Multilayered Defense-in-Depth Architecture for Cryptocurrency Wallet

Hossein Rezaeighaleh
Department of Computer Science
University of Central Florida
Orlando, USA
e-mail: rezaei@knights.ucf.edu

Cliff C. Zou
Department of Computer Science
University of Central Florida
Orlando, USA
e-mail: czou@cs.ucf.edu

2020 IEEE 6th International Conference on Computer and Communications (ICCC) | 978-1-7281-8635-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICCC51575.2020.9345013

***Abstract*—A significant challenge in blockchain and cryptocurrencies is protecting private keys from potential hackers because nobody can rollback a transaction made with a stolen key once the blockchain network confirms the transaction. The technical solution to protect private keys is cryptocurrency wallets, a piece of software, hardware, or a combination of them to manage the keys. In this paper, we propose a multilayered architecture for cryptocurrency wallets based on a Defense-in-Depth strategy to protect private keys with a balance between convenience and security. The user protects the private keys in three restricted layers with different protection mechanisms. So, a single breach cannot threaten the entire fund, and it saves time for the user to respond. We implement a proof-of-concept of our proposed architecture on both a smart card hardware wallet and an Android smartphone wallet with no performance penalty. Furthermore, we analyze the security of our proposed architecture with two adversary models.**

Keywords-blockchain, cryptocurrency, wallet, defense-in-depth, bitcoin.

I. Introduction

Cryptocurrencies are not currency until the owners can purchase something with them. Today, a user can perform various electronic commerce transactions like paying a bill [1] booking a hotel or flight [2] purchasing online products [3], and paying taxes [4] with cryptocurrency.

As blockchain and cryptocurrencies become increasingly popular and practical in electronic commerce, they also become more attractive targets for hackers. Every week, we read the news of stealing money from exchanges, servers, and cryptocurrency owners. A big challenge in bitcoin and almost all blockchains is protecting the private keys of cryptocurrency owners. Blockchain usually uses elliptic-curve asymmetric cryptography to control the ownership of coins or accounts. For example, when Alice signs a transaction with her private key to transfer coins to Bob, the blockchain network will verify the signature of the transaction with Alice's public key. After being confirmed by the blockchain network, the transaction, unlike the traditional bank transfer, cannot be rolled back by anyone.

Consequently, the private key has full control of the crypto funds, and the most crucial task of a user is keeping her private keys safe. It is one of the fundamental challenges in cryptocurrencies [5]. Existing systems require a particular software or hardware called cryptocurrency wallet to store the private keys and sign the transactions. Cryptocurrency wallets have a spectrum of forms from online wallets to cold wallets, while many experts believe the most secure one is the hardware wallet. The hardware wallets are good but not enough because they are hard to use in comparison to hot wallets (i.e., software wallets) and smartphone wallets. We need an appropriate setup when using hardware wallets to achieve a balance between convenience and security.

Defense-in-Depth (DiD) is an approach in IT security that usually conveys multiple layers with various security mechanisms to protect a system from attacks in several steps. DiD applies to all IT systems and is a standard solution for network security. In this paper, we propose a multi-layer architecture that provides a Defense-in-Depth design for cryptocurrency wallets. We propose a layered deployment of wallets that delivers a balance between convenience with security for cryptocurrencies. The user protects the private keys in three restricted layers with different protection mechanisms. So, a single breach cannot threaten the entire fund, and it saves time for the user to respond. This paper provides the following research contributions:

- Proposing a layered architecture for cryptocurrency wallets that is secure yet convenient for average users
- Implementing a proof-of-concept on a hardware wallet and an android wallet and evaluating its performance
- Providing adversary models to analyze the security of the proposed layered architecture

In the rest of this paper, in section 0 we review previous works to use in our proposed architecture. Next, we explain our proposed layered architecture for cryptocurrency wallets in section III and our proof-of-concept implementation in section IV. Finally, in section V, we provide adversary models to analyze the security of our proposed model and finish with the conclusion in section VI.

