

# Quantitative Risk Assessment on a Decentralized Cryptocurrency Wallet With a Bayesian Network

1<sup>st</sup> Byeongcheol Yoo  
School of Cybersecurity, Korea  
University  
Seoul, Korea  
architectophile@gmail.com,

2<sup>nd</sup> Seungjoo Kim  
School of Cybersecurity, Korea  
University  
Seoul, Korea  
skim71@korea.ac.kr

**Abstract**—Since the creation of the first Bitcoin blockchain in 2009, the number of cryptocurrency users has steadily increased. However, the number of hacking attacks targeting assets stored in these users' cryptocurrency wallets is also increasing. Therefore, we evaluate the security of the wallets currently on the market to ensure that they are safe. We first conduct threat modeling to identify threats to cryptocurrency wallets and identify the security requirements. Second, based on the derived security requirements, we utilize attack trees and Bayesian network analysis to quantitatively measure the risks inherent in each wallet and compare them. According to the results, the average total risk in software wallets is 1.22 times greater than that in hardware wallets. In the comparison of different hardware wallets, we found that the total risk inherent to the Trezor One wallet, which has a general-purpose MCU, is 1.11 times greater than that of the Ledger Nano S wallet, which has a secure element. However, use of a secure element in a cryptocurrency wallet has been shown to be less effective at reducing risks.

**Keywords**—*risk assessment, threat modeling, cryptocurrency wallet, attack tree, Bayesian network, CVSS*

## I. INTRODUCTION

Since the creation of the first Bitcoin [1] blockchain in 2009, the number of cryptocurrency users has been steadily increasing. At the time of writing, there are thought to be more than 50 million cryptocurrency users worldwide. The number of different types of cryptocurrency has increased exponentially as the issuance and management of cryptocurrencies has become easier thanks to smart contracts [2], which were first introduced on the Ethereum blockchain [3]. Currently, there are more than 5000 cryptocurrencies worldwide, and the total market cap of cryptocurrencies is over \$900 billion.

Unfortunately, with the increase in cryptocurrency users, the threat of hacking attacks on cryptocurrency assets is also increasing. At the time of writing, the cumulative losses from cryptocurrency hacking are estimated to be over \$11 billion. As cryptocurrency exchanges are increasingly the targets of hacking attacks, the number of users who try to manage assets directly from personal cryptocurrency wallets rather than exchanges is increasing. However, accordingly, malicious code attacks targeting users' personal cryptocurrency wallets are also on the rise.

Cryptocurrency wallet services are very different from traditional financial services. The decisive difference between the two arises from two characteristics of blockchain: anonymity and decentralization. The address of a cryptocurrency account on the blockchain is created using a user's public key. By providing a signature generated from the private key that corresponds to the public key, the user proves that he is the owner of the account. This method provides very good anonymity because account information on the blockchain is not linked to the user's personal identity information. Furthermore, the transaction history ledger on the blockchain is recorded and managed in a decentralized manner. Therefore, since the same transaction ledger is managed by a large number of distributed nodes, it is very difficult to change the contents recorded on the blockchain unless someone controls more than 51% of the blockchain network. Therefore, the fact that the transaction history is recorded on the blockchain provides immutability and integrity. However, the threat of attacks against cryptocurrency wallets is increasing due to the nature of blockchain technology. This is because, even if a hacker steals a user's private key and transfers the user's cryptocurrency assets to the hacker's account, the owner of the account remains anonymous. Traditional banking systems verify a user's identity before opening an account. Also, since all account transactions are recorded and managed through the centralized system of the bank, it is easy to identify the user by the account number. However, since there is no centralized system in blockchain, users can open a new account by generating a new key pair whenever they want. In addition, due to the immutability of the blockchain ledger, cryptocurrency assets that have been sent to a hacker cannot be returned. In the traditional banking system, if money is withdrawn due to a hacking incident, it is possible to request suspension of payment for the deposited bank account. By contrast, there is no centrally managed system on the blockchain, so a hacker cannot be forcibly prevented from using their accounts. Moreover, if a user loses his or her private key, the assets in the account cannot be retrieved. This is in contrast to the traditional banking system where a user's account can be retrieved at any time once the user's identity has been verified. Therefore, a new and systematic method is needed to evaluate the security of cryptocurrency wallets, which have completely different characteristics than traditional financial services.

For this reason, several studies have been conducted to analyze the security of cryptocurrency wallets [4]-[8]. However, these studies often summarize commonly known cryptocurrency wallet attack vectors or identify the vulnerabilities of specific platform wallets rather than performing a systematic security analysis. The authors of [9], [10] analyzed threats to wallets in a more systematic manner through threat modeling. However, the studies lacked a logical basis or explanation for the analytical methods and processes that were employed, and since these studies were limited to specific platform wallets, most of the methods cannot easily be applied to other types of wallets. Most of all, there has been a lack of research on quantitative measurement and evaluation of the risks present in cryptocurrency wallets.

Therefore, in this paper, we present a methodology to systematically evaluate the risk of cryptocurrency wallets. We use a threat modeling method to identify threats that are present in cryptocurrency wallets and derive security requirements. We also quantitatively measure the risk inherent in real cryptocurrency wallets using attack trees, Bayesian network modeling, and a variety of risk assessment factors including CVSS metrics. In addition, the risk of each wallet is measured and compared to ensure that hardware wallets, which are generally known to be secure, are actually safer than software wallets. The risks of different hardware wallets are then compared with each other to gauge the relative safety of a hardware wallet with a secure element compared to a hardware wallet with a general-purpose MCU.

#### A. Scope of Study

Since this paper builds a general-purpose threat model for cryptocurrency wallets that can be applied to various platforms, it does not cover all threats caused by inherent vulnerabilities that appear only in a specific operating system or operating environment. However, certain attack vectors that are common in a specific operating environment have been included in the threat model. For example, in embedded systems, firmware update attacks or attacks using debugger pins are reasonably common, and in mobile operating systems, attacks on rooted mobile devices are common.

In this paper, we mainly focus on threats that exist within the system boundary of a user's wallet application or hardware wallet device. Therefore, threats to entities or systems outside the boundaries of the user's wallet system are not analyzed in detail. For example, the work does not specifically analyze how to attack external servers that communicate with wallet applications or how to attack blockchain networks.

Furthermore, we only analyze decentralized cryptocurrency wallets that directly manage private keys in users' devices. Centralized cryptocurrency wallets that manage users' private keys on a separate central server were excluded from the study.

## II. BACKGROUND

### A. Cryptocurrency Wallet

A cryptocurrency wallet refers to a type of software or hardware that stores a user's cryptocurrency and allows the user to transfer or receive cryptocurrency. Cryptocurrency wallets

provide information (address, balance, transaction details, etc.) on accounts held by users. In addition, a wallet generates new transaction data when a user transfers cryptocurrency to other accounts, generates a signature for the transaction, and broadcasts it to the blockchain network.

### B. Taxonomy of Cryptocurrency Wallets

Before analyzing the security of cryptocurrency wallets, we will classify cryptocurrency wallets according to the characteristics and types of each wallet. Table I shows the wallet classification results. Wallets can largely be divided into two types: decentralized wallets and centralized wallets. A decentralized wallet is a wallet that manages the full lifecycle (creation, change, use, destruction) of cryptographic keys directly in a user's device. On the other hand, a centralized wallet refers to a wallet that manages the lifecycle of cryptographic keys in a centralized server.

We only focus on decentralized wallets in this paper. This is because a decentralized wallet directly manages cryptographic keys on a user's device, which creates completely novel threats that are not present in the existing financial services sector due to the inherent characteristics of the blockchain.

Decentralized wallets can be divided into two categories: hot wallets and cold wallets. A hot wallet refers to a cryptocurrency wallet that has a network interface and can be connected to the internet at any time. For example, wallets that are installed and used on mobile devices (smartphones) or computers are classified as hot wallets. Since hot wallets are generally implemented in software form, they are also called software wallets. On the other hand, a cold wallet refers to a cryptocurrency wallet that does not have a network interface or has a physically separated key storage space that is not connected to the internet. A cold wallet is also called a hardware wallet because it is mainly implemented as a separate hardware device.

TABLE I. TAXONOMY OF CRYPTOCURRENCY WALLETS.

Criteria	Key Management	Network Connection	Platform
Wallet	Decentralized Wallet	Hot Wallet (Software Wallet)	Mobile, PC, Web, Chrome Extension
		Cold Wallet (Hardware Wallet)	Embedded System, PC with no network connection
	Centralized Wallet	-	Crypto Exchange, Cloud

### C. Related Works

As interest in cryptocurrency has grown, studies that use threat modeling to evaluate the security of cryptocurrency wallets have begun to emerge [9] [10]. Electric Coin Company, which developed Zcash, created a threat model for their ECC wallet in [9] using the Invariant-Centric Threat Modeling method proposed in [11]. An invariant refers to a security property that has been analyzed, on which a user can safely rely. However, [9] does not provide a detailed description of how the ECC wallet verified the security invariants in the threat model. Moreover, even if the invariants have been sufficiently verified, it is unreasonable to say that the wallet guarantees 100% of the

security invariants. Therefore, creating a list of security invariants but not quantitatively measuring the risk of the wallet raises questions regarding the accuracy of the security evaluation.

The Whonix team, which is developing a security operating system, created a threat model for cryptocurrency hardware wallets in [10]. That study mainly focused on the importance of the secure display of a hardware wallet, and analyzed what threats occur when an external host is infected with malicious code. The threat model in [10] is written in language that is very easy for users to understand to make it a convenient reference for users of hardware wallets. However, the types of threats that are considered in the threat model are very limited, and the focus of the study is on the security display, so the analysis of more diverse and specific threats, such as firmware modification or physical attacks, is insufficient.

As the number of people who use cold wallets to safely store cryptocurrency assets has increased, researchers have become interested in performing security analyses of cold wallets. M. Guri [8] analyzed the security of cold wallets in isolated (air-gapped) computers and showed that even cold wallets can leak private keys through covert channels. The cold wallet assumed in this paper is not a hardware wallet, but a software wallet installed on a host with no internet connection. M. Guri explained that, when transferring signed transactions to an external host, the cold wallet host can become infected with malicious code through an infected USB device. M. Guri also introduced several types of bridgware [12], which is malicious code that leaks data from air-gapped computers through various covert channels.

D. Nedospasov et al. [13] discussed the overall security vulnerabilities of the Ledger and Trezor hardware wallets. In [13], they demonstrated installation of an RF trigger on the Ledger Nano S through a supply chain attack, followed by remote approval of a transaction using an antenna without the user's consent. They also demonstrated an attack that exploits the vulnerability of the Ledger Nano S bootloader to install arbitrary firmware on the proxy MCU. In addition, they introduced a firmware compression method that can bypass firmware verification in the secure element of the Ledger Nano S. A method of remotely stealing a PIN code entered into the Ledger Blue by sniffing the radio signal generated upon entry of the code was introduced as well. In addition, they demonstrated the possibility of bypassing the memory read protection mechanism of the STM32 MCU through a glitching attack on a Trezor wallet, and stole a recovery phrase and PIN code stored in RAM using a JTAG debugger. Therefore, previous authors have shown that, even if a hardware wallet is used, hacking is possible if the attacker has physical access and a sufficient amount of time.

Various security analyses have been conducted on software wallets as well. D. He et al. [7] analyzed the vulnerability of cryptocurrency wallets running on the Android operating system, derived attack vectors, and analyzed the security of two Android wallets that are available on the market. Experiments have shown that keyboard input, screen touch input, and screen output data can be collected from Android wallets to obtain sensitive

and confidential information, such as private keys or recovery phrases.

A. R. Sai et al. [4] analyzed the security and privacy of mobile wallets. In that paper, threats to mobile wallets were identified based on the OWASP Mobile Top 10. Source code static analysis, network traffic inspection, and automatic vulnerability analysis tools were used to analyze cryptocurrency wallets and traditional financial applications. However, since the study was conducted to identify general OWASP mobile vulnerabilities, it did not consider the inherent threats to cryptocurrency wallets that arise from the nature of the blockchain.

Er-Rajy, L., et al. [5] introduced the different types of threats to Bitcoin wallets. For example, threats and actual cases of private key stealing attacks using malicious code such as worms or Trojans were described. The paper also introduced threats that involve interfering with the connection of the wallet application to the blockchain network, such as denial of service attacks.

In addition to analyzing the security of cryptocurrency wallets, researchers have also conducted studies on methods of improving the security of such wallets. [14] and [15] proposed methods using ARM's TrustZone technology. G. Miraje et al. [14] improved the security of the BitSafe wallet, an open source wallet, by using it as a base wallet and applying TrustZone technology to it. They implemented a BitSafe wallet by porting the Trusted OS on the actual ARM development board. They moved the key storage space of the BitSafe wallet to TEE to ensure that the key would remain encrypted at all times. However, when executing wallet functions using TEE, reading and writing data took a few to several thousand times longer than in a general environment.

W. Dai et al. [15] proposed a method of implementing a Simplified Payment Verification (SPV) wallet based on TrustZone. In the paper, SPV was operated in the Secure Execution Environment (SEE) to securely verify transactions recorded on the blockchain. Also, block headers were encrypted in SEE so that they could not be read in the Normal Execution Environment (NEE). Thus, even if Rich OS were to become infected with malicious code, SPV could safely be executed. W. Dai et al. implemented SBLWT on an actual Raspberry Pi development board. In this method, all functions requiring security, such as key generation, signature generation, and account address generation, are operated within the SEE of TrustZone, and private keys are stored in the secure storage. In addition, implementing a secure display and secure touchscreen driver ensures that the user's input and output can safely be processed within the SEE.

Y. Liu et al. [16] proposed a key generation method to ensure that keys could be used safely in cryptocurrency wallets. In this method, private keys are generated by SHA256 hashing together a random seed and a user's secret phrase string. The method improves the security of key management, but the same function can be implemented by using a passphrase in BIP39 [17]. Furthermore, in the method discussed in [16], if the seed is lost, the private key cannot be recovered even if the user knows the secret phrase. Therefore, it is less convenient than the method of storing only one master seed in BIP32 [18] because all seeds must be backed up in case they are lost.

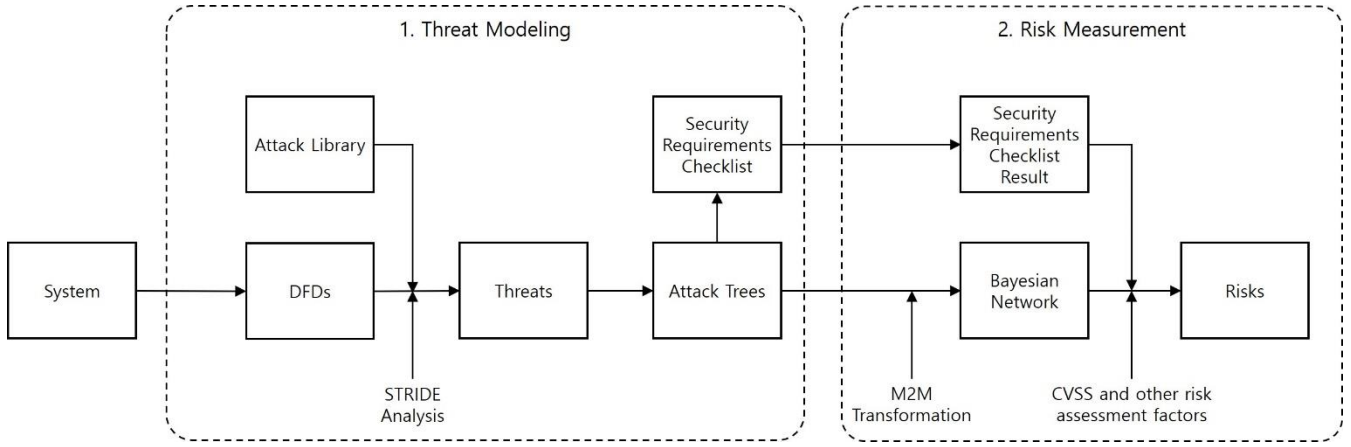


Fig. 1. Overview of the risk assessment process.

### III. RISK ASSESSMENT METHODOLOGY

#### A. Threat Modeling

Methods for assessing risk on cryptocurrency wallet systems can largely be divided into threat modeling and risk measurement phases. Fig. 1 represents the overall risk assessment process. In the first phase (the threat modeling phase), we present the cryptocurrency wallet system as a data flow diagram and then identify the inherent threats in the system through STRIDE analysis. Then, we create attack trees and use them to create a checklist of security requirements. Second, in the risk measurement phase, we transform the attack trees into a Bayesian network and measure the risk to each wallet through CVSS and various risk measurement factors based on the results of the security requirement checklist analysis.

We use threat modeling to identify inherent threats to cryptocurrency wallet systems. In threat modeling, we systematically model a target system to identify security threats. We use STRIDE (Microsoft), as it is the most well-established among various available threat modeling methodologies. STRIDE identifies possible threats to each component of the system from the attacker's perspective based on six types of threats: Spoofing, Tampering, Repudiation, Information Disclosure, and Elevation of Privilege. This enables systematic identification of the threats present in the target system. Other threat modeling methodologies such as Trike and PASTA could be used in place of STRIDE. Trike classifies actors in various roles as well as assets that exist in the system, and identifies threats by creating a matrix of intended actions that each actor can use to access various assets. This methodology is suitable when there are users in various roles, and various assets to be protected such as within a server, so the situation is complicated. However, a personal cryptocurrency wallet is designed to be used by only one user, and most of the assets to be protected are cryptocurrencies or private keys, which are relatively straightforward, so the Trike methodology is not suitable here. Moreover, while Trike identifies all threats that involve Denial of Service and Privilege Escalation attacks, STRIDE is able to more specifically classify threats from six perspectives. PASTA is a risk-centric threat modeling methodology composed of seven steps. Similar to STRIDE, it breaks down the components of a system and creates a DFD to identify threats. However, PASTA identifies threats by deriving an attack scenario based

on known attack vectors or previous knowledge and expertise. Since the attack types are not systematically classified, like they are in STRIDE, the results may differ greatly between analysts. Therefore, we use STRIDE for systematic threat modeling of cryptocurrency wallets.

Use of STRIDE requires that we build and model data flow diagrams (DFDs) by segmenting the system in question into process units. By segmenting the system into process units capable of performing independent functions, we can specifically analyze the threats that are present in each component.

After deriving the inherent threats in the system, we build attack trees to identify the attack goals that could be achieved using those threats. This allows us to derive security requirements to eliminate or mitigate the threats in each attack tree.

TABLE II. ELEMENTS OF A DATA FLOW DIAGRAM.

