# Not Your MPC, Not Your Coin

Yuto Takei
Mercari, Inc.

***Abstract*—A multi-party computation (MPC) wallet is being developed as a cryptocurrency wallet that enables the distribution of signing authority to multiple parties. Many vendors are developing MPC wallets based on different cryptographic protocols, which can be readily installed by users, i.e., cryptocurrency exchanges. However, for users, simply introducing the wallet system provided by vendors poses a risk of secret key shares being stolen without their knowledge. We will illustrate methods to secretly steal key shares by exploiting the communication channels with steganography or by modifying the protocol to achieve a fault injection attack. One of the methods is using the error correction capability in QR codes as a covert channel, and another method is embedding hidden bits in the legitimate communication of the underlying MPC protocol. Such attacks can be executed by sophisticated threat actors targeting the vendor's codebase or through external software dependencies. We will propose measures that user exchanges can implement to address these risks and provide guidance for the secure use of MPC wallets.**

***Index Terms*—cryptocurrency, multi-party computation, QR code, security, threshold signature, wallet.**

## I. Introduction

Protecting cryptocurrency wallets is a great responsibility for exchanges that hold a large amount of assets. To minimize the risk of asset loss or theft, significant efforts have been made in the industry to secure transaction signing keys. In recent years, a Multi-Party Computation (MPC) wallet has been developed and started to be used in commercial scenarios. The MPC wallet combines Distributed Key Generation (DKG) and the Threshold Signature Scheme (TSS) techniques at the cryptographic level, allowing multiple parties to own private key shares and enabling the collective signing of cryptocurrency transactions.

An MPC wallet can be a quick solution to distribute signing powers among multiple people while preventing individual misuse. However, user exchanges should understand the security model of the operating environment. In fact, certain risks can be easily overlooked, such as by using a wallet provided by a vendor without a security audit or risk assessment. While wallet developers with a firm production management are less likely to be compromised, one still cannot overlook the possibility of sophisticated targeted attacks or internal threats.

The purpose of this paper is to demonstrate potential attack vectors associated with vendor-provided MPC wallets and possible mitigations from the user's perspective. Our proposed attacks exploit the system configuration to extract the user exchange's key. The explanations are initially provided using a simple example between two parties constructing Schnorr signatures, and we later mention the extension to realistic protocols.

The contributions of this paper are as follows.

- We provide a simple toy model of an MPC wallet and show techniques to hide messages in communication to extract the user key.
- In the same model, we modify the cryptographic protocols and explain a method to steal keys in a way that appears normal in communication.
- We provide the direction of countermeasures for the secure use of MPC wallets.

The remaining sections of this paper are structured as follows. Section II explains the needs and reasons for using MPC wallets, as well as an overview of the current landscape surrounding MPC wallets. Section III presents prior studies on cryptology that serve as the foundation for MPC wallets, along with studies on usage and potential attacks against MPC wallets. Section IV introduces a simplified target model of the attacks proposed in Section V, which proposes attack methods against this model. Section VI discusses the possibility of extending these attacks to commercially available MPC wallets. Section VII offers recommendations for users and vendors to mitigate risks against these attacks.

## II. Background

### A. Distributing Signing Authority

To transfer cryptocurrency, the asset owner needs to sign the transaction. There are several methods to distribute the owner's authority among multiple parties.

(A) Use of multi-signature addresses or smart contracts
(B) Secret sharing with a trusted dealer
(C) Combination of DKG and TSS

(A) requires multiple signers to provide their signatures to validate the transaction. In Bitcoin, this can be achieved by using an address with a Pay-to-Witness-Script-Hash format script. In Ethereum, smart contract wallets can be used to initiate transfers only if the signatures provided in calldata are valid. These techniques simply rely on well-established cryptography like ECDSA, but they can only be used when the underlying blockchain has certain functionalities.

