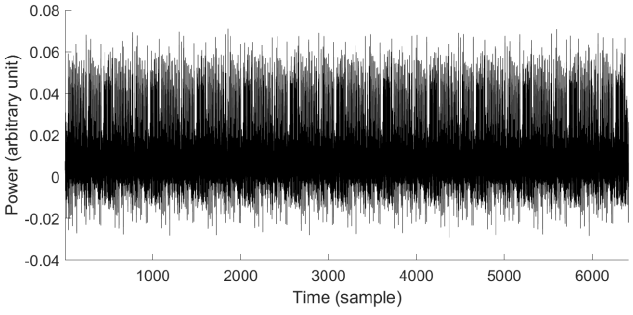



**FIGURE 7.** A power trace of the first 12 rounds of HMAC-SHA-2.

power traces of the HMAC-SHA-2 implementation (Algorithm 1) using ChipWhisperer [15]. The device under test is equipped with an STM32F415RGT6 evaluation board with a 32-bit ARM Cortex-M4 processor and running at 7.37 MHz clock frequency. We assume that the input block $X^{(1)}$ is containing the fixed master seed $M$ the attacker want to recover. Then we enter various keys $K$ into HMAC-SHA-2 to obtain the intermediate hash $H^{(1)}$ and the corresponding power trace.

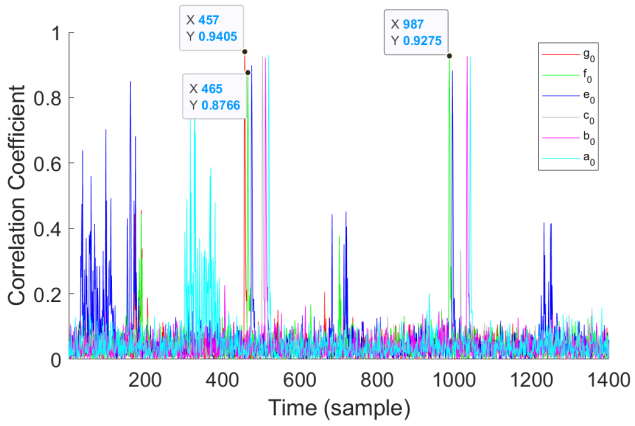



**FIGURE 8.** Correlation coefficients with known values. The number of traces is 500.

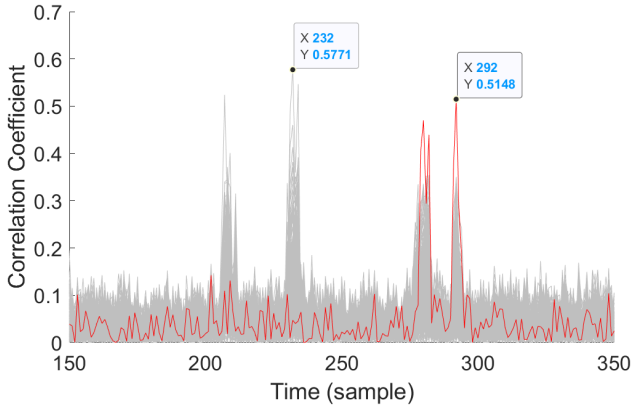



**FIGURE 9.** The basic MSR attack with $U_1$ as an intermediate value. The red solid line is for the correct guess, and 255 gray solid lines are for the incorrect guesses. The number of traces is 500.

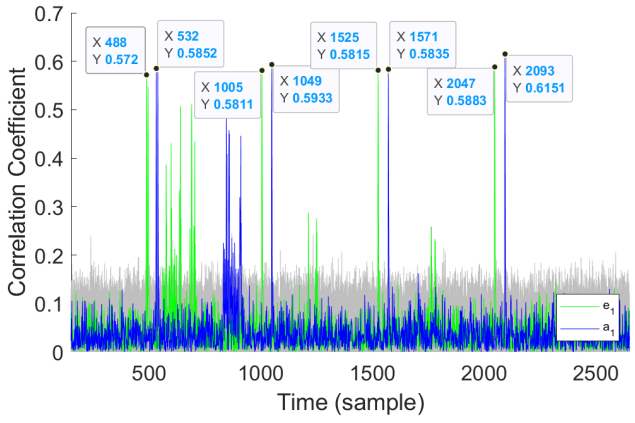



**FIGURE 10.** The MSR attack advanced by the diffusion effects. The green and blue solid lines are the correlation coefficient graphs of $e_1$ and $a_1$, respectively, obtained with the correct guess. The number of traces is 500.