Element	Symbol	Description
External Entity		Any entity(people or system) outside of the application
Data Store		A location where data are stored
Process		A task that handles data within the application
Data Flow		A path that data take between external entities, processes and data stores
Trust Boundary		The change of trust levels as data flow through the system

#### 1) Creating data flow diagrams

We create Data Flow Diagrams (DFD) to analyze the structure of a cryptocurrency wallet. Table II shows the elements that are required to create a DFD. An external entity represents a person or system outside of the target system. In general, it represents a user who uses an application or an external server. A data store represents a space in which data is stored in the application. For example, a data store could be the hard disk of

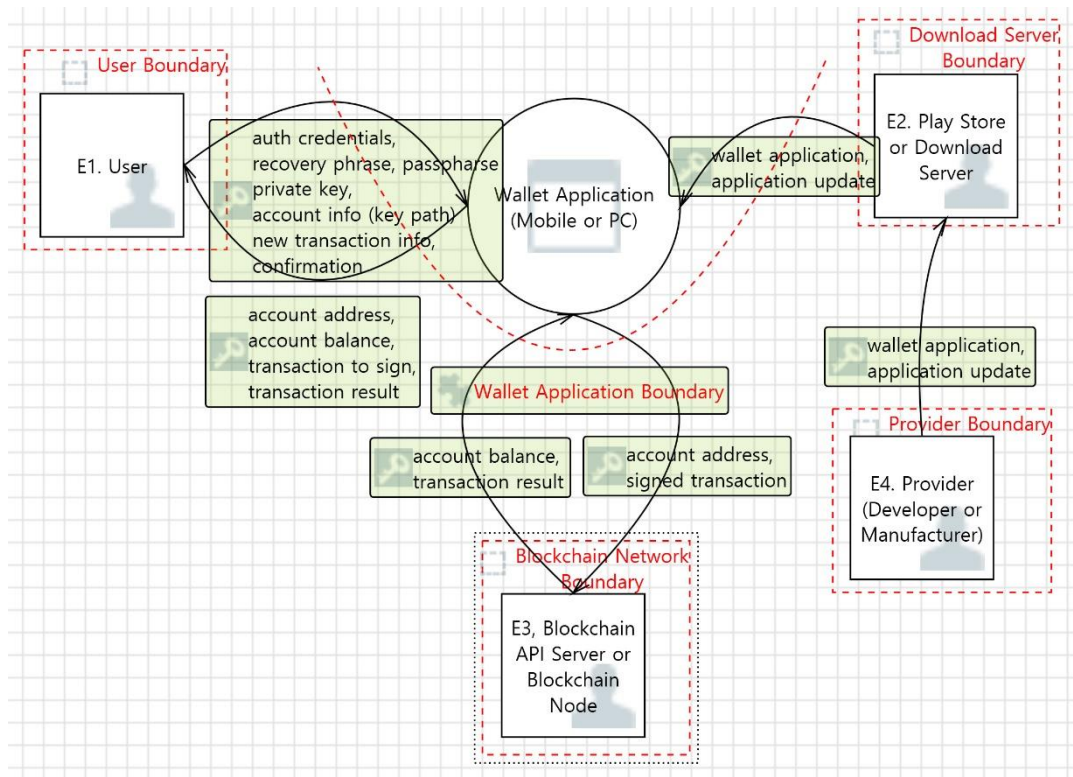


Fig. 2. Level 0 data flow diagram of a hot wallet.

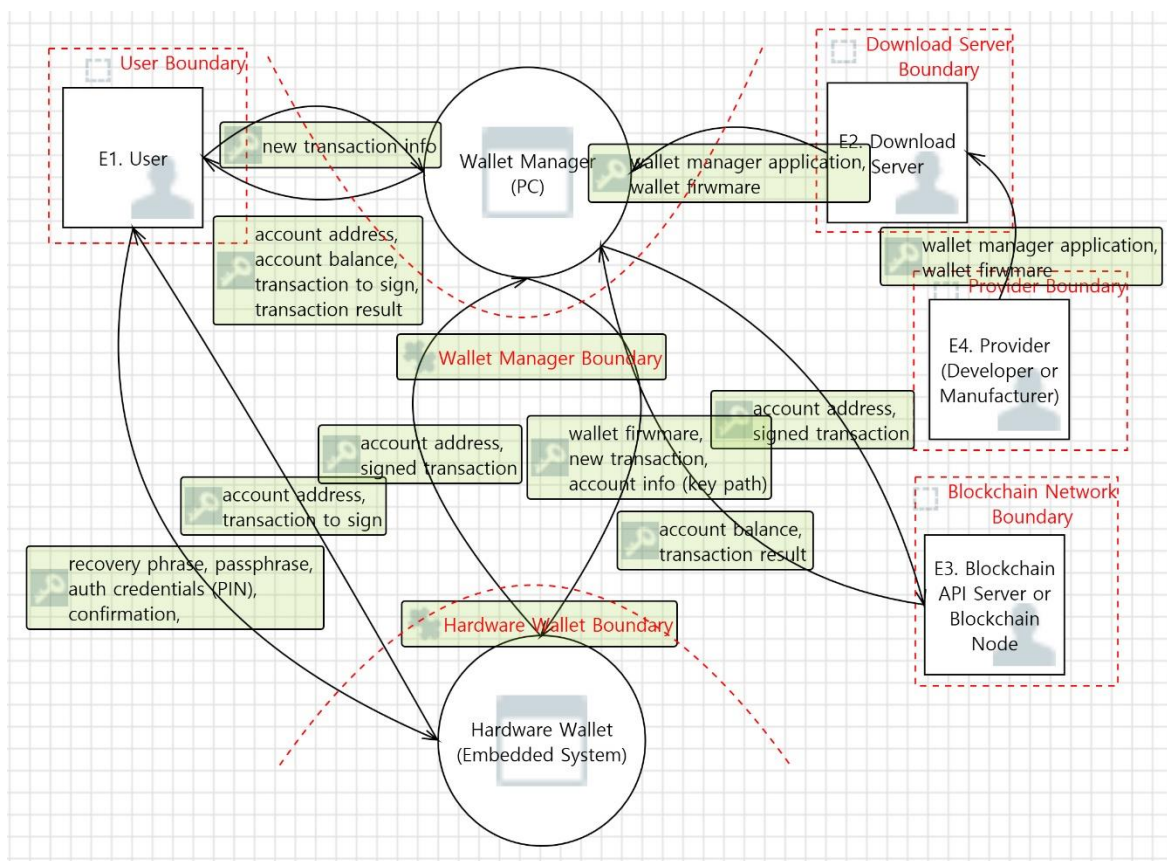


Fig. 3. Level 0 data flow diagram of a cold wallet.



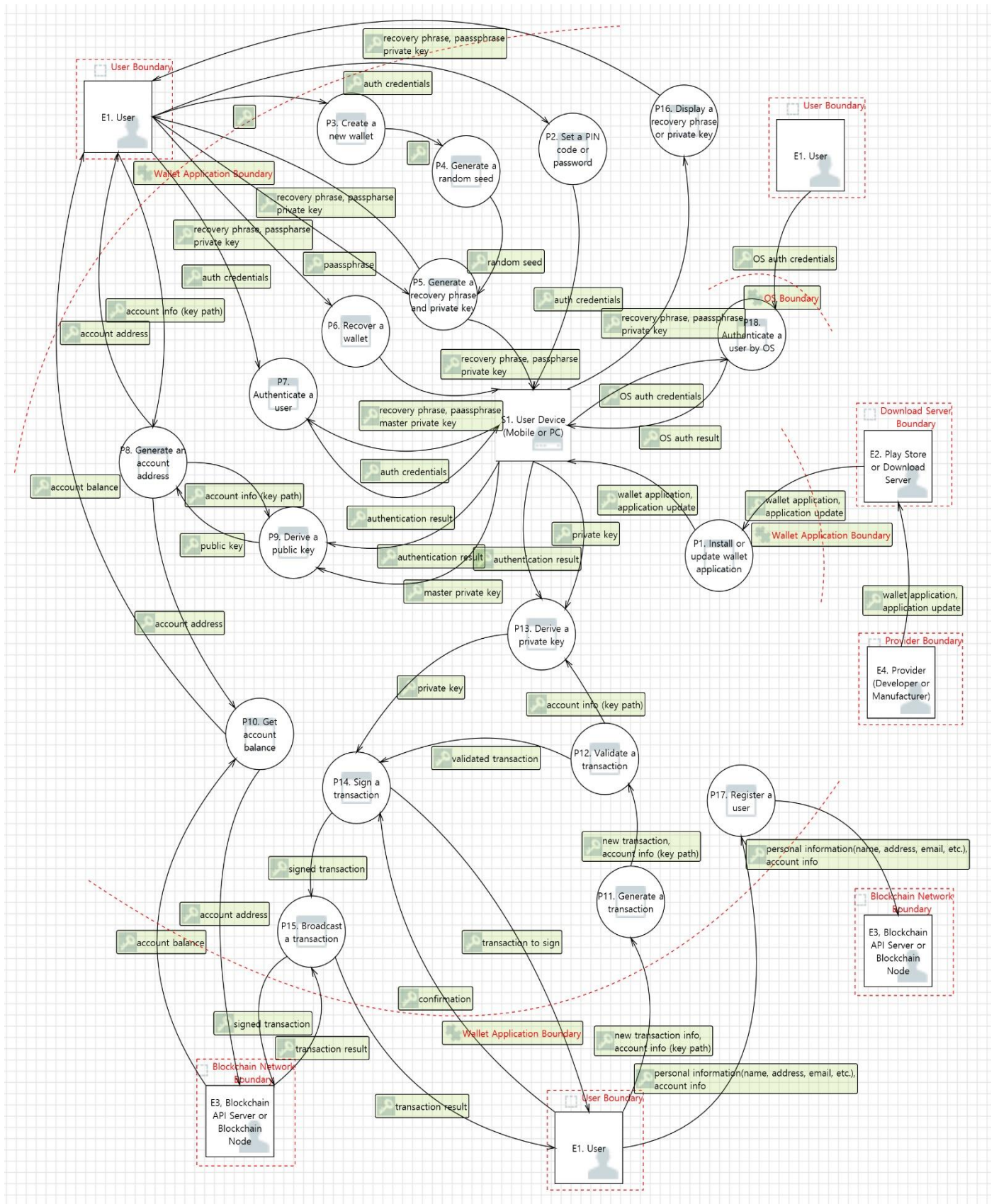


Fig. 4. Level 2 data flow diagram of a hot wallet.

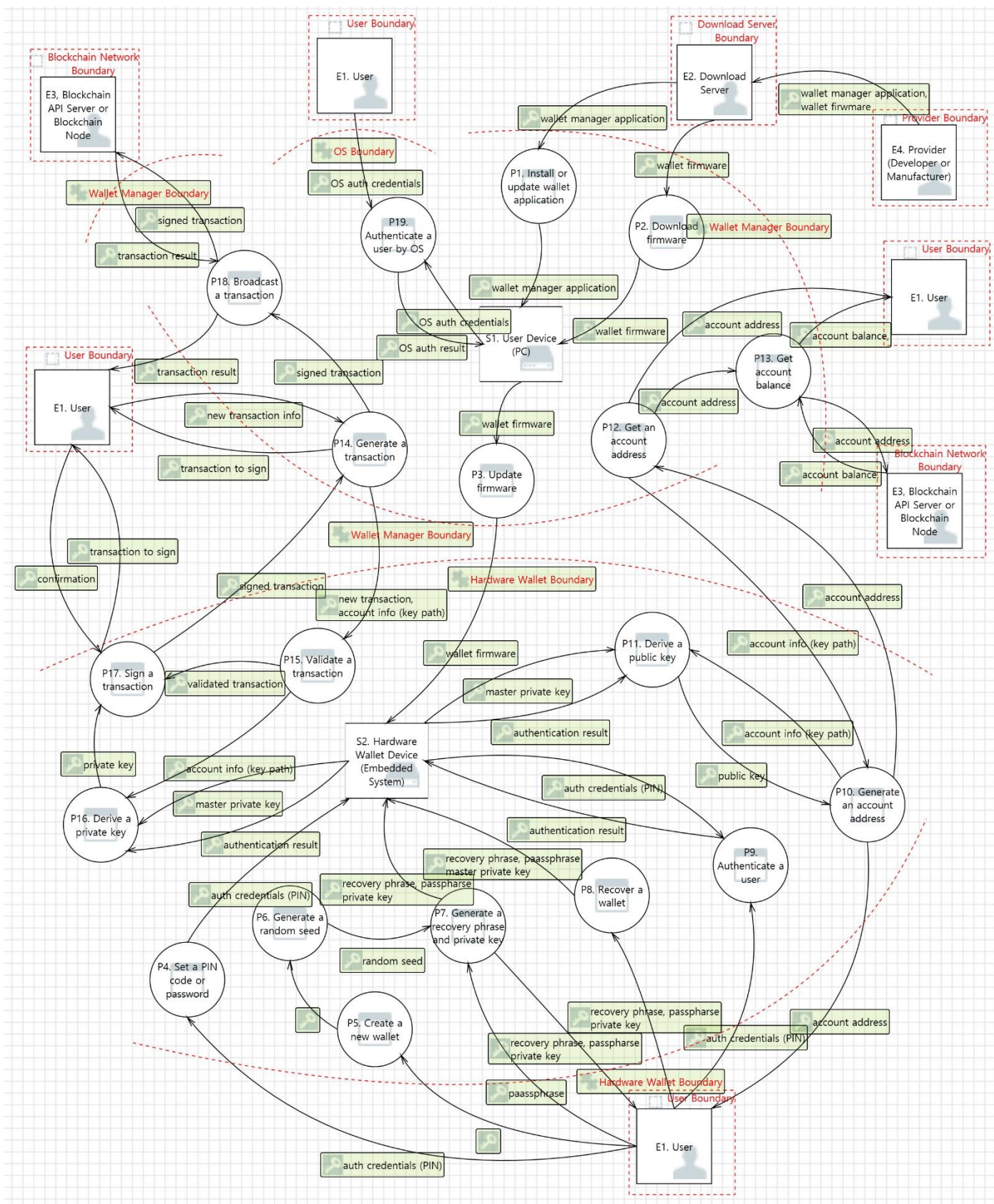


Fig. 5. Level 2 data flow diagram of a cold wallet.



a PC or the flash memory of a smartphone.

A process represents the task of processing data in the application. It refers to a process or execution code that processes input data and generates output data. A data flow represents the flow of data exchanged between external entities, data stores, and processes. It involves both input data and output data. Finally, a trust boundary represents the boundary at which the level of trust between elements changes. Data flows across trust boundaries are considered essentially untrustworthy. For example, when a system communicates with an external server through a network, the data that is exchanged is not trusted, as represented by a trust boundary in the diagram.

A data flow diagram is written differently depending on the degree of abstraction. Level 0 represents the target system in its most abstract form. The higher the level, the more specifically the system is modeled. At level 2, which is usually the highest level, the system is expressed in a very specific form, with each process functioning independently. We created DFDs using the Threat Modeling Tool provided by Microsoft, as shown in Figs. 2, 3, 4, and 5.

Fig. 2 shows a level 0 DFD of a hot wallet. At the heart of the DFD is the cryptocurrency wallet application. A user downloads the wallet from the download server and installs it on their own device. The user then creates and manages a cryptocurrency account by sending a command to the wallet. At this time, the wallet application, which is connected to the blockchain API server or blockchain node, downloads the user's cryptocurrency asset information and transmits a new transaction created by the user to the blockchain network.

Fig. 3 shows a level 0 DFD of a cold wallet. The most notable difference from the diagram of a hot wallet is that the wallet manager application in the middle is connected to a separate hardware wallet. The wallet manager is installed on the user's device; it receives the user's commands and sends those commands to the hardware wallet. The wallet manager manages the wallet and displays asset details in a convenient manner using the input/output interfaces of the host. At this time, the wallet manager is connected to the blockchain API server or blockchain node. It downloads the user's cryptocurrency asset information and transmits a new transaction created by the user to the blockchain network. The hardware wallet securely generates and stores cryptocurrency keys. When a signature request for a new transaction is delivered, the user's approval is confirmed through a button on the hardware wallet, and a signature is created and delivered to the wallet manager.

Figs. 4 and 5 show the more detailed level 2 DFDs of hot and cold wallets, respectively. Using these DFDs, it is possible to identify threats that exist in the wallet. The diagrams show, in detail, where important data such as user authentication information, recovery phrases, and private keys can flow. For example, if an attacker can intercept the recovery phrase data output during the initial wallet creation process through a passive attack, the attacker can then use the key derivation function to steal the user's private keys for all cryptocurrency assets.

## 2) STRIDE analysis

STRIDE is a threat modeling methodology that classifies possible threats to each component into six types: Spoofing (S), Tampering (T), Repudiation (R), Information Disclosure (I), Denial of Service (D), and Elevation of Privilege (E).

The STRIDE analysis procedure is as follows. First, we create an attack library by collecting known attack vectors, threats or vulnerabilities that can threaten a cryptocurrency wallet system. Table III shows the attack library that was created by referring to journals, conferences, books, CVE, CAPEC and so on. Second, based on the attack library in Table III, we identify possible threats for the 6 types of STRIDE targeting all components of the DFDs mentioned in Chapter A-1. For example, for E1.User in Fig. 4, an attacker can disguise himself as a user and use a wallet. Therefore, the H1 threat in Table IV arises from the possibility of a spoofing attack in which the attacker uses the Bypassing Authentication attack vector in the attack library. Importantly, the completeness of the threat modeling procedure is guaranteed and traceability is ensured because the number of the referenced attack library is recorded. Tables IV and V show the results of STRIDE analysis of each component created based on the level 2 DFD of a hot and cold wallet, respectively. As a result of the analysis, a total of 112 threats were identified for a hot wallet, and 103 threats were identified for a cold wallet.

### 3) Attack trees

The attack tree method was first proposed by B. Schneier [19]. It is a threat modeling method that can systematically derive various attack paths for a target system. We create attack scenarios to achieve each goal of the attack trees based on the threats identified using the STRIDE analysis in Chapter A-2. In an attack tree, the root node represents the final goal, and the sub-goals (intermediate goals) are represented as child nodes. The last leaf node through several branch nodes in the middle represents an independent threat. The advantage of an attack tree is that it can logically and systematically derive various specific attack vectors that are necessary to achieve an attack goal. We divided the final attack goals into three main categories: stealing cryptocurrency, denial of service, and privacy breach.

#### a) Stealing cryptocurrency

G1 in Table VI shows the "stealing cryptocurrency" attack tree. The first sub-goal (S1) is to obtain a user's private key. If the wallet is a hierarchical deterministic wallet [17], stealing a recovery phrase is equivalent to stealing a private key. To achieve this sub-goal, an attack may monitor the input/output data to/from a wallet by installing malware on a user device, sniff data copied to the clipboard, or read a private key stored in the device. Most of these attacks can be performed by installing malware on the user's device. If an attacker can physically access the user's device, they can bypass user authentication or steal a private key through physical attacks, or they can directly read a recovery phrase or private key through a shoulder-surfing attack. In addition, it is possible to alter the wallet application to generate a private key known to the attacker in advance through a supply chain attack, or, if a weak random number generator is used in the wallet, discover the private key using the nonce value of the signature [18]. It is also possible to use a side-channel attack to find a private key [19]. However, a side-channel attack requires an attacker to physically access the user's wallet and



TABLE III. ATTACK LIBRARY FOR A CRYPTOCURRENCY WALLET.