(B) and (C) can generate a digital signature that appears to be from a single signing key. This means that there are no specific technical requirements on the blockchain, and it prevents the transaction fee increase associated with the size of the signature data and computational costs of verification. However, in (B), a trusted dealer is required. Since a trusted dealer would have full knowledge of the signing key, user exchanges may not be willing to accept the risk of a vendor playing the dealer role. In contrast, (C) does not require

such a dealer. As the signing key is never computationally recovered, there is no single point of weakness from a security perspective. Compared to (B), this method is relatively new, and the protocols are quite complex, which inherently comes with the risk of unknown attack methods being discovered or vulnerabilities in their implementation or usage (as this paper suggests). There is also the possibility that the security assumptions of some protocols may become compromised.

### B. Multi-Party Computation and Wallets

Multi-party computation (MPC), as the name suggests, refers to a type of protocol where multiple parties collectively perform a specific computation. The protocols need to protect the privacy of each party's secret and ensure the correctness of the computation result. Some protocols allow a certain number of parties to be adversarial, meaning they may deviate from the protocol. An adversarial party may actively spread inconsistent messages to manipulate the result or collude with other adversaries to leak secret information. Communication channels between parties may also vary among protocols. When assuming a synchronous network, all messages are guaranteed to reach the destination party within a specific time frame, or otherwise in asynchronous cases, no such guarantees can be made even among honest parties.

In the context of cryptocurrency, MPC is used to implement (C), i.e., to distribute secret shares among multiple subsystems and generate a single signature while keeping each one's share private. A distributed system with such a mechanism is called an *MPC wallet*. Even if some stakeholders who control a few subsystems are misbehaving, an MPC wallet needs to ensure that they cannot forge a signature, nor steal secret

Multiple vendors began providing MPC wallets from 2019. As different protocols are being proposed by several research groups, the protocols behind those implementations may vary. Like other cryptocurrency wallets, MPC wallets have KeyGen and Sign protocols, and there is a need for message exchanges among the parties in both of them. Messages can be encoded in any format and exchanged over any channels.

In many protocols currently known, multiple rounds of communication among parties are necessary, and research efforts are made to reduce the number of rounds to improve operational convenience, particularly for Sign.

## III. Related Work

### A. Threshold Signature Scheme

The threshold signature scheme (TSS) enables the parties to create a signature without revealing each party's secret.

Schnorr signatures [1] are considered to be TSS-friendly due to their cryptographic structure. In the past, Stinson et al. presented a method to thresholdize Schnorr signatures to a $(t, n)$-setting [2]. In recent years, several protocols have been developed, such as MuSig [3], FROST [4], and their variants.

EdDSA is one variant of Schnorr signatures and has many similarities to standard Schnorr signatures in cryptographic structures. There are several recent blockchains that natively use EdDSA. One of the few differences is that EdDSA derandomizes the nonce, which is determined from the public key and message digest. Takahashi et al. have proposed an attack method against schemes including Schnorr signatures that can recover the key from slight biases in the nonce across multiple signatures [5]. Although derandomization of the nonce prevents those attacks in a regular setting, a malicious party can extract secret shares in an MPC setting. Therefore, the use of Pseudo-Random Functions (PRF) is proposed for thresholdized Schnorr signature schemes [6], [7].

In contrast, ECDSA has a different signature structure than EdDSA, and it has been used natively in the Bitcoin and Ethereum blockchains. Several protocols have been proposed [8]–[11] to thresholdize ECDSA, and they use different MPC techniques under various security assumptions.

### B. Studies on MPC Wallets and Attacks Against Them

There have been many studies on MPC wallets. Kondi et al. proposed a protocol for an MPC wallet that incorporates the technique of proactive secret sharing [12]. It can specifically address cases where a few shares below the threshold may be stolen by malicious third parties. Battagliola et al. proposed another MPC wallet protocol with a $(2, 3)$-threshold that explicitly operates with one party being completely offline for emergency recovery [13]. Takei et al. have conducted a survey and analysis on various wallet techniques at cryptocurrency exchanges and have also discussed the risks associated with using MPC wallets [14]. They have proposed the implementation of cold wallets using QR codes. Zhang et al. have implemented an MPC cold wallet that communicates via QR codes [15].

