Enhancing Cold Wallet Security with Native Multi-Signature schemes in Centralized Exchanges

Shahriar Ebrahimi
Nobitex Exchange, Iran.

Parisa Hasanizadeh
Nobitex Exchange, Iran.

Seyed Mohammad Aghamirmohammadali
Nobitex Exchange, Iran.

Amirali Akbari
Nobitex Exchange, Iran.

Abstract

Currently, one of the most widely used protocols to secure cryptocurrency assets in centralized exchanges is categorizing wallets into *cold* and *hot*. While *cold* wallets hold user deposits, *hot* wallets are responsible for addressing withdrawal requests. However, this method has some shortcomings such as: 1) availability of private keys in at least one *cold* device, and 2) exposure of all private keys to one trusted *cold* wallet admin. To overcome such issues, we design a new protocol for managing *cold* wallet assets by employing native multi-signature schemes. The proposed *cold* wallet system, involves at least two distinct devices and their corresponding admins for both wallet creation and signature generation. The method ensures that no final private key is stored on any device. To this end, no individual authority can spend from exchange assets. Moreover, we provide details regarding practical implementation of the proposed method and compare it against state-of-the-art. Furthermore, we extend the application of the proposed method to an scalable scenario where users are directly involved in wallet generation and signing process of cold wallets in an MPC manner.

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Key management; Security and privacy → Digital signatures

Keywords and phrases Cold wallet security, centralized exchange, multi-signature, Multi-party computation (MPC)

Digital Object Identifier 10.4230/OASIcs.Nobitex.2021.23

Acknowledgements This research is fully funded by Nobitex crypto-exchange.

1 Introduction

Currently, centralized exchanges play a big role in cryptocurrency world and provide multiple advantages over decentralized exchanges (DEX) [1, 2], such as higher liquidity, lower fee, and advanced trading tools. However, the main drawback of centralized exchanges is that users have to trust a third party to mange their cryptocurrency assets. To this end, the first responsibility of any centralized exchange is to ensure security of user cryptocurrency funds. The state-of-the-art protocol for managing wallet private keys in exchanges is to keep users deposits in *cold* wallet system, while handling withdrawals by *hot* wallets. The *cold* wallet is usually consisted from series of air-gapped devices that hold wallets private keys and a secure *cold* gateway that are responsible for charging hot wallets. There is no standard regarding best practices in *cold* wallet management, and therefore, in order to gain users trust, exchanges usually publicly announce some details regarding their *cold* wallet protocol [3].

In this paper, we analyze the state-of-the-art cold/hot wallet management protocols in exchanges. We furthermore point-out the shortcomings [4, 5, 6, 7] of the basic protocol and propose our practical method in order to solve such shortcomings. The proposed method is based on the native multi-signature protocols [8] in underlying public-key infrastructure (PKI) of the cryptocurrency, such as ECDSA [9] and Schnorr [10, 11], and does not effect the transaction structure or size on the blockchain. Moreover, we analyze the security of the proposed *cold* wallet architecture and reduce it to the security of the underlying PKI. We furthermore extend the application of the proposed method to a scenario where users are directly involved in wallet creation and signing process in a multi-party computation (MPC) setup. The extended protocol ensures that no individual authority in the exchange can spend user cryptocurrency funds without users direct involvements with their own private shares of the wallet. Finally, we evaluate communication and computation overhead of the proposed method and provide different solutions to increase scalability of its extended application.

The rest of the paper is organized as follows. Section 2 provides required background to follow the paper. Section 3 details the state-of-the-art hot/cold wallet management protocol and points out its shortcomings. Section 4 describes the proposed enhanced hot/cold wallet system based on the native multi-signature schemes. Section 5 evaluates the proposed method against the state-of-the-art in terms of complexity and security. Section 6 discusses the advantages of the proposed method compared to the state-of-the-art and extends it to a scenario where users take part in controlling exchange wallets. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Digital Signatures in Elliptic Curve Cryptography

Currently, the security of popular cryptocurencies, such as Bitcoin and Ethereum, are based on elliptic curve cryptography (ECC). To this end, the main focus of the paper is on ECC signatures. However, the idea behind the proposed method is also applicable on other PKIs, such as lattice-based ones. In this section, we describe the abstract computations in elliptic curve digital signature schemes as is shown in Table 1.