We acquired 500 power traces using a ChipWhisperer-Lite oscilloscope with a sampling rate of 7.37 MHz, as same as the clock frequency. We filtered high-frequency noise by using a hardware low-pass filter whose cut frequency is 20MHz. Also, although the compression function consists of

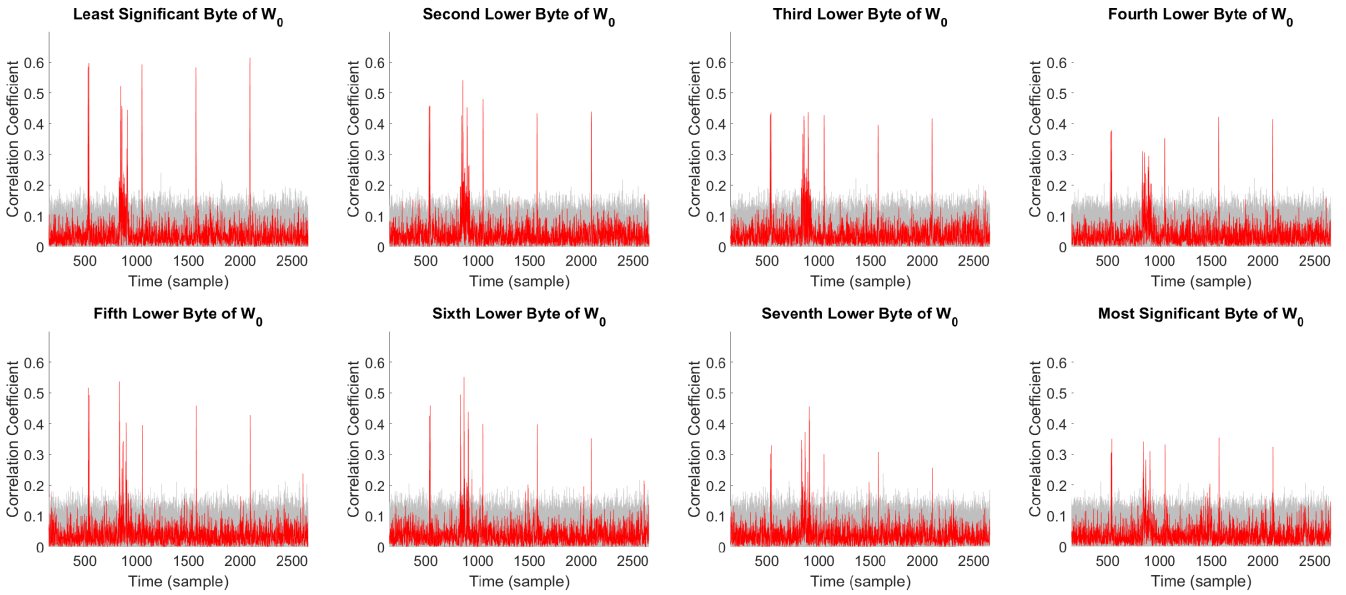



**FIGURE 11.** The 8-byte full extraction result for $W_0$. The sub-figure in row 1, column 1 is the result for the least significant byte of $W_0$, which is the same graph as $a_1$ in Fig. 10. The number of traces is 500 for each distinct attack.

80 rounds in total, we truncated the traces to include only 12 rounds by discarding the later rounds that are unnecessary for the attack. The processed trace has a length of 6416 and is displayed in Fig. 7, which seems to have a repeating pattern of 12 times.

First and foremost, we analyze the basic characteristics of the collected traces. Fig. 8 shows the correlation coefficients with the lower 32-bit of the known values, namely $g_0, f_0, e_0$, $c_0$, $b_0$, and $a_0$. Six peaks appear in the range of 400 to 600, which are likely caused by the assignments in Steps 12, 13, 14, 16, 17, and 18 of the first round. The interval between two adjacent peaks is 8, indicating the time taken for a simple assignment. We can infer that the processing of Step 10 ($U_1$) occurs before the first peak ($g_0$). We can also infer that the processing of Step 15 ($d_0 \boxplus U_1$) is in the gap between the third peak ($e_0$) and fourth peak ($c_0$). In the range of 900 to 1100, there are four peaks attributed to Steps 12, 13, 16, and 17 of the second round. Two peaks of Step 12 in one round and the next round are spaced 530 apart, which corresponds to the length of one round.