From a cryptanalysis perspective, Aumasson et al. have illustrated three methods for attacking TSS wallets by exploiting flaws in subprotocols [16]. Some attacks target the sequence of key reshare subprotocols while others target the lack of proper verification for the range proof. Makriyannis et al. have demonstrated a vulnerability in commercial MPC wallets [17].

## IV. Preliminaries

In this section, we introduce an MPC wallet system that will be the subject of attacks. To simplify the explanation of the attack methods, the protocol we introduce here is a toy example involving two stakeholders: a wallet vendor V-Inc and a user exchange U-Inc. The two parties in MPC are vendor (party index $i = 1$) and user ($i = 2$). They use additive secret sharing for DKG and generate Schnorr signatures on secp256k1 using TSS.

- $G$ is the generator for a cyclic group with an order of a large prime number.
- $n$ is the order of the secp256k1 curve.
- $|X|$ represents the bit length of message $X$.
- $A \parallel B$ concatenates binary representations of $A$ and $B$.
- $H(X)$ is a message digest for binary representation of $X$.
- $X[i{:}j]$ refers to the substring of the bit-string representation of $X$, starting from the inclusive $i$-th bit up to, but not including, the $j$-th.

### A. Security Model

The vendor runs on the V-Inc's server, while the user runs in the wallet subsystem provided by V-Inc, for example, as a software program or a dedicated hardware device. The U-Inc has operator who uses the subsystem.

vendor and user are assumed to behave honestly at the cryptographic level and not deviate from the protocol. Hence, the protocol described later will omit the proof of knowledge on their secrets and the blame phase to eliminate adversaries.

A communication channel between vendor and user is synchronous and can be observed by operator, i.e., messages will reach their destination within a time bound. operator will not modify messages unless otherwise explicitly mentioned.

### B. Protocol

We define two subprotocols: KeyGen and Sign (figure 1).

> KeyGen (one round): vendor picks a secret share $s_1$ randomly from $[1, \ldots, n)$, calculates a public share $P_1 = s_1 G$, and sends $P_1$ to user. Then, user also picks $s_2$ randomly and sends back $P_2$ to vendor in the same manner.
> Both parties at the end will know the public verification key $Y = P_1 + P_2$.

> Sign (two rounds): For round 1, vendor picks a nonce $r_1$ from $[1, \ldots, n)$ and sends $r_1 G$ to user. Then, user also picks a nonce $r_2$ and sends $r_2 G$ to the vendor.
>
> For round 2, vendor sends the message-to-sign $M$ and $R = r_1 G + r_2 G$ to user, and user computes the challenge $c = H(R \parallel Y \parallel M)$ and sends back the portion of the signature $z_2 = r_2 + c s_2$.
>
> Finally, vendor computes $c$, $z_1$ and $z = z_1 + z_2$, and outputs the signature $(R, z)$.

vendor virtually plays the role of aggregating the signatures in Sign, but the aggregator here is not a trusted dealer. A signature $(R', z')$ generated by Sign can be verified in the same way as regular Schnorr signatures by checking $z'G \stackrel{?}{=} R' + H(R' \parallel Y \parallel M)Y$.

## V. Proposed Attacks

In this section, we demonstrate two potential methods of extracting U-Inc's key from user. There are several rounds of bidirectional communication between vendor and user in both KeyGen and Sign. During these message exchanges, user may leak some information about the secret $s_2$ secretively.

Note that V-Inc itself may not need to be malicious for such attacks to be successful; on the contrary, that may be quite rare in reality. It is more likely that the attack is done by an individual employee in V-Inc, or by a malicious developer of a third-party dependency in V-Inc's codebase, or by an external threat actor conducting sophisticated targeted attacks.

