Pragmatic Analysis of Key Management for Cryptocurrency Custodians

Yuto Takei
Mercari, Inc., and
Tokyo Institute of Technology

Kazuyuki Shudo
Kyoto University

***Abstract*—We discuss key management for cryptocurrencies from the perspective of security and risk management. While we found many earlier research studies about the implementation and security of wallets, few of them provide a comprehensive analysis of real-world integration as a complex system. We particularly focus on cryptocurrency custodians, who have to tightly control the risk to meet business needs as well as regulatory requirements. Unlike individual use cases, to manage substantial amounts of various assets, they typically need more complex wallet configurations and operations, such as multiple layers of hot and cold wallets in combination with different types of implementations. Therefore, we discuss the suitability of various wallet techniques, including software, hardware, HSMs, smart contracts, or cryptographic methods. We also address several open challenges for custodians mentioned in earlier work. Furthermore, as the ultimate example, we propose Extreme-Cold, a reference cold wallet in an air-gapped environment. It is resistant to side-channel attacks studied in earlier research. The risk assessment we conduct on Extreme-Cold demonstrates the effectiveness of our systematized knowledge.**

***Index Terms*—cryptocurrency, wallet, security, cryptography, key management system, digital signature**

I. Introduction

Cryptocurrencies have gained worldwide popularity as financial assets, with their market capitalization often surpassing one trillion dollars between 2021 and 2024. Owners of cryptocurrencies must effectively manage their private keys, which are essential for initiating transactions. This objective becomes even more critical for custodians of cryptocurrencies, such as exchanges or payment service providers, considering the significant scale of assets they handle.

The management of cryptographic keys has been a subject of study for a long time, even before the birth of cryptocurrencies [1]. It has been known that there is a dilemma between measures to prevent damage or loss of cryptographic keys through replication, and measures to enhance encryption and authentication to prevent leakage or unauthorized use of keys [2]. Therefore, organizations responsible for handling cryptographic keys must find a balance and invest appropriately.

While there have been numerous studies on the management of private keys in the context of cryptocurrencies [3], many of these studies have focused on wallets suited for individuals. Some researchers propose wallet implementations, while others discuss attack methods against existing wallets.

However, when considering cryptocurrency custodians, it is necessary to take into account not only the perspective of standalone wallets but also broader factors such as physical and human elements. Additionally, the business environment related to wallet management must be considered in certain cases. Jaroucheh et al. conducted a review from the perspective of custodians, addressing token classification and wallet technologies [4]. In their study, they highlighted several open challenges in wallet technology, such as regulatory differences, unknown security risks, transparency of wallet implementation, and cost. Many cryptocurrency custodians, including the ones the author has affiliated with, have already encountered and discussed these considerations in reality, even in the absence of extensive existing research literature.

Hence, this paper examines the pragmatics of cryptocurrency custodianship and aims to fill the gaps in the literature on effective practices for private key management.

The main contributions of this paper are as follows:

- Investigating literature on cryptographic key management from before the emergence of cryptocurrencies, and exploring the connections to wallet technologies.
- Examining wallet technologies from the enterprise scale managing substantial assets, and analyzing their security.
- Considering management strategy across multiple wallets, such as hot and cold, common for exchanges.
- Proposing a reference cold wallet implementation, Extreme-Cold and conducting actual risk analysis based on this implementation.

This paper is structured as follows. In Section II, we first review the historical development of cryptographic key management techniques, ranging from previous literature to recent approaches in the general Internet industry. In Section III, we model the signing system of cryptocurrency custodians and consider various approaches to evaluate its security and potential threats. In Section IV, we consider unique requirements for cryptocurrency exchanges, such as regulations of solvency and audits. In Section V, we explore different wallet options, analyzing the strengths, weaknesses, characteristics, and risks associated with each technology. In Section VI, we discuss the evaluation criteria for building new wallets in cryptocurrency custodians. In Section VII, we propose a cold wallet configuration called Extreme-Cold for demonstration, which operates in a highly secure and self-managed environment

to enhance risk management transparency. In Section VIII, we provide the evaluation of EXTREME-COLD, based on the aspects proposed earlier. Finally, in Section IX, we conclude by showing the future direction of the research.

II. BACKGROUND

A. Genesis of Cryptographic Key Management

Cryptographic key management has been a significant concern even before the development of cryptocurrency. Initially, it was mainly studied for military purposes. With the rise of computers, the importance of key management became even more apparent. This is evident from literature dating back to the 1970s, which is still relevant today.

In a study by Popek et al., various cryptographic techniques were examined known at the time in 1979, along with their applications and associated issues [5]. They mentioned the importance of keeping cryptographic systems simple in design. They also highlighted the need for key backups and the protection of them as secure as the originals. In the same year, Blakley classified incidents related to human involvement in key management into three types: abnegation, betrayal, or combination incidents. They formulated the number of key copies an organization should keep based on their protection requirements [2]. Additionally, Shamir introduced a scheme for securely distributing a secret value among multiple parties, which is widely used today [6]. Later, Fumy et al. provided a systematic approach to designing key management services (KMS) [1]. They discussed technical requirements throughout the key's lifecycle, including generation, distribution, activation, and deletion. They also emphasized the importance of prohibiting plaintext access to private keys.

In the financial sector, the need for key management practices led to the compilation of ANSI X9.17 by practitioners in the US banking industry in 1985. This standard defines key management practices and encryption techniques for both manual and automated processes [7]. Based on this, the US government established FIPS 171, which provides guidelines for cryptography usage in general government information systems. This standard has been continuously updated and is currently known as NIST SP 800-57.

B. Key Management for Internet Infrastructures

The widespread use of the Internet has made key management essential in the communication industry. One notable scenario is Pretty Good Privacy (PGP), which is an ad-hoc messaging network [8]. In PGP, each participant generates a key pair and exchanges the public key with their peers to enable sender authentication and message encryption. This concept is sometimes referred to as the Web of Trust, and several research studies have been conducted, such as [9].

On the other hand, Public Key Infrastructure (PKI) and DNS Security Extensions (DNSSEC) are well-known technologies that propagate trust hierarchically within a specific authoritative domain. Participants in these systems maintain a predefined set of trusted institutions, known as trust anchors. Users can trust lower entities certified by these trusted institutions, forming a Chain of Trust.

PKI, standardized by X.509 [10], defines the commonly-used format of a digital certificate. Users keep self-signed certificates issued by Certificate Authorities (CAs) as trust anchors, and CAs sign certificates for subordinate entities using their private keys. If a CA's private key is compromised, all subordinate certificates become untrustworthy. There was an incident in the past, where public CAs operating on online servers were breached, resulting in significant damage [11]. Currently, CAs issuing public TLS certificates are required to comply with the Baseline Requirements (BR) specified by the CA/B Forum for operational safety [12]. The BR mandates that root CAs operate offline or in an air-gapped environment, with explicit human operation required for signing certificates.

DNS Security Extensions (DNSSEC) [13] are a set of protocols used to digitally sign and validate DNS zones. It establishes a chain of trust starting from the digest of the Root Key Signing Key (KSK) public key. The Root KSK is managed in an air-gapped environment by the Internet Assigned Numbers Authority (IANA). They have made all operations involving the Root KSK public to ensure transparency in internet operations [14], which serves as a leading example of critical key management.

The US government has published several documents in the field of key management. NIST SP 800-57, mentioned earlier, provides comprehensive guidance on key management [15]. It categorizes different types of cryptographic keys and provides guidance on their lifecycle management. NIST SP 800-130 [16], which is closely related to the previous one, focuses on the Cryptographic Key Management System (CKMS) and includes design requirements. There are also standards for cryptographic modules and digital signature algorithms: FIPS 140 [17] and FIPS 186 [18] mentioned later in Section V-C.

The IETF provides a decision guideline in RFC 4107 [19] to determine whether key management should be automated or manual. It strongly recommends an automated approach, except in rare situations where the encrypted data has low monetary value or where encryption frequency is low.

C. Context in Cryptocurrency and Blockchain

The history of cryptocurrency begins with the invention of Bitcoin [20]. In many cryptocurrency systems, a blockchain (or a distributed ledger) is used to record the associations between addresses and their cryptocurrency balances. An address is derived from a public key, and transactions are recorded on the blockchain to transfer cryptocurrency from one address to another. Each transaction requires a signature from the private signing key associated with the originating address(es). A mechanism to keep signing keys is often called a wallet.

Managing signing keys securely and accurately retrieving recipient addresses is crucial for cryptocurrency owners to successfully transfer funds. This technical landscape is similar to the era of PGP, and it is often described as decentralized. Some cryptocurrencies even offer enhanced anonymity to further promote decentralization [21].

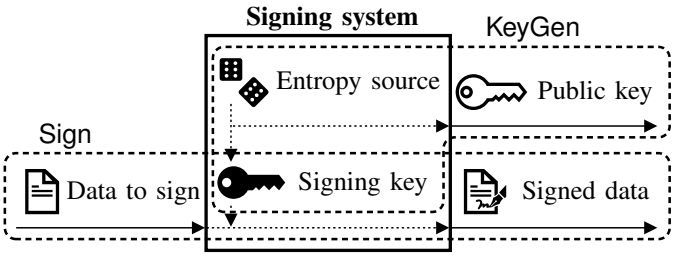

Fig. 1. Model of Signing System (SIG-SYS)

Eskandari et al. conducted one of the first analyses of personal key management in Bitcoin [3]. Bonneau et al. extensively examined the architecture of Bitcoin, including the challenges of key management, and discussed general research topics [22]. There are also other studies discussing wallets for cryptocurrencies in general [23], [24].

Secure key management in cryptocurrency custodians has been developed and improved over time, driven by major incidents. There have been cases where cryptocurrencies were unlawfully transferred due to private key leakage [25], [26], cases where signing keys were misappropriated to divert funds [27], [28], and cases where cryptocurrencies became permanently frozen because encrypted signing keys could no longer be decrypted [29]. However, it is important to note that new incidents continue to occur. Oosthoek et al. conducted a comprehensive review of past incidents [30].

D. Other Recent Studies

Many systems are operating in cloud infrastructures in recent years, including cryptocurrency exchanges. Chandramouli et al. discussed the challenges of cryptographic key management over virtual machines, storage, and databases in the cloud [31]. Similarly, Kuzminykh et al. compared and analyzed various KMS products [32]. These studies may be beneficial when designing an online wallet architecture.

Xiao et al. evaluated the security of complex authentication from a reliability engineering perspective [33]. They formulated the successful attack rate based on the Mean Time Between Failure (MTBF) of a single key. This approach may be extended to evaluate the security of a multi-signature wallet.

Rana et al. extensively examined the architecture of KMS for various types of cryptographic algorithms [34].

III. PROTECTING SIGNING SYSTEM

A. Model

We introduce an abstract model of a signing system, referred to as SIG-SYS, to discuss its security characteristics. As depicted in Figure 1, SIG-SYS holds the cryptographic key internally, which digitally signs data.

This model does not depend on any specific signing algorithms or curves, as long as the protocol itself is secure. It can accommodate algorithms like secp256k1 used in Bitcoin, or non-standard signing methods. It is important to note that even if the cryptographic keys are securely handled, potential flaws in the signing protocols can still lead to key leakage.

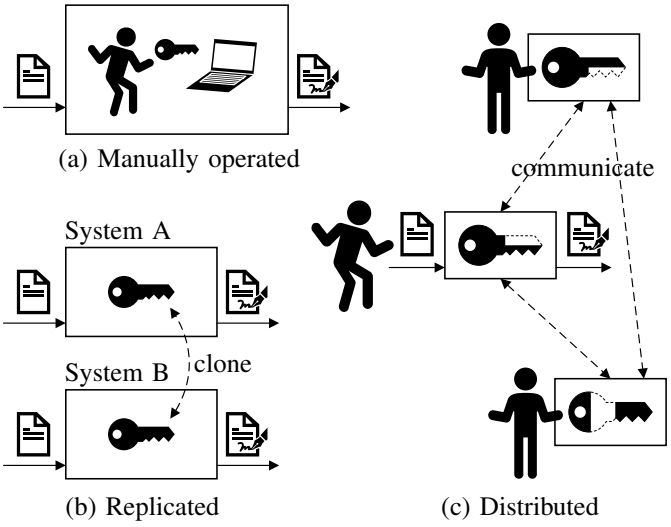

Fig. 2. Variants of SIG-SYS

From an external perspective, SIG-SYS can be viewed as a black box. It provides two interfaces:

- KeyGen generates a private signing key (referred to as sk) from the entropy source. It does not take any input. It outputs the corresponding public key (pk).
- Sign uses sk to sign the input and produces the signature. It may also perform additional verifications on the input.

To ensure the security of these interfaces, appropriate mechanisms should be implemented to authorize users or external systems. They can be integrated within SIG-SYS or implemented as additional security layers outside it.