Category	Attack	Num	Title	Author	Type
Bypassing Authentication	Brute-force attack on a PIN or password	1	Brute-force and dictionary attack on hashed real-world passwords	L. Bošnjak et al.	Conference
			Fast dictionary attacks on passwords using time-space tradeoff	Narayanan et al.	Conference
	Buffer overflow (code reuse)	2	Jump-oriented programming: a new class of code-reuse attack	Bletsch, Tyler, et al.	Conference
			DisARM: mitigating buffer overflow attacks on embedded devices	Habibi, Javid, et al.	Conference
	Evil maid attack	3	Android data storage security: A review	Altuwaijri et al.	Journal
	Fake biometrics	4	Image Quality Assessment for Fake Biometric Detection: Application to Iris, Fingerprint, and Face Recognition.	J. Galbally et al.	Journal
	Physical access	5	CAPEC-507: Physical Theft	MITRE	CAPEC
	Shoulder-surfing attack	6	Understanding Shoulder Surfing in the Wild: Stories from Users and Observers	Eiband et al.	Conference
			CAPEC-508: Shoulder Surfing	MITRE	CAPEC
	SQL injection	7	Prevention of SQL Injection attack using query transformation and hashing	D. Kar and S. Panigrahi	Conference
Cryptanalysis	Brute-force attack on a private or secret key	8	The Security of DSA and ECDSA	Vaudenay, Serge	Conference
			DFA Mechanism on the AES Key Schedule	J. Takahashi et al.	Conference
	ECDSA weak signature	9	Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies	Breitner et al.	Conference
	ECDSA nonce reuse	10	Identifying Key Leakage of Bitcoin Users	Brengel M and Rossow C	Book
DoS attack	ICMP flooding (ping of death)	11	Denial of Service via Algorithmic Complexity Attacks	Crosby et al.	Conference
			CAPEC-487: ICMP Flood	MITRE	CAPEC
	TCP SYN flooding	12	TCP SYN flooding attacks and common mitigation (RFC 4987)	Eddy, Wesley	RFC
			CAPEC-482: TCP Flood	MITRE	CAPEC
Malware	Botnet	13	Your botnet is my botnet: analysis of a botnet takeover	Stone-Gross, Brett, et al.	Conference
	Clipboard hijacker	14	Attacks on android clipboard	Zhang, Xiao, and Wenliang Du	Conference
			Simple Clipboard Malware Attack Detection and Analysis from the User-Machine Interaction View	Wieczoreka, Michal	Conference
			CAPEC-637: Collect Data from Clipboard	MITRE	CAPEC
	Keylogger (keyboard, mouse, screen touch input logger)	15	A framework for detection and prevention of novel keylogger spyware attacks	M. Wazid et al.	Conference
			TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion	Cai, Liang, and Hao Chen	Conference
			CAPEC-568: Capture Credentials via Keylogger	MITRE	CAPEC
	Network packet sniffer	16	CAPEC-158: Sniffing Network Traffic	MITRE	CAPEC
	Ransomware	17	Automated Behavioral Analysis of Malware: A Case Study of WannaCry Ransomware	Q. Chen and R. A. Bridges	Conference
	Screen recorder, screen capture	18	Detection and Elimination of Spyware and Ransomware by Intercepting Kernel-Level System Routines	D. Javaheri et al.	Journal
			CAPEC-648: Collect Data from Screen Capture	MITRE	CAPEC
	Trojan	19	Malware Behavior Analysis: Learning and Understanding Current Malware Threats	M. F. Zolkipli and A. Jantan	Conference
	USB packet sniffer	20	Security analysis and improvement of USB technology	D. Noyes et al.	Conference
Malware Installation	Malicious document files (MS Word, Excel, PDF)	21	Malicious PDF detection using metadata and structural features	Smutz et al.	Conference
			The evolution of malware	Touchette, Fred	Technical report

Malware Installation	Malvertising	22	Malvertising exploiting web advertising	Sood, Aditya K., and Richard J. Enbody	Journal
	Phishing (Email, SMS, social media)	23	A Taxonomy of Attacks and a Survey of Defence Mechanisms for Semantic Social Engineering Attacks	Ryan Heartfield and George Loukas	Journal
	Removable media infection (USB)	24	USB_SEC: A secure application to manage removable media	S. T. Reddy et al.	Conference
			BeatCoin: Leaking Private Keys from Air-Gapped Cryptocurrency Wallets	M. Guri	Conference
	Rogue AP	25	Detecting and eliminating Rogue Access Points	V. S. S. Sriram et al.	Conference
	Supply chain attack	26	wallet.fail	Dmitry Nedospasov et al.	Conference
			Breaking the Ledger Security Model	Saleem Rashd	Public
			CVE-2018-1000851	MITRE	CVE
	Web-based infection (drive-by download)	27	The Ghost in the Browser: Analysis of Web-based Malware	Provos, N et al.	Journal
Man-in-the-middle Attack	ARP spoofing	28	Detecting ARP spoofing: An active technique	Ramachandran et al.	Conference
	DNS spoofing	29	DNS Protection against Spoofing and Poisoning Attacks	M. A. Hussain et al.	Conference
	IP address spoofing	30	Proposed methods of IP spoofing detection & prevention	Rashid et al.	Journal
Physical Attack	Cold boot attack	31	Android data storage security: A review	Altuwaijri et al.	Journal
			On the Practicability of Cold Boot Attacks	M. Gruhn and T. Müller	Conference
	Connect a debugger (JTAG, SWD)	32	wallet.fail	Dmitry Nedospasov et al.	Conference
	Fault injection (glitching)	33	wallet.fail	Dmitry Nedospasov et al.	Conference
			Implementing practical electrical glitching attacks	Giller, Brett	Conference
	Microscopy	34	Reverse engineering flash EEPROM memories using scanning electron microscopy	Courbon et al.	Conference
	Probing	35	A layout-driven framework to assess vulnerability of ICs to microprobing attacks	Q. Shi, N et al.	Conference
	Side-channel attack	36	Extracting the Private Key from a TREZOR	Jochen Hoenicke	Public
			CVE-2019-14354	MITRE	CVE
			CVE-2019-14353	MITRE	CVE
Privilege Escalation	Android root toolkit	37	Android rooting: Methods, detection, and evasion	Sun, San-Tsai et al.	Conference
			Rooting attack detection method on the Android-based smart phone.	Won-Jun Jang et al.	Conference
	Buffer overflow (code injection)	38	Code injection attacks on harvard-architecture devices	Francillon et al.	Conference
			A framework for on-device privilege escalation exploit execution on Android	Höbarth et al.	Conference
			CVE-2018-4918	MITRE	CVE
	Row Hammer attack	39	Android data storage security: A review	Altuwaijri et al.	Journal
			The Rowhammer Attack Injection Methodology	K. S. Yim	Conference

perform as many cryptographic operations as necessary. This is not a real threat because the attacker would already have to know the wallet's PIN code to accomplish this task. Thus, side-channel attacks are not considered in this paper.

The second sub-goal (S2) is to send cryptocurrencies from the user's wallet device to an attacker. To do this, the attacker

may trick the user into approving the wrong transaction, or physically access the device and bypass user authentication. The simplest way to deceive the user is to replace the destination address with the attacker's address when the user copies the other party's account address to the clipboard to create a transaction. Since most cryptocurrency account address formats are difficult for users to read, such an address change attack is a

TABLE IV. STRIDE ANALYSIS RESULT OF A HOT WALLET.

Component	Name	STRIDE	Threat	Description	Attack Library
Entity	E1. User	S	H1	Impersonate a user by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		S	H2	Impersonate a user by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
		R	H3	Repudiate attacks by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		R	H4	Repudiate attacks by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
	E2. Download Server	S	H5	Impersonate a provider by bypassing authentication.	1, 2, 3, 4, 5, 6, 7, 33
		R	H6	Repudiate attacks by bypassing authentication.	1, 2, 3, 4, 5, 6, 7, 33
		D	H7	Execute DDoS attacks (botnets, flooding attacks).	11, 12, 13
		D	H8	Execute DoS attacks by installing ransomware on the download server.	17, 21, 22, 23, 24, 25, 26, 27
		D	H9	Execute DoS attacks using SQL injection.	7
	E3. Blockchain Node or API Server	S	H10	Impersonate a normal blockchain node using MITM attacks.	28, 29, 30
		R	H11	Repudiate attacks using MITM attacks.	28, 29, 30
		D	H12	Prevent the wallet from accessing the blockchain node using MITM attacks.	28, 29, 30
		D	H13	Execute DDoS attacks (botnets, flooding attacks).	11, 12, 13
		D	H14	Execute DoS attacks by installing ransomware on the blockchain node or API server.	17, 21, 22, 23, 24, 25, 26, 27
		D	H15	Execute DoS attacks using SQL injection.	7
Data Store	S1. User Device (Mobile or PC)	T	H16	Modify auth credentials, a recovery phrase, passphrase or private key using a clipboard hijacker.	14, 21, 22, 23, 24, 25, 26, 27
		T	H17	Modify a wallet application, auth credentials, a recovery phrase, passphrase, private key, or transaction information by getting root or admin privilege.	37, 38, 39
		T	H18	Install malware using social engineering (malicious files, malvertising, phishing, drive-by download attack).	21, 22, 24, 27
		T	H19	Install malware using a Rogue AP.	25
		T	H20	Install malware using supply chain attack.	26
		T	H21	Install malware using a removable media (USB drive).	24
		I	H22	Obtain auth credentials, a recovery phrase, passphrase or private key using physical attacks.	31, 33, 34, 35, 36
		I	H23	Obtain auth credentials, recovery phrase, passphrase, private key or transaction information by getting root or admin privilege.	37, 38, 39
		I	H24	Find a recovery phrase, passphrase or private key using a brute-force attack.	8
		D	H25	Delete the wallet application or key files by getting root or admin privilege.	37, 38, 39
		D	H26	Delete the wallet application or key files using factory reset or disk formatting by accessing the wallet physically.	5
		D	H27	Encrypt the wallet application or key files by installing ransomware.	17, 21, 22, 23, 24, 25, 26, 27
Process	P1. Install or update the wallet application	S	H28	Impersonate the download server using MITM attacks.	28, 29, 30
		T	H29	Install a modified wallet application by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
		T	H30	Install a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	H31	Install a modified wallet application using MITM attacks.	28, 29, 30
		D	H32	Prevent a user from installing a wallet application by executing DoS attacks on the download server.	7, 11, 12, 13, 17
		D	H33	Prevent a user from installing a wallet application using MITM attacks.	28, 29, 30
	P2. Set a PIN code or password	S	H34	Set a PIN code or password by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
		I	H35	Obtain a PIN code or password using screen recording malware.	18, 19
		I	H36	Obtain a PIN code or password using keylogger malware.	15, 19
		I	H37	Obtain a PIN code or password using shoulder-surfing attack.	6
		I	H38	Obtain a PIN code or password by getting root or admin privilege.	37, 38, 39
	P3. Create a new wallet	S	H39	Create a new wallet by bypassing OS authentication.	1, 2, 3, 4, 5, 6
	P4. Generate a random seed	I	H40	Obtain a known random seed by installing a modified wallet application using social engineering and supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	H41	Find a random seed using a brute-force attack (with a weak random number generator).	8

Component	Name	STRIDE	Threat	Description	Attack Library
Process	P5. Generate a recovery phrase and private key	I	H42	Obtain a known recovery phrase or private key by installing a modified wallet application using social engineering and supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	H43	Obtain a recovery phrase, passphrase or private key by installing a screen recorder.	18
		I	H44	Obtain a recovery phrase, passphrase or private key using shoulder-surfing attack.	6
		I	H45	Obtain a recovery phrase, passphrase or private key by installing a clipboard hijacker.	14
		I	H46	Obtain a passphrase by installing a keylogger.	15
	P6. Recover a wallet	S	H47	Recover a new wallet by bypassing OS authentication.	1, 2, 3, 4, 5, 6
		T	H48	Modify a recovery phrase, passphrase or private key by installing a clipboard hijacker.	14
		I	H49	Obtain a recovery phrase, passphrase, or private key by installing a screen recorder.	18
		I	H50	Obtain a recovery phrase, passphrase, or private key by installing a clipboard hijacker.	14
		I	H51	Obtain a recovery phrase, passphrase, or private key by installing a keylogger.	15
		I	H52	Obtain a recovery phrase, passphrase, or private key using shoulder-surfing attack.	5
	P7. Authenticate a user	S	H53	Bypass wallet user authentication using a brute-force attack (guessing, dictionary attack).	1
		S	H54	Bypass wallet user authentication using a buffer overflow (code reuse) attack.	2
		S	H55	Bypass wallet user authentication using evil maid attack.	3
		S	H56	Bypass wallet user authentication using fake biometrics.	4
		S	H57	Bypass wallet user authentication by accessing the wallet when it is unlocked.	5
		S	H58	Bypass wallet user authentication using shoulder-surfing attack.	6
		S	H59	Bypass wallet user authentication using physical attacks (e.g., fault injection(glitching)).	33
		I	H60	Obtain auth credentials by installing a keylogger.	15, 19
		I	H61	Obtain auth credentials by installing a screen recorder.	18, 19
		D	H62	Try the wrong PIN or password consecutively until the wallet is locked.	5
	P8. Generate an account address	S	H63	Generate an account address by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		T	H64	Generate a fake address by modifying the wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	H65	Replace an address with a fake address by installing a clipboard modifier.	14
		I	H66	Obtain an account address by installing a screen recorder.	18
		I	H67	Obtain an account address by installing a clipboard hijacker.	14
	P9. Derive a public key	I	H68	Generate a known public key by installing a modified wallet using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
	P10. Get account balance	T	H69	Modify an account address or account balance using MITM attacks.	28, 29, 30
		T	H70	Modify account balance by installing malware on the blockchain node or API server.	21, 22, 23, 24, 25, 26, 27
		I	H71	Obtain an account address or account balance using MITM attacks.	28, 29, 30
		I	H72	Obtain an account balance by installing a screen recorder.	18
		I	H73	Obtain an account address or account balance by installing a packet sniffer.	16
		D	H74	Prevent the wallet fetching account balance using MITM attacks.	28, 29, 30
		D	H75	Prevent the wallet fetching account balance by executing DoS attacks on the blockchain node or API server.	11, 12, 13, 17
	P11. Generate a transaction	S	H76	Generate a transaction by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		T	H77	Modify the destination address by installing a clipboard modifier.	14
		T	H78	Modify transaction information by installing a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	H79	Observe transaction information by installing a screen recorder.	18
		I	H80	Observe transaction information by installing a clipboard hijacker.	14
		I	H81	Observe transaction information by installing a keylogger.	15
	P12. Validate a transaction	T	H82	Modify transaction information by installing a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
	P13. Derive a private key	I	H83	Derive a known private key by installing a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27



Component	Name	STRIDE	Threat	Description	Attack Library
Process	P14. Sign a transaction	S	H84	Sign a transaction by bypassing user confirmation by accessing the wallet application.	5
		T	H85	Modify a transaction by installing a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	H86	Obtain transaction information using shoulder-surfing attack.	6
		I	H87	Compute a private key using an ECDSA nonce exploitation.	9, 10
	P15. Broadcast a transaction	S	H88	Impersonate a normal blockchain node or API server using MITM attacks.	28, 29, 30
		T	H89	Modify a transaction using MITM attacks.	28, 29, 30
		I	H90	Obtain transaction information using MITM attacks.	28, 29, 30
		I	H91	Obtain transaction information by installing a screen recorder.	18
		D	H92	Prevent the wallet from broadcasting a transaction using MITM attacks.	28, 29, 30
		D	H93	Prevent the wallet from broadcasting a transaction by executing DoS attacks on the blockchain node or API server.	11, 12, 13, 17
	P16. Display a recovery phrase or private key	S	H94	Obtain a recovery phrase or private key by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		T	H95	Modify a recovery phrase or private key using a clipboard modifier.	14
		T	H96	Modify a recovery phrase or private key by installing a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	H97	Obtain a recovery phrase or private key using a screen recorder.	18
		I	H98	Obtain a recovery phrase or private key using a clipboard hijacker.	14
	P17. Register a user	S	H99	Impersonate a provider using MITM attacks.	28, 29, 30
		T	H100	Modify personal information using MITM attacks.	28, 29, 30
		I	H101	Obtain personal information using MITM attacks.	28, 29, 30
		I	H102	Obtain personal information using a packet sniffer.	16
		I	H103	Obtain personal information using screen recorder malware.	18
		I	H104	Obtain personal information using a clipboard hijacker.	14
		I	H105	Obtain personal information using a keylogger.	15
	P18. Authenticate a user by OS	S	H106	Bypass OS authentication using a brute-force attack (guessing, dictionary attack).	1
		S	H107	Bypass OS authentication using a buffer overflow (code reuse) attack.	2
		S	H108	Bypass OS authentication using evil maid attack.	3
		S	H109	Bypass OS authentication using fake biometrics.	4
		S	H110	Bypass OS authentication by accessing the wallet when it is unlocked.	5
		S	H111	Bypass OS authentication using shoulder-surfing attack.	6
		S	H112	Bypass OS authentication using physical attacks (e.g., fault injection(glitching)).	33

very efficient method. Alternatively, a supply chain attack that causes the wallet to create a tampered transaction is also possible. Also, if the attacker can bypass user authentication when physically accessing the wallet, the attacker can send cryptocurrency to himself using the wallet.

The third sub-goal (S3) is to intercept cryptocurrency that is sent to the user. The easiest way of doing this is to replace the user's address on the clipboard with the attacker's address, similar to the tactic used in S2. This is a very simple and efficient method; the attacker simply replaces the address stored on the clipboard with the attacker's own account address when the user transmits his account address in an attempt to receive cryptocurrency from a third party. Another method is to show the attacker's account address as if it were the user's account address. To do so, the attacker may tamper with the wallet application through a supply chain attack.

#### *b) Denial of service*

G2 in Table VI shows the attack tree of the denial of service attack goal. The first sub-goal (S4) is to prevent access to the

user's private key. To do so, the attacker may launch an attack that deletes the private key. The stored key can be deleted by installing malware on the user's device or by acquiring root or administrator privileges. Alternatively, if physical access is possible, the key can be deleted by bypassing user authentication or by formatting a disk or factory reset. In addition, some attacks deliberately fail user authentication in succession and induce the wallet to delete keys itself, or encrypt keys using ransomware.

The second sub-goal (S5) is to prevent the user from accessing the wallet application. It is possible to delete or encrypt the application. This is similar to the attacks in S4.

The third sub-goal (S6) is to prevent the wallet from accessing the blockchain network. To do so, it is possible to disrupt the connection to the blockchain network through a man-in-the-middle attack. Alternatively, if the wallet exchanges blockchain data through a specific blockchain API server, the attacker may interfere with the connection by performing a denial of service attack on the server.

TABLE V. STRIDE ANALYSIS RESULT OF A COLD WALLET.