2.1.1 ECDSA

The process of signing a message using ECDSA starts with choosing a random $(log\, q)$-bit vector k from $\mathbb{Z}_q$. Multiplying the secret vector k to the curve's generator G, results in the public two-dimensional point R that is used later for verification of the signature. The first dimension of R is directly used in the signature s. The signature is calculated as $s = k^{-1}.(H(m) + r.x)\ mod\, q$, where x is the private key of the signer. Finally the signer outputs the pair (r, s) as the signature. Note that for every signature, the k value is generated randomly and therefore, the scheme ensures that signing the same message by one private key results in different signatures.

In verification process, the verifier computes two terms $u_1 = H(m).s^{-1}\ mod\, q$ and $u_2 = r.s^{-1}\ mod\, q$. Finally, the phrase $u_1.G + u_2.P$ should be equal as R. Following equation demonstrates the correctness of the verification process:

$$u_1.G+u_2.P = u_1.G+u_2.(x.G) = (H(m).s^{-1}+r.s^{-1}.x)\times G = (H(m)+r.x)\left(k^{-1}(H(m)+r.x)\right)^{-1}\times G$$

$$= (H(m)+r.x)\times(H(m)+r.x)^{-1}\times\left(k^{-1}\right)^{-1}\times G = k\times G = R$$

Table 1 Elliptic Curve (EC) Signature Algorithms

	ECDSA	Schnorr
Signature generation	$k \leftarrow \mathbb{Z}_q$ $R = (r_x, r_y) = k.G$ $r = r_x \, mod \, q$ $s = k^{-1}.(H(m) + r.x) \, mod \, q$ $Sig = (r, s)$	$k \leftarrow \mathbb{Z}_l$ $R = k.G$ $e = H(R\|P\|m)$ $s = (k + x.e) \, mod \, l$ $Sig = (e, s)$
Verification	$u_1 = H(m).s^{-1} \, mod \, q$ $u_2 = r.s^{-1} \, mod \, q$ $(r'_x, r'_y) = u_1.G + u_2.P$ Verify: $r'_x == r$	$R' = s \times G - e \times P$ Verify: $H(R'\|P\|m) == e$

Table 2 Paillier Homomorphic Cryptosystem

Key Generation	Encryption	Decryption
$p, q \leftarrow$Primes $n = p.q, \; g = n + 1$ $\lambda = (p-1).(q-1)$ $\mu = \lambda^{-1} \, mod \, n$	$r \leftarrow \mathbb{Z}_n^*, \; gcd(r, n) = 1$ $c = g^m.r^n \, mod \, n^2$	$m = L(c^\lambda \, mod \, n^2).\mu \, mod \, n$ note: $L(x) = \frac{x-1}{n}$

2.1.2 Schnorr

The Schnorr signature variant over ECC has multiple standards. We stick to the latest one [12, 13] using Ristretto sub-groups over twisted Edward curves, i.e. Sr25519. The process of signature generation starts with randomly choosing one-time secret vector k from $\mathbb{Z}_l$ and calculating its public related point R. Vector e is constructed by hashing a concatenation of R, P and m values. The s value is simply calculated as $(k + x.e) \, mod \, l$. Note that in contrast with ECDSA, k and s are used in a linear manner. This property allows Schnorr signatures to be aggregated easily to construct a multi-party signature. Finally, the signer outputs (e, s) pair as signature.

In order to verify a signature, one simply calculates $R' = s \times G - e \times P$. In case of a valid signature, R' should be equal to the R calculated during the signature generation process. Therefore, final verification step is to ensure that $H(R'|P|m)$ and e are equal. The correctness of the scheme is as follows:

$$R' = s \times G - e \times P = (k + x.e) \times G - e \times (x.G) = ((k + x.e) - e.x) \times G = k.G = R$$

2.2 Paillier Cryptosystem

Pailliar [14] is a probabilistic additively homomorphic public key cryptosystem. For any two encrypted messages m_1 and m_2, such as $Enc(m_1)$ and $Enc(m_2)$, the encrypted summation can be directly calculated by multiplication of two ciphertexts as follows: $Enc(m_1 + m_2) = Enc(m_1) \times Enc(m_2)$. Table 2 shows detailed computations in three phases of Paillier cryptosystem. The correctness of the the homomorphic property of the scheme is as follows:

$$Enc_{(m_1)} \times Enc_{(m_2)} = g^{m_1}.{r_1}^n \times g^{m_2}.{r_2}^n = g^{(m_1+m_2)}.(r_1.r_2)^n = g^{(m_1+m_2)}.(r')^n = Enc_{(m_1+m_2)}$$