II. Related Works

In this section, we review two related works that we will use to create a multi-layer architecture for cryptocurrency wallets.

A. Wallet Backup

Existing cryptocurrency wallets usually use the paper backup. The wallet generates a mnemonic word list to convert the master seed from digital form to physical form as a

978-1-7281-8635-1/20/$31.00 ©2020 IEEE

backup[6]. The user may either save these words in a computer file or writes them down on a piece of paper. In our previous paper [7], we suggest a new mechanism to back up a wallet on another wallet directly with the elliptic-curve Diffie-Hellman key agreement.

In contrast to the paper-based backup, our scheme uses Elliptic-Cure Cryptography (ECC) to transfer the keys to another wallet. So, the user does not need to either write a list of words or remember a complex long passphrase.

Our new scheme uses elliptic-curve cryptography to back up the keys. It employs a crafted version of the Elliptic-Curve Diffie-Hellman (ECDH) key agreement protocol [8] for backup and recovery. The problem of ECDH is the Man-In-The-Middle attack where a hacker replaces the public key of the backup wallet by a fake public key, and the main wallet cannot distinguish the original backup public key from the fake one. To solve this problem, we employ the side-channel user visual confirmation (verification code, aka vcode). Existing hardware wallets use a similar method to confirm transaction information like the receiver address, amount, and fee before signing [9][10].

In the backup process, there are two wallets: the main wallet and the backup wallet. Before start, the main wallet has generated and stored the master seed, and the goal of our proposed backup process is to transfer an encrypted copy of the master seed from the main wallet to the backup wallet. We assume both wallets have a screen and (at least) one physical button. Also, we assume the backup channel is an untrusted terminal, like a smartphone that may be compromised by a hacker. The vcode is displayed on the hardware wallets' screen for user verification. The values shown on the two wallets' screen should be identical.

B. *Super-Wallet and Sub-Wallet*

Storing all funds on only one wallet and use that for daily spending is risky because it is possible that the wallet crashes, gets damaged, or stolen. So, the authors of [5] propose a simple but useful idea called super-wallet and sub-wallet. The user stores the large fund on the super-wallet and refills a small fund to the sub-wallet frequently or as needed. So, she uses the sub-wallet for daily spending and the super-wallet as a saving account.

The sub-wallet/super-wallet model proposed in [5] is simple. The user has two regular wallets and uses one of them as the super-wallet and another one as the sub-wallet. One disadvantage of this model is that it requires one transaction per refill, which means that the user pays a miner fee and waits for the network confirmations for each small refilling. Also, the user must get backup of both wallets, and the sending transaction is vulnerable to the MITM (Man-In-The-Middle) attack for receiving address injection like other regular sending transactions.

To resolve these challenges in the super-wallet/sub-wallet model, we propose a new scheme that we call the Deterministic Sub-walle [11]. In our model, the root of key trees of the super-wallet and sub-wallet are linked. In other words, the sub-wallet seed is derived from the super-wallet seed. It means that the super-wallet seed can generate the entire key tree of the super-wallet and also all the sub-wallets. So, the super-wallet does not require to get the sub-wallet address from the external source, and it generates them internally. It also eliminates the sub-wallet backup process because the super-wallet backup is enough, and it can regenerate the sub-wallet seeds.

To link the super-wallet seed (*mSeed*) to the sub-wallet seeds (*subSeed*) we use the following derivation function in [11]. This formula is similar to the existing master key generation function in [11] with some modifications. The xxxx is the four-digit hexadecimal index of the sub-wallet starting from zero. The output of this function is a 512-bit deterministic pseudo-random value, which can be used as a regular seed to generate the entire key tree of the sub-wallet.

subSeed = HMAC-SHA512(key="Sub-wallet xxxx", data=mSeed)

We use a modified version of our proposed mechanism [7] to transport a sub-wallet seed from the super-wallet to the sub-wallet. The modified version uses the same steps but transports a sub-wallet seed instead of a master seed. It uses ECDH to encrypt the seed and vcode as the side-channel user visual confirmation. After that, the super-wallet creates a refill transaction and publishes it on the blockchain to send funds to the sub-wallet addresses.