Component	Name	STRIDE	Threat	Description	Attack Library
Entity	E1. User	S	C1	Impersonate a user by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		S	C2	Impersonate a user by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
		R	C3	Repudiate attacks by bypassing wallet user authentication.	1, 2, 3, 4, 5, 6, 33
		R	C4	Repudiate attacks by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
	E2. Download Server	S	C5	Impersonate a provider by bypassing authentication.	1, 2, 3, 4, 5, 6, 7, 33
		R	C6	Repudiate attacks by bypassing authentication.	1, 2, 3, 4, 5, 6, 7, 33
		D	C7	Execute DDoS attacks (botnets, flooding attacks).	11, 12, 13
		D	C8	Execute DoS attacks by installing ransomware on the download server.	17, 21, 22, 23, 24, 25, 26, 27
		D	C9	Execute DoS attacks using SQL injection.	7
	E3. Blockchain Node or API Server	S	C10	Impersonate a normal blockchain node using MITM attacks.	28, 29, 30
		R	C11	Repudiate attacks using MITM attacks.	28, 29, 30
		D	C12	Prevent the wallet from accessing the blockchain node using MITM attacks.	28, 29, 30
		D	C13	Execute DDoS attacks (botnets, flooding attacks).	11, 12, 13
		D	C14	Execute DoS attacks by installing ransomware on the blockchain node or API server.	17, 21, 22, 23, 24, 25, 26, 27
		D	C15	Execute DoS attacks using SQL injection.	7
Data Store	S1. User Device (PC)	T	C16	Modify the wallet manager by getting root or admin privilege.	37, 38, 39
		T	C17	Install malware using social engineering (malicious files, malvertising, phishing, drive-by download attack).	21, 22, 24, 27
		T	C18	Install malware using a rogue AP.	25
		T	C19	Install malware using supply chain attack.	26
		T	C20	Install malware using a removable media (USB drive).	24
		D	C21	Delete the wallet manager by getting root or admin privilege.	37, 38, 39
		D	C22	Encrypt the wallet manager by installing ransomware.	17, 21, 22, 23, 24, 25, 26, 27
		D	C23	Delete the wallet manager using disk formatting by accessing the wallet physically.	5
	S2. Hardware Wallet	T	C24	Modify firmware, authentication credentials, a recovery phrase, passphrase or private key by connecting a debugger (JTAG, SWD)	32
		I	C25	Obtain authentication credentials, a recovery phrase, passphrase or private key using physical attacks (e.g., fault injection, probing, microscoping or cold boot attack).	31, 33, 34, 35, 36
		I	C26	Obtain authentication credentials, a recovery phrase, passphrase or private key by connecting a debugger (JTAG, SWD)	32
		I	C27	Find a recovery phrase, passphrase or private key using a brute-force attack.	8
		D	C28	Delete the wallet application or key files using factory reset by accessing the wallet physically.	5
		D	C29	Delete the wallet application or key files by connecting a debugger (JTAG, SWD)	32
Process	P1. Install or update the wallet application	T	C30	Install a modified wallet application by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
		T	C31	Install a modified wallet application using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	C32	Install a modified wallet application using MITM attacks.	28, 29, 30
		D	C33	Prevent a user from installing a wallet application by executing DoS attacks on the download server.	7, 11, 12, 13, 17
		D	C34	Prevent a user from installing a wallet application using MITM attacks.	28, 29, 30
	P2. Download firmware	S	C35	Impersonate the download server using MITM attacks.	28, 29, 30
		T	C36	Download a modified firmware by bypassing OS authentication.	1, 2, 3, 4, 5, 6, 33
		T	C37	Download a modified firmware using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	C38	Download a modified firmware application using MITM attacks.	28, 29, 30
	P2. Download firmware	D	C39	Prevent a user from downloading firmware by executing DoS attacks on the download server.	7, 11, 12, 13, 17
		D	C40	Prevent a user from downloading firmware using MITM attacks.	28, 29, 30
	P3. Update firmware	T	C41	Provide a modified firmware by installing a modified wallet application using social engineering and supply chain attack.	21, 22, 23, 24, 25, 26, 27
	P4. Set a PIN code or password	S	C42	Set a PIN code or password by accessing the wallet.	5
	P5. Create a new wallet	S	C43	Create a new wallet by accessing the hardware wallet.	5

Component	Name	STRIDE	Threat	Description	Attack Library
Process	P6. Generate a random seed	I	C44	Obtain a known random seed by installing a modified firmware using social engineering and supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	C45	Find a random seed using a brute-force attack (with a weak random number generator).	8
	P7. Generate a recovery phrase and private key	I	C46	Obtain a known recovery phrase or private key by installing a modified firmware using social engineering and supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	C47	Obtain a recovery phrase, passphrase or private key using shoulder-surfing attack.	6
	P8. Recover a wallet	S	C48	Recover a new wallet by accessing the hardware wallet.	5
		I	C49	Obtain a recovery phrase, passphrase, or private key using shoulder-surfing attack.	5
	P9. Authenticate a user	S	C50	Bypass wallet user authentication using a brute-force attack (guessing, dictionary attack).	1
		S	C51	Bypass wallet user authentication using a buffer overflow (code reuse) attack.	2
		S	C52	Bypass wallet user authentication using evil maid attack.	3
		S	C53	Bypass wallet user authentication by accessing the wallet when it is unlocked.	5
		S	C54	Bypass wallet user authentication using shoulder-surfing attack.	6
		S	C55	Bypass wallet user authentication using physical attacks (e.g., fault injection (glitching)).	33
		D	C56	Try the wrong PIN or password consecutively until the wallet is locked.	5
	P10. Generate an account address	S	C57	Generate an account address by installing a modified wallet manager.	21, 22, 23, 24, 25, 26, 27
		T	C58	Generate a fake address by modifying firmware using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	C59	Generate a known address by modifying firmware using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
	P11. Derive a public key	I	C60	Generate a known public key by installing a modified firmware using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
	P12. Get an account address	S	C61	Get an account address from a fake device using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	C62	Get a fake address by modifying the wallet manager using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	C63	Replace an address with a fake address by installing a clipboard modifier.	14
		I	C64	Obtain an account address by installing a screen recorder.	18
		I	C65	Obtain an account address by installing a clipboard hijacker.	14
		I	C66	Obtain an account address by installing a USB packet sniffer.	20
	P13. Get account balance	T	C67	Modify an account address or account balance using MITM attacks.	28, 29, 30
		T	C68	Modify account balance by installing malware on the blockchain node or API server.	21, 22, 23, 24, 25, 26, 27
		I	C69	Obtain an account address or account balance using MITM attacks.	28, 29, 30
		I	C70	Obtain account balance by installing a screen recorder.	18, 19
		I	C71	Obtain an account address or account balance by installing a packet sniffer.	16, 19
		D	C72	Prevent the wallet fetching account balance using MITM attacks.	28, 29, 30
		D	C73	Prevent the wallet fetching account balance by executing DoS attacks on the blockchain node.	11, 12, 13, 17
	P14. Generate a transaction	S	C74	Generate a transaction by bypassing wallet manager user authentication.	1, 2, 3, 4, 5, 6, 33
		S	C75	Impersonate a genuine hardware wallet using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	C76	Modify the destination address by installing a clipboard modifier.	14
		T	C77	Modify transaction information by installing a modified wallet manager using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	C78	Observe transaction information by installing a screen recorder.	18
		I	C79	Observe transaction information by installing a clipboard hijacker.	14
		I	C80	Observe transaction information or a signed transaction by installing a USB packet sniffer.	20
		I	C81	Observe transaction information by installing a keylogger.	15
	P15. Validate a transaction	S	C82	Impersonate a genuine wallet manager by installing a modified wallet manager using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		T	C83	Modify transaction information by installing a modified wallet manager using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	C84	Observe transaction information or a signed transaction by installing a USB packet sniffer.	20

Component	Name	STRIDE	Threat	Description	Attack Library
	P16. Derive a private key	I	C85	Derive a known private key by installing a modified firmware using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
	P17. Sign a transaction	S	C86	Sign a transaction by bypassing user confirmation by accessing the wallet application.	5
		T	C87	Modify a transaction by installing a modified firmware using social engineering or supply chain attack.	21, 22, 23, 24, 25, 26, 27
		I	C88	Obtain a private key using side channel attacks.	36
		I	C89	Compute a private key using an ECDSA nonce exploitation.	9, 10
		I	C90	Obtain transaction information using shoulder-surfing attack.	6
	P18. Broadcast a transaction	S	C91	Impersonate a normal blockchain node or API server using MITM attacks.	28, 29, 30
		T	C92	Modify a transaction using MITM attacks.	28, 29, 30
		I	C93	Obtain transaction information using MITM attacks.	28, 29, 30
		I	C94	Obtain transaction information by installing a screen recorder.	18
		D	C95	Prevent the wallet from broadcasting a transaction using MITM attacks.	28, 29, 30
		D	C96	Prevent the wallet from broadcasting a transaction by executing DoS attacks on the blockchain node or API server.	11, 12, 13, 17
	P19. Authenticate a user by OS	S	C97	Bypass OS authentication using a brute-force attack (guessing, dictionary attack).	1
		S	C98	Bypass OS authentication using a buffer overflow (code reuse) attack.	2
		S	C99	Bypass OS authentication using evil maid attack.	3
		S	C100	Bypass OS authentication using fake biometrics.	4
		S	C101	Bypass OS authentication by accessing the wallet when it is unlocked.	5
		S	C102	Bypass OS authentication using shoulder-surfing attack.	6
		S	C103	Bypass OS authentication using physical attacks (e.g., fault injection (glitching)).	33

### c) Privacy breach

G3 in Table VI shows the attack tree of the privacy breach attack. The first sub-goal (S7) is to obtain information related to the user's cryptocurrency account. To do so, the attacker can install malware to monitor the input/output data to/from the wallet or sniff network packets. Due to the nature of most blockchains, only the user's account information must be found to find not only the account balance but also data on transactions with other accounts.

The second sub-goal (S8) is to steal the user's identifiable personal information. Attacks similar to those in S7 can be used. However, in general, decentralized cryptocurrency wallets do not require or store user's personal information, so leakage of personal information is not a severe threat.

### B. Risk Measurement Methodology

We measure the risks to wallets by converting the attack trees into a Bayesian network and utilizing CVSS Exploitability metrics and various risk measurement factors.

First, we refer to the risk management or risk assessment standards for various cybersecurity threats created by standard organizations or national organizations from around the world in order to develop a methodology for measuring the risks to wallets. A total of 12 standards that are commonly mentioned in papers related to risk management or risk assessment frameworks were referenced: OCTAVE Allegro, NIST RMF, ISO/IEC 27005, CVSS, SP 800-30, FAIR, CRAMM, EBIOS, ISRAM, CORAS, COBIT 5, and MEHARI. Among them, high-level standards such as NIST RMF (SP 800-37) and ISO/IEC 27005 present structured procedures and methods for continuous risk management throughout the life cycle of the

target system. However, the specific low-level method that is used to measure risk is largely determined by the analyst's expertise. On the other hand, SP800-30, CVSS, FAIR, EBIOS, etc. provide certain factors for measuring risks. For example, CVSS uses the Exploitability metric to calculate a severity score that indicates the possibility of exploitation of a vulnerability and the Impact metric to represent the consequences of a successful attack. FAIR quantitatively measures the overall risk to a wallet by calculating the Loss Event Frequency factor, which indicates the probability of occurrence of a threat, and the Loss Magnitude factor, which represents the amount of damage caused by that threat. In EBIOS, the Likelihood factor, which indicates the likelihood of an attack, is calculated by considering the attacker's Motivation, Resources, and Activity, and then the risk is calculated as a matrix along with the Severity factor (representing the severity of the attack). However, all of these methods have a drawback in that they do not reflect the structural context in which an attack is achieved by combining individual threats when measuring risk. Even in two situations where the same threat exists, the calculated risk should differ depending on the structure of the target system and whether the threat can be combined with any other threats. To this end, CVSS uses the Scope metric to calculate scores differently when the threat also affects other components, but this method is inaccurate because it does not actually consider the structural context in which the components exist and how they are connected. In the previous chapter, we created attack trees to reveal how different threats are connected. Therefore, there is a need for a risk measurement methodology that reflects the combined structure of each threat using the attack tree as an input.



TABLE VI. ATTACK TREES FOR A CRYPTOCURRENCY WALLET.

Description					STRIDE Analysis	Node		
Stealing Cryptocurrency						G1		
1	Obtain a private key					S1		
OR	1.1	Eavesdrop input data				B1		
	OR	1.1.1	Keylogger malware			B2		
		AND	1.1.1.1	Install a malware (keylogger, screen touch input logger)			B3	
			OR	1.1.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)		H18	T1
				1.1.1.1.2	Rogue AP		H19	T2
				1.1.1.1.3	Supply chain attack		H20	T3
				1.1.1.1.4	Removable media (USB drive)		H21	T4
			1.1.1.2	Execute keylogging attack		H51	T5	
	1.2	Eavesdrop output data				B4		
	OR	1.2.1	Screen capture malware			B5		
		AND	1.2.1.1	Install a malware (screen recorder)			B6	
			OR	1.2.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)		H18	T6
				1.2.1.1.2	Rogue AP		H19	T7
				1.2.1.1.3	Supply chain attack		H20	T8
				1.2.1.1.4	Removable media (USB drive)		H21	T9
			1.2.1.2	Execute screen capture attack		H43, H49, H97	T10	
	1.3	Observe output data directly on the screen				B7		
	OR	1.3.1	Shoulder-surfing attack (smartphone, surveillance camera)			H44, H52, C47, C49	T11	
		1.3.2	Open the wallet and obtain secret data				B8	
		AND	1.3.2.1	Bypass OS authentication			B9	
			OR	1.3.2.2.1	Brute-force attack (guessing, dictionary attack)		H2, H4, H106	T12
				1.3.2.2.2	Buffer overflow (code reuse)		H2, H4, H107	T13
				1.3.2.2.3	Evil maid attack		H2, H4, H108	T14
				1.3.2.2.4	Fake biometrics		H2, H4, H109	T15
				1.3.2.2.5	Physical access when the host is open		H2, H4, H110	T16
				1.3.2.2.6	Shoulder-surfing attack		H2, H4, H111	T17
				1.3.2.2.7	Physical attack (fault injection(glitching))		H2, H4, H112	T18
			1.3.2.2	Bypass wallet user authentication			B10	
			OR	1.3.2.2.1	Brute-force attack (guessing, dictionary attack)		H1, H3, H53, H94	T19
				1.3.2.2.2	Buffer overflow (code reuse)		H1, H3, H54, H94	T20
				1.3.2.2.3	Evil maid attack		H1, H3, H55, H94	T21
				1.3.2.2.4	Fake biometrics		H1, H3, H56, H94	T22
				1.3.2.2.5	Physical access when the wallet is open		H1, H3, H57, H94	T23
				1.3.2.2.6	Shoulder-surfing attack		H1, H3, H37, H58, H94	T24
		1.3.2.2.7		Physical attack (fault injection(glitching))		H1, H3, H59, H94	T25	
		1.3.2.2.8	Obtain authentication credentials using a malware (keylogger, screen recorder, trojan)		H35, H36, H37, H60, H61, H94	T26		
		1.3.2.3	Obtain a passphrase				B11	
		OR	1.3.2.3.1	Shoulder-surfing attack			H44, H52	T27
			1.3.2.3.2	Brute-force attack			H24, C27	T28
			1.3.2.3.3	Physical attack (fault injection, probing, microscoping, cold boot attack)			H22, C25	T29

		AND	1.3.2.3.4	Connect a debugger (JTAG, SWD)		C26	T30		
			1.3.2.3.5	Install a malware				B12	
			OR	1.3.2.3.5.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)			H18	T31
				1.3.2.3.5.2	Rogue AP			H19	T32
				1.3.2.3.5.3	Supply chain attack			H20	T33
				1.3.2.3.5.4	Removable media (USB drive)			H21	T34
			1.3.2.3.6	Execute malware attack				B13	
			OR	1.3.2.3.6.1	Screen recorder malware			H43, H49	T35
				1.3.2.3.6.2	Clipboard hijacker			H45, H50	T36
				1.3.2.3.6.3	Keylogger malware			H46, H51	T37
1.4	Obtain a private key or recovery phrase at rest						B14		
AND	1.4.1	Obtain data at rest (Flash, HDD, SDD)					B15		
	OR	1.4.1.1	Connect a debugger (JTAG, SWD)			C26	T38		
		1.4.1.2	Physical attack (fault injection, probing, microscoping, cold boot attack)			H22, C25	T39		
		1.4.1.3	Get root or admin privilege				B16		
		OR	1.4.1.3.1	Android root toolkit			H23	T40	
			1.4.1.3.2	Buffer overflow (code injection)			H23	T41	
			1.4.1.3.3	Row Hammer attack			H23	T42	
	1.4.2	Decrypt a private key or recovery phrase				T43			
1.5	Obtain data in transit (RAM)					B17			
OR	1.5.1	Cold boot attack			H22, C25	T44			
	1.5.2	Clipboard hijacker malware					B18		
	AND	1.5.2.1	Install a malware (clipboard hijacker)				B19		
		OR	1.5.2.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)			H18	T45	
			1.5.2.1.2	Rogue AP			H19	T46	
			1.5.2.1.3	Supply chain attack			H20	T47	
			1.5.2.1.4	Removable media (USB drive)			H21	T48	
	1.4.2.2	Execute clipboard hijacking attack			H45, H50, H98	T49			
	1.5.3	Gain root or admin privilege				B20			
	OR	1.5.3.1	Android root toolkit			H23	T50		
		1.5.3.2	Buffer overflow (code injection)			H23	T51		
		1.5.3.3	Row Hammer attack			H23	T52		
1.6	Make the wallet use a known private key					B21			
OR	1.6.1	Wallet application modification					B22		
	AND	1.6.1.1	Application reverse engineering			H29, H30, H31, H83	T53		
		1.6.1.2	Install a modified application				B23		
		OR	1.6.1.2.1	Social engineering (phishing)			H30, H40, H42, H96	T54	
			1.6.1.2.2	Supply chain attack			H5, H6, H30, H40, H42, H96	T55	
			1.6.1.2.3	MITM attack				B24	
			OR	1.6.1.2.3.1	ARP spoofing			H28, H31	T56
		1.6.1.2.3.2		DNS spoofing and poisoning			H28, H31	T57	
		1.6.1.2.3.3		IP address spoofing			H28, H31	T58	
		1.6.1.2.4	Bypass OS authentication				B25		
		OR	1.6.1.2.4.1	Brute-force attack (guessing, dictionary attack)			H29, H106	T59	
	1.6.1.2.4.2		Buffer overflow (code reuse)			H29, H107	T60		

					1.6.1.2.4.3	Evil maid attack	H29, H108	T61					
					1.6.1.2.4.4	Fake biometrics	H29, H109	T62					
					1.6.1.2.4.5	Physical access when the host is open	H29, H110	T63					
					1.6.1.2.4.6	Shoulder-surfing attack	H29, H111	T64					
					1.6.1.2.4.7	Physical attack (fault injection(glitching))	H29, H112	T65					
					OR	1.6.1.3	Gain root or admin privilege					B26	
						OR	1.6.1.3.1	Android root toolkit			H17	T66	
							1.6.1.3.2	Buffer overflow (code injection)			H17	T67	
							1.6.1.3.3	Row Hammer attack			H17	T68	
					1.6.2	Firmware modification						B27	
					AND	1.6.2.1	Firmware reverse engineering				C36, C37, C38, C41, C85	T69	
						1.6.2.2	Install a modified firmware					B28	
						OR	1.6.2.2.1	Social engineering (phishing)			C37, C44, C46	T70	
							1.6.2.2.2	Supply chain attack			C37, C44, C46	T71	
							1.6.2.2.3	MITM attack				B29	
							OR	1.6.2.2.3.1	ARP spoofing		C35, C38	T72	
								1.6.2.2.3.2	DNS spoofing and poisoning		C35, C38	T73	
								1.6.2.2.3.3	IP address spoofing		C35, C38	T74	
							1.6.2.2.4	Bypass OS authentication			C36	B30	
							OR	1.6.2.2.4.1	Brute-force attack (guessing, dictionary attack)			C36, C97	T75
								1.6.2.2.4.2	Buffer overflow (code reuse)			C36, C98	T76
								1.6.2.2.4.3	Evil maid attack			C36, C99	T77
								1.6.2.2.4.4	Fake biometrics			C36, C100	T78
								1.6.2.2.4.5	Physical access when the host is open			C36, C101	T79
								1.6.2.2.4.6	Shoulder-surfing attack			C36, C102	T80
								1.6.2.2.4.7	Physical attack (fault injection(glitching))			C36, C103	T81
						OR	1.6.2.3	Connect a debugger (JTAG, SWD)				C24	T82
						1.6.3	Create or recover a wallet using a known private key						B31
						OR	1.6.3.1	Replace a recovery phrase or private key with an adversary's using a clipboard malware					B32
							AND	1.6.3.1.1	Install a malware (clipboard hijacker)				B33
					OR			1.6.3.1.1.1	Social engineering		H18	T83	
								1.6.3.1.1.2	Rogue AP		H19	T84	
								1.6.3.1.1.3	Supply chain attack		H20	T85	
								1.6.3.1.1.4	Removable media (USB drive)		H21	T86	
					1.6.3.1.2		Execute clipboard data modification attack				H16, H48, H95	T87	
					1.6.3.2		Access the target device and create or recover a wallet					B34	
					AND		1.6.3.2.1	Access the target device physically			H39, H47, H110, C43, C48	T88	
							1.6.3.2.2	Bypass OS authentication				B35	
							OR	1.6.3.2.2.1	Brute-force attack (guessing, dictionary attack)		H2, H4, H106	T89	
								1.6.3.2.2.2	Buffer overflow (code reuse)		H2, H4, H107	T90	
								1.6.3.2.2.3	Evil maid attack		H2, H4, H108	T91	
								1.6.3.2.2.4	Fake biometrics		H2, H4, H109	T92	
								1.6.3.2.2.5	Physical access when the host is open		H2, H4, H110	T93	
								1.6.3.2.2.6	Shoulder-surfing attack		H2, H4, H111	T94	
								1.6.3.2.2.7	Physical attack (fault injection(glitching))		H2, H4, H112	T95	