### A. Exploiting Communication Channels

When considering the actual implementation of an MPC wallet system, the wallet subsystems may exchange information in various ways. Some of the commercially available products adopts following methods for example:

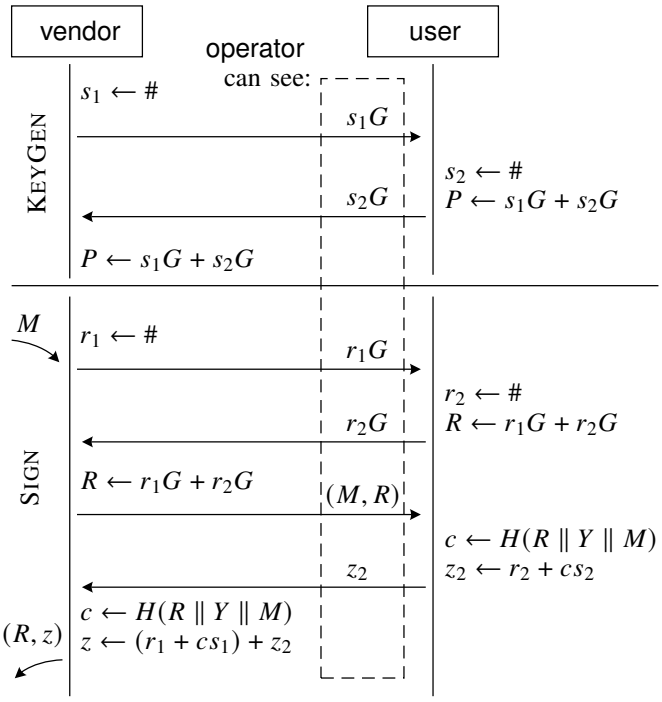



Fig. 1. An Example MPC Wallet Protocol

(a) Human operators use the vendor's website on an online computer. They transfer JSON files containing transactions to sign via USB sticks to offline computers that hold secret shares.

(b) The user exchange prepares multiple smartphone devices with secret shares: some are online and others are offline. These devices communicate with each other via QR codes and in-device cameras.

(c) The user installs wallet apps on their tablet devices. The app securely stores the secret share, and they interact with the vendor's API.

We, as an attacker, modify Sign subprotocol as follows.

> SignModTuple (modifies only round 1 of Sign. The rest is omitted.) : vendor picks a nonce $r_1$ from $[1, \ldots, n)$, and sends $r_1 G$ and a tuple $(\texttt{query}, p, q)$ to user. Then, user also picks a nonce $r_2$ and sends $r_2 G$ and $(\texttt{response}, s_2[p{:}q])$ to vendor.

The modification of the subprotocol is that both parties have started sending new tuples as underlined. The vendor stores the received $s_2[p{:}q]$ values. By signing multiple signatures while changing the values of $p$ and $q$ to cover all the bits in $s_2$, the vendor gains complete knowledge of $s_2$. Finally, the vendor can recover the complete signing key $s$ by $s_1 + s_2$. These changes can also be introduced in round 2 of Sign or during KeyGen.

*1) Steganography using QR Codes:* We now consider (b) method. QR codes are standardized as ISO/IEC 18004 and

**Algorithm 1** EncodeStego($M, S$)

$QR \leftarrow$ GenerateQR($M$)
$X \leftarrow |S| \parallel S$
**for** $i = 0 \rightarrow \lceil |X|/c \rceil$ **do**
  $k \leftarrow X[ci : c(i+1)]$
  **if** $k \neq 0$ **then**
    $p \leftarrow i(2^c - 1) + k$
    $t \leftarrow \lfloor (p - 1 + \text{Rand}())b \rfloor$
    Flip $t$-th module of $QR$
  **end if**
**end for**
**return** $QR$

**Algorithm 2** DecodeStego($QR$)

$M \leftarrow$ DecodeQR($QR$)
$D \leftarrow$ Diff($QR$, GenerateQR($M$))
$X \leftarrow [0, \ldots]$
**for all** $t \in D$ **do**
  $p \leftarrow \lfloor t/b \rfloor$
  $i \leftarrow \lfloor p/(2^c - 1) \rfloor$
  $k \leftarrow p - i \cdot (2^c - 1) + 1$
  $X[ci : c(i+1)] \leftarrow k$
**end for**
$(len, S) \leftarrow$ Decompose($X$)
**return** $(M, S)$

are used worldwide in various scenarios, including electronic payments. They can be used to transfer data between air-gapped devices in proximity with only a camera and a display. Animating QR codes may allow even larger data transmission. As mentioned in Section III, there are several studies on cryptocurrency wallets that use QR codes.