In addition, SIG-SYS may define Destroy for destroying sk, which is not suitable for cryptocurrencies. To the best of the author's knowledge, there are no blockchains that allow the decommissioning of addresses. We do not recommend the destruction of sk and will not discuss it further in detail.

Figure 2 shows different variants of the SIG-SYS. As depicted in (a), SIG-SYS may not always be fully automated. In certain cases, such as an air-gapped system, manual or mechanical mechanisms may be required.

Furthermore, SIG-SYS may consist of multiple components rather than being monolithic. For example, one component may hold sk and transmit the encrypted version of it to another component for higher availability, as shown in (b). It is also possible to distribute sk across multiple terminals, as depicted in (c), using secret split techniques discussed in Section V-G.

B. Security Analysis using Formal Approaches

The following two requirements should be met:

1) Any bit of sk is kept within the SIG-SYS's boundary.
2) Data should only be signed by sk if and only if requested via Sign, and no other signature is generated.

These requirements can be verified using formal methods, especially for synchronous software implementations.

For distributed implementation with multiple processes, Lamport's method for proving the correctness of multi-process

software [35] can be employed. This involves demonstrating the following *safety* properties, which ensure that the system does not reach undesirable states:

- sk remains unchanged and is not lost.
- Any output from SIG-SYS does not contain any unencrypted bits of sk.
- The output from Sign corresponds to its input.

as well as the following *liveness* properties, which ensure that the system eventually reaches the desired states:

- sk will eventually be generated after KeyGen is called.
- A digital signature by sk will eventually be generated for a message m after Sign is called with the input m.

However, it is often challenging and costly to thoroughly formalize cryptographic processes.

C. Security Analysis using CIA Triad

As an alternative to formal methods, the security of an information system can be evaluated using the principles of confidentiality, integrity, and availability, commonly known as the CIA triad [36], [37]. Each represents a specific requirement: (*C*) ensuring that secret information does not leave the system's boundary, (*I*) preventing unauthorized modification of information and the system's processes, and (*A*) ensuring that the system can operate when necessary. For example, Warkentin et al. [38] examined the security of blockchain applications for the public sector using the CIA triad.

SIG-SYS can employ multiple layers of authentication and encryption to prevent the leakage or misuse of sk, enhancing (*C*), or sk can be duplicated for backup to avoid damage or loss, enhancing (*I*) and (*A*). These measures are in conflict with each other, and a rational balance needs to be maintained.

The following considerations should be taken into account:

1) Confidentiality of sk: Insufficient entropy or improper implementation of random number algorithms are common causes of vulnerabilities. Predictable random numbers can be exploited in attacks, where the seed value can be guessed. Such attacks on random number generators (RNGs) have been studied extensively on various operating systems [39], [40]. Side-channel attacks, which recover keys from electromagnetic radiation during signing, also exist for RNGs [41]. Weak keys for cryptocurrency wallets can result in the loss of funds, and there have been reported cases of successful attacks [42].

2) Integrity of SIG-SYS*:* Bits of sk may be embedded as steganography in the output of Sign if SIG-SYS is maliciously tampered with, including a malicious act of operators. This risk has been suggested by Guri as a method of stealing a cryptocurrency wallet from an air-gapped environment [43].

3) Integrity of KeyGen's Output pk: To prevent attackers from stealing funds by modifying the output of KeyGen within SIG-SYS's boundaries, the user must ensure the authenticity of pk. One can test a small amount withdrawal from the pk's address as the simplest countermeasure.

4) Integrity of Sign's Input m: Similar to the above, an attacker may tamper with the pre-signed data to redirect the cryptocurrency transfer. The user should not rely on SIG-SYS's integrity, and rather validate the output of Sign.

D. Threats

There are various threats that compromise the security of SIG-SYS. We can analyze threats by their origin:

1) External Threats to the System: In a hostile environment, attackers may try to generate fraudulent Sign requests by bypassing the authentication of SIG-SYS or by stealing the private key through logical or physical means. Natural disasters can also damage the availability of SIG-SYS.

2) Internal Threats to the System: Internal components of SIG-SYS may also be adversarial. If SIG-SYS is implemented in software or hardware, there may be intentional or unintentional defects introduced by the developers. If humans are involved in the process of SIG-SYS, it is important to mitigate risks of misconduct such as information leakage, denial of tasks, and destructive acts, as suggested by Blakley [2].

E. Security Considerations in the Use of Cryptography

1) Hidden Number Problem: In ECDSA [44], an attacker can calculate the private key from multiple signatures with the same nonce. This type of attack, known as the Hidden Number Problem [45], has been studied in other cryptographic algorithms as well. The nonce reuse problem in ECDSA is addressed in RFC 6979, which proposes a deterministic method to choose the nonce from the message to sign [46].

2) Noncanonical Encoding: Transaction malleability, as seen in Bitcoin, allows an attacker to produce different transactions while preserving the digital signature. One technique used for this type of attack is to exploit the fact that both (r, s) and $(r, -s)$ are equally valid ECDSA signatures. A group of attackers successfully forged transactions depositing Bitcoin into Mt.Gox [47]. Cryptocurrency exchanges need to perform strict validations and ensure the canonicality of such different instances of transactions or signatures.

IV. SPECIFICS TO CRYPTOCURRENCY EXCHANGES

A. Architecture and Operations

Most cryptocurrency exchanges facilitate the swap between legal currency and cryptocurrency, as well as process users' deposits and withdrawals. They manage wallets to keep customers' funds typically in their custody. They may also operate blockchain nodes to monitor transactions, as discussed later in Section V-A. There is an IEEE Standard about the modern cryptocurrency exchange's architecture [48].

Some exchange operators rely on external blockchain APIs to simplify their architecture. They need to ensure the accuracy of incoming information and the availability of the system, for example, by connecting to multiple APIs and data sources.

1) Deposit: Exchanges assign a unique deposit address to each customer upon request and monitor transfers to deposit addresses. The customer's account balance is updated accordingly once the deposits are confirmed.

Certain blockchains allow additional data to be attached to transactions, such as a message in Symbol, a tag in Ripple, and a memo in Stellar. Exchanges may specify unique data per customer while providing the same deposit address. This approach helps simplify the wallet structure and reduce

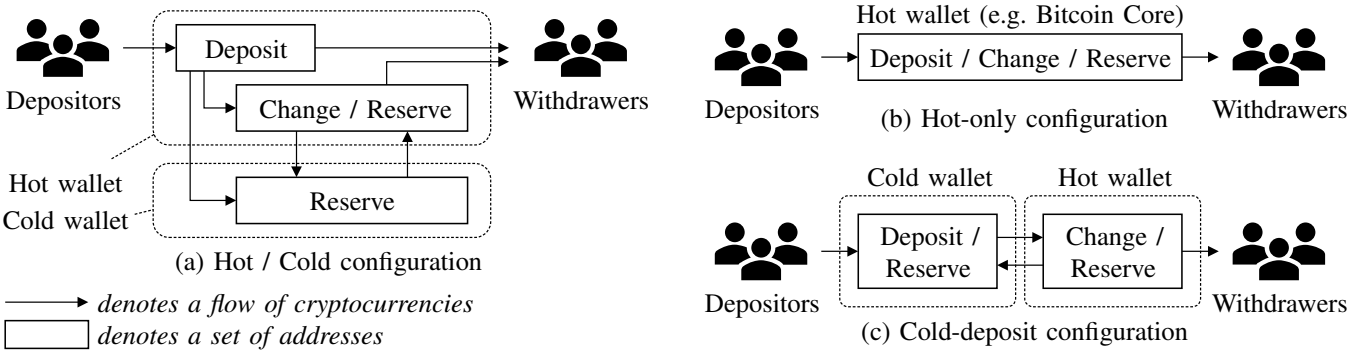

Fig. 3. Various Wallet Management Configurations

transaction costs. However, such information can also be used for tracking by third parties [49], and some exchanges offer multiple deposit addresses to address privacy concerns [50].

2) Withdrawal: When a customer requests a withdrawal, exchanges initiate an outgoing transaction from their wallet. It is common for exchanges to automate these withdrawals, but they are also required to screen for suspicious requests to prevent financial crimes, which may take several hours.

3) Wallet Management: The configuration of wallets in modern exchanges is depicted in Figure 3 (a). Hot wallets are online signing systems that are connected to other systems in exchanges. They are designed for immediate withdrawals. On the other hand, cold wallets are offline or air-gapped systems that aim to provide more secure storage. This configuration offers several advantages:

- Different security policies can be applied based on the *temperature* of the wallet.
- For Bitcoin-like cryptocurrencies, exchanges can save on transaction fees by optimizing the selection of unspent transaction outputs (UTXOs).

Figure 3 (b) illustrates the configuration of a Bitcoin exchange in its early days based on the author's experience. The exchange hosted its wallet using software and the wallet was connected online during blockchain synchronization.

As an alternative to the typical configuration, exchanges can use cold wallets for customer deposit addresses, as shown in Figure 3 (c). Since there is usually no immediate need to withdraw from these addresses, cold wallets provide a suitable option. However, exchanges may still need to prove their control over these addresses for compliance reasons [51].

It is important to carefully design the wallet management configuration as it can be difficult to change afterward.

B. Regulations

Cryptocurrency exchanges are regulated by authorities in many countries. Although the regulations differ according to jurisdiction, they cover various aspects of exchange operations.

One of them is the solvency requirement, also known as Proof of Reserve. This was introduced following the Mt.Gox incident, where they did not have enough Bitcoin to pay their customers. Dagher et al. proposed a method using Merkle Trees to provide proof of an exchange's balance [52]. Several exchanges have extended this method with zero-knowledge proof techniques to show their solvency [53], [54]. In the US, an accountants' organization has established auditing standards [55]. Some auditors even require exchanges to demonstrate the actual withdrawal to verify the possession of private keys.

Another regulation is the requirement for cold wallets. Similar to the air-gap practice of PKI, some countries mandate that the majority of a cryptocurrency exchange's assets be stored in cold wallets. For instance, in Japan, more than 95% of assets [56] must be managed in cold wallets. Cold wallets are defined there as electronic devices that have never been connected to the Internet [57]. This means that devices requiring online activation or attestation cannot be used as cold wallets in Japanese exchanges. This strict regulation played a crucial role in safeguarding the assets of Japanese customers during the collapse of FTX in 2022 [58].

Furthermore, additional requirements may be applied, such as the involvement of multiple individuals for initiating cryptocurrency transfers or creating backups for signing keys.

V. Analysis of Known Types of Wallets

A. Software Wallets

1) Overview: A software wallet is the simplest type of wallet implementation. It can generate and manage private keys on a computer and sign transactions. The idea of software wallets can be traced back to the original Bitcoin client written in C++ by Satoshi Nakamoto. Nowadays, there are various developers who provide software wallets that support different types of cryptocurrencies or tokens. Software wallets can come in different forms, such as desktop applications, browser extensions, and mobile apps for smartphones.

Software wallets are closely connected to blockchain nodes. In many blockchain systems, referencing block information is essential when creating withdrawal transactions. As a result, many blockchain clients combine both node and wallet functionalities, although they can still operate independently if needed. For example, Bitcoin Core can create and manage multiple wallets and disable wallet functionality for security purposes [59]. There are different types of nodes, each with different relationships and risks regarding wallet functionality.

A *full node* is the most common type of node. It maintains a complete history of all blocks and transactions, as well as the latest execution state, such as the UTXOs or the world state. It is capable of verifying the consistency of new transactions, making it essential for confirming deposits at exchanges. If fully synchronized, wallets can generate valid transactions. However, running a full node may require a significant amount of storage space. As an example, the Bitcoin blockchain has exceeded 512 GB as of December 2023 [60].

A *lightweight node* is suitable for implementing wallets on systems with limited capacity, such as smartphones. It retains only the recent blocks and transactions, discarding older information to fit within a certain storage capacity. However, this prevents the node from independently verifying the validity of new transactions, and a wallet could accept invalid transactions addressed to itself unless relying on a trusted full node. In Bitcoin-based cryptocurrencies, the Simple Payment Verification (SPV) protocol allows for the extraction of transactions related to specific monitored addresses from other full nodes using a Bloom filter. It retains only headers of all past blocks, and fetches blocks containing transactions of interest as necessary [20]. There are other distributed ledgers like Ripple, which do not require synchronization of past ledgers.

An *archive node* retains all past states in a blockchain architecture with complex state, such as Ethereum, while regular nodes retain only the most recent state. Archive nodes can restore past states and execute function calls on them, such as checking the owner of a specific NFT at a particular point in the past. Archive nodes are primarily used for data analysis and require high performance. For example, Ethereum's archive nodes require around 16TB of storage space as of December 2023 [61]. Considering their purpose, it is highly unlikely to operate wallet functionality on archive nodes.