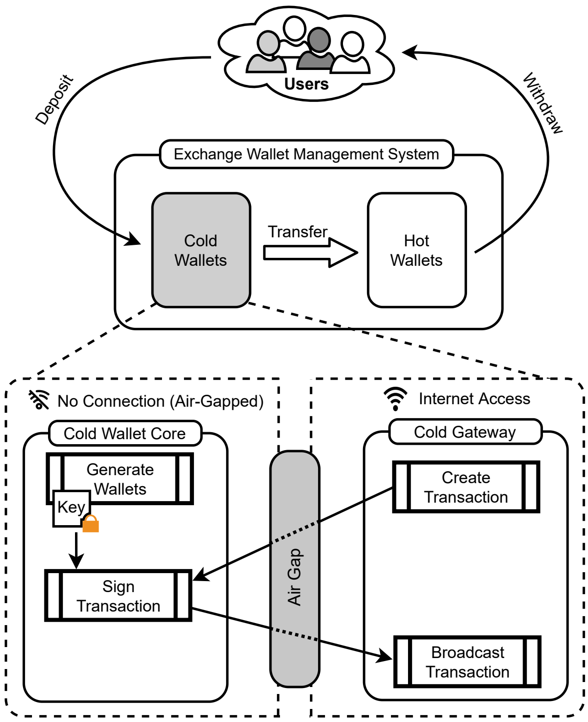

Figure 1 Basic Cold Wallet Management System

3 State-of-the-art Cold Wallet Protocol

3.1 Overview

In order to protect private keys of cryptocurrency wallets, centralized exchanges classify their wallets into two sub-classes: 1) hot wallets and 2) cold wallets. Figure 1 provides overall structure of hot/cold wallet management system. The *hot* wallet is responsible for withdrawal requests of users. The destination addresses in *hot* wallet transactions are controlled by users themselves, which are usually users local wallets or accounts on other exchanges. Note that the number of transactions in *hot* wallet system is high and, therefore, we require to have fast transaction creation, signing, and broadcast process. The security of *hot* wallet system is considered to be compromised, since the signing process of transactions are done in a system that is also connected to the internet. This results in possible exposure of *hot* wallet private keys upon a successful breach to the *hot* server. Thus, to limit security risks in *hot* wallet system, the cryptocurrency balance of *hot* wallets are kept limited (around 2-5% of total deposit).

On the other hand, the *cold* wallet system is responsible for constantly charging *hot* wallets balance. While the *cold* wallet system contains more than 90% of total deposit, it demands certain level of security. To this end, as is shown in Figure 1, the *cold* system is usually divided into two sub-systems, namely: 1) cold wallet *core* (or *cold-storage*) and 2) cold *gateway*. The cold wallet *core* is responsible for generating and managing wallet private keys and signing transactions. Moreover, the *gateway* has access to the internet and can create and broadcast transactions to the blockchain network. Note that there is an airgapped connection between the two subsystems.

It is important to make sure that no attacker can gain access to exchange users wallets private keys, even after a successful breach. To this end, the cold wallet *core* (or *cold-storage*) is isolated from any connection to any network. This mechanism ensures that signing any transaction from cold wallets requires a physical access to at least one airgapped and physically secured device.

3.2 Shortcomings

Although the general cold wallet mechanism satisfies many of the security requirements in the exchange, it still has some fundamental shortcomings as follows:

3.2.1 Availability of Wallet Private Keys in at Least One Device

Although the cold-storage mechanism, ensures that no external connection is possible to the device, however, the authority can access wallet private keys through direct physical contact with the air-gapped device. Even in scenarios where admin has no direct access to the keys (in hardware-based signing mechanisms, such as HSM), the keys can be extracted with different side-channel analysis, such as fault-injection attacks or simple/differential power analysis (SPA/DPA) [15].

3.2.2 Systematic Attacks on Key Derivation Mechanisms:

Exchanges require private key management mechanism to decrease overall complexity and security costs of the *cold-storage*. Currently there are multiple key derivation standards, such as BIP32 [16], that allow derivation of unlimited recoverable private keys from a few master keys. However, previous studies [4, 6] proposed successful attacks on different scenarios that are based on the nonlinear relation among master and its child keys. Thus, although such derivation methods are necessary for managing large amount of wallets in exchanges, however, there is a risk that an attacker can forge valid signatures for all of child keys in case of accessing to only one of the child private keys.

3.2.3 Possible Threat from a Malicious or under-pressure Admin

Since all private keys are available in cold-storage, the cold-storage admin(s) can sign and broadcast different transactions without submitting them to the cold gateway for broadcast. Therefore, in different scenarios (corrupt or under-pressure admin), unlimited number of unauthorised transactions can be signed by cold-storage admin(s). Note that the cold-storage is air-gapped and hence, has no connection to any system, which makes it impossible to monitor admin(s) actions online.