III. Proposed Multi-Layer Wallet

To protect the private keys from attackers, we introduce a defense-in-depth architecture for cryptocurrency wallets. Our proposed architecture has three layers with different usage and protection mechanisms, which makes a balance between usability and security. Figure 1 demonstrates this architecture. It has three layers, including *offline layer*, *protected layer*, and *online layer*.

The protected layer consists of a *superior wallet*. This wallet conveys the master seed, which generates the entire key tree and all addresses. The offline layer has at least one *backup wallet* where it is a clone of the superior wallet. We use our previously proposed method in [7] for encrypted wallet-to-wallet cloning. The online layer can have multiple *spending wallets* for regular daily purchases. A spending wallet has a subordinate seed from the superior wallet with a limited fund. We use our previously proposed mechanism in [11] for key derivation to generating subordinate seeds and seed transferring from superior wallet to a spending wallet.

We have revised our previous algorithm [11] to support our new proposed architecture. Firstly, we modify the derivation function as follows where *swSeed* stands for spending wallet seed, mSeed stands for master seed, and xxxx indicates the spending wallet index starts from zero in 4-digit hex number format (0000). The superior wallet uses the derivation function only when it creates a new seed for a spending wallet.

swSeed = HMAC-SHA512(key="swSeed xxxx", data=mSeed)

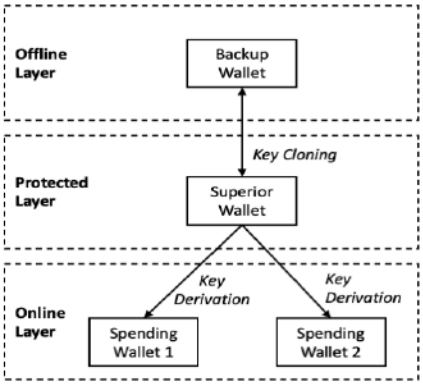

Figure 1. The proposed multi-layer defense-in-depth architecture for cryptocurrency wallets.

Secondly, we also modify the refilling address selection policy. On the original work [11], the wallet only refills the first address index of each derived seed. However, in our new proposed architecture, the superior wallet uses multiple addresses of a spending wallet seed. For each refilling, it searches the blockchain to find the first unused address to send the fund.

The offline layer is designed to be offline and does not need any connection to the blockchain network. It gets online if and only if an incident occurs for the superior wallet and needs an emergency response. If the superior wallet is compromised by an attack or is lost, the backup wallet generates a brand-new master seed. It creates a blockchain transaction to transfer all available funds of the last master seed to an address under the new master seed. It avoids any unintended transfer from the superior wallet as soon as possible. We recommend a secure hardware wallet with a secure element, a trusted display, and an embedded button for the backup wallet.

The protected layer has only one superior wallet. This wallet only refills the spending wallets. It calculates the spending wallet addresses internally, so it does not send any fund to other addresses that are vulnerable to MITM attack for receiving address injection. Similar to the backup wallet, we recommend a secure hardware wallet for the superior wallet too.

Finally, the online layer can have multiple spending wallets. These wallets can be software wallets like smartphone wallets or hot wallets (third-party hosted wallets). Spending wallets do not need a backup because the superior wallet can recreate their seeds[11].

These three layers provide a balance between security and usability. While the user stores her large fund on the superior wallet and creates a clone of it on the backup wallet, she enjoys the convenience of a smartphone wallet or hot wallet to purchase online and pay her expenses.