OR	1.7	Find a private key using a computational method			B36		
	OR	1.7.1	Brute-force attack on a private key		H24, H41, C27, C45	T96	
		1.7.2	Calculate a private key from signatures			B37	
		OR	ECDSA weak signature		H87, C89	T97	
			ECDSA nonce reuse		H87, C89	T98	
2	Make the wallet send cryptocurrency to an adversary					S2	
OR	2.1	Manipulate the recipient address or amount of the transaction				B38	
	AND	2.1.1	Modify the clipboard data			B39	
		OR	2.1.1.1	Install a malware (clipboard data modifier)			B40
			OR	2.1.1.1.1	Social engineering	H18, C17	T99
				2.1.1.1.2	Rogue AP	H19, C18	T100
				2.1.1.1.3	Supply chain attack	H20, C19	T101
				2.1.1.1.4	Removable media (USB drive)	H21, C20	T102
		2.1.1.2	Clipboard data modification attack		H65, H77, C63, C76	T103	
		2.1.1.3	Bypass user confirmation			B41	
		OR	2.1.1.3.1	Access the target wallet and bypass user confirmation		H84, C86	T104
		2.1.2	Modify wallet application (wallet manager) and manipulate transaction data			B42	
		AND	2.1.2.1	Application reverse engineering		H78, H82, H85, C77, C83	T105
			2.1.2.2	Install the modified application			B43
			OR	2.1.2.2.1	Social engineering (phishing)		H30, C31
	2.1.2.2.2			Supply chain attack		H30, C5, C6, C31	T107
	2.1.2.2.3			MITM attack		H31, C32	B44
	OR		2.1.2.2.3.1	ARP spoofing		H31	T108
			2.1.2.2.3.2	DNS spoofing and poisoning		H31	T109
			2.1.2.2.3.3	IP address spoofing		H31	T110
	2.1.2.2.4		Bypass OS authentication		H29, C30	B45	
	OR		2.1.2.2.4.1	Brute-force attack (guessing, dictionary attack)		H106, C97	T111
			2.1.2.2.4.2	Buffer overflow (code reuse)		H107, C98	T112
			2.1.2.2.4.3	Evil maid attack		H108, C99	T113
			2.1.2.2.4.4	Fake biometrics		H109, C100	T114
			2.1.2.2.4.5	Physical access when the host is open		H110, C101	T115
			2.1.2.2.4.6	Shoulder-surfing attack		H111, C102	T116
			2.1.2.2.4.7	Physical attack (fault injection(glitching))		H112, C103	T117
	2.1.2.3		Bypass user confirmation			B46	
	OR	2.1.2.3.1	Access the target wallet and bypass user confirmation		C86	T118	
	2.1.3	Wallet firmware modification				B47	
	AND	2.1.3.1	Firmware reverse engineering		C41, C87	T119	
		2.1.3.2	Install the modified firmware			B48	
		OR	2.1.3.2.1	Social engineering (phishing)		C37	T120
			2.1.3.2.2	Supply chain attack		C37	T121
			2.1.3.2.3	MITM attack		C38	B49
		OR	2.1.3.2.3.1	ARP spoofing		C38	T122
			2.1.3.2.3.2	DNS spoofing and poisoning		C38	T123
			2.1.3.2.3.3	IP address spoofing		C38	T124



AND			2.1.3.2.4	Bypass OS authentication		C36	B50	
			OR	2.1.3.2.4.1	Brute-force attack (guessing, dictionary attack)	C97	T125	
				2.1.3.2.4.2	Buffer overflow (code reuse)	C98	T126	
				2.1.3.2.4.3	Evil maid attack	C99	T127	
				2.1.3.2.4.4	Fake biometrics	C100	T128	
				2.1.3.2.4.5	Physical access when the host is open	C101	T129	
				2.1.3.2.4.6	Shoulder-surfing attack	C102	T130	
				2.1.3.2.4.7	Physical attack (fault injection(glitching))	C103	T131	
		2.1.4	Get root or admin privilege					B51
		OR	2.1.4.1	Android root toolkit			H17, C16	T132
			2.1.4.2	Buffer overflow (code injection)			H17, C16	T133
			2.1.4.3	Row Hammer attack			H17, C16	T134
	2.2	Send cryptocurrency using the target wallet directly					B52	
		2.2.1	Bypass OS authentication					B53
		OR	2.2.1.1	Brute-force attack (guessing, dictionary attack)			H106	T135
			2.2.1.2	Buffer overflow (code reuse)			H107	T136
			2.2.1.3	Evil maid attack			H108	T137
			2.2.1.4	Fake biometrics			H109	T138
			2.2.1.5	Physical access when the host is open			H110	T139
			2.2.1.6	Shoulder-surfing attack			H111	T140
			2.2.1.7	Physical attack (fault injection(glitching))			H112	T141
		2.2.2	Bypass wallet user authentication					B54
		OR	2.2.2.1	Brute-force attack (guessing, dictionary attack)			H53, H76, C1, C3, C50, C74	T142
			2.2.2.2	Buffer overflow (code reuse)			H54, H76, C1, C3, C51, C74	T143
			2.2.2.3	Evil maid attack			H55, H76, C1, C3, C52, C74	T144
			2.2.2.4	Fake biometrics			H56, H76	T145
			2.2.2.5	Physical access when the wallet is open			H57, H76, C1, C3, C53, C74	T146
			2.2.2.6	Shoulder-surfing attack			H37, H58, H76, C1, C3, C54, C74	T147
			2.2.2.7	Physical attack (fault injection(glitching))			H59, H76, C1, C3, C55, C74	T148
			2.2.2.8	Obtain authentication credentials using a malware (keylogger, screen recorder)			H35, H36, H37, H60, H61, H76	T149
		2.3.3	Obtain a passphrase					B55
		OR	2.3.3.1	Shoulder-surfing attack			H44, H52	T150
			2.3.3.2	Brute-force attack			H24, C27	T151
			2.3.3.3	Physical attack (fault injection, probing, microscoping, cold boot attack)			H22, C25	T152
			2.3.3.4	Connect a debugger (JTAG, SWD)			C26	T153
3	Intercept cryptocurrency					S3		
OR	3.1	Show an adversary's address as a user's address					B56	
	OR	3.1.1	Get root or admin privilege					B57
		OR	3.1.1.1	Android root toolkit			H17, C16	T154
			3.1.1.2	Buffer overflow (code injection)			H17, C16	T155
			3.1.1.3	Row Hammer attack			H17, C16	T156

		3.1.2	Wallet application (or wallet manager) modification			B58			
		AND	3.1.2.1	Application reverse engineering			H64, H68, C62	T157	
			3.1.2.2	Install a modified application				B59	
			OR	3.1.2.2.1	Social engineering (phishing)		H30, C31	T158	
				3.1.2.2.2	Supply chain attack		H30, C5, C6, C31	T159	
				3.1.2.2.3	MITM attack				B60
				OR	3.1.2.2.3.1	ARP spoofing		H31, C32	T160
					3.1.2.2.3.2	DNS spoofing and poisoning		H31, C32	T161
					3.1.2.2.3.3	IP address spoofing		H31, C32	T162
				3.1.2.2.4	Bypass OS authentication			H29, C30	B61
				OR	3.1.2.2.4.1	Brute-force attack (guessing, dictionary attack)		H106, C2, C4, C97	T163
					3.1.2.2.4.2	Buffer overflow (code reuse)		H107, C2, C4, C98	T164
					3.1.2.2.4.3	Evil maid attack		H108, C2, C4, C99	T165
					3.1.2.2.4.4	Fake biometrics		H109, C2, C4, C100	T166
					3.1.2.2.4.5	Physical access when the host is open		H110, C2, C4, C101	T167
					3.1.2.2.4.6	Shoulder-surfing attack		H111, C2, C4, C102	T168
					3.1.2.2.4.7	Physical attack (fault injection(glitching))		H112, C2, C4, C103	T169
		3.1.3	Wallet firmware modification				B62		
		AND	3.1.3.1	Firmware reverse engineering			C58, C60	T170	
			3.1.3.2	Install the modified firmware				B63	
			OR	3.1.3.2.1	Social engineering (phishing)		C37, C59, C60, C61	T171	
				3.1.3.2.2	Supply chain attack		C37, C59, C60, C61	T172	
				3.1.3.2.3	MITM attack				B64
				OR	3.1.3.2.3.1	ARP spoofing		C38	T173
					3.1.3.2.3.2	DNS spoofing and poisoning		C38	T174
					3.1.3.2.3.3	IP address spoofing		C38	T175
				3.1.3.2.4	Bypass OS authentication				B65
				OR	3.1.3.2.4.1	Brute-force attack (guessing, dictionary attack)		C36, C97	T176
					3.1.3.2.4.2	Buffer overflow (code reuse)		C36, C98	T177
					3.1.3.2.4.3	Evil maid attack		C36, C99	T178
					3.1.3.2.4.4	Fake biometrics		C36, C100	T179
					3.1.3.2.4.5	Physical access when the host is open		C36, C101	T180
					3.1.3.2.4.6	Shoulder-surfing attack		C36, C102	T181
					3.1.3.2.4.7	Physical attack (fault injection(glitching))		C36, C103	T182
		3.1.4	Replace a user’s address with an adversary’s using a clipboard hijacker				B66		
		AND	3.1.4.1	Install a malware				B67	
			OR	3.1.4.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)		H18	T183	
				3.1.4.1.2	Rogue AP		H19	T184	
				3.1.4.1.3	Supply chain attack		H20	T185	
				3.1.4.1.4	Removable media (USB drive)		H21	T186	
			3.1.4.2	Execute clipboard hijacking attack			H65	T187	
Denial of Service						G2			
4	Prevent a user from using a private key					S4			
OR	4.1	Delete a private key				B68			
	OR	4.1.1	Open the wallet and delete a private key			B69			

		AND	1.3.2.1	Bypass OS authentication			B70	
			OR	1.3.2.2.1	Brute-force attack (guessing, dictionary attack)		H106	T188
				1.3.2.2.2	Buffer overflow (code reuse)		H107	T189
				1.3.2.2.3	Evil maid attack		H108	T190
				1.3.2.2.4	Fake biometrics		H109	T191
				1.3.2.2.5	Physical access when the host is open		H110	T192
				1.3.2.2.6	Shoulder-surfing attack		H111	T193
				1.3.2.2.7	Physical attack (fault injection(glitching))		H112	T194
		1.3.2.2	Bypass wallet user authentication				B71	
		OR	1.3.2.2.1	Brute-force attack (guessing, dictionary attack)		H53, C50	T195	
			1.3.2.2.2	Buffer overflow (code reuse)		H54, C51	T196	
			1.3.2.2.3	Evil maid attack		H55, C52	T197	
			1.3.2.2.4	Fake biometrics		H56	T198	
			1.3.2.2.5	Physical access when the wallet is open		H57, C53	T199	
			1.3.2.2.6	Shoulder-surfing attack		H58, C54	T200	
			1.3.2.2.7	Physical attack (fault injection(glitching))		H59, C55	T201	
		1.3.2.2.8	Obtain authentication credentials using a malware (keylogger, screen recorder)		H35, H36, H37, H60, H61	T202		
		4.1.2	Delete files at rest (HDD, SSD, Flash)				B72	
		OR	4.1.2.1	Delete key files using factory reset or disk formatting		H26, C28	T203	
			4.1.2.2	Get root or admin privilege				B73
			OR	4.1.2.2.1	Android root toolkit		H25	T204
				4.1.2.2.2	Buffer overflow (code injection)		H25	T205
				4.1.2.2.3	Row Hammer attack		H25	T206
			4.1.2.3	Connect a debugger (JTAG, SWD)		C29	T207	
	4.2	Encrypt a private key				B74		
	OR	4.2.1	Ransomware attack				B75	
		AND	4.2.1.1	Install a malware (ransomware)			B76	
			OR	4.2.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)		H18	T208
				4.2.1.1.2	Rogue AP		H19	T209
				4.2.1.1.3	Supply chain attack		H20	T210
				4.2.1.1.4	Removable media (USB drive)		H21	T211
		4.2.1.2	Execute ransomware attack		H27	T212		
4.3	Lock the wallet				B77			
AND	4.3.1	Bypass OS authentication				B78		
	OR	4.3.1.1	Brute-force attack (guessing, dictionary attack)		H106	T213		
		4.3.1.2	Buffer overflow (code reuse)		H107	T214		
		4.3.1.3	Evil maid attack		H108	T215		
		4.3.1.4	Fake biometrics		H109	T216		
		4.3.1.5	Physical access when the host is open		H110	T217		
		4.3.1.6	Shoulder-surfing attack		H111	T218		
		4.3.1.7	Physical attack (fault injection(glitching))		H112	T219		
	4.3.2	Keep trying the wrong PIN or password until the wallet is locked		H62, C56	T220			
5	Prevent a user from using the wallet				S5			
	5.1	Delete the wallet application (or wallet manager)			B79			

OR	OR	5.1.1	Bypass OS authentication and uninstall the wallet application (or wallet manager)			B80		
		OR	5.1.1.1	Brute-force attack (guessing, dictionary attack)		H106, C97	T221	
			5.1.1.2	Buffer overflow (code reuse)		H107, C98	T222	
			5.1.1.3	Evil maid attack		H108, C99	T223	
			5.1.1.4	Fake biometrics		H109, C100	T224	
			5.1.1.5	Physical access when the host is open		H110, C101	T225	
			5.1.1.6	Shoulder-surfing attack		H111, C102	T226	
			5.1.1.7	Physical attack (fault injection(glitching))		H112, C103	T227	
		5.1.2	Delete files at rest (HDD, SSD, Flash)			B81		
		OR	5.1.2.1	Delete the wallet application (or wallet manager) using factory reset or disk formatting		H26, C23	T228	
			5.1.2.2	Get root or admin privilege			B82	
			OR	4.1.2.2.1	Android root toolkit	H25	T229	
				4.1.2.2.2	Buffer overflow (code injection)	H25, C21	T230	
				4.1.2.2.3	Row Hammer attack	H25, C21	T231	
			5.1.2.3	Connect a debugger (JTAG, SWD)		C29	T232	
	5.2	Encrypt a wallet application (or wallet manager)				B83		
	OR	AND	5.2.1	Ransomware attack			B84	
			OR	5.2.1.1	Install a malware (ransomware)		B85	
				OR	5.2.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18, C17	T233
					5.2.1.1.2	Rogue AP	H19, C18	T234
					5.2.1.1.3	Supply chain attack	H20, C19	T235
					5.2.1.1.4	Removable media (USB drive)	H21, C20	T236
			5.2.1.2	Execute ransomware attack		H27, C22	T237	
		5.3	Prevent a user downloading or updating the wallet application or firmware				B86	
		OR	5.3.1	DoS attacks on the download server			B87	
			OR	5.3.1.1	Resource starvation (botnet, flooding attacks)		H7, H32, C7, C33, C39	T238
	5.3.1.2			Ransomware attack		H8, H32, C8, C33, C39	T239	
	5.3.1.3			SQL injection		H9, H32, C9, C33, C39	T240	
	5.3.2		Man-in-the-middle attacks between the wallet and the download server			B88		
	OR		5.3.2.1	ARP spoofing		H33, C34, C40	T241	
			5.3.2.2	DNS spoofing and poisoning		H33, C34, C40	T242	
		5.3.2.3	IP address spoofing		H33, C34, C40	T243		
6	Prevent a wallet application (or wallet manager) from accessing the blockchain network or API server					S6		
OR	6.1	Man-in-the-middle attacks between the wallet application and the blockchain network				B89		
	OR	6.1.1	ARP spoofing		H10, H11, H12, H69, H74, H88, H89, H92, H100, C10, C11, C12, C67, C68, C72, C91, C92, C95	T244		
		6.1.2	DNS spoofing and poisoning		H10, H11, H12, H69, H74, H88, H89, H92, H100, C10, C11, C12, C67, C68, C72, C91, C92, C95	T245		
		6.1.3	IP address spoofing		H10, H11, H12, H69, H74, H88, H89, H92, H100, C10, C11, C12, C67, C68, C72, C91, C92, C95	T246		

	6.2	DoS attacks on the blockchain node or API server				B90		
	OR	6.2.1	Resource starvation (botnet, flooding attacks)			H13, H70, H75, H93, C13, C73, C96 T247		
		6.2.2	Ransomware attack			H14, H70, H75, H93, C14, C73, C96 T248		
		6.2.3	SQL injection			H15, H70, H75, H93, C15, C73, C96 T249		
Privacy Breach						G3		
7	Obtain user account information					S7		
OR	7.1	Obtain account information from the wallet application (or wallet manager)				B91		
	AND	7.1.1	Bypass OS authentication			B92		
		OR	7.1.1.1	Brute-force attack (guessing, dictionary attack)		H106, C97	T250	
			7.1.1.2	Buffer overflow (code reuse)		H107, C98	T251	
			7.1.1.3	Evil maid attack		H108, C99	T252	
			7.1.1.4	Fake biometrics		H109, C100	T253	
			7.1.1.5	Physical access when the host is open		H110, C101	T254	
			7.1.1.6	Shoulder-surfing attack		H111, C102	T255	
			7.1.1.7	Physical attack (fault injection(glitching))		H112, C103	T256	
		7.1.2	Bypass wallet user authentication			B93		
		OR	7.1.2.1	Brute-force attack (guessing, dictionary attack)		H53, H63, C50	T257	
			7.1.2.2	Buffer overflow (code reuse)		H54, H63, C51	T258	
			7.1.2.3	Evil maid attack		H55, H63, C52	T259	
			7.1.2.4	Fake biometrics		H56, H6,	T260	
			7.1.2.5	Physical access when the wallet is open		H57, H63, C53	T261	
			7.1.2.6	Shoulder-surfing attack		H58, H63, C54	T262	
			7.1.2.7	Physical attack (fault injection(glitching))		H59, H63, C55	T263	
		7.1.2.8	Obtain authentication credentials using a malware (keylogger, screen recorder)		H35, H36, H37, H60, H61 H63,	T264		
		7.1.3	Obtain a passphrase			B94		
		OR	7.1.3.1	Shoulder-surfing attack		H44, H52	T265	
			7.1.3.2	Brute-force attack		H24, C27	T266	
			7.1.3.3	Physical attack (fault injection, probing, microscoping, cold boot attack)		H22, C25	T267	
			7.1.3.4	Connect a debugger (JTAG, SWD)		C26	T268	
	7.2	Obtain account information when a user uses the wallet				B95		
	OR	7.2.1	Eavesdrop input data				B96	
OR		7.2.1.1	Keylogger malware (keyboard, mouse, touch screen input logger) malware			B97		
		AND	OR	7.2.1.1.1	Install a malware (keylogger, screen touch input logger)		B98	
				7.2.1.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)		H18, C17	T269
				7.2.1.1.1.2	Rogue AP		H19, C18	T270
				7.2.1.1.1.3	Supply chain attack		H20, C19	T271
				7.2.1.1.1.4	Removable media (USB drive)		H21, C20	T272
			7.2.1.1.2	Execute keylogger attack		H81, C81	T273	
7.2.2		Eavesdrop output data				B99		
OR		7.2.2.1	Screen recorder malware			B100		
		AND	7.2.2.1.1	Install a malware (screen recorder)		B101		