There are other types of nodes with different network configurations or blockchain designs. For example, mining nodes specialize in creating blocks in proof-of-work blockchains, relaying nodes specialize in block propagation, and consensus nodes in Ethereum specialize in attesting and signing the verification results of newly proposed blocks. They are not directly related to wallet functionality.

Conversely, some wallet implementations, such as browser extensions or smartphone apps, may not be accompanied by nodes. Similar to lightweight nodes, those nodeless wallets rely on trusted external nodes via APIs to query cryptocurrency balances and broadcast transactions. For example, Khan et al. developed a nodeless Android wallet that simply uses QR codes for data exchange [62].

2) Advantages and Disadvantages: Using a software wallet offers several benefits: *(a)* users only require a regular computer, eliminating the need for any special devices, *(b)* software wallets typically have a fast update cycle, which allows custodians to effortlessly stay updated with new cryptocurrencies and tokens, the latest DeFi services, and technical updates of blockchains, and *(c)* any issues or malfunctions that arise can be swiftly addressed and resolved.

There are also drawbacks: *(d)* software wallets are primarily designed for individuals and may lack enterprise features, such as complex approval workflows, *(e)* they are often designed to be online and the communication is encrypted, making security control challenging, and *(f)* they are most susceptible to the security of running environments.

3) Risks: There are at least two categories of risks associated with software wallets: *(x)* caused by the software itself and *(y)* caused by the user's actions or behavior.

x-1) Malicious intent of the developer: The developer or committer may program the wallet to transfer users' cryptocurrency without consent or send private keys to external parties. They can achieve this by hiding backdoors in the codebase or through other open-source software used by the wallet.

x-2) Accidental malfunctioning: sk may become corrupted or disclosed to external parties due to unknown defects. This can happen due to classical bugs, cryptographic vulnerabilities, or other reasons.

y-1) Improper settings or incorrect usage: The potential consequences of errors in software wallet configuration should not be underestimated, as they can lead to exploitation by attackers. For instance, Bui et al. demonstrated unauthorized access to the API of the node without proper authentication settings [63]. Also, user interface issues may lead to incorrect manipulation of the wallet. Extensive research has been conducted by Voskobojnikov et al. on the usability concerns associated with software wallets [64].

y-2) Contaminated running environment: Attackers may use malware to directly access files or memory where secrets are stored or incorporate keylogging techniques. Horst et al. conducted an experiment to steal private keys by analyzing the memory of software wallet processes [65]. Volety et al. focused on similar attacks using brute force methods [66]. He et al. analyzed various attack vectors targeting Android wallets [67]. Other malware uses an attack called *address poisoning* [68], which exploits human negligence in fully verifying the displayed address. Ivanov et al. showcased malware that substitutes the user-copied destination address on the clipboard with the attacker's address that partially resembles the original one [69]. One possible countermeasure for wallet developers is the implementation of visually distinguishable representations, akin to the concept of SSH key art [70].

y-3) Using counterfeit software: It is often difficult for users to differentiate phishing software from the legitimate one, due to similar names or appearances. This is particularly prevalent in browser extension wallets [71].

To mitigate risks from *(x)* and *(y)*, one can consider using verified releases of open-source software wallets with a mature community of developers. To minimize the potential impact of unknown vulnerabilities, it is also advisable to perform code audits on the codebase and adopt a defense-in-depth approach by implementing multiple layers of protection.

B. Auxiliary Wallets

1) Overview: Several offline methods are known to keep wallet information outside of computers. These methods include writing the information on paper, which is referred to

as a *paper wallet*, engraving it on a metal plate, known as a *metal wallet*, or memorizing it, known as a *brain wallet*. We will refer to these methods collectively as auxiliary wallets, since there is no single term that encompasses all of them.

Auxiliary wallets can hold two types of information:

- Cryptocurrency address or public key
- Private key or information needed to derive it

The former is only used to display one's own address when receiving payments from others, and we will not discuss it further. The latter can be one of the following:

PK: private key in hexadecimal, Base58, or QR code
MNEMO: a sequence of 12 to 24 words (mnemonic)
PWD: password or passphrase
ENC: encrypted or sharded information of the above

2) Common Advantages and Disadvantages: There are common advantages shared by auxiliary wallets, regardless of the type of information they hold: *(a)* these wallets can serve as backups for software or hardware wallets, protecting against the risk of wallet loss due to defects or attacks, and *(b)* the risk of direct theft via the Internet is low.

However, *(c)* auxiliary wallets cannot be used standalone and must be loaded into other wallets for use. As a result, the auxiliary wallet shares the same risk profile as them.

3) Characteristics by Type of Memory: Physical methods like paper or metal wallets are highly durable and most commonly used for PK or MNEMO. QR codes have error correction capability, making them resilient to partial damage. Similarly, MNEMO can be restored even with the loss of a few characters. However, physical wallets require protection against theft or loss and have lower confidentiality due to the ease of recovery from accidental video recordings (e.g., [72]).

The brain wallet ensures security as long as the person who remembers it and their memory are safe. They are commonly used for MNEMO or PWD. While using a sufficiently secure password enhances the wallet's security, sharing it with others can be inconvenient. Additionally, it can be practically difficult to segregate the information among multiple individuals. For example, a malicious one may be able to recover the other half if simply splitting it in half. While these wallets excel in confidentiality compared to physical wallets, availability may be compromised. To complement this, Sans et al. suggested storing MNEMO on a special blockchain using the owner's email address as the lookup key [73]. This approach can be theoretically applied to other types of information as well, while it relies on the security of the mail system and the specific blockchain, making it less universally applicable.

4) Characteristics by Type of Information: Modern software wallets encode the seed using MNEMO, while initially PK was the primary option in the early days of Bitcoin. The transition was due to privacy concerns as reusing cryptocurrency addresses is generally discouraged [74]. BIP-32 defines an algorithm for a hierarchical deterministic (HD) wallet and can derive a series of signing keys and addresses deterministically from a seed using the HMAC algorithm [75].

Each word for MNEMO is selected from a pre-defined set of 2048 words and represents 11 bits of information, as defined in BIP-39 [76]. Excluding the checksum, a specific permutation of 12 words encodes 256 bits of data. The use of recognizable words is similar to concepts like PGP Word [77]. Although mnemonics can be easily memorized, there is a risk of theft if observed by individuals with exceptional instant memory.

Fig. 4. Example of an Encrypted Paper Wallet

PWD, in conjunction with the type of function and salt, can uniquely derive a wallet by a key derivation function such as PBKDF2 or scrypt. However, in reality, weak passwords have been commonly used, resulting in immediate attacks [78].

The ENC, example shown in Figure 4, is generated by encryption or secret sharing algorithms. This approach enhances protection against theft or loss, but necessitates the secure management of both the ciphertext and the password separately, resulting in an increased number of concerns.

C. Use of a Hardware Security Module

1) Overview: Devices equipped with secure cryptoprocessors are called Hardware Security Modules (HSMs). HSMs are implemented as self-contained units with well-defined interfaces, and they fit into the model depicted in Figure 1.

In the PKI industry, HSMs have been widely used for cryptographic key management. CAs operate HSMs in secure environments usually through closed air-gapped networks. HSMs are also utilized in the communication and financial sectors for high-performance online processing. Cloud vendors provide virtual HSM appliances to meet such demand.

Shbair et al. implemented key management for Ethereum using a cloud HSM [79]. Alrubei et al. employed an HSM for a private blockchain tailored for enterprise purposes [80]. There are also public cases where cryptocurrency custodians employ HSMs [81]. Dornseifer et al. provide a tutorial for signing Ethereum transactions using AWS KMS [82].

2) Advantages:

a) Tamper Resistance: When an HSM detects tampering attempts, including electrical analysis, the decryption key for the internal storage is promptly erased. The administrator can recover the decryption key from an external backup medium, only after verifying the integrity of the HSM [83]. This provides an equilibrium between confidentiality and availability.

b) Reliability: HSMs that have undergone security audit and examination can be considered reliable. The quality of HSMs is ensured through established standards such as

ISO/IEC 19790 [84], FIPS 140-3 [17], and the Common Criteria. These standards specify requirements from various aspects including interface, authentication, operating environments, and more. In the US, the Cryptographic Module Validation Program (CMVP) [85] verifies and certifies the implementation of cryptographic modules. While the primary purpose of CMVP is to approve the use of modules by government agencies, certified HSMs are also widely trusted and utilized for non-governmental purposes.

c) Resilience against Attacks: Although there are known attacks against HSMs, such as the method proposed by Bardou et al. for deducing keys from publicly accessible interfaces [86], the range of attack methods is relatively restricted. The requirement for a well-defined interface and physical resilience is a substantial deterrent to potential attackers.

3) Disadvantages:

d) Lack of Cryptographic Algorithm Support: Cryptocurrencies often use ECDSA on the secp256k1 curve or EdDSA on the Curve25519. There are few certified HSMs available that support these algorithms, primarily because those curves were approved by the US government in 2023 [18] after feedback from the industry [87]. Additionally, we could not find HSMs on the market that support sr25519 used in Polkadot, or BLS signatures [88] adopted in Ethereum's beacon chain. Other researchers also pointed out the limited availability of HSMs with fairly new cryptographic algorithms [89].

e) Lack of Mechanisms for Cryptocurrency: Modern wallets often use HD wallets to derive multiple cryptocurrency addresses, but few HSMs support it. Mnemonics as in BIP-39 are also rarely supported. This is because those cryptocurrency-specific features are currently out of the scope of HSM's standards. Some HSMs offer the ability to define custom mechanisms, but it is the user's responsibility to ensure that such customization does not undermine security [90].

f) Storage Capacity: HSMs are often constrained by their internal storage capacity, which is often tens of thousands of keys per unit in higher-grade HSMs. This limitation can pose practical challenges for custodians with multiple cryptocurrencies and seeds to manage a large number of users. Han et al. proposed a method to expand the storage capacity of HSMs by combining them with Intel SGX [91]. Utilizing HSMs in the cloud can also alleviate these constraints.

g) Cost: Enterprise-grade products often exceed ten thousand USD in price per unit.

D. Hardware Wallets

1) Overview: Hardware wallets are specialized devices designed for the secure storage of cryptocurrencies. These devices come in various types. Some basic hardware wallet models consist of a simple circuit with a microcontroller, while more advanced models have a similar structure to HSMs. Many products are designed to be used via a USB interface from a computer with vendor-provided software. Arapinis et al. provided a formalized model of hardware wallets [92].

One early example of a hardware wallet was proposed by Bamert et al. [93]. It communicates over Bluetooth with payment terminals, such as point-of-sale (POS) devices. The device receives an unsigned transaction from the payment terminal, signs the transaction, and returns it to the terminal. Another implementation, proposed by Rezaeighaleh et al. [94], is a smartcard wallet that can be used together with an Android smartphone. They also suggested a method for backing up the key between multiple cards using a classical key exchange technique. Furthermore, there are numerous commercial hardware wallet products available.

2) Advantages and Disadvantages: Most hardware wallets are designed to be user-friendly. The benefits of using them include: *(a)* the intuitive nature of storing the private key in a physical device, *(b)* ready-to-use software that many vendors provide, *(c)* specialized firmware for cryptocurrency use, and *(d)* faster firmware update cycles to keep up with the emergence of new cryptocurrencies. These benefits come from fewer design constraints for vendors compared to HSMs.

Drawbacks include *(e)* the requirement of trust in the vendor for the product's integrity, *(f)* the lack of enterprise features, similar to software wallets, and *(g)* the risk of complications in the segregation of signing power in an enterprise setup.

3) Risks: Hardware wallets share similar risks with software wallets. The following are particular observations:

x-1) Risks in Firmware and Circuitry: Hardware wallets may come with firmware that behaves differently from the user's expectation. One manufacturer of hardware wallets announced a key backup service and proposed an optional firmware update. This announcement raised concerns among users, as it implied the existence of a function to extract signing keys from the device, which was initially advertised as impossible [95]. The firmware's source code needs to be publicly available and audited to address these concerns, and preferably, the user should be able to program the user-built firmware. Verification of the hardware circuitry may be necessary to take the utmost precaution.

y-2) Susceptibility to Attacks: Unlike FIPS-certified HSMs, which adhere to widely accepted technical standards, the security of hardware wallets is not standardized, and their quality can vary. For example, several attempts were made on KeepKey and Trezor through electrical analysis and signal manipulation by different researchers [96]–[98]. Ledger Nano S, which internally uses a TEE (described in the next section), was examined by Volokitin, and they demonstrated the memory access across the protection boundary [99].

y-3) Risks in Procurement: Volokitin suggests that genuine hardware wallets can be compromised by preloading malicious code during shipping [99]. There have been instances where cryptocurrencies were stolen through counterfeit hardware wallets that resemble reputable products [100].