Receiving funds does not need private keys, so there are two possible options. If the user context does not have privacy concerns, she can generate an address under master seed on the superior wallet to share with others. If the context is sensitive to privacy, the superior wallet creates an extended public key to generates hierarchical deterministic addresses outside of the superior wallet without exposing the master seed or any private keys[12].

For better understanding, we explain an example setup. Alice has 10 Bitcoin (BTC) equals to $100k (we assume the bitcoin price is $10,000 for simplicity). She stores her fund into the superior wallet, which is a secure hardware wallet and keeps it safe at her home. She creates a backup wallet, which is a secure hardware wallet too, and put it in a safe deposit box in a bank that is physically secure. Then, she installs a wallet app on her smartphone and makes it a spending wallet under the superior wallet and refills 0.5 BTC ($5K) into it. To receive her salary, she gets a receiving address from the superior wallet and shares it with her employer. She gets paid bi-weekly with bitcoin without requiring using the superior wallet. Alice uses the smartphone to buy a coffee, pay the bills, and purchase from online stores. When the spending wallet has a low balance, she refills it using the superior wallet.

For convenience, Alice uses a type of hardware wallet for superior wallets and backup wallets that support Bluetooth or NFC, and she can do backup and refilling operations using a smartphone. However, she may use an offline laptop or another offline smartphone for better protection to do the backup and refilling.

Now, we consider two possible security incidents and how the defense-in-depth architecture mitigates them. First, assume an incident in the online layer, for example, Alice loses her smartphone or recognizes a malware program on her smartphone. In this scenario, only the spending wallet is at risk with a maximum of 0.5 BTC amount. To respond to this incident, she uses superior wallet to transfer the fund of the suspected spending wallet to an address under the master seed. Then, she can reset her smartphone or get a new one, and the superior wallet generates a brand-new spending wallet seed and transfers the seed to the smartphone.

Secondly, an incident can occur in the protected layer. For example, Alice may lose the superior wallet because of the physical robbery in her home. Since she uses a secure hardware wallet for the superior wallet, it is password protected and, if an attacker tries password guessing more than the retry counter (i.e., five times), the wallet will be blocked permanently. On the other hand, for responding to this incident, Alice uses the backup wallet to generate a brand-new master seed and create a blockchain transaction to transfer all funds from the previous master seed to an address under the new master seed. She should do that as soon as possible before any breach of the suspected superior wallet. She also must create a new backup and regenerate the subordinate spending wallets.

IV. Proof-of-Concept

To evaluate our proposed architecture on bitcoin, we implement the backup wallet and the superior wallet on a hardware wallet device from scratch that supports fundamental functionalities of hierarchical deterministic wallets, according to BIP-32 [12] and BIP-44[13]. We use a secure element for key operations such as key generation and digital signature.

TABLE I. ADVERSARY MODEL I: MALICIOUS APP WITH DANGEROUS PERMISSION

Assumptions	Goals	Capabilities
• Android 8.1.0 • Internet access • NFC access • Knowledge of the low-level wallet protocol (APDUs)	• Capture the master seed or sub seed • Inject the adversary address to receive the fund	• Record the screen or log the pressed buttons to capture the password • Sniff the low-level packets to capture the master seed or spending seed • Inject the adversary address into spending wallet refill transaction to receive the fund (MITM) • Replace the backup or spending wallet original transport public key with the adversary public key to extract the master seed or spending seed (MITM)

We choose a smart card that has essential parts of a secure hardware wallet. It has a secure element for cryptography operations and key storage, a screen to display sensitive information to the user, and a button to get confirmation from the user. Figure 2 demonstrates a picture of our test device. This device is in credit card size and has NFC and contact interfaces to communicate.

Our test smart card has the following specification; Java Card 3.0.5, Global Platform 2.2.1, e-paper display 256x256 pixel, 2.5 KB memory, 170 KB storage, contact and NFC interfaces, support for SHA256, SHA512, HMAC, AES256, ECC256, and ECDH algorithm.