Next, we experiment with the basic MSR attack by letting the least significant byte of $U_1$ be the intermediate value for CPA. Fig. 9 shows the correlation coefficients of the correct guess in the red line and the other 255 correlation coefficients in the gray lines. The timing of the peaks in this figure corresponds to what we predicted from our previous analysis. However, the ghost peaks have a value of 0.5771, which exceeds the maximum value of 0.5148 calculated with the correct guess. As expected in Section III, the basic MSR attack with $U_1$ fails to find the correct master seed.

To address the ghost peak issue, we previously presented two ways to advance the MSR attack in Section III. The first way utilizes other variables affected by more diffusion effect. The candidates are two working variables $e_1 = d_0 \boxplus U_1$ and $a_1 = U_1 \boxplus V_1$. Fig. 10 shows two correlation coefficient graphs of $e_1$ in green and $a_1$ in blue. Compared to the basic attack, we can see that the ghost peaks disappear and the maximum value rises to 0.6151. As mentioned in Section III, we can only attempt to extract $W_1$ after we have fully restored the 8 bytes of $W_0$. Therefore, we also present the results for the remaining 7 bytes in Fig. 11.

The second way is the horizontal approach that makes sub-traces from each trace, increasing the number of traces used in the analysis, or, decreasing the number of queries required for the attack. In Fig. 10, four pairs of peaks appear with an interval of about $522 = 530 - 8$. This is because two identities $e_1 = f_2 = g_3 = h_4$ and $a_1 = b_2 = c_3 = d_4$ hold, as mentioned in Section III. In addition to that, $e_1$ and $a_1$ are called by other operations, causing peaks at various points. For example, the multiple green peaks in the 500-700 range are due to $e_1$ being input to the $\Sigma_1$ and $Ch$ functions when calculating $U_2$ in Step 10. We set a threshold of 0.3, approximately half of the maximum correlation coefficients, and selected 20 points above this threshold as PoIs. Consequently, We can obtain sub-traces by truncating each trace with a margin of three points before and after to encompass the corresponding PoIs. In this way, we can maintain the attack performance while requiring 20 times fewer queries. The results of one-byte CPA with only 500 sub-traces obtained from $500/20 = 25$ traces are depicted in Fig. 12.

Lastly, we compare the performance of the proposed attacks by presenting the number of queries required to extract the master seed. Fig. 13 shows the maximum correlation coefficients with respect to the number of traces

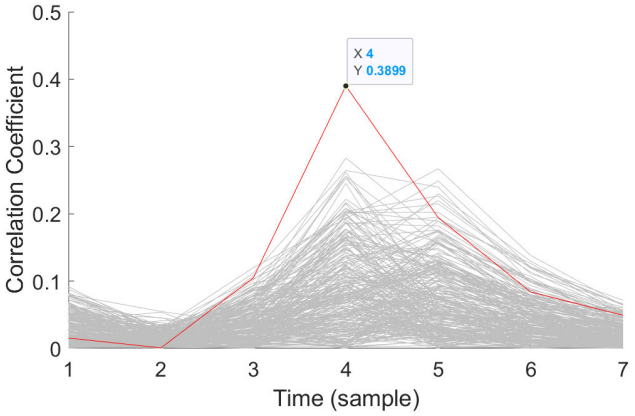



**FIGURE 12.** The MSR attack advanced by the horizontal approach. The number of sub-traces is 500 and of traces is 25.

used for each attack. The first row is for the basic MSR attack, which shows the maximum correlation coefficient over time when $U_1$ of the first round is taken as the intermediate value. The red line, calculated with the correct guess, rises above the other gray lines when the number of traces is 64, and at 500, the correlation coefficient is 0.5148, as same as in Fig. 9. Again, however, the basic MSR attack should be interpreted from an attacker's perspective (not an analyst's) as a failure to extract the master seed due to the presence of indistinguishable ghost peaks. The second row is for an advanced attack, which is the same as the basic attack but with the intermediate values changed to $a_1$ instead of $U_1$. Because of this change, the attack succeeds with only 21 traces, and at 500, the correlation coefficient is 0.6151, as same as in Fig. 10. The third row is for an even more advanced attack, which applies a horizontal approach that exploits multiple samples on a single trace. Here, the $x$-axis is not the number of traces but the number of sub-traces, so the query required for the attack is actually the number of sub-traces divided by the number of PoIs. Thus, this graph shows that the attack can be successful with only 74 sub-traces, or just four queries. On the other hand, a steep decrease in the graph indicates points where a 'bad' sub-trace, whose correlation coefficient was just above the threshold, contributes to the calculation.