E. TEE-based Wallets

One approach to creating a wallet on a computer without relying on a dedicated cryptographic processor involves the utilization of a Trusted Execution Environment (TEE). By establishing a TEE on a regular CPU, the signing system can be built under this protection mechanism. Within the

TEE, verified software with digital signatures operates in logical isolation from other applications [101]. Depending on the implementation, the hardware memory is partitioned and encrypted. Some researchers investigated Arm TrustZone or Intel SGX for implementing wallets [102], [103].

Advantages of TEE-based wallets include *(a)* the security compared to software wallets, mitigating the risk of counterfeiting and vulnerabilities to some extent, and *(b)* their versatility, as they can be deployed on various platforms.

However, there are some disadvantages; *(c)* the cost of developing TEE-based software can be significant due to programming constraints, *(d)* users need to trust the processor vendor, similar to hardware wallets, and *(e)* certain security precautions need to be made as there have been reported instances of attacks [104]. Schaik et al. stated the ability to decrypt transactions on the Secret Network, a confidential blockchain platform [105].

F. Smart Contract Wallets

1) Overview: On some blockchains such as Ethereum [106], one can write a program known as a smart contract to directly manipulate cryptocurrencies. It can be used for managing assets automatically on behalf of the user.

One example is Safe Contracts [107], an open-source contract for on-chain treasury management. It allows users to specify various conditions for cryptocurrency withdrawals, such as the authorization settings of requesters, the definition of a maximum withdrawal limit, and the restriction of the withdrawal destination address. Similarly, Uniswap, a decentralized exchange (DEX), operates entirely through a smart contract governing the deposit and withdrawal processes. This design minimizes human intervention.

Vitalik et al. have proposed the idea of social recovery wallets [108], designed with individual use in mind. This mechanism allows a specific signing key to be used for regular withdrawals, while a pre-registered set of other keys, called guardians, can reset to a different key on behalf of the wallet owner in case of an emergency. This concept can also be extended to wallet contracts used by custodians.

2) Risks: A reentrancy bug in The DAO, one of the earliest DeFi projects, allowed attackers to drain funds in 2016 [109]. The following year, a vulnerability in Parity, a smart contract wallet widely used by individuals, caused a freeze on funds. These cases highlight the significant risk caused by implementation errors in smart contracts. Praitheeshan et al. conducted a systematic analysis of smart contract vulnerabilities [110].

Given the inherent risks associated with contract defects, it is crucial to ensure the safety of smart contract wallets [111]. Bhargavan et al. proposed the use of F* for the formal proof of smart contracts [112]. Jiang et al. introduced a tool called ContractFuzzer for automated vulnerability discovery [113]. He et al. emphasized the importance of code audit by employing tools such as Oyente and Mythril [114]. Praitheeshan et al. used these tools to evaluate smart contracts used by custodians [115]. Cassez et al. formally evaluated the Ethereum virtual machine using Dafny [116].

Nevertheless, key management remains essential to initiate withdrawals from the smart contract wallets. They do not eliminate the necessity of key management.

G. Splitting Signing Power

Various techniques have been developed to partition the signing key into multiple components, enabling the distribution of signing power among multiple individuals, either through cryptographic or systematic mechanisms. The generalized approach requires M participants out of N shares.

When implemented correctly, these techniques offer redundancy of signers and prevent a few individuals from having complete control over the signing power. However, as discussed in Section III-C, setting $N = M$ increases the risks against availability, and having N significantly larger than M does against confidentiality.

The following are the notable techniques to split the power:

1) Multi-Signatures: In Bitcoin-based cryptocurrencies, one can create M-of-N multi-signature addresses. Another cryptocurrency, Flow, allows users to register any number of public keys for an account with weights. Withdrawals from Flow accounts can only be made if accompanied by digital signatures that exceed a certain weight threshold [117]. These are implemented natively on the blockchain in a systematic way to simply verify M signatures but have the drawback to users where the transaction size increases linearly with M. Multi-signature is not universally available in key management since Ethereum and some others do not natively support it.

2) Secret Sharing: Shamir proposed a method to split a secret into multiple shares [6]. This method requires a trusted party, as it divides the complete secret during the initial splitting and combines the M shares to recover the secret. The trusted party must be safeguarded when used in SIG-SYS.

There is also a variant that does not rely on a trusted party. Pedersen introduced Distributed Key Generation (DKG) in the construction of Verifiable Secret Sharing schemes, where participants can maintain their secret shares while proving the integrity of the combined secret [118].

3) Threshold Signature Scheme: Bitcoin has introduced the Schnorr signature [119], allowing for the non-interactive combination of multiple independent digital signatures without increasing the signature size [120]. A multi-signature scheme built on this has also been proposed [121], [122], although it is limited to a configuration where all participants must sign.

There are several other researches on interactive threshold signature schemes that do not rely on a trusted third party, extending DKG [123]–[125]. These multi-party computation schemes are suitable for scenarios where key shares are generated and managed individually by different officers. However, those known methods require multiple rounds of communication among signers to create a single signature. Additionally, there have been frequent studies on attack methods, such as [126]. Some implementations were reported to be vulnerable due to insufficient verification [127]. We think that further examination is necessary to adopt these techniques in cryptocurrency custodians.

H. Custodial Wallet

Custodial wallets involve the delegation of signing keys or key shares to a third party. For individuals, this typically entails depositing cryptocurrency into their personal account at a cryptocurrency exchange. Likewise, exchanges and institutional investors may entrust their key management to custodial service providers instead of handling the keys themselves.

In custodial wallets, the security of signing keys relies on the goodwill of the service provider. Antonopoulos argues that relying on custodial services to store cryptocurrency introduces a risk that is beyond the control of individual users [128]. He advocates for the use of hardware wallets, which emphasize the importance of owning the private keys.

Enterprise users of custodial wallets typically evaluate the risk based on the financial statements and SOC2 reports of the service providers. However, they may lack a comprehensive understanding of the detailed security measures. To mitigate this opaque risk, enterprise users may seek legal remedies such as insurance coverage for the funds held in custody.

VI. Evaluation Criteria of Signing Systems

Cryptocurrency custodians have the flexibility to combine various wallet techniques discussed earlier to build their own signing system. We provide examples of evaluation criteria that encompass common tasks performed by custodians.

A. Withdrawal

Evaluating withdrawal scenarios is critical, as it is likely to be the most frequently performed task. The requirements may vary depending on the wallet configuration or business needs. The following are typical scenarios for withdrawals.

- *Scheduled withdrawals* conducted on a regular basis from a cold wallet to a hot wallet, or the other way around. This process maintains a certain reserve in the hot wallet, ensuring that automatic withdrawals can be made to users.
- *Unplanned withdrawals* conducted when the balance in the hot wallet falls below a predetermined threshold, usually during times of cryptocurrency price instability.

The following enumerate some perspectives for evaluation:

1) Security: The potential attack vectors should be examined, including through defined interfaces or physical threats.

2) Turnaround time: The total amount of time required for a single signing attempt should be assessed. Cold wallets generally take longer turnaround times.

3) Scalability: The capacity, which refers to the number of keys that a system can hold, as well as performance, which refers to the number of signing attempts that can be processed per unit of time, should be measured.

4) Cost: Custodians may aim to reduce the ongoing expenses related to equipment and maintenance, while usually justifiable at a certain level of initial investment.

B. Prediction and Response to Abnormality

Even with sufficient measures and risk control in place, system failures or security breaches can still occur. It is important to predict such signs and evaluate the impact, as well as estimate the time and cost required for recovery.

1) Ease of Incident Detection: The architecture of the signing system should be evaluated for its ability to detect abnormalities. For example, a hot wallet can have a separate system to monitor the outputs and automatically disconnect it if needed, while a cold wallet can have more basic techniques like surveillance cameras.

2) Complexity of Emergency Response: The signing system can have a panic button, even if a significant portion of the assets were to be drained under the attacker gaining control of the key. There might be a possibility to minimize the damage by promptly and securely evacuating the remaining assets using uncompromised keys.

C. Updating the System

Blockchain systems frequently undergo updates to their protocols, often in the form of soft forks to ensure compatibility. Some notable examples include Segregated Witness in Bitcoin [129] and the updates to transaction fee specifications in Ethereum [130]. These updates involve changes to the transaction format. Another significant update was The Merge in Ethereum, which had a substantial impact on the blockchain architecture [131]. The Nem to Symbol migration also falls into this category. On the other hand, the Taproot update in Bitcoin did not immediately affect the withdrawal process [132], and various other updates occur regularly.

Updating software and hardware firmware is important for addressing vulnerabilities and adopting the latest specifications. However, it should be done cautiously to avoid compromising system security. Custodians should consider postponing updates unless security is immediately at risk. This helps minimize vulnerabilities introduced during the update.

1) Frequency: The frequency of mandatory system updates should be anticipated while designing wallets. This includes operating system or firmware updates for hot wallets and facility maintenance for cold wallets.

2) Ease of Updates: The complexity and safety of the update process should be evaluated, which includes verifying the integrity of the software or establishing a clear procedure.

3) Vendor Dependency: Custodians should assess whether it is acceptable to be locked into a specific vendor due to unique features offered by certain wallet products. This evaluation should consider trust, cost, support, and others.

VII. Implementation Example: Extreme-Cold

A. Overview

In this section, we will introduce a reference implementation of a cold wallet called Extreme-Cold. This implementation is based on the background and security considerations discussed so far. The main objective of this demonstration is to ensure maximum transparency in risk management.

To fully explore the concepts discussed in the first half of this paper, we assume a complex environment and a scenario with human involvement, as depicted in Figure 2 (a). While there are similarities to the Faraday cage method employed by a US exchange mentioned by Simonite [133], no specific details were provided in his article. The method described

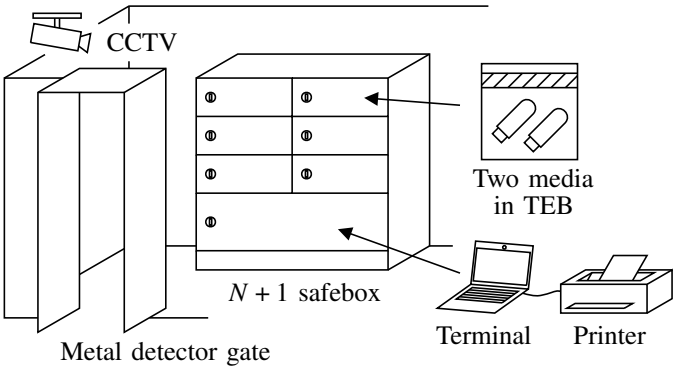

Fig. 5. Example of KMF Setup

here is based on the author's own considerations. Previous studies have explored the extraction of private keys from air-gapped environments [43], [134], and we will also discuss countermeasures against such attacks.

B. Facilities and Equipment

1) Key Management Facility: To effectively manage the signing keys, a physically secure room is essential, namely key management facility (KMF). Figure 5 illustrates the example configuration. It is necessary to establish two or more KMFs in multiple geographically separate facilities due to the potential impact of disasters, as well as the possibility of destructive behavior by staff members within a KMF.

It is crucial to take appropriate physical and logical security measures for KMFs. For example, the US federal government has established structural requirements for high-security facilities [135]. We compiled a checklist for KMFs in Table I.

A KMF needs to equip a locker with $N + 1$ or more compartments. Among them, N compartments are used to securely store N secret shares, which are recorded on separate storage media. The other compartment, preferably a larger one, is used to store an operating terminal and a printer. The key to each compartment with a secret share should be solely held by a corresponding treasurer, while the key to the compartment with the devices can be duplicated among all treasurers.

To mitigate the risk of side-channel attacks and unauthorized data extraction, it is recommended to enforce a policy that prohibits the presence of personal electronic devices. This policy can be enforced through the use of metal detectors.

2) Operating Terminal and Printer: In each KMF, a trusted computer, an *operating terminal*, and a printer need to be equipped for handling the private key. These devices can be any commonly available products, but they must be procured through a secure supply chain. Upon preparation, certain components of the operating terminal, such as the battery, storage, communication modules, and audio devices, must be removed. This eliminates the risk of side-channel attacks and removes the capability to persist data across multiple signing sessions. The exchange of information between the operating terminal and outside the KMF is accomplished through QR codes that are printed on paper. Therefore, a built-in camera is required for the operating terminal.