Since the secure element is a resource-constraint device with limited memory and processing ability, our code must use the minimum amount of memory. We use the sharing memory technique and allocate the entire memory to only two arrays. We pass these arrays with the maintained indexes to the functions that require arrays, and it minimizes the heap consumption.

Furthermore, we do not use a very nested function and any recursive call, and it minimizes stack memory usage. We use the Java Card framework [14] to program the secure element. It is a limited version of Java Virtual Machine with fewer features to run on microcontrollers and secure elements. We compile the code with the Java Development Kit, convert it to a Card Application (CAP), and load it into the secure element.

One of our implementation challenges is the public key derivation. In ECC, a public key calculates by multiplying the private key and the Generator point (G) [15]. Unfortunately, our secure element (and many others) does not support EEC multiplication, and its software implementation has no acceptable performance due to the limited resources of the secure element. However, Java Card API and our secure element support Elliptic-Curve Diffie-Hellman (ECDH) key agreement.

In ECDH, each party calculates a secret by multiplying its private key and the other party public key. An ECC public key is an elliptic-curve point aka EC point. Therefore, the ECDH function mathematically multiplies a scaler and an EC point. We use the ECDH function with the private key as the scaler and the Generator point (G) as the EC point. Thus, the result of ECDH is the public key.

For the spending wallet, we develop a mobile app to test our prototype with a smartphone. We use a Google Pixel smartphone with an NFC antenna and the following specifications: Google Pixel G-2PW4100 smartphone, quad-core Qualcomm Snapdragon 821 processor with two 2.15 GHz cores and two 1.6 GHz cores, 4 GB memory, 32 GB storage, and Android 8.1.0.

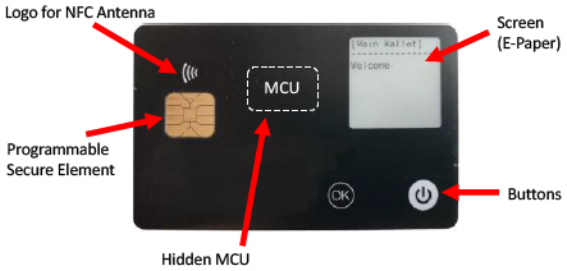

Figure 2. Test device with secure element, screen, and button to create a hardware wallet

According to our evaluation, the total execution time for creating a backup on the test smart card takes less than one second to complete based on our prototype[7]. The derivation mechanism and refilling a spending wallet also can complete around one second [11].

V. SECURITY ANALYSIS

In this section, we analyze the security aspect of our proposed architecture and the implemented proof-of-concept on hardware wallets and smartphones. Firstly, we argue about the security advantages of our proposed architecture in comparison to the existing solutions. Next, we provide appropriate adversary models to investigate the possible major attacks and countermeasures.

A. Security Advantages

No Paper Backup: Spending wallets do not need any backup, and the superior wallet has one or more identical backup on other hardware wallets. Therefore, all backups are in digital format, and there is no physical backup on a paper that is vulnerable to traditional attacks.

Less Vulnerable to Lose Large Amount: In our architecture, we split the fund between two layers. The protected layer stores a large amount and is used rarely, while the online layer stores a small amount and is used frequently. Therefore, a spending wallet is more exposed to the network and accessible for attacks; however, it has a small fund at risk. On the other hand, the superior wallet is less accessible on the network, and hence, more secure to possible attacks.

Control of spending wallets: The superior wallet can regenerate the spending wallet seed and all corresponding keys. Therefore, if a spending wallet is lost or stolen, the user can use the superior wallet to recover all spending wallet keys

and transfer their funds to a brand-new address and empties the spending wallet.

B. Adversary Models

Authors of [16] survey security analyses on several papers and propose a comprehensive adversary model to employ in future security researches. This model defines three aspects of an adversary, including Assumptions, Goals, and Capabilities. The assumptions describe the environment, resources, and equipment of the adversary. The goals identify the intentions of the adversary and explain why he targets the system. The capabilities are the abilities and actions that the adversary performs to achieve his goals.