In this section, we presented experimental results that validate the basic and advanced attacks. While the basic MSR attack can be easily derived from research in the field of SCA, it suffers from the problem of ghost peaks, fails to find the correct key, and is not feasible due to the properties of hardware wallets that do not allow unrestricted queries. However, our experiments on the advanced MSR attack show that it can be conducted with a small number of queries and can also solve the ghost peak issue. As a result, it is possible to recover the master seed from a lost or stolen wallet without knowing the valid PIN, implying that an attacker could access and steal all the assets currently stored in the wallet, as well as any future assets that may be added.

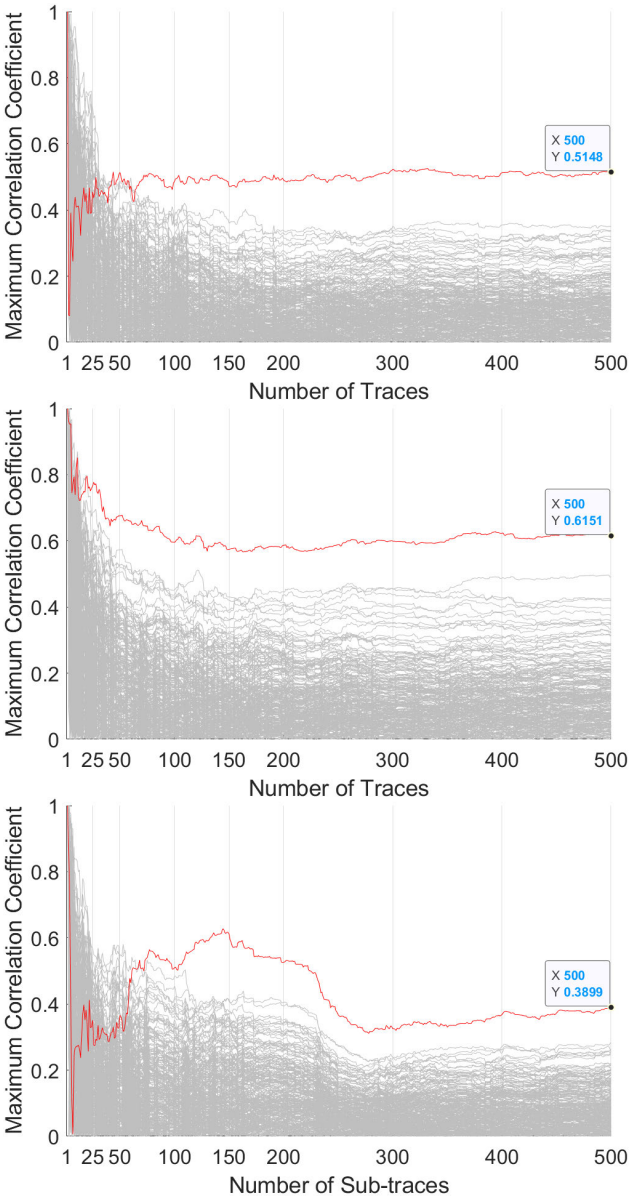



**FIGURE 13.** Correlation coefficients with respect to the number of traces for each attack.

## V. CONCLUSION

This paper proposes a practical SCA attack on hardware wallets aimed at extracting master seeds in a non-profiling, non-invasive environment. We exploit the power consumption of HMAC-SHA-2 during root node creation that does not require any valid credentials. In our experimental setup, we successfully extracted the 512-bit master seed with just four queries. As a result, it implies that by restoring the master seed from a lost or stolen wallet without knowing the wallet's valid PIN, an attacker could steal all the property stored (and to be stored) in that wallet. It reminds us that the SCA attack is a significant threat even in the cryptocurrency ecosystem. it would be a good research direction to develop attacks that

are possible with fewer queries, or to demonstrate attacks on real hardware wallets instead of test boards.

We look forward to the maturation of blockchain technology. We conclude this paper by proposing the following countermeasures to prevent SCA attacks. It would be a good research direction to investigate more efficient countermeasures not mentioned here.