QR codes use Reed-Solomon coding and have high error correction capability. We propose a method of embedding hidden data by intentionally flipping some pixels (called *modules* in the QR code specification).

As algorithm constants, we determine the block size $b$ and the chunk size $c$. We denote the data to be regularly encoded as $M$, and to be hidden as $S$.

EncodeStego : Generate a QR code by encoding $M$ as usual. Let $X$ be the concatenation of the binary representation of the length of $S$ and $S$ itself. Slice $X$ into chunks of $c$ bits from the beginning. Each chunk corresponds to $(2^c - 1)b$ modules on the QR code in order. If the chunk has any value other than 0, randomly reverse one of the modules between the $(k-1)b$-th and $kb$-th within that corresponding range.

DecodeStego : Decode the scanned QR code using the regular method to obtain message $M$. Then, find a set of flipped positions between a regular QR code encoded with $M$, and the QR code used to extract the hidden data. From the indices of the flipped modules, execute the inverse operations of encoding to restore $X$, an array of the $c$-bit chunks with the entire length prefixed. Read the first bits to get $|S|$, and recover the hidden data $S$.

The pseudocodes are shown in Algorithm 1 and 2.

Note that the same $b$, $c$, and QR code generation algorithm must be used on both the encoding and decoding sides. The amount of data that can be hidden with this method is proportional to the length of $M$. For example, to embed $m$ bits, approximately $\lceil m/c \rceil (2^c - 1)b$ data zone modules are necessary as a minimum, and the error rate of modules will become $1/2^c b$ at maximum.

QR codes have four levels of error correction capability, and the highest level, `H`, can recover up to approximately 30% of overall scanning errors. It is seemingly most efficient to choose $b = 1, c = 2$. In reality, we are exchanging error correction capability to hide the data, and the generated QR code is more susceptible to scanning errors, or the generated QR code may become unreadable due to the arrangement of modules. We consider it acceptable in this situation since the devices communicate in close proximity when used as cryptocurrency wallets. Nonetheless, the length of the data to hide should be decided conservatively.

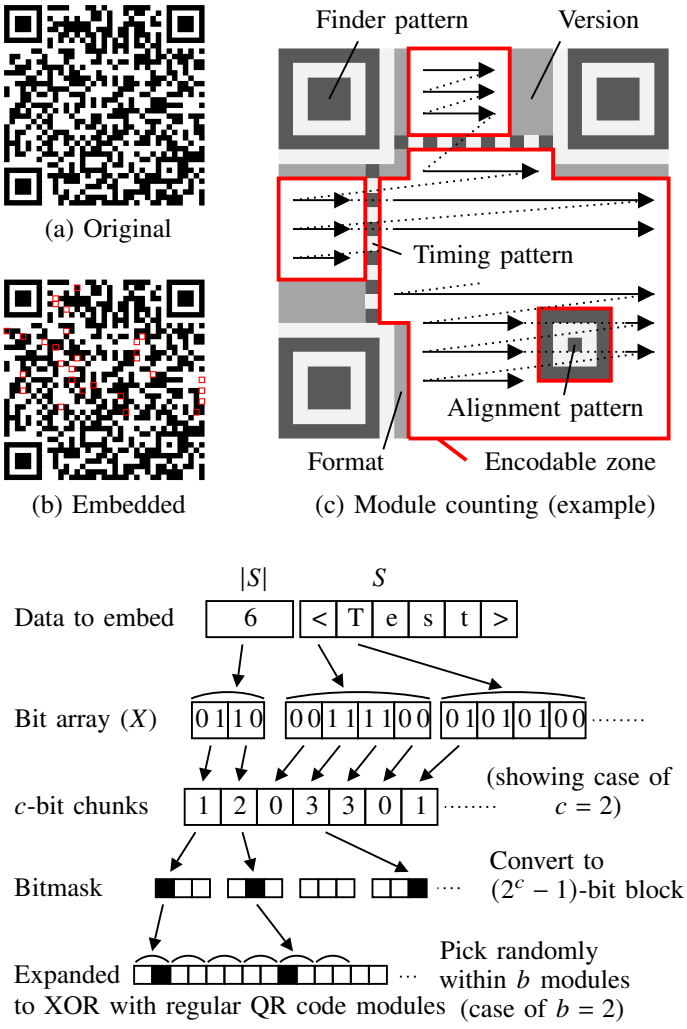
As an example, Figure 2 shows (a) a normal QR code and (b) a QR code with steganography embedded "`A hidden message.`" The flipped modules are framed in red. It is created with parameters $b = 3, c = 3$, and $|S|$ in $X$ is a fixed 9-bit size. This size of QR code, called *version 5*, has 37 modules squared, excluding the outer area called the quiet zone, and can encode binary data up to 44 bytes. With 1,043 data zone modules, this method can hide up to approximately 18 bytes. (c) shows the overview of the EncodeStego algorithm, and (d) shows one example of ordering modules with steganography in a QR code, left-to-right, top-to-bottom in non-reserved data zones. The order of data zone modules can be arbitrarily decided in this method, unrelated to that of QR code specifications.