TABLE I
SECURITY CONSIDERATIONS OF KMF

Physical perspectives	Fire	Does the KMF have a gaseous fire suppression system?
	Water damage	Does the KMF locate on a higher floor to prevent flooding? Is waterproofing treatment made against leaks?
	Structure	Does the KMF meet building code against seismic or hurricane damage?
	Power supply	Are there emergency power sources for access control, monitoring, and lighting? Is the lightning protection in place for electrical system?
	Interior elements	Is the KMF finished by non-combustible and robust materials that can withstand destructive acts?
	Windows	Is the KMF free of windows? Are there wire mesh or other physical blockade in place for any openings?
	Access routes	Is there only one normal access route to the KMF? Are doors and emergency exits designed to be unlocked only from the inside during emergencies?
	Signs	Are there no visible indications of the KMF inside or outside the building?
	Alarms	Does the building have an alarm system for emergencies?
Logical perspectives	Authentication	Are all persons entering the KMF authenticated by multiple factors such as ID cards, passcodes or biometric? Are records retained for a certain period?
	Access control	Is there a technical control in place to prohibit single occupancy in the KMF?
	Surveillance	Can the KMF be monitored by cameras, incuding during power outages? Is the video footage retained?
	Metal detection	Is there a metal detector to prevent unauthorized bringing in or taking out of electronic equipment?

The operating terminal executes programs for KeyGen and Sign. These programs can be implemented in any arbitrary way. We provide our implementation for reference [136], which is a custom Linux image with a key management program. IANA has also been developing COEN [137] in the same manner to operate the HSM from the offline terminal.

The integrity of these devices is critical. Therefore, the devices must be securely stored in lockers to prevent tampering while not in use, and the hash value of the running software must be verified when operating.

3) Storage Media: A private key is divided into N shares in our approach. Each of those shares corresponds to a different treasurer one-to-one. Each share is recorded on a storage medium, e.g., a USB memory stick, and duplicated into at least two copies for redundancy. These copies are then securely placed in a tamper-evident bag (TEB) and stored in the corresponding treasurer's compartment when not in use. Additionally, an exact duplicate set of TEBs is prepared for each KMF as a backup.

C. Human Involvement

Our approach requires the participation of M out of N treasurers to conduct the signing process. Each treasurer is accountable for securing their specific portion of the key, which is stored in individual compartments among every KMF. Such a control policy that requires multiple persons for a process, commonly referred to as two-person integrity (or more), is not only adopted by US government agencies but also by other high-security facilities.

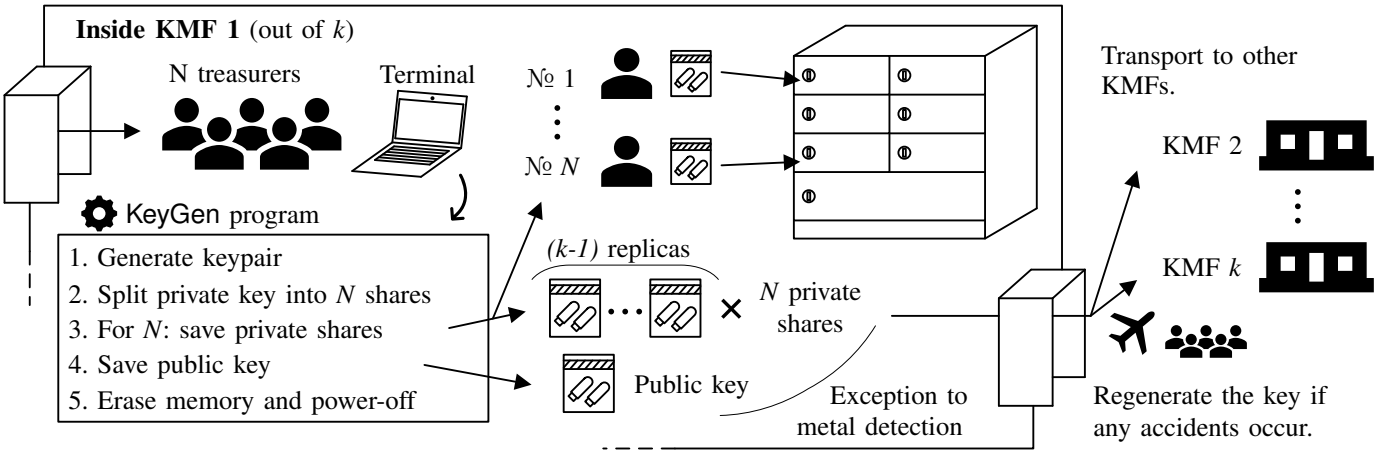

Fig. 6. Outline of KeyGen process

TABLE II
EXAMPLES OF FRAUD TRIANGLE

Motivation	• Greed to a large amount of cryptocurrency. • Discontent or resentment towards the company.
Opportunity	The physical presence to the operating terminal with the signing key loaded.
Justification	• "My salary is not high. I shouldn't be blamed for a procedural error." • "I am not treated fairly by the management. They deserve what they get."

1) Prevention of Fraudulent Behavior and Collusion: Cressey identified the conditions for fraudulent behavior, some of which include the motivation to commit fraud, the opportunity to commit fraud, and the presence of rationalization or justification [138]. These are now widely known as the fraud triangle [139]. Table II shows examples of elements of fraud that can occur in a KMF.

When the likelihood p of an individual attempting fraud is uniform across all members, one can calculate the probability that M out of N members collude at a specific point in time as a simple binomial probability shown in (1). This is useful for quantifying the potential fraud risk.

$$F = \binom{N}{M} p^M (1-p)^{N-M} \tag{1}$$

In reality, it may be infeasible to assume that p is equal to all treasurers due to environmental reasons, such as professional or personal relationships, salary disparities, and hierarchical differences. A sufficient M may be alternatively decided based on the number of approvers required for legal currency disbursement, for instance.

2) Prevention of Operational Errors: As a separate perspective from intentional misbehavior, there is a possibility that humans may make mistakes due to unintentional errors when performing tasks. Additionally, there is a possibility of forgetting procedures due to the psychological pressure of handling important equipment. Therefore, we have prepared a work progress checklist.

D. KeyGen – Key Generation

As depicted in Figure 6, all N treasurers gather in one KMF to generate a private signing key and split it into N shares on the operating terminal. Once all the shares are physically transported to the other KMF(s), KeyGen is considered complete.

1) Generating Random Number: As stated in Section III-A, the entropy source used to generate a private signing key should ideally be completely random. We adopted OpenSSL's `CTR_DRBG` implementation, which is one of the deterministic random bit generation algorithms that conforms to NIST SP 800-90A, hence FIPS 140-2. Other options include using a true random number generator device or a stream cipher.

Regarding `CTR_DRBG`, while Campagna illustrated a vulnerability where the expected randomness is not achieved when the key length is other than 112 bits [140], it can be resolved fairly easily by initializing the algorithm with new seeds until reaching the desired length since most cryptocurrencies use signing keys with no more than a few hundred bits. Woodage et al. discussed vulnerabilities when the internal state of RNGs can be observed and examined the OpenSSL implementation [141]. However, this will not be a major issue, considering that KeyGen in our method is done on the air-gapped operating terminal in the KMF. Strenzke also examined OpenSSL's random algorithm implementation [142].

2) Sharding the Secret: We split the signing key using Shamir's method, which allows us to divide the key in a way that is not dependent on the type of blockchain or the signing algorithm being used. The benefit of this approach is the simplicity of the algorithm, hence a lower risk of implementation errors. There is no need for VSS or MPC since we trust the operating terminal's integrity.

3) Distribution of the Private Signing Key: All treasurers transport the signing key to each KMF(s). Once all KMFs have successfully equipped the generated key in their respective compartments without any issues, the corresponding cryptocurrency address can be considered ready for use. Any risk of transportation accidents is discussed in Section VIII-A3.

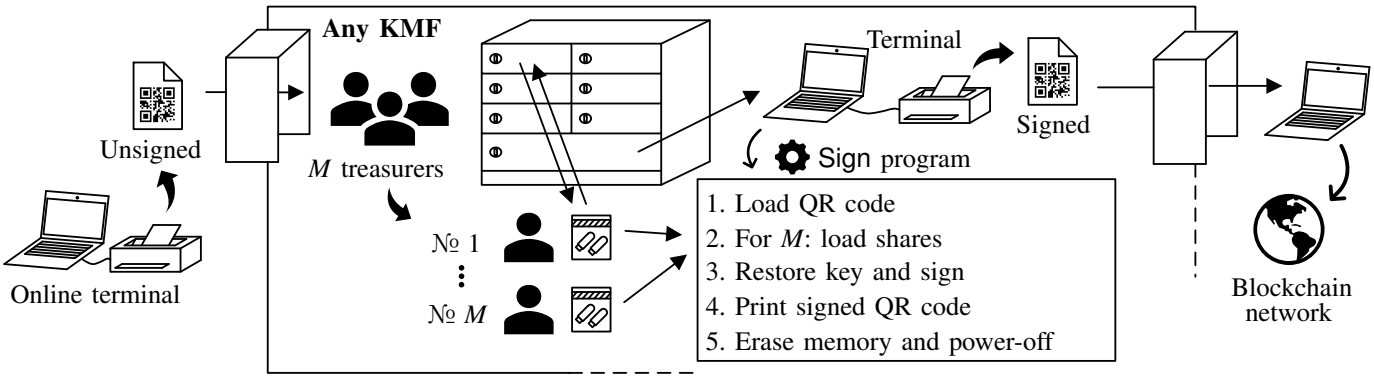

Fig. 7. Outline of Sign process

E. Sign – Signing Transactions

As depicted in Figure 7, the treasurers bring the unsigned transaction into the KMF to conduct a withdrawal. This serves as the input for Sign. The signing process takes place on the operating terminal under the supervision of multiple treasurers. Once completed, the signed transaction is taken outside with the treasurers and broadcasted to the blockchain network.

1) Preparing the Unsigned Transaction Outside KMF: One of the treasurers prepares the withdrawal transaction to be signed outside the KMF using a fully-synchronized online blockchain node. The transaction is converted into a QR code and printed on paper, preferably along with human-readable details for verification by other treasurers.

2) Entering the KMF: Any combination of M treasurers enter the KMF to sign the transaction together. Entry and exit to the KMF are prohibited throughout the entire process to prevent unauthorized actions, such as tampering with QR codes or taking out the running operating terminal with the signing key loaded.

3) Signing Process: The treasurers use an operating terminal to sign the transaction.

1) Scan the QR code and decode the transaction.
2) Verify the amount and destination of the transfer.
3) Restore the signing key from M key shards.
4) Sign the transaction.
5) Encode the signed transaction into a QR code.
6) Print the QR code on a sheet of paper.
7) Power off the operating terminal.

In step 2 above, the treasurers must ensure that the encoded data in the QR code exactly matches the expected withdrawal details, i.e., the destination and the amount, without any additional metadata or information. In step 3, each of the M treasurers retrieves their respective key shard, loads it into the operating terminal, returns the shard to its compartment, and securely locks it.

4) Exiting the KMF and Broadcasting the Transaction: After the printed QR code is carried out from the KMF, one of the treasurers scans and loads it to an online blockchain node, and broadcasts it to the blockchain network. The sheet with the QR code may be disposed of.

F. Response to Abnormal Situations

1) Failure of a Single Medium: If a treasurer's medium is faulty, the treasurer can duplicate the other medium in the same compartment for recovery. The faulty medium should be completely destroyed. If both media are faulty, the shard needs to be recreated by using other treasurers' media, similar to the KeyGen procedure. Any further failures than $N - M$ sets should be treated as equivalent to the next case, where the key is unrecoverable and a new key needs to be generated.

2) Disaster Affecting KMF: If more than $N - M$ sets of treasurer keys are damaged, it can be assumed that the key in the affected KMF is essentially lost.

The safest approach is to establish a new KMF, either temporarily or permanently, and generate a new key by the KeyGen process. The funds in the impacted address should be transferred to the new address to minimize any remaining risk. Additionally, stakeholders who know the old address, such as liquidity providers, should be notified of the new address.

Alternatively, the integrity of the impacted KMF can be restored, and the key can be duplicated from another reliable KMF. The transportation of the live operational key will be discussed in the following section.

3) Loss or Theft of Media: If any media are unexpectedly lost, regardless of the number of affected shards, the key should be considered compromised. In such cases, it is important to generate a new signing key. The recovery procedure does not differ from the previous case.

G. Alternative Implementations and Comparison

1) Using Simpler KMF: Building a robust KMF can be a challenging task and may not be feasible in some cases. Despite the theoretical increase in vulnerability to side-channel attacks or even the risk of physical brute-force intrusions, it may still be possible to utilize lightweight facilities if proper alternative controls are implemented.

It is particularly crucial to consider measures to prevent theft. In the original method, the key was stored on a medium and was not bound to any cryptoprocessors, unlike the security offered by an HSM or a hardware wallet. With those devices, the key would remain protected even if the device were stolen, ensuring that it does not become compromised immediately.

2) Using an HSM: There are a few general-purpose HSMs available on the market that support cryptocurrencies. When using such HSMs, the seed for key derivation is generated and stored internally within the HSM, and the signing key is ephemerally derived on demand. The benefits of using HSMs in EXTREME-COLD are that the physical operation can be simplified and there is no longer a need for $N + 1$ lockers; instead, only a safe to store the HSM is required. Furthermore, the KeyGen and Sign software only need to send commands to the HSM using Cryptoki in PKCS #11 [143].