			OR	7.2.2.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18, C17	T274				
				7.2.2.1.1.2	Rogue AP	H19, C18	T275				
				7.2.2.1.1.3	Supply chain attack	H20, C19	T276				
				7.2.2.1.1.4	Removable media (USB drive)	H21, C20	T277				
			7.2.2.1.2	Execute screen recording attack	H66, H72, H79, H91, C64, C70, C78, C94	T278					
		7.2.3	Eavesdrop clipboard data						B102		
		OR	7.2.3.1	Clipboard hijacker						B103	
			AND	7.2.3.1.1	Install a malware (clipboard hijacker)						B104
				OR	7.2.3.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18, C17	T279			
					7.2.3.1.1.2	Rogue AP	H19, C18	T280			
					7.2.3.1.1.3	Supply chain attack	H20, C19	T281			
					7.2.3.1.1.4	Removable media (USB drive)	H21, C20	T282			
				7.2.3.1.2	Execute clipboard hijacking attack	H67, H80, C65, C79	T283				
		7.2.4	Eavesdrop network traffic						B105		
		OR	7.2.4.1	Network packet sniffer						B106	
			AND	7.2.4.1.1	Install a malware (network packet sniffer)						B107
				OR	7.2.4.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18, C17	T284			
					7.2.4.1.1.2	Rogue AP	H19, C18	T285			
					7.2.4.1.1.3	Supply chain attack	H20, C19	T286			
					7.2.4.1.1.4	Removable media (USB drive)	H21, C20	T287			
			7.2.4.1.2	Execute network packet sniffing	H73, C71	T288					
			7.2.4.2	Man-in-the-middle attack						B108	
			OR	7.2.4.2.1	ARP spoofing	H71, H90, C69, C93	T289				
				7.2.4.2.2	DNS spoofing and poisoning	H71, H90, C69, C93	T290				
				7.2.4.2.3	IP address spoofing	H71, H90, C69, C93	T291				
		7.2.5	Eavesdrop peripheral data						B109		
			7.2.5.1	USB packet sniffer						B110	
			AND	7.2.5.1.1	Install a malware (USB packet sniffer)						B111
				OR	7.2.5.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	C17	T292			
					7.2.5.1.1.2	Rogue AP	C18	T293			
					7.2.5.1.1.3	Supply chain attack	C19	T294			
					7.2.5.1.1.4	Removable media (USB drive)	C20	T295			
				7.2.5.1.2	Execute USB packet sniffing attack	C66, C80, C84	T296				
			7.2.6	Shoulder-surfing attack						H86, C90	T297
8	Obtain a user's personally identifiable information						S8				
OR	8.1	Obtain personal information from the wallet application (or wallet manager)						B112			
	OR	8.1.1	Bypass OS authentication						B113		
		OR	8.1.1.1	Brute-force attack (guessing, dictionary attack)			H106, C97	T298			
			8.1.1.2	Buffer overflow (code reuse)			H107, C98	T299			
			8.1.1.3	Evil maid attack			H108, C99	T300			
			8.1.1.4	Fake biometrics			H109, C100	T301			
			8.1.1.5	Physical access when the host is open			H110, C101	T302			
			8.1.1.6	Shoulder-surfing attack			H111, C102	T303			
			8.1.1.7	Physical attack (fault injection(glitching))			H112, C103	T304			
	8.1.2	Bypass wallet user authentication						B114			

OR	OR	8.1.2.1	Brute-force attack (guessing, dictionary attack)			H53, C50	T305		
		8.1.2.2	Buffer overflow (code reuse)			H54, C51	T306		
		8.1.2.3	Evil maid attack			H55, C52	T307		
		8.1.2.4	Fake biometrics			H56	T308		
		8.1.2.5	Physical access when the wallet is open			H57, C53	T309		
		8.1.2.6	Shoulder-surfing attack			H58, C54	T310		
		8.1.2.7	Physical attack (fault injection(glitching))			H59, C55	T311		
		8.1.2.8	Obtain authentication credentials using a malware (keylogger, screen recorder)			H35, H36, H37, H60, H61	T312		
	8.2	Obtain personal information when a user uses the wallet					B115		
	OR	8.2.1	Eavesdrop input data					B116	
		OR	8.2.1.1	Keylogger (keyboard, mouse, touch screen input logger) malware				B117	
			AND	OR	8.2.1.1.1	Install a malware (keylogger, screen touch input logger)			B118
					OR	8.2.1.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18	T313
						8.2.1.1.1.2	Rogue AP	H19	T314
						8.2.1.1.1.3	Supply chain attack	H20	T315
				8.2.1.1.1.4	Removable media (USB drive)	H21	T316		
			8.2.1.1.2	Execute keylogger attack			H105	T317	
		8.2.2	Eavesdrop output data					B119	
		OR	8.2.2.1	Screen recorder malware				B120	
			AND	OR	8.2.2.1.1	Install a malware (screen recorder)			B121
					OR	8.2.2.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18	T318
						8.2.2.1.1.2	Rogue AP	H19	T319
						8.2.2.1.1.3	Supply chain attack	H20	T320
				8.2.2.1.1.4	Removable media (USB drive)	H21	T321		
			8.2.2.1.2	Execute screen recording attack			H103	T322	
		8.2.3	Eavesdrop clipboard data					B122	
		OR	8.2.3.1	Clipboard hijacker				B123	
			AND	OR	8.2.3.1.1	Install a malware (clipboard hijacker)			B124
					OR	8.2.3.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18	T323
						8.2.3.1.1.2	Rogue AP	H19	T324
						8.2.3.1.1.3	Supply chain attack	H20	T325
				8.2.3.1.1.4	Removable media (USB drive)	H21	T326		
			8.2.3.1.2	Execute clipboard hijacking attack			H104	T327	
		8.2.4	Eavesdrop network traffic					B125	
		OR	8.2.4.1	Network packet sniffer				B126	
			AND	OR	8.2.4.1.1	Install a malware (network packet sniffer)			B127
					OR	8.2.4.1.1.1	Social engineering (malicious files, malvertising, phishing, drive-by download attack)	H18	T328
	8.2.4.1.1.2					Rogue AP	H19	T329	
	8.2.4.1.1.3					Supply chain attack	H20	T330	
	8.2.4.1.1.4			Removable media (USB drive)	H21	T331			
	8.2.4.1.2		Execute network packet sniffing attack			H102	T332		
	8.2.4.2		Man-in-the-middle attack				B128		
	OR		8.2.4.2.1	ARP spoofing			H99, H101	T333	
			8.2.4.2.2	DNS spoofing and poisoning			H99, H101	T334	
			8.2.4.2.3	IP address spoofing			H99, H101	T335	

### 1) Transforming attack trees into a Bayesian network.

A Bayesian network is a probability graph model with a directed acyclic graph (DAG) structure. Each node of the Bayesian network represents a random variable, while the directional edges connecting different nodes represent the conditional dependencies between the random variables. Using a Bayesian network, it is possible to make probabilistic Bayesian inferences based on the given evidence. For example, one can calculate the probability of having a specific disease when certain symptoms appear. Therefore, Bayesian networks are widely used in various fields such as artificial intelligence, medicine (for disease diagnosis), and document classification; they are especially useful in the cybersecurity field for threat detection and spam filtering. [22] and [23] convert a fault tree into a Bayesian network to calculate the probability of system errors, and [24] converts different attack trees into a Bayesian network to calculate the probability of successful attacks. Therefore, we convert the attack trees we created above into a Bayesian network through the Model-to-Model (M2M) transformation method used in [22-24] to calculate the probability of successful attacks and measure the risks that appear when different threats are combined in a cryptocurrency wallet. Based on the calculated probability of attacks, the risk of each attack goal is measured using the general risk calculation formula (1).

$$\text{Risk} = \text{Probability} \times \text{Impact} \quad (1)$$

Bayesian networks are generally defined on discrete random variables. In this study, among them, the Bayesian network defined on the Bernoulli random variable is used. A Bernoulli random variable is a random variable that has only the values 1 or 0, as a result of an event. For example, when an independent coin toss is performed consecutively, random variables that have a value of 1 if it comes up heads and 0 if it comes up tails in each trial can be defined. Similarly, we transform the attack trees into a Bayesian network to represent each node as a Bernoulli random variable. When a threat or attack of each node occurs, the corresponding random variable has a value of 1, and when it does not occur, it has a value of 0. And the cause-effect relationship between nodes is expressed through directional edges, and a conditional probability table (CPT) is defined for each node to represent the conditional probability distribution when parent nodes are given. We can calculate the joint probability distribution using this conditional probability distribution and the well-known chain rule of Equation (2). A node that does not have a parent node is called a root node, and a prior probability is given to these nodes. The details of calculating the prior probability of the root node are described in Chapter B-2.

$$\Pr(x_1, \dots, x_n) = \prod_{i=1}^n \Pr(x_i | \text{Parent}(x_i)) \quad (2)$$

Fig. 6 shows the Bayesian network transformation method for AND and OR operations of the attack tree. As shown in the figure, the parent-child relationship is reversed after conversion. And each node A, B, and C represents a Bernoulli random variable, with a value of 1 indicating that the corresponding threat has occurred, and a value of 0 indicating that the corresponding threat has not occurred. Looking at the

conditional probability table (CPT), in the case of an OR operation, the probability that the value of the child node will be 1 becomes 1.0 if the value of one or more parent nodes is 1, And in the case of an AND operation, the probability that the value of the child node will be 1 becomes 1.0 only if the values of all parent nodes become 1.

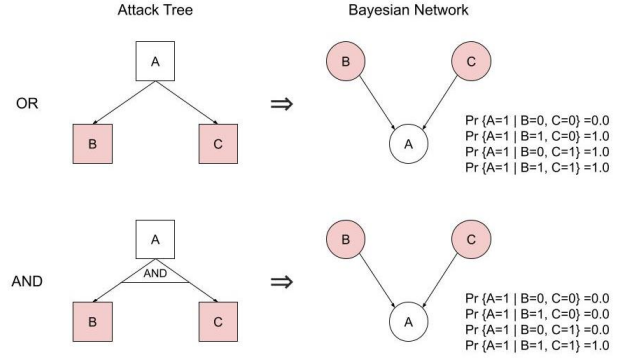


Fig. 6. Attack tree-to-Bayesian network translation using the M2M transformation method. Conditional probability tables (CPTs) are located next to the BN models.

The left side of Fig. 7 shows the attack trees for goals G1, G2, and G3 created in Chapter A-3. On the right is the Bayesian network that is derived from the attack trees through the M2M method. Looking at the Bayesian network, it can be seen that the parent-child relationship between nodes has been reversed compared to that of the attack trees. Also, the OR and AND operations have different conditional probability tables.

Note that the Bayesian network can combine nodes that are the same. Since the Bayesian network allows different child nodes to share the same parent node, nodes representing the same threat can be combined into a single node. The nodes indicated by dotted lines in Fig. 7 are those that can be combined with other nodes. For example, the B1 node of the G1 attack tree can be used to steal a user's private key by intercepting input data. In addition, the B96 node of G3 can be used to steal a user's account information by tapping the input data. Therefore, B1 and B96 can be used to steal cryptocurrency or user information by intercepting input data. Thus, B1 and B96 can be combined and represented as one node. In the Bayesian network on the right, they are combined into one node. Similarly, B9 and B70, and B10 and B71 can be used in attacks to steal cryptocurrency by finding the user's private key or in denial of service attacks by deleting the user's private key by bypassing user authentication of the operating system and wallet, respectively. Therefore, these pairs of nodes have been combined into one node in the transformed Bayesian network. By connecting nodes that are shared by each attack tree, it is possible to measure the overall risk for each attack goal in the Bayesian network.

### 2) Calculating the prior probabilities of threats.

In order to create a Bayesian network, it is necessary to calculate the prior probability that each threat node, which is the root node, will occur. In general, prior probabilities are calculated based on statistical data or set by subjective judgment based on expert knowledge. Since there is not yet a sufficient amount of statistical data on threats to cryptocurrency wallets,



TABLE VII. RISK ASSESSMENT METRICS.

Method	CVSS Exploitability	SP 800-30	OCTAVE Allegro	FAIR	EBIOS	CRAMM	MEHARI	CC Attack Potential
Likelihood	Attack Vector	-	-	-	Resources	-	-	Window of Opportunity
	Access Complexity	-	-	Contact Frequency	Activity	-	-	-
		Adversary Targeting	-	-	Activity	-	-	Knowledge of TOE
	Privilege Required	-	-	Difficulty	Resources	-	-	-
	User Interaction	-	-	Difficulty	Resources	-	-	-
	-	-	-	-	-	-	-	Elapsed Time
	-	Adversary Capability	-	Threat Capability	Resources	-	-	Expertise
	-	-	-	Probability of Action	Resources	-	-	Equipment
	-	-	-	-	-	-	-	-
	-	Adversary Intent	-	Probability of Action	Motivation	-	-	-
	-	Range of Effects	-	-	-	-	-	-
Impact	Integrity, Confidentiality, Availability	Vulnerability Severity	Reputation & Customer Confidence	Reputation	-	Integrity, Confidentiality, Availability	Integrity, Confidentiality, Availability	-
			Financial	Response, Replacement, Competitive advantage				
			Productivity	Productivity				
			Safety & Health	-				
			Fines & Legal Penalties	Fines and judgements				

we use a number of criteria to calculate the likelihood of threats in various standards and then calculate the prior probability of each threat.

We selected 7 of the 12 previously investigated standards that provided criteria for risk assessment. Although it is not a risk assessment standard, the Attack Potential factor, which is used to calculate the attack potential of a given vulnerability in the CC evaluation, was added. The results are shown in Table VII. In Table VII, likelihood is a combination of various standard factors that expresses the likelihood of a threat, and factors with a common meaning are shown in the same row. In addition, factors that overlap with each other are integrated; these are summarized in Table VIII.

We use the CVSS Exploitability metrics as the basic metrics to calculate the prior probability of threat occurrence. In the case of SP 800-30, FAIR, and EBIOS, there is no standard method to calculate the value of each element of likelihood, so it is determined by the subjective judgment of the analyst. Although CC's Attack Potential provides criteria for calculating the value of each metric, it is not suitable because the calculated Attack Potential score does not indicate the likelihood of an attack, but instead indicates the degree of resistance to an attack.

On the other hand, the Exploitability metrics of CVSS indicate the likelihood of a vulnerability occurring, and criteria

for calculating each value have been developed. In addition, since each metric has a value of [0, 1], it can be used to calculate joint probability when it is assumed to be a probability value. Therefore, we use the CVSS Exploitability metrics as the basic metrics to calculate the prior probability of each threat node. Similarly, N. Poolsappasit et al. [25] proposed a risk management method using a Bayesian attack graph, in which CVSS base metrics were used to calculate the conditional probability of each node. Furthermore, H. Zhang et al. [26] proposed a method of calculating the conditional probability of each threat by replacing the CVSS v2.0 metrics in the method in [25] with the CVSS v3.0 metrics. We use the CVSS v3.1 exploitability metrics in a similar manner as [26], and calculate the prior probability of each threat using the appended measurement factors based on Table VII.

Table VIII shows the calculated values of the CVSS metrics that are used to calculate the prior probability of threats. Here, in the case of Access Complexity, a Medium value is added for more precise calculation. Also, based on Table VII, three metrics are added. The Time Complexity (TC) metric indicates how much time complexity is required for an attack to be successful given that the attack can be performed. The Expertise (EX) metric represents the level of expertise or competence the attacker must possess. Lastly, the Equipment (EQ) metric represents the level of equipment or software required for an

attack. In addition, among the Likelihood factors in Table VII, Adversary intent or Motivation, which indicates the attacker's intention, is determined based on an actual attacker's profile. This factor was excluded since this study does not set a specific attacker. In addition, Range of Effects, which shows the ripple effect of the attack on other systems, is excluded because this study only considers a single system, i.e., the user's personal cryptocurrency wallet. We appended TC, EX, and EQ metrics to the probability computation formula of [26] to calculate the prior probability  $\Pr(T)$  of each threat node and defined Equation (3) by adjusting the constant value to 3.44 so that the maximum probability is approximately 1.0.

$$\Pr(T) = 3.44 \times AV \times AC \times PR \times UI \times TC \times EX \times EQ \quad (3)$$

Thus, the maximum probability is close to 1, as follows.

$$\Pr(T) = 3.44 \times 0.85 \times 0.77 \times 0.85 \times 0.85 \times 0.85 \times 0.85 \times 0.85 \approx 1.00$$

Also, the minimum probability is zero, as follows.

$$\Pr(T) = 3.44 \times 0.2 \times 0.44 \times 0.27 \times 0.62 \times 0 \times 0.33 \times 0.3 = 0.00$$

For the TC metric, when an attack could be launched immediately, there is no time constraint, so the same weight as used for PR and UI was applied and the metric value was set to 0.85. If an attack requires a great deal of time, such as a brute-force attack, the probability of attack success according to the attack time is calculated as follows. First, it is assumed that the average user changes their wallet device or account every 5 years, and that 90% of people will have changed their wallet or account after 9 years. Assuming that this probability distribution follows the normal distribution, it has a normally distributed curve with mean  $m = 5$  and standard deviation  $\sigma = 3.1$ , as shown in Fig. 8. If there are no restrictions on the remaining metrics and the required attack time is 5 years, the probability of a successful attack is 50%. Therefore, the value of the TC metric can be calculated inversely using Equation (3).

$$\Pr(T) = 3.44 \times AV(N) \times AC(L) \times PR(N) \times UI(N) \times TC \times EX(L) \times EQ(S)$$

$$= 3.44 \times 0.85 \times 0.77 \times 0.85 \times 0.85 \times TC \times 0.85 \times 0.85 = 0.5$$

Therefore, if TC takes 5 years, the metric value is 0.42. The remaining metric values were also calculated in the same manner.

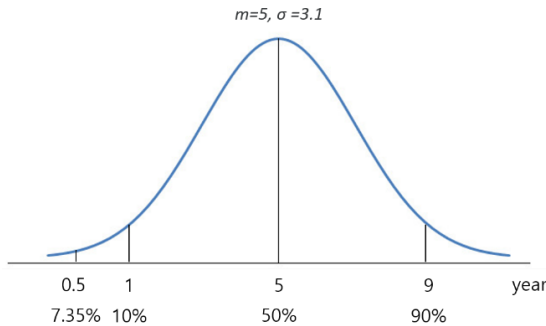


Fig. 8. Cryptocurrency wallet change cycle (assuming a normal distribution).

TABLE VIII. CVSS EXPLOITABILITY METRICS AND APPENDED METRICS WITH METRIC VALUES.