The authors of [16] discuss various adversary models for diverse environments like personal computers, networks, and cryptography parties. We use the models of the smartphone environment to measure the security of our final prototype on an Android smartphone.

1) Malicious app adversary model

The adversary model has different properties in various fields of study, and the authors of [16] provide several adversary models for smartphone applications. Their proposed Malicious App Adversary Model is appropriate for our conditions. This model includes three sub-models based on the app permissions: Zero Permission Adversary only has access to the list of installed apps and files stored on external storage. Normal Permission Adversary adds Internet access, Bluetooth, and NFC interfaces. Finally, Dangerous Permission Adversary has access to all resources such as camera, microphone, contact, and SMS. In this paper, we use the Dangerous Permission Adversary model to assume maximum power for the attacker that is defined in Table I.

According to Table I., the adversary could capture the user's password by recording the screen or log the pressed buttons. Even though some solutions exist for this attack like Trezor [17] that uses a blind visual matrix to avoid entering a plain password on the host, we use a physical button on the hardware wallet for confirmation.

Also, the adversary may sniff the transmitted messages between hardware wallets and the smartphone app to eavesdrop the master seed or spending seed. Our mechanism is secure against this attack because the smartphone only transmits public information, including the superior wallet, the backup wallet and spending wallet public keys, and encrypted master seed or encrypted seed under an AES 256-bit key. Therefore, the attacker does not have access to any private data.

Another capability of the adversary is making an MITM attack to replace the receiver address by his injected address in the transaction. The classic super-wallet/sub-wallet model [5] is vulnerable to this attack because the super-wallet needs to get the sub-wallet address from the host like a smartphone. However, in our architecture, we use the deterministic sub-wallet that prevents this attack since the spending wallet seeds are derived from the superior wallet master seed, and the superior wallet generates the receiving addresses internally. Therefore, there is no need to get the receiving addresses from the external source, and the hacker has no chance to replace them.

Last but not least, the adversary may make an MITM attack to intercept the messages between the superior wallet and the backup wallet or the superior wallet and the spending wallet. Then, he replaces the backup wallet public key or the spending wallet public key by the adversary public key in ECDH key agreement, and he can recover the transferred seed.

To defend against this attack, we have used a side-channel verification code (vcode) in our mechanism. Both wallets compute their vcodes of the public key and display the vcode on their screens (see the hardware wallet shown in Figure 2). The user visually inspects and confirms the equality of these two vcodes by pressing a physical button on the superior wallet. Existing hardware wallets use a similar method to confirm transaction information like receiver address, amount, and fee before signing them. Therefore, during the wallet transfer operation, if a hacker injects his public key to the superior wallet, the user will be able to detect such an attack due to the mismatch of the two vcodes shown on two wallets' screen and reject this MITM attack.

2) Physical access adversary model

Another possible adversary model for our proposed architecture is an adversary with physical access to the superior wallet (or backup wallet). In this case, the adversary can do anything directly with the hardware wallet without the need to install a malicious app on the remote user's smartphone. Table II. demonstrates the Physical Access Adversary Model.

TABLE II. ADVERSARY MODEL II: PHYSICAL ACCESS

Assumptions	Goals	Capabilities
• Access to the hardware wallet device • Knowledge of the low-level wallet protocol (APDUs)	• Sign a transaction and send the fund to the adversary address	• Make a brute-force attack to guess the password and sign a transaction to transfer the fund

In this adversary model, the adversary can make a brute-force attack to obtain the hardware wallet password (PIN code) and sign his desired transaction. Our proposed architecture recommends a hardware wallet with a secure password for the superior wallet that has a fixed password retry counter, usually between 3 and 15. After that, the secure element locks permanently. It is a standard mechanism for secure elements. Therefore, if a hacker finds the superior wallet, he can only try a limited number of guessed passwords and could not make a brute-force attack. For instance, if the PIN code length is four digits and the retry counter is 10, the chance to find the PIN code is 0.001 (tries / possible PINs = $10/10^4 = 0.001$). On the other hand, the user has time to use her backup wallet to transfer all funds to a brand-new seed as soon as possible.