However, using HSMs does not necessarily simplify human involvement. FIPS-140 defines Security Levels, and typically Level 3 or Level 4 is used for high-security environments, which require strict authentication to access the protected storage of the HSM. In an M-of-N authentication configuration, the protected storage of the HSM is encrypted with the keys that are distributed and stored in smartcards or dedicated tokens. This process is no different from our approach and presents similar challenges when personnel changes occur.

In a real-world example, an HSM certified as Level 4 is used in Root KSK management. The decryption key to the Root KSK on the HSM is divided into multiple smartcards held by Crypto Officers (COs) using a 3-of-7 configuration. Every time an HSM is operated or any CO is replaced, three or more COs travel to the facility.

3) Use of a hardware wallet: Instead of using storage media or HSMs, an alternative option is to utilize hardware wallets, accepting the risks discussed in the earlier section.

Most hardware wallet products generate an HD wallet using a mnemonic phrase, and they often provide a function to display the mnemonics for backup purposes. To align with this product design, a custodian should choose a 24-word mnemonic phrase to minimize the risk of any treasurer remembering it. The mnemonic phrase should then be backed up as auxiliary wallets, preferably on physical materials. Treasurers should ensure that the wallet remains out of the view of CCTV cameras. It is also worth noting that implementing M-of-N authentication can be technically challenging, especially since hardware wallets are typically protected by a single password. $N + 1$ lockers may be unsuitable as well.

In terms of scalability, many hardware wallet products have limitations on the number of cryptocurrencies that can be supported by a single unit due to capacity constraints. Therefore, if a custodian intends to host many keys across multiple blockchains, they should consider procuring multiple units to distribute the keys accordingly.

4) Storage Media options: Appropriate storage media for storing the key may need to be procured based on the organization's choice. To enhance confidentiality, one can use removable media with hardware encryption, as demonstrated in [144]. For higher durability, one can use industrial-grade devices with more stable memory cells. To maintain firm integrity, one can use write-once memory, such as the one described in [145]. Unfortunately, during our investigation, we could not find any products that possess all these features. Certain compromises may be necessary in reality.

5) KeyGen and Sign program on TEE: In our approach, we implemented software that runs as a regular program on the operating terminal. However, it is important to note that the key in its complete form will exist in memory during the operation as we utilized Shamir's secret sharing scheme. This introduces a security risk if a treasurer were to take such a terminal outside of the KMF with the power still on. Implementing the software to operate in a TEE may help reduce such risk of extracting the key from the running terminal, although the development cost may increase.

VIII. EVALUATION OF EXTREME-COLD

A. Security

We are confident that EXTREME-COLD has effectively mitigated various known attacks, including side channels that exploit visual, electromagnetic, or audio vulnerabilities, which previous studies have identified as potential attack vectors. In the following paragraphs, we will discuss the additional considerations and the reasoning behind our approach.

1) Metal detection: The implementation of a metal detection gate serves as a preventive measure against the inadvertent or deliberate introduction or removal of items to or from the KMF. For example, smartphones, which can be used for side-channel attacks, and removable media, which can be swapped with genuine ones in the compartment, should be prohibited.

There are some exceptions. Physical keys of the lockers, unless they are dial locks, and media of software and keys in transit should be allowed. For the latter, the media should be securely enclosed in a plastic case and placed inside the TEB. This is to prevent the use of metal probes to read the media without damaging the TEB.

2) Treasurer's Misconduct: KeyGen is performed by N treasurers, and Sign is performed by M treasurers together. There is still a possibility that one of them may engage in misconduct during the process.

In order to maintain the confidentiality of the signing key, it is recommended to start the KeyGen process from the beginning if any misconduct is suspected. Similarly, in the case of Sign, a responsible treasurer should forcefully shut down the operating terminal to clear the loaded key from memory.

3) Physical Transportation of the Key during KeyGen: To mitigate the risk of accidents during transit, such as loss or unauthorized access outside the KMF, treasurers should ensure the integrity of all key shares in the TEB at each KMF before putting the key into operation. Treasurers at every KMF are required to verify the absence of any defects in the TEBs by cross-checking with one another. In the event that any defects are detected, the key must be regenerated from scratch. This transportation method for critical keys, yet naive, is also employed in the management of Root KSK in DNSSEC [14].

4) Physical Transportation of the Operational Key after Anomaly: In principle, it is not safe to transport the operational key outside the KMF, yet the well-established method of key distribution can be used to duplicate the key from the operational KMF_{src} to the damaged KMF_{dst}.

After the safety of $\mathrm{KMF}_{\mathrm{dst}}$ is confirmed, M treasurers together can perform the following method:

1) Generate an encryption keypair (ek, dk) in $\mathrm{KMF}_{\mathrm{dst}}$.
2) To prevent the substitution of ek, have multiple treasurers transport it from $\mathrm{KMF}_{\mathrm{dst}}$ to $\mathrm{KMF}_{\mathrm{src}}$.
3) Once the integrity of ek is confirmed in $\mathrm{KMF}_{\mathrm{src}}$, encrypt sk with ek into a cipher c on the terminal.
4) Safely transport c back to $\mathrm{KMF}_{\mathrm{dst}}$ and decrypt with dk. Reshard the resulting sk to an M-of-N configuration, and store them in the compartments.

This method of key transportation is similar to the replication of keys between HSMs, where the destination HSM generates a wrapping key to create a secure transport channel between HSMs. Similar to Diffie-Hellman key exchange, it is vulnerable to man-in-the-middle attacks, so the integrity of ek must be independently ensured.

5) Exfiltration of Key Information by Malicious Software: Naor et al. proposed a method of performing secret sharing using visual methods [146]. As an application of this, steganography using single or multiple QR codes has been studied to embed confidential information [147], [148]. It is challenging to differentiate QR codes with hidden information from regular ones, as the only indication is the presence of more error cells in the resulting QR codes. It is critical to ensure the integrity of QR-code processing algorithms to prevent the embedding of private keys during the generation and printing of signed transactions.

6) CCTV Recorded Footage: Despite having strict physical security measures in place for KMF, there is still an exploitable information channel in the form of the surveillance camera. The surveillance camera footage is crucial as evidence for investigating any potential misconduct, so it cannot be excluded from the requirements. To ensure the secrecy of the keyboard input for the KMF, we have implemented a screen filter on the operation terminal and a cover to hide the keyboard input.

B. Turn-around time

We have built a mockup facility and conducted an experiment. Our setup targeted the Bitcoin Testnet with a configuration of three treasurers threshold out of four ($N = 4$, $M = 3$).

We have learned from the experiment that KeyGen is quite time-consuming. On average around three attempts, it took about 50 minutes to generate a signing key in the first KMF, and about 35 minutes to store it in the second KMF (we reused the same room for budget reason) except the transportation. We observed several factors for this long duration, such as *(1)* the necessity to rigorously verify the integrity of the media and the key, and *(2)* the mental pressure on the participants of performing important but unfamiliar tasks.

In terms of Sign, out of 15 attempts, the first two attempts to sign a single transaction took over an hour from entry to exit at the KMF. However, as the participants got familiar with the process, they were able to finish it in 30 minutes for the last six attempts. We consider this acceptable as cold wallet operability in normal cryptocurrency exchanges.

C. Scalability

In EXTREME-COLD, an HD wallet can be used to generate multiple cold addresses from a single private key, although we have not experimented. It is technically possible to store multiple keys on a single medium, allowing the management of various addresses from different seeds and blockchains.

In terms of signing performance, this method allows for the simultaneous signing of multiple transactions in batches. One way to streamline the process is by sequentially scanning QR codes that encode transaction data. However, if frequent withdrawals from cold storage are necessary, it may be worth considering the much use of a hot wallet.

D. Cost

The initial setup cost of the equipment, including computers, printers, storage media, and CCTVs, amounted to approximately 3,000 USD in our experiment. These devices typically have a long lifespan, which makes the cost justifiable.

The facility rent for the KMF is a major recurring expense that greatly impacts the overall cost of this design. If the KMF is implemented as a colocation in a data center, the monthly cost can vary from a few thousand dollars. One way to reduce the cost is by using typical racks as lockers. Another option is to utilize an office room, which we have chosen.

E. Ease of Predicting Fraud

The KMF facility may be equipped with traditional intrusion prevention measures like door sensors and CCTV, which can promptly identify intruders in real-time. These measures also serve as an audit trail in case of any fraudulent activities.

Additionally, this approach incorporates cross-checking behavior into various operational procedures, such as sealing media in TEBs, storing them in compartments, and using metal detectors. This ensures that the responsible treasurer can easily detect any inappropriate behavior or misconduct.

F. Updates

1) Replacement of Treasurers (Changing N): Staff replacements are a common occurrence in organizations. When adding a new treasurer, the existing M treasurers are required to restore the key at the KMF and create a new shard for the new treasurer. On the other hand, if a treasurer retires and the number of treasurers decreases, but not below M, simply destroying the contents of the retiree's compartment will suffice. These actions need to be carried out in each KMF. In cases of staff replacement, the process can be as simple as the departing person handing over the locker's key or dial combination to the new one, though the verification of the contents of the locker is encouraged.

2) Change of Threshold (Changing M): Changing the required number of treasurers from M_{old} to M_{new} is more complicated due to the properties of Shamir's secret sharing. To implement this change, all N people need to be present, restore the keys with M_{old} people, and then recreate shards so that the threshold becomes M_{new}. Additionally, all N old media in each compartment must be destroyed. The same should be done for all KMFs, respectively.

3) Addition of Signing Keys: KeyGen must be done every time when adding a new signing key, for example, for a new wallet, a new cryptocurrency, or a new blockchain. This process does not impact the existing keys.

4) Software Updates: The potential impact caused by protocol updates on cryptocurrencies or blockchains is straightforward. The organization needs to *(1)* create updated software for KeyGen or Sign, *(2)* check the correctness of the algorithm, and *(3)* ensure its integrity with hash or code-signing. *(4)* The updated software should be brought into the KMF using a storage medium, such as USB memory sticks in a TEB, to replace the old media. The old media should be destroyed.

G. Vendor Dependence

This approach is independent of any particular vendors for equipment or devices. The physical construction of the KMF may pose a significant challenge if there is a need for migration, but it can be established in any location. The operating terminal, printer, media, TEBs, and other equipment used in the KMF do not have specific requirements and can be compatible with a variety of products.

IX. Conclusion

Cryptocurrencies have gained widespread usage, with cryptocurrency custodians playing a significant role in this ecosystem. While some enthusiasts oppose the trend towards centralization, the majority of users currently entrust their cryptocurrencies to exchanges in reality. Cryptocurrency custodians must therefore securely manage customers' assets.

In this paper, we have discussed the security and risk management of wallet implementation by modeling it as SIG-SYS. We have also heavily focused on cold wallet practices as a rational approach for handling significant assets. However, several open issues still need to be addressed, for example:

1) Implementing a Lightweight Cold Wallet: We acknowledge that the operations involved in the referential example EXTREME-COLD are resource-intensive. Further research is needed to explore lighter management methods that allow companies to easily implement cold wallets while ensuring security and effectively controlling risks.

2) Efficient Wallet Operations: We have not discussed the asset management aspect across wallets, particularly in terms of balancing or timing transfers between hot and cold layers. We believe that different types of businesses have different practices in this regard, such as centralized exchanges with consumers, liquidity providers that deal with large-lot transfers, or decentralized exchanges. Additionally, lending or staking options of cryptocurrencies may also impact these practices. Unfortunately, we could not find any relevant literature on this specific topic.

3) Non-Fungible Types of Assets: Our method focused on major cryptocurrencies, such as native tokens of popular blockchains, or ERC-20 and compatible tokens that can be managed using regular wallets. However, non-fungible tokens (NFTs) have gained attention since 2021, and other derived tokens are actively being developed [149]. An example is real-world assets (RWAs) backed by off-chain assets, such as gold, bonds, and real estate [150]. These types of assets are inherently unique and may require different practices.

References

[1] W. Fumy and P. Landrock, "Principles of key management," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 785–793, 1993.

[2] G. R. Blakley, "Safeguarding cryptographic keys," in *1979 International Workshop on Managing Requirements Knowledge (MARK)*, 1979, pp. 313–318.

[3] S. Eskandari, D. Barrera, E. Stobert, and J. Clark, "A first look at the usability of bitcoin key management," 2015. [Online]. Available: https://www.ndss-symposium.org/ndss2015/ndss-2015-usec-programme/first-look-usability-bitcoin-key-management/

[4] Z. Jaroucheh and B. Ghaleb, "Crypto assets custody: Taxonomy, components, and open challenges," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023, pp. 1–6.