Metric		Metric Value				
CVSS Exploitability metric	Attack Vector (AV)	Network (N)	Adjacent (A)	Local (L)	Physical (P)	
		0.85	0.62	0.55	0.2	
	Access Complexity (AC)	Low (L)	Medium (M)	High (H)	-	
		0.77	0.62	0.44		
	Privilege Required (PR)	None (N)	Low (L)	High (H)	-	
		0.85	0.62	0.27		
	User Interaction (UI)	None (N)	Required (R)	-		-
		0.85	0.62			
Appended metric	Time Complexity (TC)	None (N)	$\leq 6$ m (M)	$\leq 5$ y (H)	$\leq 10$ y (E)	$> 10$ y (X)
		0.85	0.78	0.42	0.05	0
	Expertise (EX)	Layman (L)	Proficient (P)	Expert (E)	Multiple Experts (M)	
		0.85	0.53	0.39	0.33	
	Equipment (EQ)	Standard (S)	Specialized (P)	Bespoke (B)	Multiple Bespoke (M)	
		0.85	0.47	0.35	0.30	

The Expertise and Equipment metrics are added by referring to Attack Potential. We used the scores that were used in Attack Potential to calculate the value of each metric. Similar to the calculation of Time Complexity, the metric value was set to 0.85 for the lowest criterion without attack restrictions. Based on this value, the other metric values are determined according to the ratio of the Attack Potential scores. Here, since the score of the lowest criterion of each metric is 0, 1 is added to all values to calculate the ratio between factors. Based on the Attack Potential scores calculated when the rest of the metrics are not limited (with the minimum score), the metric values to be used for the threat occurrence probability calculation were derived. For example, Expertise's minimum score for Attack Potential is 1 point for the Layman criterion. When the remaining four metrics of Attack Potential take their minimum values, and if Expertise is Layman, the score is 5 points. When this score is converted to the metric value of the CVSS calculation method, it is set to 0.85. For the rest of the criteria, the value of the metric is calculated according to the attack potential score ratio. Here, the Attack Potential score is calculated by lowering the metric value by the rate at which the score is raised, as the higher the probability of attack success, the lower the score. For example, the score of Expertise's Proficient criterion is 8 points, which is 1.6 times higher than the score for the Layman criterion, so the value of the metric representing the threat occurrence probability is set to 0.53, as follows.

$$\Pr(T) = 0.85 \times 5 \div 8 = 0.53$$

The remaining Expertise and Equipment metric values are calculated in the same way. These values are shown in Table IX.

TABLE IX. CVSS EXPLOITABILITY METRICS AND APPENDED METRICS WITH METRIC VALUES

Metric		Metric Value		
		Attack Potential value	Attack Potential score when other metrics have the lowest values	Converted metric value
Expertise (EX)	Layman	1	5	0.85
	Proficient	4	8	0.53
	Expert	7	11	0.39
	Multiple Experts	9	13	0.33
Equipment (EQ)	Standard	1	5	0.85
	Specialized	5	9	0.47
	Bespoke	8	12	0.35
	Multiple Bespoke	10	14	0.30

We define a new function F in Equation (4) to simplify the prior probability calculation in Equation (3).

$$F(AV, AC, PR, UI, TC, EX, EQ) = 3.44 \times AV \times AC \times PR \times UI \times TC \times EX \times EQ \quad (4)$$

Based on Table VIII and Equation (4), the prior probabilities of some threat nodes in the Bayesian network of Fig. 7 are calculated, as follows. At this time, the wallet system was assumed to be a mobile wallet.

$$\Pr(T1) = F(N, L, L, R, M, P, S) = 0.30$$

$$\Pr(T2) = F(L, L, N, R, M, P, S) = 0.27$$

$$\Pr(T3) = F(N, H, N, R, M, E, S) = 0.18$$

$$\Pr(T4) = F(P, H, N, N, N, L, S) = 0.13$$

$$\Pr(T5) = F(N, L, L, R, N, L, S) = 0.53$$

$$\Pr(T11) = F(P, H, N, R, N, L, S) = 0.10$$

$$\Pr(T12) = F(P, L, N, N, E, L, P) = 0.01$$

$$\Pr(T18) = F(P, H, N, N, M, E, P) = 0.03$$

Nodes T1 – T4 represent the threats that an attacker will install keylogger malware. T1 uses social engineering techniques such as e-mail and SMS phishing, the system can be attacked remotely, the system can easily be reproduced, and user interaction is required to download the malware. T2 installs a malicious AP on the local network, requires an interaction for the user to access the AP, and requires some expertise to operate it. T3 is a supply chain attack that can be performed remotely; in order to succeed, a lot of prior knowledge about the supply chain system and a considerable level of expertise are required. T4 uses a removable storage medium such as a USB flash drive, requires physical access, and can only proceed at certain times

when the user is not using the host. T5 uses an installed keylogger. The keylogger malware mainly operates in the background and can continuously transmit user input data to the attacker remotely. T11 is a shoulder-surfing attack on the user's device, for which the attacker must be physically located in the same space as the user and an attack is possible only while the user is using the device. T12 accesses a device and obtains user authentication information through a brute-force attack. Although there is no user interaction, the attack generally requires a considerable amount of time and a specific tool or device. T18 bypasses user authentication through a physical attack such as a fault injection. It is difficult to reproduce this kind of attack, it usually takes several months or more to succeed, and a high level of expertise and special equipment are required. The prior probability of each threat node calculated in this manner is shown in the default prior item in Fig. 7.

If a specific security control is applied to the system in Fig. 7, the probability of occurrence of each threat will change. For example, assume that the device is separated from the external network and the private key is stored in a secure element to enhance physical security. Because the device is separated from the external network, threats T1 and T2 are limited to the local network, so the AV is changed to Local. Also, since the amount of contact between the system and the outside world is reduced, the conditions for performing the attack are more difficult, so the AC is changed to High. In addition, since T5 also needs to access the local network to collect data using malware, AV is changed to Local. Furthermore, as the system's physical security is strengthened by the use of a secure element, the attack time of T18 increases, so the TC is changed to Extreme, and the EQ is changed to Bespoke because more specialized attack equipment is needed. There are no changes to the remaining nodes (T2, T4, T11, and T12). Therefore, the prior probabilities after security controls are applied can be calculated as follows. These values are shown in the changed prior item in Fig. 7.

$$\Pr(T1) = F(L, H, L, R, M, P, S) = 0.11$$

$$\Pr(T2) = F(L, L, N, R, M, P, S) = 0.27$$

$$\Pr(T3) = F(L, H, N, R, M, E, S) = 0.11$$

$$\Pr(T4) = F(P, H, N, N, N, L, S) = 0.13$$

$$\Pr(T5) = F(L, L, L, R, N, L, S) = 0.34$$

$$\Pr(T11) = F(P, H, N, R, N, L, S) = 0.10$$

$$\Pr(T12) = F(P, L, N, N, E, L, P) = 0.01$$

$$\Pr(T18) = F(P, H, N, N, E, E, B) = 0.00$$

An examination of the changed prior probabilities shows that the nodes with the largest changes are T1 and T5. This is because the attack surface of the threats that install malware or steal data using malware is limited to the local network when network separation security controls are applied. In the case of node T18, the difference between the prior probabilities before and after applying the secure element is not large. This is because the possibility of a basic attack is low, as a physical attack itself is very difficult to execute and generally requires expertise and special equipment.



### 3) Calculating the marginal probabilities of sub-goals

In order to measure the risk of each wallet, we calculate the probability of each sub-goal of the attack trees. To this end, the joint probability is calculated using the conditional probability of all nodes in the path to each sub-goal. Therefore, the joint probability of each sub-goal node is calculated using the chain rule of Equation (2).

Then, by summing the joint probabilities obtained through Equation (2), when the state of the sub-goal node  $S$  is 1, the marginal probability  $\Pr(S=1)$  is calculated. In this case, the CPT of Fig. 6 is used to calculate the conditional probability for the AND and OR operation of each node. The CPTs of several nodes (B10, B2, and B7) are shown in Fig. 7. In the figure, the default marg and changed marg items of each sub-goal node represent the marginal probabilities before and after applying security controls to the system, respectively. For example, for node S1, the marginal probability decreases from 0.42 to 0.27 when the security controls are applied.

### 4) Calculating the risks of attack goals.

We measure the risk of each attack goal using Equation (1) based on the calculated marginal probabilities of each sub-goal. In Equation (1), Probability is the marginal probability of the sub-goal for each wallet. Impact is the scale of the damage that the sub-goal inflicts on users. The financial and reputation factors among the impact factors in Table VII are used to calculate the impact of each attack goal. Three factors, Integrity, Confidentiality, and Availability, are not relevant to financial damage due to the nature of cryptocurrency wallets. In addition, the Productivity and Fines & Legal Penalties factors apply to companies and are not relevant to personal wallets. Therefore, as shown in Table X, Impact is calculated using the Financial and Reputation factors. For example, S1 is an attack that steals assets by finding a user's private key. We determined that the Financial factor corresponds to Catastrophic(5) and the Reputation factor corresponds to Minor(2), therefore the Impact is set to 7. In S7, the attacker steals the user's asset information, and Impact is set to 4 after determining that the Financial factor corresponds to Insignificant(1) and the Reputation factor corresponds to Moderate(3). In this way, Impact is calculated for the remaining sub-goals.

TABLE X. IMPACT CALCULATION SCALE AND VALUES

Impact	Insignificant	Minor	Moderate	Major	Catastrophic
Financial	1	2	3	4	5
Reputation	1	2	3	4	5

When calculating risk, we calculate the probability of each sub-goal, not the probability of the final goal, because the amount of damage that is done is different for each sub-goal. Therefore, the risk is calculated by subdividing the goal into sub-goals, and then summing the risk of each to measure the risk of the final attack goal.

### C. Creating a security requirements checklist

As explained in Chapter B-2, the prior probability of a threat varies depending on whether security controls are applied to the wallet system. Therefore, when evaluating the risk to an actual

cryptocurrency wallet, the prior probability of a threat must be calculated differently according to the security controls applied to the system. To this end, based on the threat modeling result in Chapter A, a checklist of security requirements is derived, as shown in Table XI.

In Table XI, the domain is divided into a common area and specific platform areas (Embedded System and Mobile). This is done to denote whether the security requirement is commonly required for all wallets or only for specific platform wallets. For example, in the Embedded System area, the security requirements for the debugger pin are applicable only to an embedded system. In the checklist, the Impacted Node represents the nodes that are affected according to the security requirements. For example, Authentication a. in the Common area represents the requirement to prevent PIN or password exposure when using a wallet. If the security requirement is satisfied, the probabilities of occurrence of nodes T24, T147, T200, T262, and T310 of the attack tree are reduced. In the table (AC) indicates the metric in Table VIII that changes depending on whether or not the security requirement is satisfied. Removed Node represents a node that is removed according to the security requirements. For example, Authentication d. in the Common area represents a security requirement that applies an additional authentication method to a hierarchical deterministic wallet using a passphrase. If the security requirement is not satisfied, the B11, B55, and B94 nodes of the attack tree are deleted. In this case, (O) indicates a node that is removed when the security requirement is satisfied. Conversely, (X) means a node that is removed when the security requirement is not satisfied.

## IV. RESULTS OF RISK ASSESSMENT

The cryptocurrency wallets that are subjected to risk assessment in this paper are made with open source code or have reference documents that provide sufficient information on the structure or design of the wallet while also having a sufficient number of users. We selected two types of cold wallet and four types of hot wallet. Among cold wallets, the Ledger Nano S and Trezor One hardware wallets, which are the most popular worldwide, were selected for analysis. Among hot wallets, we selected and analyzed the Bread and Trust Wallets for mobile wallets, and the Copay and Electrum for PC wallets. The Android and Windows operating systems (OSs) were selected as the OSs for the mobile and PC wallets, respectively, as they are the most commonly used OSs worldwide.

### A. Results of the security requirements checklist

Table XII reveals whether the wallets on the market satisfy all security requirements based on the security requirements checklist in Table XI.

#### 1) Ledger Nano S

The Ledger Nano S is a hardware wallet that contains a secure element. The Ledger wallet was found to be the wallet that satisfies the most security requirements. Essentially, it is safe from malicious code infection attacks due to the characteristics of the embedded system and network separation. In particular, since it has a secure element it is safe from physical attacks. However, we found that Output.b in the common area was not satisfied, which indicates whether or not caution is exercised when the wallet shows the recovery phrase corresponding to the

TABLE XI. CRYPTOCURRENCY WALLET SECURITY REQUIREMENTS CHECKLIST

Domain	Category	Security Requirement	Impacted Node	Removed Node
Common	Authentication	a. Does the wallet hide the PIN or password on the screen?	T24(AC), T147(AC), T200(AC), T262(AC), T310(AC)	
		b. Does the wallet get disabled after a certain amount of consecutive unsuccessful authentication attempts?	T19(AC), T142(AC), T195(AC), T257(AC), T305(AC)	
		c. Does the wallet get locked if it is not used for a certain period of time?	T23(AC), T146(AC), T199(AC), T261(AC), T309(AC)	
		d. Can a passphrase be added to the recovery phrase to create a hidden wallet?		B11(X), B55(X), B94(X),
		e. Is there any protection mechanism for authentication credentials (e.g., encryption, hash, or secure element)?	T21(TC, EQ), T144(TC, EQ), T197(TC, EQ), T259(TC, EQ), T307(TC, EQ)	
		f. Is there any defense mechanism for physical attacks (e.g., fault injection) on the user authentication process?	T25(TC, EX, EQ), T29(TC, EX, EQ), T39(TC, EX, EQ), T148(TC, EX, EQ), T152(TC, EX, EQ), T201(TC, EX, EQ), T263(TC, EX, EQ), T267(TC, EX, EQ), T311(TC, EX, EQ)	
	Output	a. Is there a mechanism to prevent screen capture when a private key or recovery phrase is displayed?	T10(AC, PR), T26(AC, PR), T35(AC, PR), T149(AC, PR), T202(AC, PR), T264(AC, PR), T312(AC, PR), T322(AC, PR),	
		b. Does the wallet deliver a warning message about the risk of exposing a private key or recovery phrase before they are displayed?	T11(AC), T27(AC), T150(AC)	
		c. Is user authentication required before displaying a private key or recovery phrase at the request of a user?		B10(X)
		d. Is there a mechanism to prevent screen capture when account or personal information is displayed?	T278(AC, PR)	
	Input	a. Is there a defense mechanism for keylogging attacks when a private key or recovery phrase is entered by a user?	T5(AC, PR), T26(AC, PR), T37(AC, PR), T149(AC, PR), T202(AC, PR), T264(AC, PR), T312(AC, PR), T317(AC, PR)	
	Copy	a. Is it forbidden to copy a private key or recovery phrase to the clipboard?	T36(AC, PR), T49(AC, PR)	
	Key Generation	a. Is a proven random number generator used to generate a seed or a private key?	T96(TC, EQ)	
		b. Is more than 112-bit entropy used to generate a master seed?	T96(TC, EQ)	
	Key Management	a. Is an encryption key that provides more than 112 bits of security length used to encrypt a private key or recovery phrase?		T43(X)
		b. Is there an access control mechanism for the encrypted private key or recovery phrase?	T38(AC, TC), T39(AC, TC)	
		c. Is there any defense mechanism for physical attacks (e.g., microprobing or reverse engineering) on the device?	T25(TC, EX, EQ), T29(TC, EX, EQ), T39(TC, EX, EQ), T42(TC, EX, EQ), T44(TC, EX, EQ), T52(TC, EX, EQ), T68(TC, EX, EQ), T34(TC, EX, EQ), T148(TC, EX, EQ), T152(TC, EX, EQ), T156(TC, EX, EQ), T199(TC, EX, EQ), T201(TC, EX, EQ), T206(TC, EX, EQ), T227(TC, EX, EQ), T231(TC, EX, EQ), T263(TC, EX, EQ), T267(TC, EX, EQ), T311(TC, EX, EQ)	
	Transaction	a. Is the detail of a new transaction displayed and user confirmation is required before signing the transaction?		B41(X), B46(X)
		b. Is user authentication required before signing a new transaction?		B54(X)
		c. Is a proven random number generator used to generate a signature?	T97(TC, EQ), T98(TC, EQ)	
	Application	a. Is there any integrity verification mechanism for the wallet application or wallet manager?	T54(PR, EX), T56(PR, EX), T57(PR, EX), T58(PR, EX), T106(PR, EX), T108(PR, EX), T109(PR, EX), T110(PR, EX), T158(PR, EX), T160(PR, EX), T161(PR, EX), T162(PR, EX)	

TABLE XII. CRYPTOCURRENCY WALLET SECURITY REQUIREMENTS:  
RESULTS OF THE ANALYSIS

Domain	Category	Security Requirement	Ledger Nano S	Trezor One	Bread Wallet	Trust Wallet	Copay Wallet	Electrum Wallet
Common	Authentication	a. Does the wallet hide the PIN or password on the screen?	O	O	O	O	O	O
		b. Does the wallet get disabled after a certain amount of consecutive unsuccessful authentication attempts?	O	O	△	O	X	X
		c. Does the wallet get locked if it is not used for a certain period of time?	O	X	X	O	X	X
		d. Can a passphrase be added to the recovery phrase to create a hidden wallet?	O	O	X	X	X	X
		e. Is there any protection mechanism for authentication credentials (e.g., encryption, hash)?	O	O	O	O	O	O
		f. Is there any defense mechanism for physical attacks on the user authentication process?	O	X	X	X	X	X
	Output	a. Is there a mechanism to prevent screen capture when a private key or recovery phrase is displayed?	O	O	X	O	X	X
		b. Does the wallet deliver a warning message about the risk of exposing a private key or recovery phrase before they are displayed?	X	O	X	O	O	O
		c. Is user authentication required before displaying a private key or recovery phrase at the request of a user?	O	O	O	O	O	O
	Input	a. Is there a defense mechanism for keylogging attacks when a private key is entered?	O	O	X	X	X	X
	Copy	a. Is it forbidden to copy a private key or recovery phrase to the clipboard?	O	O	X	O	X	X
	Key Generation	a. Is a proven random number generator used to generate a seed or a private key?	O	O	O	O	O	O
		b. Is more than 112-bit entropy used to generate a master seed?	O	O	O	O	O	O
	Key Management	a. Is an encryption key that provides more than 112 bits of security length used to encrypt a private key or recovery phrase?	O	O	O	O	O	O
		b. Is there an access control mechanism for the encrypted private key or recovery phrase?	O	O	O	O	X	X
		c. Is there any defense mechanism for physical attacks on the device?	O	X	O	O	X	X
	Transaction	a. Is the detail of a new transaction displayed and user confirmation is required before signing the transaction?	O	O	O	O	O	O
		b. Is user authentication required before signing a new transaction?	O	O	O	O	O	O
		c. Is a proven random number generator used to generate a signature?	O	O	O	O	O	O
	Application	a. Is there any integrity verification mechanism for the wallet application or wallet manager?	O	O	O	O	O	O
	Network	a. Is data transmitted across networks through secure channels (e.g., HTTPS)?	O	O	O	O	O	O
		b. Does the wallet device keep offline (air-gapped) when it is not used?	O	O	X	X	X	X
	Recovery	a. Are there any instructions explaining the importance of backing up private keys or a recovery phrase?	O	O	△	O	O	O
		b. Is there a mechanism to check if the user has backed up a private key or recovery phrase?	O	X	O	O	O	O
	Privacy	a. Is personally identifiable user information is not entered or stored in the wallet?	O	O	O	O	△	O
		b. Is user authentication required before displaying an account address or balance?	O	O	O	O	X	O
Embedded System	Output	a. Is there an output interface to display an account address or transaction information for user confirmation?	O	O	-	-	-	-
	Firmware	a. Is there any firmware integrity verification mechanism?	O	O	-	-	-	-
	Debugger	a. Are debugger pins removed or disabled (e.g., JTAG)?	O	O	-	-	-	-
	Communication	a. Is there a secure communication mechanism between the host and the wallet device?	O	X	-	-	-	-
	Authentication	a. Is there a mechanism for checking the authenticity of the wallet device that is connected to the host?	O	X	-	-	-	-
	Authorization	a. Is there an authorization mechanism for the wallet manager that is installed on the host?	O	O	-	-	-	-
Mobile	Privilege Escalation	a. Is there a mechanism to check if the device is rooted?	-	-	O	O	-	-

private key to the user. Therefore, since it does not mitigate the risk of exposing recovery phrases, it may be vulnerable to a shoulder-surfing attack.