3.2.4 Corrupted Transactions from a Compromised Cold *Gateway*

In most of the transactions, the raw transaction data is clearly verifiable offline. Therefore, the cold-storage can verify the transaction's final hex data by hashing the raw transaction. However, in some cases such as complex smart-contract transactions or privacy preserving platforms, such as *z-address* payments in *tron* blockchain [17], it is not possible to ensure the validity of the given data to the cold core. To this end, it can be possible for a compromised cold gateway system to produce malicious transactions that can be used for extracting certain information regarding a targeted private key or simply result in withdrawals to attackers wallet.

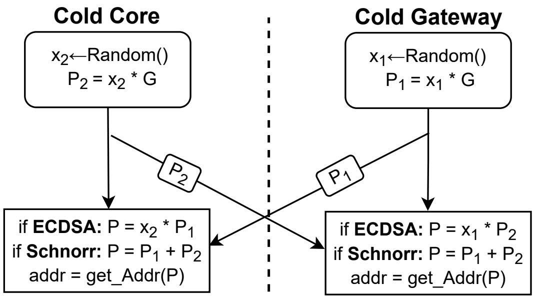

Figure 2 Proposed MPC Wallet Generation in Cold Wallet

3.2.5 No direct (off-chain) mechanism for users to get involved in transaction signing process

One of the other shortcomings of state-of-the-art cold wallet system is that the exchange is always in full control over all wallet private keys. The only possible solution for user involvement in transaction authentications is by on-chain multi-signature wallets that are provided by the blockchain platform itself. However, such mechanisms differ within different blockchains and may not be supported by all of the exchanges or wallet providers. Moreover, on-chain multi-signature transactions have extended data size, which results in higher fee per transaction. In addition, due to their more obvious on-chain relations, they can be used for mapping individuals to an exchange wallet, which violates users privacy.

4 Enhanced Cold Wallet Protocol

In order to address general shortcomings of the basic architecture (Figure 1), various native multi-signature protocols can be employed between cold-storage and cold-gateway subsystems. The proposed method is based on the native multi-party signature mechanisms over the underlying PKI in the blockchain. We employ the multi-signature variants of Schnorr [10, 11] and ECDSA [9, 8]. Since in Schnorr signing algorithm, private key x and k are employed in a linear manner, distributing the signature over more than one party is easy ($x_{golden} = x_1 + x_2$, $k_{final} = k_1 + k_2$). Note that Schnorr signatures and their related public keys can be easily aggregated to construct multi-party shared values [10, 11]. However, in ECDSA, k and x are required to be shared in a multiplicative manner among parties such that $x_{golden} = x_1 \times x_2$, $k_{final} = k_1 \times k_2$ [8]. This results in a far more complex protocol to establish multi-party ECDSA [8]. The rest of the section provides details of wallet creation and singnature generation in the enhanced cold wallet protocol.

4.1 Multi-Party Wallet Creation

In order to construct a shared wallet without violating privacy of the parties, each party starts the protocol by generating its key pair locally. Figure 2 presents the proposed multi-party wallet creation protocol. Note that after passing public keys to the other party, each side can calculate the shared public key P without knowing other party's secret key. To this end, both sides can reach to the same cryptocurrency address without violating any privacy. It is important to point out that by using this protocol, the exact private key ($x = x_1 \times x_2 \, mod \, p$ in ECDSA and $x = x_1 + x_2 \, mod l$ in Schnorr) is never calculated and therefore, is not available on any scenario during the entire execution of the protocol. This feature prevents extracting

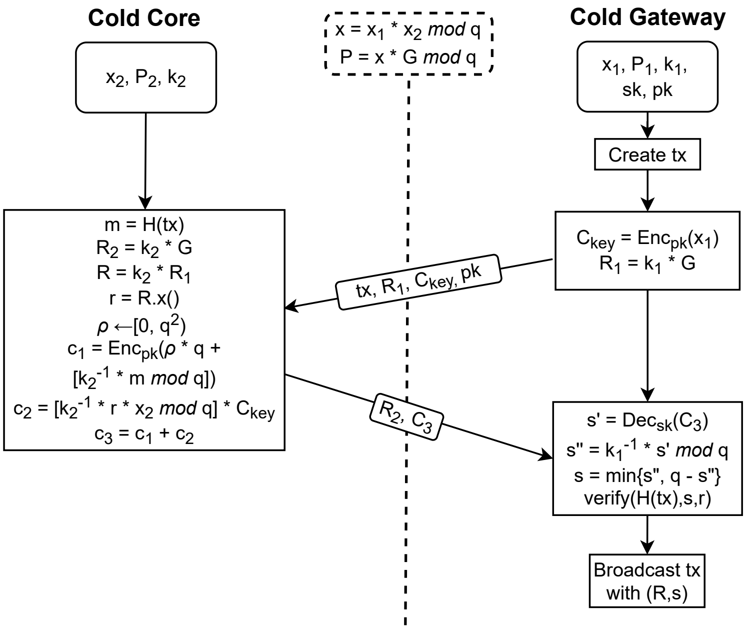

Figure 3 Proposed Protocol for 2PC-ECDSA in Cold Wallet

main private key (x) by employing side-channel analysis [18, 15] or through eavesdropping communications because only public variables are shared with the other party.