### A. FOR WALLET MANUFACTURERS

Ensure that a hardware wallet prompts the user for credentials before executing any action based on a cryptographic algorithm, even if that action is not for signing a previously received transaction. Also, consider masking which is a common countermeasure used in cryptographic algorithms. This is done by dividing the intermediate value into multiple shares so that the expected intermediate value is not involved in the computation.

### B. FOR HARDWARE ENGINEERS

Introducing intentional power noise can be an inefficient but effective countermeasure. This can disrupt the correlation between power consumption and sensitive data, making it harder for attackers to extract information.

### C. FOR WALLET USERS

Employ hidden wallets protected by strong passphrases. While this may be inconvenient for users, it offers an extra layer of security to safeguard against potential attacks. Most fundamentally, do not lose your hardware wallet and use it in a trusted location only.

## REFERENCES


[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Nov. 16, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] T. Dzetkulic, P. Rusnak, and J. Hoenicke. (2022). *Trezor Firmware*. [Online]. Available: https://github.com/trezor/trezor-firmware/blob/master/crypto/ecdsa.c

[3] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, p. 15, May 2020.

[4] P. Wuille et al. (2012). *Hierarchical Deterministic Wallets*. Accessed: Nov. 6, 2022. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[5] M. Palatinus et al. (2014). *Purpose Field for Deterministic Wallets*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki

[6] (2014). *Multi-Account Hierarchy for Deterministic Wallets*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki

[7] D. Weigl et al. (2016). *Derivation Scheme for P2wpkh-nested-in-p2sh Based Accounts*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0049.mediawiki

[8] P. Rusnak. (2017). *Derivation Scheme for P2wpkh Based Accounts*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0084.mediawiki

[9] Z. Kazemi, A. Papadimitriou, I. Souvatzoglou, E. Aerabi, M. M. Ahmed, D. Hely, and V. Beroulle, "On a low cost fault injection framework for security assessment of cyber-physical systems: Clock glitch attacks," in *Proc. IEEE 4th Int. Verification Secur. Workshop (IVSW)*, Jul. 2019, pp. 7–12.

[10] C. Bozzato, R. Focardi, and F. Palmarini, "Shaping the glitch: Optimizing voltage fault injection attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 199–224, Feb. 2019.

[11] M. S. Kelly and K. Mayes, "High precision laser fault injection using low-cost components," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Dec. 2020, pp. 219–228.

[12] M. A. Elmohr, H. Liao, and C. H. Gebotys, "EM fault injection on ARM and RISC-V," in *Proc. 21st Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2020, pp. 206–212.

[13] T. G. Malkin, F.-X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Fault Diagnosis and Tolerance in Cryptography*. Berlin, Germany: Springer, 2006, pp. 159–172.

[14] D. Nedospasov, J. Datko, and T. Roth. (2018). *Wallet.Fail*. [Online]. Available: https://wallet.fail

[15] C. O. Flynn, "MIN()imum failure: EMFI attacks against USB stacks," in *Proc. 13th USENIX Workshop Offensive Technol.*, Aug. 2019, pp. 1–10.

[16] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2004, pp. 16–29.

[17] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1999, pp. 388–397.

[18] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2001, pp. 251–261.

[19] R. McEvoy, M. Tunstall, C. C. Murphy, and W. P. Marnane, "Differential power analysis of HMAC based on SHA-2, and countermeasures," in *Information Security Applications*. Berlin, Germany: Springer, 2007, pp. 317–332.

[20] C. H. Gebotys, B. A. White, and E. Mateos, "Preaveraging and carry propagate approaches to side-channel analysis of HMAC-SHA256," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 1, pp. 1–19, Feb. 2016.

[21] D. Oku, M. Yanagisawa, and N. Togawa, "A robust scan-based side-channel attack method against HMAC-SHA-256 circuits," in *Proc. IEEE 7th Int. Conf. Consum. Electron.*, Sep. 2017, pp. 79–84.

[22] J.-W. Ma, X.-G. Guan, T. Zhou, and T. Sun, "A new countermeasure against side channel attack for HMAC-SM3 hardware," in *Proc. IEEE 12th Int. Conf. ASIC (ASICON)*, Oct. 2017, pp. 327–330.

[23] Y. Belenky, I. Dushar, V. Teper, H. Chernyshchyk, L. Azriel, and Y. Kreimer, "First full-fledged side channel attack on HMAC-SHA-2," in *Constructive Side-Channel Analysis and Secure Design*. Springer, 2021, pp. 31–52.