### 2) *Trezor One*

The Trezor One wallet is a hardware wallet that has a general-purpose built-in MCU. Like the Ledger Nano S, the Trezor wallet was found to basically be safe from malicious code infection attacks due to the characteristics of the embedded system and network separation. However, the Trezor wallet is vulnerable to physical attacks because the keys are stored in a general-purpose MCU. In addition, since there is no automatic lock function when the wallet is not used for a certain period of time, an attacker may be able to access and use the wallet while the user is away.

### 3) *Bread*

Bread is a mobile wallet that does not satisfy the security requirements for screen capture, keylogger, and clipboard data stealing attacks. Therefore, we confirmed that the wallet is generally vulnerable to malicious code attacks. It is also vulnerable to physical access attacks because it lacks an automatic locking function. However, private keys are stored in Android's Keystore system and managed securely.

### 4) *Trust Wallet*

Trust Wallet is a mobile wallet that uses security controls to prevent screen capture and clipboard data theft attacks. Therefore, it is safer than the Bread wallet against malicious code infection attacks. However, there is no security mechanism for keylogger attacks, so there is a threat that input data will be stolen remains. Also, the wallet automatic locking function is applied, and the private keys are stored in Android's Keystore system and managed safely.

### 5) *Copay*

As a PC wallet, Copay was found to not satisfy the security requirements for screen capture, keylogger, and clipboard data stealing attacks. Therefore, it is generally vulnerable to malicious code attacks. It is also vulnerable to physical access attacks because it lacks an automatic locking function. In particular, although the user's private key is encrypted and stored, it is stored in the general storage space of the PC, so it may be stolen by malware, making the private key vulnerable to brute force attacks. In addition, even without user authentication, the address and balance of the cryptocurrency account can be discovered, and there is a function to register an email address to receive notifications about deposits and withdrawals to/from the account, so there is a threat of privacy invasion.

### 6) *Electrum*

As a PC wallet, Electrum, like the Copay wallet, does not satisfy the security requirements for screen capture, keylogger, and clipboard data stealing attacks. Therefore, it is generally vulnerable to malicious code attacks. It is also vulnerable to physical access attacks because it lacks an automatic locking function. In addition, although the user's private key is encrypted and stored, it is stored in a general storage space such as the PC's HDD or SSD, so the private key is vulnerable to brute force attacks if it is stolen by malware.

## B. *Risk measurement results*

We combined all of the attack trees and converted them into a Bayesian network as described in Chapter III. Then, we measured the risks to actual cryptocurrency wallets based on the findings of the security requirements checklist. Table XIII shows the risk measurement results of all six cryptocurrency wallets that were analyzed.

Regarding the first attack goal (G1, stealing cryptocurrency), the Ledger Nano S is the least vulnerable wallet. The next best is the Trezor One. Therefore, as is generally known, hardware wallets are at lower risk of cryptocurrency theft than software wallets. The Bread wallet was found to have the highest risk, at 1.49 times higher than the Ledger Nano S. On average, the risk to software wallets was 1.41 times greater than that to hardware wallets. Among the sub-goals of G1, the risk to hardware and software wallets differed most clearly at S1, the private key stealing sub-goal. The most effective way to achieve S1 is to launch an attack using malicious code. Therefore, since an embedded hardware wallet that is stored separately from a network is relatively safe from malicious code attacks, the risk of S1 differs greatly between different types of wallets. Furthermore, the risk of S2 is lower than that of S1 for all wallets. This is because, in order for an attacker to transfer cryptocurrency to herself using a wallet, the attacker may physically access the wallet and bypass user authentication or install a wallet that has been tampered with through a supply chain attack. These methods have a low probability of attack success. Therefore, the risk of cryptocurrency being stolen in this manner is low. For the next sub-goal, S3, the risks posed to hardware wallets and software wallets were similar. This is because one of the most effective ways to intercept a user's cryptocurrency is to change the address on the clipboard. Even if the user uses a hardware wallet, the host's wallet manager program should be used to copy the account address. Therefore, regardless of whether a user chooses a hardware or software wallet, the risk of host malware infection is similar, so the risk of S3 is similar across wallet types.

Notably, for G1, the risk to the Trezor One with its general-purpose MCU is only 1.05 times greater than the risk to the Ledger Nano S with a built-in secure element. This difference is not large. This is because, as previously mentioned, the most effective attack method that is used to steal cryptocurrency is malware infection, and the use of hardware wallets removes many high-risk threats. An attack that accesses a hardware wallet and bypasses user authentication (through either a physical attack or extraction of the private key stored in memory) has a very low probability of success even if no secure element is used. Therefore, even if a secure element is used, the resulting reduction in the risk of cryptocurrency theft is small.

For the second attack goal, G2 (denial of service), similar risks were observed across all wallets except for the Trezor One. Locking or deleting a wallet is easily accomplished if a wallet can be accessed, and since the wallet manager application that is used to manage hardware wallets is installed on a general PC, DoS attacks using malicious codes are possible. Therefore, most wallets have a similar degree of risk of G2. However, the risk of G2 to the Trezor One is particularly high, because the Trezor One does not check whether the user has backed up his recovery

TABLE XIII. RISK ASSESSMENT OF CRYPTOCURRENCY WALLETS WITH THREE GOALS AND EIGHT SUB-GOALS (P: MARGINAL PROBABILITY, I: IMPACT, R: RISK)

Wallet		G1. Stealing Cryptocurrency				G2. Denial of Service				G3. Privacy Breach			Total Risk
		S1	S2	S3	Sum	S4	S5	S6	Sum	S7	S8	Sum	
Ledger Nano S	P	0.37	0.32	0.63	-	0.51	0.76	0.39	-	0.88	0	-	-
	I	7	7	6	-	4	4	3	-	3	0	-	-
	R	2.59	2.24	3.78	8.61	2.04	3.04	1.17	6.25	2.64	0	2.64	17.5
Trezor One	P	0.4	0.35	0.63	-	0.55	0.77	0.39	-	0.89	0	-	-
	I	7	7	6	-	5	5	3	-	3	0	-	-
	R	2.8	2.45	3.78	9.03	2.75	3.85	1.17	7.77	2.67	0	2.67	19.47
Bread Wallet	P	0.93	0.41	0.57	-	0.58	0.82	0.39	-	0.89	0	-	-
	I	7	7	6	-	4	4	3	-	3	0	-	-
	R	6.51	2.87	3.42	12.8	2.32	3.28	1.17	6.77	2.67	0	2.67	22.24
Trust Wallet	P	0.84	0.35	0.57	-	0.57	0.82	0.39	-	0.84	0	-	-
	I	7	7	6	-	4	4	3	-	3	0	-	-
	R	5.88	2.45	3.42	11.75	2.28	3.28	1.17	6.73	2.52	0	2.52	21
Coplay Wallet	P	0.92	0.4	0.56	-	0.57	0.81	0.39	-	0.95	0.92	-	-
	I	7	7	6	-	4	4	3	-	3	3	-	-
	R	6.44	2.8	3.36	12.6	2.28	3.24	1.17	6.69	2.85	1.84	4.69	24.9
Electrum Wallet	P	0.92	0.4	0.56	-	0.57	0.81	0.39	-	0.91	0	-	-
	I	7	7	6	-	4	4	3	-	3	0	-	-
	R	6.44	2.8	3.36	12.6	2.28	3.24	1.17	6.69	2.73	0	2.73	22.02

phrases. If the user forgets the PIN or loses the wallet itself without first writing down and storing the recovery phrase, it is impossible to recover the asset. Therefore, the risk of G2 is greater for the Trezor One, as the impact on S4 and S5 is high (Table XII).

For the third attack goal, G3 (privacy breach), all wallets except the Copay wallet show low risk. S7's risk of stealing user account information is similar for both hardware and software wallets. This is because, even if a hardware wallet is used, a wallet manager must be installed on the host, so the threat of account information exposure is similar to that of a software wallet. However, the risk that personal information will be stolen (S8) is 0 for all wallets except the Copay wallet. The S8 node was deleted because all wallets except the Copay wallet do not receive or store any identifiable personal information. On the other hand, for the Copay wallet, the S8 node was not deleted because the user can enter an email address to receive messages when certain account events occur. Therefore, for G3, the Copay wallet is at particularly high risk.

### C. Results of risk assessment

We evaluated the overall risk of each wallet by summing all of the risks of each attack goal. Fig. 9 graphically displays the total risk results that are listed in Table XIII. Among the 6 wallets, the Ledger Nano S has the lowest risk, at 17.5, while the Trezor One has the second lowest risk, at 19.47. Therefore, we can conclude that hardware wallets are generally more secure than software wallets. The average overall risk posed to software wallets is 1.22 times greater than the risk posed to hardware wallets.

Among software wallets, the Trust Wallet was found to have the lowest risk, at 21. On the other hand, the Copay wallet has the highest risk among all wallets, at 24.9. Thus, the Trust Wallet has been confirmed to be the safest among all four software wallets, as it applies security controls such as screen capture prevention and rooting detection. In addition, the Copay wallet has the highest overall risk due to the significant increase in the risk of G3 arising from registration of the user's e-mail address.

The overall risk to the Trezor One, which has a general-purpose MCU, is 1.11 times greater than that to the Ledger Nano S with its built-in secure element. However, in the case of the Trezor One, the risk of G2 is relatively high due to the lack of a backup function for recovery phrases, which greatly increases the overall risk. Therefore, considering that the risk to the Trezor One is only 1.05 times greater than that to the Ledger Nano S for goal G1, where the role of a secure element is the most important, and that the risk to the Trezor One is only 1.01 times greater than that to the Ledger Nano S for goal G3, we can conclude that use of a secure element does not significantly diminish the overall risk to the wallet.

## V. CONCLUSION

In this paper, we derived the security requirements of cryptocurrency wallets through threat modeling and evaluated the risks to each of 6 wallets on the market by converting the attack trees into a Bayesian network. We found that hardware wallets are more secure than software wallets, as is generally known. In addition, the use of a secure element in hardware wallets was shown to be less effective at reducing the overall risk. This is because the possibility of serious threats that must

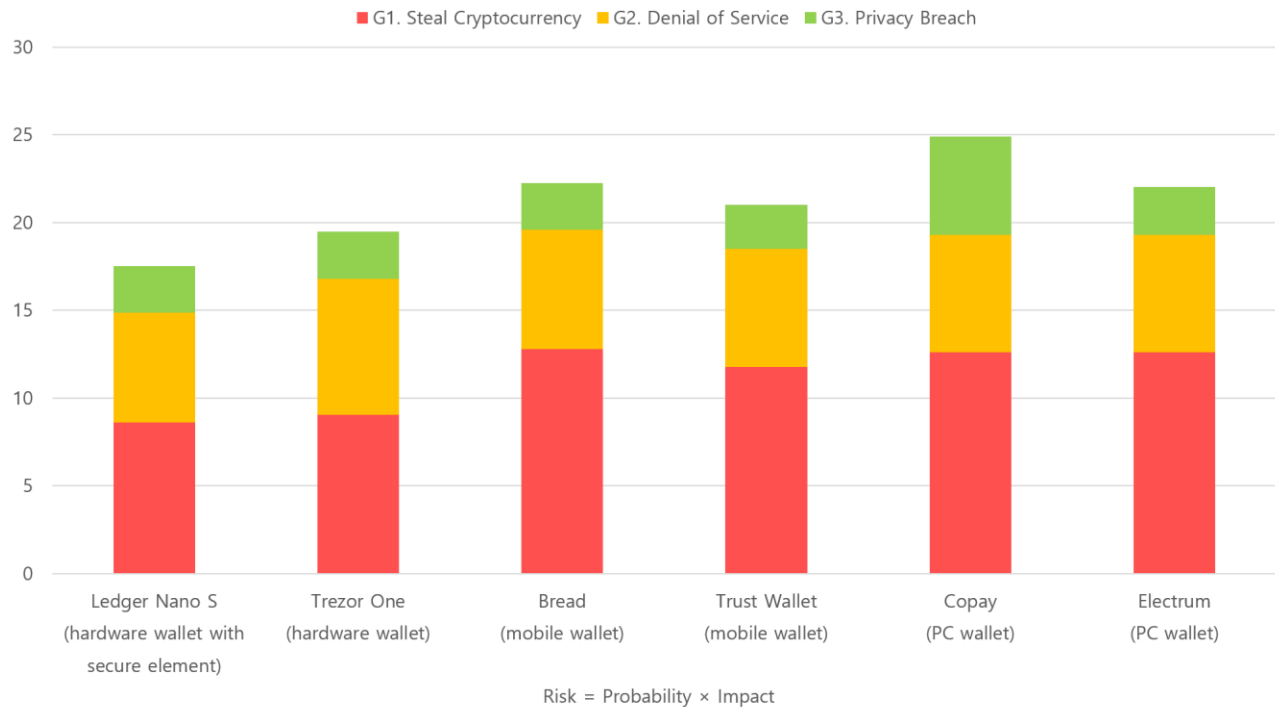


Fig. 9. Risk assessment results for 6 cryptocurrency wallets and three attack goals.

be guarded against through use of a secure element is low under normal circumstances. However, if the value of the cryptocurrency assets stored in a wallet is high enough for an attacker to physically access the wallet and perform a physical attack using specialized equipment, the impact of a successful attack will be very high and the risk will also increase. Therefore, it is important to choose a suitable wallet in consideration of the value of the assets to be protected.

The cryptocurrency wallet security requirements checklist that was derived through this study can be referenced by wallet developers and used for security-by-design purposes. In addition, since the vulnerabilities of cryptocurrency wallets can be quantitatively measured through the methodology presented in this study, improved risk management has been made possible. Importantly, risk assessors can adjust the probability and impact of threats by applying security controls according to a specific operating environment. In addition, the risk assessment methodology used in this study can be widely applied to various systems other than cryptocurrency wallets.

#### REFERENCES

- [1] Nakamoto, Satoshi, "Bitcoin: A peer-to-peer electronic cash system," May 2009. [Online] Available: <https://bitcoin.org/bitcoin.pdf>.
- [2] Buterin, Vitalik, "A next-generation smart contract and decentralized application platform," 2014. [Online] Available: [https://cryptorating.eu/whitepapers/Ethereum/Ethereum\\_white\\_paper.pdf](https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf).
- [3] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," Dec. 2020. [Online] Available: <https://ethereum.github.io/yellowpaper/paper.pdf>.
- [4] A. R. Sai, J. Buckley and A. Le Gear, "Privacy and Security Analysis of Cryptocurrency Mobile Applications," 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ), Miami Beach, FL, USA, pp. 1-6, doi: 10.1109/MOBISECSERV.2019.8686583. 2019.
- [5] Er-Rajy, L., et al., "Blockchain: Bitcoin wallet cryptography security, challenges and countermeasures," in Journal of Internet Banking and Commerce 22.3: 1-29. 2017.
- [6] E. Almutairi and S. Al-Megren, "Usability and Security Analysis of the KeepKey Wallet," 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Seoul, Korea (South), pp. 149-153. 2019.
- [7] D. He et al., "Security Analysis of Cryptocurrency Wallets in Android-Based Applications," in IEEE Network, vol. 34, no. 6, pp. 114-119, November/December, doi: 10.1109/MNET.011.2000025. 2020.
- [8] M. Guri, "BeatCoin: Leaking Private Keys from Air-Gapped Cryptocurrency Wallets," in 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, pp. 1308-1316, doi: 10.1109/Cybermatics\_2018.2018.00227. 2018.
- [9] Electric Coin Company, "Wallet App Threat Model," [Online] Available: [https://zcash.readthedocs.io/en/latest/rtd\\_pages/wallet\\_threat\\_model.html](https://zcash.readthedocs.io/en/latest/rtd_pages/wallet_threat_model.html). 2019.
- [10] SWhonix, "Cryptocurrency Hardware Wallet: Threat Model," [Online] Available: [https://www.whonix.org/wiki/Hardware\\_Wallet\\_Security](https://www.whonix.org/wiki/Hardware_Wallet_Security). Dec. 2020.
- [11] Taylor Hornby, "Invariant-Centric Threat Modeling," [Online] Available: <https://github.com/defuse/ictm>. Oct. 2019.
- [12] M. Guri and Y. Elovici, "Bridgware: The air-gap malware," Commun. ACM, vol. 61, no. 4, pp. 74-82, [Online]. Available: <http://doi.acm.org/10.1145/3177230>. Mar. 2018.
- [13] Dmitry Nedospasov, Thomas Roth and Josh Datko, "wallet.fail", in 35th Chaos Communication Congress, [Online] Available: <https://wallet.fail>. 2018.
- [14] G. Miraje, M. Paulo and S. Leonel, "Trustzone-backed bitcoin wallet," in Proc. The Fourth Workshop on Cryptography and Security in Computing Systems, pp.25-28, 2017.
- [15] W. Dai, J. Deng, Q. Wang, C. Cui, D. Zou and H. Jin, "SBLWT: A Secure Blockchain Lightweight Wallet Based on Trustzone," in IEEE Access, vol. 6, pp. 40638-40648, doi: 10.1109/ACCESS.2018.2856864. 2018.

- [16] Y. Liu et al., "An efficient method to enhance bitcoin wallet security," in 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, pp. 26-29. 2017.
- [17] Palatinus, Marek, et al., "BIP 39: Mnemonic code for generating deterministic keys." [Online] Available: <https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki>. 2013.
- [18] Wuille, Pieter, "BIP 32: Hierarchical deterministic wallets," [Online]. Available: <https://github.com/genjix/bips/blob/master/bip-0032.md>. 2012.
- [19] B. Schneier, "Attack Trees," Dr. Dobbs's J., Dec. 1999.
- [20] Breitner, Joachim, and Nadia Heninger, "Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies," in International Conference on Financial Cryptography and Data Security. Springer, Cham, 2019.
- [21] Jochen Hoenicke, "Extracting the Private Key from a TREZOR," [Online] Available: <https://jochen-hoenicke.de/crypto/trezor-power-analysis>. 2015.
- [22] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into Bayesian Networks," in Reliability Engineering and System Safety, 71, Rome, Italy, pp. 249-260. 2001.
- [23] Pappaterra, Mauro José. "Implementing Bayesian Networks for online threat detection." 2018.
- [24] Gribaudo, Marco, Mauro Iacono, and Stefano Marrone. "Exploiting Bayesian networks for the analysis of combined attack trees." Electronic notes in theoretical computer science 310: 91-111. 2015.
- [25] N. Poolsappasit, R. Dewri and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," in IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 1, pp. 61-74, doi: 10.1109/TDSC.2011.34. 2012.
- [26] H. Zhang, F. Lou, Y. Fu and Z. Tian, "A Conditional Probability Computation Method for Vulnerability Exploitation Based on CVSS," 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), Shenzhen, pp. 238-241, doi: 10.1109/DSC.2017.33. 2017.