[5] G. J. Popek and C. S. Kline, "Encryption and secure computer networks," *ACM Comput. Surv.*, vol. 11, no. 4, pp. 331–356, dec 1979.

[6] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, nov 1979.

[7] ASC X9, *ANSI X9.17, Financial Institution Key Management (Wholesale)*. American Bankers Association, Apr. 1985. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub171.pdf

[8] P. Zimmermann, *PGP Source Code and Internals*. Mit Press, 1995.

[9] S. Capkun, L. Buttyán, and J. P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.

[10] ITU-T, "X.509 (10/19) : Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks," Oct. 2019.

[11] R. Charette, "DigiNotar certificate authority breach crashes e-government in the Netherlands," *IEEE Spectrum*, 2011.

[12] CAB Forum, "Baseline requirements for the issuance and management of publicly-trusted certificates - version 2.0.1," p. 42, Aug. 2023.

[13] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends, "RFC 4033, DNS Security Introduction and Requirements," Mar. 2005.

[14] IANA, "Key signing ceremonies." [Online]. Available: https://www.iana.org/dnssec/ceremonies

[15] E. Barker, *SP 800-57, Recommendation for Key Management: Part 1 – General*. National Institute of Standards and Technology, May 2020.

[16] E. Barker, M. Smid, D. Branstad, and S. Chokhani, *SP 800-130, A Framework for Designing Cryptographic Key Management Systems*. National Institute of Standards and Technology, Aug. 2013.

[17] J. Wilbur L. Ross and W. Copan, *FIPS 140-3, Security Requirements for Cryptographic Modules*. National Institute of Standards and Technology, Mar. 2019.

[18] G. M. Raimondo and L. E. Locascio, *FIPS 186-5, Digital Signature Standard (DSS)*. National Institute of Standards and Technology, Feb. 2023.

[19] S. Bellovin and R. Housley, "RFC 4107, Guidelines for Cryptographic Key Management," Jun. 2005.

[20] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[21] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[22] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 104–121.

[23] S. Suratkar, M. Shirole, and S. Bhirud, "Cryptocurrency wallet: A review," in *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, 2020, pp. 1–7.

[24] Y. Erinle, Y. Kethepalli, Y. Feng, and J. Xu, "Sok: Design, vulnerabilities, and security measures of cryptocurrency wallets," Jul. 2023. [Online]. Available: https://arxiv.org/abs/2307.12874

[25] P. Alpeyev and Y. Nakamura, "How to launder $500 million in digital currency," Jan. 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-01-29/how-to-launder-500-million-in-digital-currency-quicktake-q-a

[26] W. Zhao, "Crypto exchange zaif hacked in $60 million bitcoin theft," Sep. 2018. [Online]. Available: https://www.coindesk.com/markets/2018/09/20/crypto-exchange-zaif-hacked-in-60-million-bitcoin-theft/

[27] OSC, "QuadrigaCX: A review by staff of the Ontario Securities Commission," Apr. 2020. [Online]. Available: https://www.osc.ca/quadrigacxreport/

[28] FTX, "Exhibit A – Document #1242, Attachment #1," Apr. 2023, case 22-11068-JTD. United States Bankruptcy Court, District of Delaware.

[29] M. D. Detmer, "Petition for appointment of receiver, temporary injunction, and other permanent relief," Jun. 2023, case A-23-872963-B. Eighth Judicial District Court, Clark County, Nevada.

[30] K. Oosthoek and C. Doerr, "From hodl to heist: Analysis of cyber security threats to Bitcoin exchanges," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–9.

[31] R. Chandramouli, M. Iorga, and S. Chokhani, *Cryptographic Key Management Issues and Challenges in Cloud Services*. Springer New York, 2014, pp. 1–30.

[32] I. Kuzminykh, B. Ghita, and S. Shiaeles, "Comparative analysis of cryptographic key management systems," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, O. Galinina, S. Andreev, S. Balandin, and Y. Koucheryavy, Eds. Cham: Springer International Publishing, 2020, pp. 80–94.

[33] S. Xiao, W. Gong, D. Towsley, Q. Zhang, and T. Zhu, "Reliability analysis for cryptographic key management," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 999–1004.

[34] S. Rana, F. K. Parast, B. Kelly, Y. Wang, and K. B. Kent, "A comprehensive survey of cryptography key management systems," *Journal of Information Security and Applications*, vol. 78, p. 103607, 2023.

[35] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, 1977.

[36] D. L. Evans, P. J. Bond, and A. L. Bement, Jr., *FIPS 199, Standards for Security Categorization of Federal Information and Information Systems*. National Institute of Standards and Technology, Feb. 2004.

[37] A. Avižienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[38] M. Warkentin and C. Orgeron, "Using the security triad to assess blockchain technology in public sector applications," *International Journal of Information Management*, vol. 52, p. 102090, 2020.

[39] Z. Gutterman, B. Pinkas, and T. Reinman, "Analysis of the Linux random number generator," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, 2006.

[40] L. Dorrendorf, Z. Gutterman, and B. Pinkas, "Cryptanalysis of the random number generator of the Windows operating system," in *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, nov 2009.

[41] Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung, "Practical leakage-resilient pseudorandom generators," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. Association for Computing Machinery, 2010, pp. 141–151.

[42] ISE, "Ethercombing: Finding secrets in popular places," Apr. 2019. [Online]. Available: https://www.ise.io/casestudies/ethercombing/

[43] 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)*, 2018, pp. 1308–1316.

[44] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, 2001.

[45] D. Boneh and R. Venkatesan, "Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes," in *Advances in Cryptology — CRYPTO '96*, N. Koblitz, Ed. Springer Berlin Heidelberg, 1996, pp. 129–142.

[46] T. Pornin, "RFC6979, Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)," Aug. 2013.

[47] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and MtGox," in *Computer Security - ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds. Cham: Springer International Publishing, pp. 313–326.

[48] "IEEE standard for general requirements for cryptocurrency exchanges," *IEEE Std 2140.1-2020*, 2020.

[49] Y. Tsuchiya and N. Hiramoto, "How cryptocurrency is laundered: Case study of Coincheck hacking incident," *Forensic Science International: Reports*, vol. 4, p. 100241, 2021.

[50] "Exchanges which let you create multiple deposit addresses," Nov. 2020. [Online]. Available: https://bitcointalk.org/index.php?topic=5292667

[51] T. Hardjono, A. Lipton, and A. Pentland, "Wallet attestations for virtual asset service providers and crypto-assets insurance," May 2020. [Online]. Available: https://arxiv.org/abs/2005.14689

[52] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. Association for Computing Machinery, 2015, pp. 720–731.

[53] OKX, "Proof of reserves." [Online]. Available: https://www.okx.com/proof-of-reserves

[54] Binance, "Proof of reserves." [Online]. Available: https://www.binance.com/en/proof-of-reserves

[55] AICPA, "Accounting for and auditing of digital assets, practice aid," p. 60, Jul. 2023, aU Chapter 2, IV-A.

[56] "Cabinet office order on cryptoasset exchange service providers," Mar. 2017, article 27, paragraph (2) and (3). [Online]. Available: https://www.japaneselawtranslation.go.jp/en/laws/view/4315#je_ch2at15

[57] Financial Services Agency of Japan, "Guideline for supervision of crypto-asset exchange service providers," p. 46, Jun. 2021, iI-2-2-3-2 (3) (v). [Online]. Available: https://www.fsa.go.jp/common/law/guide/kaisya/e016.pdf

[58] J. Koning, "Japan was the safest place to be an ftx customer," Dec. 2022. [Online]. Available: https://www.coindesk.com/consensus-magazine/2022/12/13/japan-was-the-safest-place-to-be-an-ftx-customer

[59] A. M. Antonopoulos, *Mastering Bitcoin*. O'Reilly Media, Inc., 2014.

[60] Blockchain.com, "Blockchain size." [Online]. Available: https://www.blockchain.com/explorer/charts/blocks-size

[61] Etherscan, "Ethereum full node sync (archive) chart." [Online]. Available: https://etherscan.io/chartsync/chainarchive

[62] A. G. Khan, A. H. Zahid, M. Hussain, and U. Riaz, "Security of cryptocurrency using hardware wallet and QR code," in *2019 International Conference on Innovative Computing (ICIC)*, 2019, pp. 1–10.

[63] T. Bui, S. P. Rao, M. Antikainen, and T. Aura, "Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet," in *Proceedings of the 12th European Workshop on Systems Security*, ser. EuroSec '19. Association for Computing Machinery, 2019.

[64] A. Voskobojnikov, O. Wiese, M. Mehrabi Koushki, V. Roth, and K. K. Beznosov, "The U in crypto stands for usable: An empirical study of user experience with mobile cryptocurrency wallets," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. Association for Computing Machinery, 2021.

[65] L. Van Der Horst, K.-K. R. Choo, and N.-A. Le-Khac, "Process memory investigation of the Bitcoin clients Electrum and Bitcoin Core," *IEEE Access*, vol. 5, pp. 22 385–22 398, 2017.

[66] T. Volety, S. Saini, T. McGhin, C. Z. Liu, and K.-K. R. Choo, "Cracking bitcoin wallets: I want what you have in the wallets," *Future Generation Computer Systems*, vol. 91, pp. 136–143, 2019.

[67] D. He, S. Li, C. Li, S. Zhu, S. Chan, W. Min, and N. Guizani, "Security analysis of cryptocurrency wallets in Android-based applications," *IEEE Network*, vol. 34, no. 6, pp. 114–119, 2020.

[68] MetaMask, "Address poisoning scams." [Online]. Available: https://support.metamask.io/hc/en-us/articles/11967455819035

[69] N. Ivanov and Q. Yan, "EthClipper: A clipboard meddling attack on hardware wallets with address verification evasion," in *2021 IEEE Conference on Communications and Network Security (CNS)*, 2021, pp. 191–199.

[70] A. Perrig and D. Song, "Hash visualization : a new technique to improve real-world security," *International Workshop on Cryptographic Techniques and E-Commerce*, vol. 25, 1999.

[71] A. Sarkar, "MetaMask scammers take over government websites to target crypto investors," Sep. 2023. [Online]. Available: https://cointelegraph.com/news/metamask-scam-government-websites-crypto-investors

[72] S. Ro, "Bloomberg's Miller Bitcoin gift stolen," Dec. 2013. [Online]. Available: https://www.businessinsider.com/bloomberg-matt-miller-bitcoin-gift-stolen-2013-12

[73] T. Sans, Z. Liu, and K. Oh, "A decentralized mnemonic backup system for non-custodial cryptocurrency wallets," in *Foundations and*

Practice of Security, G.-V. Jourdan, L. Mounier, C. Adams, F. Sèdes, and J. Garcia-Alfaro, Eds. Cham: Springer Nature Switzerland, 2023, pp. 355–370.

[74] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed. Springer Berlin Heidelberg, 2013, pp. 34–51.

[75] P. Wuille, "Hierarchical deterministic wallets." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[76] M. Palatinus, P. Rusnak, A. Voisine, and S. Bowe, "Mnemonic code for generating deterministic keys." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

[77] P. Juola and P. Zimmermann, "Whole-word phonetic distances and the PGPfone alphabet," in *Proc. 4th International Conference on Spoken Language Processing (ICSLP 1996)*, 1996, pp. 98–101.

[78] M. Vasek, J. Bonneau, R. Castellucci, C. Keith, and T. Moore, "The bitcoin brain drain: Examining the use and abuse of bitcoin brain wallets," in *Financial Cryptography and Data Security*, J. Grossklags and B. Preneel, Eds. Springer Berlin Heidelberg, 2017, pp. 609–618.

[79] W. M. Shbair, E. Gavrilov, and R. State, "HSM-based key management solution for Ethereum blockchain," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–3.

[80] S. Alrubei, E. Ball, and J. Rigelsford, "Adding hardware security into IoT-blockchain platforms," in *2022 IEEE Latin-American Conference on Communications (LATINCOM)*, 2022, pp. 1–6.

[81] Thales, "Securing digital currency with bitgo multi-signature and thales hsms - solution brief," Feb. 2020. [Online]. Available: https://cpl.thalesgroup.com/resources/encryption/bitgo-hsm-solution-brief

[82] D.-P. Dornseifer and T. von Bomhard, "How to sign ethereum eip-1559 transactions using aws kms," Feb. 2022. [Online]. Available: https://aws.amazon.com/jp/blogs/database/how-to-sign-ethereum-eip-1559-transactions-using-aws-kms/

[83] Thales, "Tamper, secure transport, and purple ped keys." [Online]. Available: https://thalesdocs.com/gphsm/luna/6.3/docs/usb/Content/overview/security_features/purple_keys_tamper_and_secure-transport.htm

[84] *ISO/IEC 19790:2012 – Information technology – Security techniques – Security requirements for cryptographic modules*. International Organization for Standardization, Aug. 2012.