We must mention that the attacks to the security element or other hardware parts and their countermeasures are out of the scope of this paper and apply to entire hardware wallets not specific for our proposed schemes.

VI. Conclusion

In this paper, we considered protecting private keys in cryptocurrency wallets for blockchain technology. Even though the most secure choice is hardware wallets, we argued that there are critical issues that should be addressed. We introduced a multi-layer architecture for cryptocurrency wallets to provide a Defense-in-Depth approach. For our proposed architecture, we implemented a proof-of-concept on a hardware wallet and Android smartphone. We also offered performance evaluation and security analysis for our proposed architecture.

Acknowledgment

This work was supported by the National Science Foundation under grant DGE-1915780 and DGE-1723587.

References

[1] AT&T, "AT&T is the First Mobile Carrier to Accept Payment in Cryptocurrency," 23 May 2019. [Online]. Available: https://about.att.com/story/2019/att_bitpay.html.

[2] Jeff Klee, "A Letter to Our Bitcoin Customers," 20 April 2018. [Online]. Available: https://www.cheapair.com/blog/a-letter-to-our-bitcoin-customers/.

[3] Microsoft, "How to use Bitcoin to add money to your Microsoft account," 5 Oct 2018. [Online]. Available: https://support.microsoft.com/en-us/help/13942/microsoft-account-how-to-use-bitcoin-to-add-money-to-your-account.

[4] P. Vigna, "Pay Taxes With Bitcoin? Ohio Says Sure," 26 Nov 2018. [Online]. Available: https://www.wsj.com/amp/articles/pay-taxes-with-bitcoin-ohio-says-sure-1543161720.

[5] S. Barber, X. Boyen, E. Shi and E. Uzun, "Bitter to Better — How to Make Bitcoin a Better Currency," in Financial Cryptography and Data Security, Berlin, 2012.

[6] M. Palatinus, P. Rusnak, A. Voisine and S. Bowe, "Mnemonic code for generating deterministic keys," 2013. [Online]. Available: https://en.bitcoin.it/wiki/BIP_0039.

[7] H. Rezaeighaleh and C. C. Zou, "New Secure Approach to Backup Cryptocurrency Wallets," in 2019 Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019.

[8] Certicom Research, "SEC 1: Elliptic Curve Cryptography," 2009.

[9] "Ledger Nano X," [Online]. Available: https://shop.ledger.com/pages/ledger-nano-x.

[10] "Trezor One," Trezor, [Online]. Available: https://shop.trezor.io/product/trezor-one-white.

[11] H. Rezaeighaleh and C. C. Zou, "Deterministic Sub-Wallet for Cryptocurrencies," in 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019.

[12] P. Wuille, "Hierarchical Deterministic Wallets," 2012. [Online]. Available: https://en.bitcoin.it/wiki/BIP_0032.

[13] M. Palatinus and P. Rusnak, "Multi-Account Hierarchy for Deterministic Wallets," 2014. [Online]. Available: https://en.bitcoin.it/wiki/BIP_0044.

[14] Oracle, "Java Card 3 Platform Runtime Environment Specification, Classic Edition Version 3.0.5," 2015.

[15] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters," 2000.

[16] D. Quang, B. Martini and C. K.-K. Raymond, "The role of the adversary model in applied security research," Computers & Security, vol. 81, pp. 156-181, March 2019.

[17] Trezor, "User manual: Entering PIN," [Online]. Available: https://wiki.trezor.io/User_manual:Entering_PIN.