[24] Y. Belenky, I. Dushar, V. Teper, V. Bugaenko, O. Karavaev, L. Azriel, and Y. Kreimer, "Carry-based differential power analysis (CDPA) and its application to attacking HMAC-SHA-2," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 1–29, Jun. 2023.

[25] S. Belaïd, L. Bettale, E. Dottax, L. Genelle, and F. Rondepierre, "Differential power analysis of HMAC SHA-2 in the Hamming weight model," in *Proc. Int. Conf. Secur. Cryptography (SECRYPT)*, Jul. 2013, pp. 1–12.

[26] S. Belaïd, L. Bettale, E. Dottax, L. Genelle, and F. Rondepierre, "Differential power analysis of HMAC SHA-1 and HMAC SHA-2 in the Hamming weight model," in *Communications in Computer and Information Science*. Cham, Switzerland: Springer, 2015, pp. 363–379.

[27] T. Zhou, Y. Zhu, N. Jing, T. Nan, W. Li, and B. Peng, "Reliable SoC design and implementation of SHA-3-HMAC algorithm with attack protection," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2020, pp. 88–93.

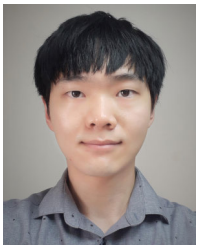
[28] M. S. Pedro, V. Servant, and C. Guillemet, "Side-channel assessment of open source hardware wallets," *Cryptology ePrint Archive*, 2019.

[29] D. Park, M. Choi, G. Kim, D. Bae, H. Kim, and S. Hong, "Stealing keys from hardware wallets: A single trace side-channel attack on elliptic curve scalar multiplication without profiling," *IEEE Access*, vol. 11, pp. 44578–44589, 2023.

[30] E. Amossys et al., "Inter-Cesti: Methodological and technical feedbacks on hardware devices evaluations," in *Proc. NIST Non-Invasive Attack Testing Workshop*, vol. 7, 2011, pp. 115–136.

[31] D. Evans, P. Bond, and K. Brown, "Advanced encryption standards," *Rivier Academic Journal*, vol. 6, no. 2, pp. 1–4, 2024.

[32] C. Gutierrez and J. Turner. (2008). *The Keyed-Hash Message Authentication Code*. [Online]. Available: https://csrc.nist.gov/files/pubs/fips/198-1/final/docs/fips-198-1_final.pdf
[33] P. Pritzker and W. May, "Secure hash standard (SHS)," *Fips Pub*, vol. 180, no. 4, Nov. 2012.



**DONGJUN PARK** received the B.S. degree in information security from Sejong University, Seoul, South Korea, in 2018, and the M.S. degree in information security from Korea University, Seoul, in 2020. He is currently pursuing the Ph.D. degree. Since 2018, he has been a Research Assistant with the Institute of Cyber Security and Privacy (ICSP), School of Cyber Security (SCS), Korea University. His research interests include cryptography, hardware security, and side-channel attacks.

**JOONSUP KIM** received the B.S. degree in mathematics from Korea University, Seoul, South Korea, in 2017, where he is currently pursuing the M.S. degree with the School of Cybersecurity. His research interests include side-channel attacks and side-channel security of deep learning neural networks.

**HEESEOK KIM** (Member, IEEE) received the B.S. degree in mathematics from Yonsei University, Seoul, South Korea, in 2006, and the M.S. and Ph.D. degrees in engineering and information security from Korea University, Seoul, in 2008 and 2011, respectively. He was a Postdoctoral Researcher with the University of Bristol, U.K., from 2011 to 2012. From 2013 to 2016, he was a Senior Researcher with Korea Institute of Science and Technology Information (KISTI). Since 2016, he has been with Korea University. His research interests include side-channel attacks, cryptography, and network security.

**SEOKHIE HONG** (Member, IEEE) received the M.S. and Ph.D. degrees in mathematics from Korea University, in 1997 and 2001, respectively. From 2000 to 2004, he was with Security Technologies Inc. From 2004 to 2005, he conducted postdoctoral research with COSIC, KU Leuven, Belgium. He joined the Graduate School of Cyber Security, Korea University. His research interests include cryptography, public and symmetric key cryptosystems, hash functions, and message authentication codes.