*2) Communication over Encrypted Channel:* Online wallets, also known as hot wallets, may work as (c). In this case, user may authenticate vendor and verify the integrity of the message. The communication in transit would be encrypted and become opaque in some cases.

The benefit of message encryption is that the secret shares of both parties are secured from any man-in-the-middle, if the security assumptions of DKG or TSS protocols are broken in the future. On the other hand, the drawback is that any party can hide arbitrary messages including secret shares in the complete blackbox, which even eliminates the need for steganography techniques. From U-Inc's perspective, this means that the source of the risk is shifting from any man-in-the-middle to V-Inc after the channel encryption.

### B. Concealing in Random Numbers

In round 1 of Sign, both parties exchange a nonce. These random numbers can also be used as communication channels. This type of attack is known as a fault injection attack. We use a few bits of $r_1 G$ and $r_2 G$.




(a) Original

(b) Embedded

(c) Module counting (example)

(d) Creating QR code with embedded data

Fig. 2. QR Code Example with Hidden Data

SIGNMODRAND (modifies only round 1 of SIGN. The rest is omitted.) : vendor picks a nonce $r_1$ such that $(r_1G)[0{:}|p|]$ represents $p$, and sends $r_1G$ to user. Then, user decodes $(r_1G)$ to recover $p$, and picks a nonce $r_2$ such that the most significant bit of $r_2G$ represents $(p+1)$-th bit of $s_2$, and sends $r_2G$ to vendor.

Assuming that $s_2$ is picked from a sufficiently random source, we can consider the probability of each bit being 0 or 1 to be equal. As a result, $r_2$ chosen by the method above is also randomly distributed. $R$ in the signature takes the form of $(r_1 + r_2)G$, where $r_2$ acts as a masking factor to conceal any bias in $r_1$.

This method exploits the actual data channel. This technique can be seen in the previous work on subliminal channels by Simmons [18]. Since legitimate communication is exploited, it becomes much harder for operator to notice. Note that this method cannot be applied in non-randomized protocols such as EdDSA, or Schnorr's deterministic signing TSS variants.

## VI. APPLICABILITY TO REAL-WORLD SCENARIOS

### A. Differences in MPC Protocols

The technique discussed in Section V-A, which exploited the communication channel, does not rely on the cryptographic structure. It allows attackers to gradually steal secret shares as long as there is round-trip communication. This attack can be extended regardless of the parameters $t$, $n$, or the protocol if all shares are managed in the same way.

To the best of the author's knowledge, there are currently no commercial MPC wallet products available that allow parties from different vendors to operate as a single wallet. In other words, when using an MPC Wallet, all secret shares must be held by the same wallet subsystem from the same vendor. If this subsystem is compromised, all key shares can be easily stolen.