[85] NIST, "Cryptographic module validation programs." [Online]. Available: https://csrc.nist.gov/projects/cryptographic-module-validation-program

[86] R. Bardou, R. Focardi, Y. Kawamoto, L. Simionato, G. Steel, and J.-K. Tsay, "Efficient padding oracle attacks on cryptographic hardware," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Springer Berlin Heidelberg, 2012, pp. 608–625.

[87] "Public comments received on draft nist sp 800-186: Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," Jan. 2020. [Online]. Available: https://csrc.nist.gov/files/pubs/sp/800/186/final/docs/sp800-186-draft-comments-received.pdf

[88] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Springer Berlin Heidelberg, 2001, pp. 514–532.

[89] K. Chalkias, F. Garillot, Y. Kondi, and V. Nikolaenko, "Non-interactive half-aggregation of eddsa and variants of schnorr signatures," in *Topics in Cryptology – CT-RSA 2021*, K. G. Paterson, Ed. Cham: Springer International Publishing, 2021, pp. 577–608.

[90] Thales, "Functionality modules." [Online]. Available: https://thalesdocs.com/gphsm/luna/7/docs/network/Content/Product_Overview/fms.htm

[91] J. Han, S. Kim, T. Kim, and D. Han, "Toward scaling hardware security module for emerging cloud services," in *Proceedings of the 4th Workshop on System Software for Trusted Execution*, ser. SysTEX '19. Association for Computing Machinery, 2019.

[92] M. Arapinis, A. Gkaniatsou, D. Karakostas, and A. Kiayias, "A formal treatment of hardware wallets," in *Financial Cryptography and Data Security*, I. Goldberg and T. Moore, Eds. Cham: Springer International Publishing, 2019, pp. 426–445.

[93] T. Bamert, C. Decker, R. Wattenhofer, and S. Welten, "Bluewallet: The secure bitcoin wallet," in *Security and Trust Management*, S. Mauw and C. D. Jensen, Eds. Cham: Springer International Publishing, 2014, pp. 65–80.

[94] H. Rezaeighaleh and C. C. Zou, "New secure approach to backup cryptocurrency wallets," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[95] A. Sarkar, "Ledger co-founder clarifies 'there is no backdoor' in 'Recover' firmware update," May 2023. [Online]. Available: https://cointelegraph.com/news/ledger-co-founder-clarifies-there-is-no-backdoor-in-recover-firmware-update

[96] J. Datko, C. Quartier, and K. Belyayev, "Breaking Bitcoin hardware wallets," Jul. 2017. [Online]. Available: https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20presentations/DEF%20CON%2025%20-%20Datko-and-Quartier-Breaking-Bitcoin-Hardware-Wallets.pdf

[97] A. Gkaniatsou, M. Arapinis, and A. Kiayias, "Low-level attacks in bitcoin wallets," in *Information Security*, P. Q. Nguyen and J. Zhou, Eds. Cham: Springer International Publishing, 2017, pp. 233–253.

[98] D. Park, M. Choi, G. Kim, D. Bae, H. Kim, and S. Hong, "Stealing keys from hardware wallets: A single trace side-channel attack on elliptic curve scalar multiplication without profiling," *IEEE Access*, vol. 11, pp. 44 578–44 589, 2023.

[99] S. Volokitin, "Software attacks on hardware wallets," Aug. 2018. [Online]. Available: https://i.blackhat.com/us-18/Wed-August-8/us-18-Volokitin-Software-Attacks-On-Hardware-Wallets-wp.pdf

[100] S. Golovanov, "Case study: fake hardware cryptowallet," May 2023. [Online]. Available: https://www.kaspersky.com/blog/fake-trezor-hardware-crypto-wallet/48155/

[101] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 57–64.

[102] M. Gentilal, P. Martins, and L. Sousa, "TrustZone-backed Bitcoin wallet," in *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, ser. CS2 '17. Association for Computing Machinery, 2017, pp. 25–28.

[103] N. Lehto, K. Halunen, O.-M. Latvala, A. Karinsalo, and J. Salonen, "CryptoVault - a secure hardware wallet for decentralized key management," in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, 2021, pp. 1–4.

[104] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security vulnerabilities of sgx and countermeasures: A survey," *ACM Comput. Surv.*, vol. 54, no. 6, jul 2021.

[105] S. van Schaik, A. Seto, T. Yurek, A. Batori, B. AlBassam, C. Garman, D. Genkin, A. Miller, E. Ronen, and Y. Yarom, "Sok: Sgx. fail: How stuff get exposed," 2022. [Online]. Available: https://sgx.fail/

[106] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," 2014.

[107] "Safe contracts." [Online]. Available: https://github.com/safe-global/safe-contracts

[108] V. Buterin, "Why we need wide adoption of social recovery wallets," Jan. 2021. [Online]. Available: https://vitalik.ca/general/2021/01/11/recovery.html

[109] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *Principles of Security and Trust*, M. Maffei and M. Ryan, Eds. Springer Berlin Heidelberg, 2017, pp. 164–186.

[110] P. Praitheeshan, L. Pan, J. Yu, J. K. Liu, and R. Doss, "Security analysis methods on ethereum smart contract vulnerabilities: A survey," Aug. 2019. [Online]. Available: http://arxiv.org/abs/1908.08605

[111] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, "Smart contracts vulnerabilities: a call for blockchain software engineering?" in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 2018, pp. 19–25.

[112] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Zanella-Béguelin, "Formal verification of smart contracts: Short paper," in *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*, ser. PLAS '16. Association for Computing Machinery, 2016, pp. 91–96.

[113] B. Jiang, Y. Liu, and W. K. Chan, "Contractfuzzer: Fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '18. Association for Computing Machinery, 2018, pp. 259–269.

[114] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, "Smart contract vulnerability analysis and security audit," *IEEE Network*, vol. 34, no. 5, pp. 276–282, 2020.

[115] P. Praitheeshan, L. Pan, and R. Doss, "Security evaluation of smart contract-based on-chain ethereum wallets," in *Network and System Security*, M. Kutyłowski, J. Zhang, and C. Chen, Eds. Cham: Springer International Publishing, 2020, pp. 22–41.

[116] F. Cassez, J. Fuller, M. K. Ghale, D. J. Pearce, and H. M. Quiles, "Formal and executable semantics of the Ethereum virtual machine in Dafny," in *Formal Methods*, M. Chechik, J.-P. Katoen, and M. Leucker, Eds., vol. 14000 LNCS. Cham: Springer International Publishing, 2023, pp. 571–583.
[117] Flow, "Accounts." [Online]. Available: https://developers.flow.com/build/basics/accounts
[118] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology — CRYPTO '91*, J. Feigenbaum, Ed. Springer Berlin Heidelberg, 1992, pp. 129–140.
[119] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology — EUROCRYPT '89*, J.-J. Quisquater and J. Vandewalle, Eds. Springer Berlin Heidelberg, 1990, pp. 688–689.
[120] P. Wuille, J. Nick, and T. Ruffing, "Schnorr signatures for secp256k1." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki
[121] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to Bitcoin," *Designs, Codes, and Cryptography*, vol. 87, 2019.
[122] J. Nick, T. Ruffing, and Y. Seurin, "MuSig2: Simple two-round Schnorr multi-signatures," in *Advances in Cryptology – CRYPTO 2021*, T. Malkin and C. Peikert, Eds. Cham: Springer International Publishing, 2021, pp. 189–221.
[123] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ECDSA with fast trustless setup," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. Association for Computing Machinery, 2018, pp. 1179–1194.
[124] Y. Lindell and A. Nof, "Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. Association for Computing Machinery, 2018, pp. 1837–1854.
[125] J. Doerner, Y. Kondi, E. Lee, and abhi shelat, "Threshold ECDSA in three rounds," Cryptology ePrint Archive, Paper 2023/765, May 2023. [Online]. Available: https://eprint.iacr.org/2023/765
[126] J.-P. Aumasson and O. Shlomovits, "Attacking threshold wallets," Cryptology ePrint Archive, Paper 2020/1052, Sep. 2020. [Online]. Available: https://eprint.iacr.org/2020/1052
[127] N. Makriyannis and O. Yomtov, "Gg18 / gg20 tss beta parameter vulnerability," Aug. 2023. [Online]. Available: https://www.cve.org/CVERecord?id=CVE-2023-33241
[128] A. M. Antonopoulos, Apr. 2016. [Online]. Available: https://twitter.com/aantonop/status/720784384572465152
[129] E. Lombrozo, J. Lau, and P. Wuille, "Segregated witness (consensus layer)." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
[130] V. Buterin, xEric Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta, "Eip-1559: Fee market change for eth 1.0 chain." [Online]. Available: https://eips.ethereum.org/EIPS/eip-1559
[131] "The merge." [Online]. Available: https://ethereum.org/roadmap/merge/
[132] P. Wuille, J. Nick, and A. Towns, "Taproot: Segwit version 1 spending rules." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki
[133] T. Simonite, "Why you need a physical vault to secure a virtual currency," Aug. 2018. [Online]. Available: https://www.wired.com/story/coinbase-physical-vault-to-secure-a-virtual-currency/
[134] A. Davenport and S. Shetty, "Air gapped wallet schemes and private key leakage in permissioned blockchain platforms," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 541–545.
[135] NCSC, "Technical specifications for construction and management of sensitive compartmented information facilities, version 1.5," Mar. 2020. [Online]. Available: https://www.dni.gov/files/Governance/IC-Tech-Specs-for-Const-and-Mgmt-of-SCIFs-v15.pdf
[136] Y. Takei, "Operating environment for EXTREME-COLD." [Online]. Available: https://github.com/takeiyuto/extreme-cold
[137] IANA, "Root ksk ceremony operating environment." [Online]. Available: https://github.com/iana-org/coen
[138] D. R. Cressey, *Other People's Money: A Study in the Social Psychology of Embezzlement*. Free Press, 1953.
[139] A. Schuchter and M. Levi, "The fraud triangle revisited," *Security Journal*, vol. 29, 2016.
[140] M. J. Campagna, "Security bounds for the nist codebook-based deterministic random bit generator," *IACR Cryptology ePrint Archive*, 2006. [Online]. Available: https://eprint.iacr.org/2006/379
[141] J. Woodage and D. Shumow, "An analysis of NIST SP 800-90A," in *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham: Springer International Publishing, 2019, pp. 151–180.
[142] F. Strenzke, "An analysis of OpenSSL's random number generator," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds., vol. 9665. Springer Berlin Heidelberg, pp. 644–669.
[143] C. Zimman and D. Bong, "PKCS #11 Cryptographic Token Interface Base Specification Version 3.0," Jun. 2020. [Online]. Available: https://docs.oasis-open.org/pkcs11/pkcs11-base/v3.0/os/pkcs11-base-v3.0-os.html
[144] L. M. Bolotin and S. B. Johnson, "Us 9,813,416 b2: Data security system with encryption," Nov. 2017. [Online]. Available: https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/9813416
[145] S. Y. Yu, C. S. Moore, J. S. Whetstone, R. Barzilai, and H. Ino, "Us 8,533,414 b2: Authentication and securing of write-once, read-many (worm) memory devices," Sep. 2013. [Online]. Available: https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/8533414
[146] M. Naor and A. Shamir, "Visual cryptography," in *Advances in Cryptology — EUROCRYPT'94*, A. De Santis, Ed. Springer Berlin Heidelberg, 1995, pp. 1–12.
[147] T. V. Bui, N. K. Vu, T. T. Nguyen, I. Echizen, and T. D. Nguyen, "Robust message hiding for QR code," in *10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2014, pp. 520–523.
[148] Y.-W. Chow, W. Susilo, G. Yang, J. G. Phillips, I. Pranata, and A. M. Barmawi, "Exploiting the error correction mechanism in QR codes for secret sharing," in *Information Security and Privacy*, J. K. Liu and R. Steinfeld, Eds. Cham: Springer International Publishing, 2016, pp. 409–425.
[149] Y. Takei, *Complete Guide to the Theory and Practice of NFTs (translated)*. Ohmsha, 5 2023.
[150] S. Jagati, "Token adoption grows as real-world assets move on-chain," Oct. 2023. [Online]. Available: https://cointelegraph.com/news/token-adoption-real-world-assets-blockchain

APPENDIX

A. Example of Signed QR Code

For demonstrating the size of the QR code of our method, we show the actual signed withdrawal transaction on Bitcoin Testnet in Figure 8. The size of data is 416 bytes, which is encoded in 69-cell QR code (version 13).

Transaction hash:
`47489ba139ca848537ba54829e62f0787d8ea2a3abb51b79085c95257dcedf6c`

Fig. 8. Example QR Code of a Signed Transaction