Moreover, this attack is not limited to specific means of communication. In Section V, we provided one possible example of hidden data communication via QR codes. The attack can still be executed even in protocols where certain parties are mostly offline, e.g., [13]. Suppose that signatures can only be generated by online nodes, the attack can be completed by stealing these online shares. Conversely, if the participation of offline nodes is necessary, the attack can be successful by exploiting the message to/from the offline nodes. This implies that even wallets that claim to be secure in a disconnected environment may still have the potential to unintentionally expose a secret key to external entities through some form of communication.

An exception to this is protocols that employ proactive secret sharing, such as [12]. In this method, regular resharing is performed, where the key shares are redistributed and old shares are removed from each node. This helps to keep the total number of compromised shares below the threshold. Yet, if there is a bug in such a reshare protocol, it can lead to a different attack where adversarial nodes can force honest nodes to delete their valid share.

### B. Eliminating Bias in Nonce

Since a nonce is used in Schnorr signatures, ECDSA, and their variants, there is a step in their TSS protocols to exchange random numbers among parties. In Section V-B, we explained that $R = r_1G + r_2G$ is not biased because $r_2$, which has a single bit from the secret share $s_2$, has sufficient randomness and works as a mask for $r_1$. We revisit this step to review the randomness of $R$ in another protocol, although an adversary who intends to steal the key would not pay attention to eliminating bias in the nonce.

With FROST [4], two random numbers, $d$ and $e$, are used as nonces. We denote the set of signers as $Q = \{1, \ldots, t\}$. Party $i$ generates random $d_i$ and $e_i$, and all parties exchange $D_i = d_iG$ and $E_i = e_iG$ with each other to obtain $B = \langle (D_1, E_1), \ldots, (D_t, E_t) \rangle$ and $\rho = H(M \parallel B)$. When each party constructs their part of the signature $z_i = r_i + cs_i$, instead of using $r_i$, they use $d_i + e_i\rho$. The signature $(R, z)$

becomes $(\prod(D_i + \rho E_i), \sum z_i)$. If $t \neq n$, the last term of $z_i$ above should be multiplied by the Lagrange coefficient $\lambda_i = \prod_{j \in Q, j \neq i} j/(j - i)$.

If an adversarial party $i$ tries to conceal some bits in this protocol, the party may exploit $E_i$ to embed the hidden data because $E_i$ is multiplied by $\rho$, the message digest of $M$ and $B$. This way, a single party, the vendor, can broadcast the index $p$ to query, and all the other compromised parties, $\mathsf{user}_i$, can respond with their $p$-th bit of secret shares $s_i$ simultaneously, without modifying the original FROST protocol and without introducing bias in the final $R$.

## VII. Recommendations for Risk Mitigations

### A. Sanitization of Data in Transit

The attack methods proposed in Section V-A focus on exfiltrating the secret by exploiting the communication channel. To prevent this type of attack, vendors should disclose the technical details about the implementations and the data structure.

User exchanges should not blindly let the communicated information pass. Rather, they should prepare a mechanism that allows for inspection and sanitization of data in transit. More specifically, the users can:

- Eliminate unnecessary fields in the data structure and decode and re-encode serialized data to prevent covert channels from being exploited.
- Introduce randomized delays in messaging to remove the timing side channel.

### B. Source Code Audit and Disclosure

To prevent the extraction of the secret through legitimate channels, as in Section V-B, conducting an audit against the wallet system's source code becomes an option. The vendors should:

- Disclose the audit reports on the source code by a third party, which state the exact versions of the tools they provide to users. This audit should consider attack vectors through third-party dependencies.
- When upgrading tools, obtain a bridge letter from the audit firm to ensure that the previous audit report remains valid.
- Open-source the related tools, or at the very least, disclose them to the user exchanges under a non-disclosure agreement.

Particularly, open-sourcing the tool may benefit vendors as it allows them to receive security vulnerability reports from external parties.

What the users should do includes:

- Independently verify the integrity of the tools provided by vendors.
- Review the audit reports, and verify the scope, version, and audit findings.
- If necessary, build and deploy the tools from the source code on their own.

If the vendor does not disclose the source code or the audit results, the Service and Organization Controls (SOC) 2 report can be an alternative to assess the vendor's organizational controls on security and integrity for their operation.

A key aspect of third-party audits is that the viewer of the audit report, a user company, is exchanging the credibility of the audited company and vendors for the credibility of the auditing firm. Hence, it is essential *for vendors* to choose an auditing firm that is confident, reputable, and trustworthy.

## VIII. Conclusion

Using an MPC wallet offers several advantages within an enterprise organization. One key benefit is the ability to transfer cryptocurrency on virtually any blockchain by distributing signing authorities without the need for a trusted third party. Nevertheless, it is critical to recognize that vendor-provided MPC wallets also come with their own set of risks. The complexity of the MPC protocol and reliance on vendor software may delay the discovery of vulnerabilities in the wallet. Many vendors offer MPC wallets as a SaaS model, which prevents air-gaps and requires bi-directional communication due to the nature of the protocols behind them. User exchanges should carefully consider these factors, especially when managing a large amount of assets on it.

We have demonstrated the possibility of attackers stealing keys by using tampered wallet software, particularly as a sophisticated threat actor on the vendor side. Techniques like steganography or exploiting the randomness within the protocols can help easily hide data. To mitigate these risks, user exchanges need to have a good understanding of the cryptographic background of the MPC wallet being used, conduct audits of the code provided by the vendor, or even develop their own tools for message inspection. MPC wallet vendors should consider open-sourcing the core cryptographic operations, or at least disclose them to the user exchanges. This not only reduces the risk of software tampering from cyber attacks but also ensures transparency.

## References


[1] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology — EUROCRYPT '89*, vol. 434 LNCS, 1990.

[2] D. R. Stinson and R. Strobl, "Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates," in *Information Security and Privacy*, vol. 2119 LNCS. Springer Berlin Heidelberg, 2001.

[3] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple Schnorr multi-signatures with applications to Bitcoin," *Designs, Codes, and Cryptography*, vol. 87, 2019.

[4] C. Komlo and I. Goldberg, "FROST: Flexible round-optimized Schnorr threshold signatures," in *Selected Areas in Cryptography*, vol. 12804 LNCS, 2021.

[5] A. Takahashi, M. Tibouchi, and M. Abe, "New Bleichenbacher records: Fault attacks on qDSA signatures," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018.

[6] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille, "MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2020.

[7] C. Bonte, N. P. Smart, and T. Tanguy, "Thresholdizing HashEdDSA: MPC to the rescue," *International Journal of Information Security*, vol. 20, 2021.

[8] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ECDSA with fast trustless setup," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2018.

[9] Y. Lindell and A. Nof, "Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2018.

[10] J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Threshold ECDSA from ECDSA assumptions: The multiparty case," in *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2019-May.

[11] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled, "UC non-interactive, proactive, threshold ECDSA with identifiable aborts," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2020.

[12] Y. Kondi, B. Magri, C. Orlandi, and O. Shlomovits, "Refresh when you wake up: Proactive threshold wallets with offline devices," in *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2021-May.

[13] M. Battagliola, R. Longo, A. Meneghetti, and M. Sala, "Threshold ECDSA with an offline recovery party," *Mediterranean Journal of Mathematics*, vol. 19, 2022.

[14] Y. Takei and K. Shudo, "Pragmatic analysis of key management for cryptocurrency custodians," in *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2024, pp. 524–542.

[15] H. Zhang, X. Zou, G. Xie, and Z. Li, "Blockchain multi-signature wallet system based on QR code communication," in *Communications in Computer and Information Science*, vol. 1736 CCIS. Springer Nature Singapore, 2022.

[16] J.-P. Aumasson and O. Shlomovits, "Attacking threshold wallets," 2020. [Online]. Available: https://eprint.iacr.org/2020/1052

[17] N. Makriyannis, O. Yomtov, and A. Galansky, "Practical key-extraction attacks in leading MPC wallets," 2023. [Online]. Available: https://eprint.iacr.org/2023/1234

[18] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology: Proceedings of Crypto 83*. Springer US, 1984.