4.2 Multi-Party Signature

Figure 3 and Figure 4 show details of multi-party signature in enhanced cold wallet system using ECDSA and Schnorr signatures, respectively. A key feature of the proposed method is that it does not require any additional rounds compared to the state-of-the-art cold wallet systems (Figure 1). This becomes especially important in air-gapped connections because of time-consuming communications between the two parties (e.g. by transferring messages with USB flashes of QR-codes). The process starts with the *gateway* by creating a transaction for the same address that was calculated during multi-party wallet creation session. The rest of the section describes the proposed cold wallet protocol based on each signing algorithm separately.

4.2.1 ECDSA

The *gateway* calculates and sends four items to the *core* (cold-storage) as follows:

- $\boldsymbol{m}$**:** Raw transaction hash.
- $\boldsymbol{pk}$**:** Paillier public key of the *gateway*.
- $\boldsymbol{C_{key}}$**:** Encrypted signing private key of *gateway* by using its own Paillier public key.
- $\boldsymbol{R_1}$**:** The public point of the random nonce k_1.

Note that all of the passed items (m, pk, C_{key}, and R_1) are considered as public values and do not compromise the security of the system. C_{key} and R_1 are both public points and do not reveal any information regarding x_1 and k_1. The purpose of calculations in *core* (cold-storage) is to securely calculate C_3 without revealing any information regarding x_2 and k_2, which are *core*'s private value. More precisely, C_3 is the homomorphic encryption of

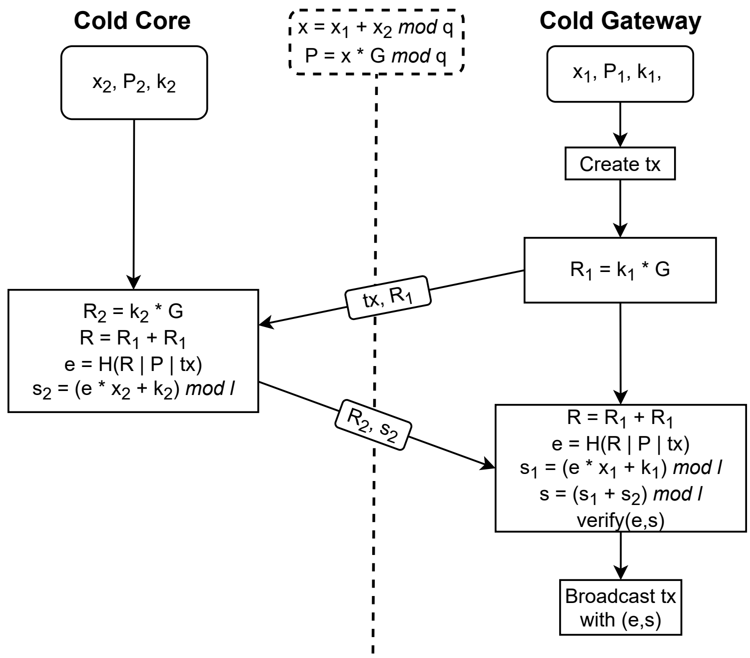

Figure 4 Proposed Protocol for 2PC-Schnorr in Cold Wallet

final signature s without k_1^{-1}, which will be multiplied later by the *gateway* itself to complete the signature before broadcast. Note that during these calculations, none of the parties will be able to calculate $x = x_1 \times x_2$ or $k = k_1 \times k_2$.

The *core* starts the signing process by calculating final $R = k_2 \times R_1$. Now the final $r = R.x$ is available for calculating the signature s. In order to achieve $c_3 = Enc_{pk}[k_2^{-1} \times (m + r.x_1.x_2)]$ without having x_1 it needs to use the homomorphic encryption of it, namely $C_{key} = Enc_{pk}(x_1)$. To this end, it calculates two phrases and returns their summation: $c_3 = Enc_{pk}[k_2^{-1} \times (m + r.x_1.x_2)] = Enc_{pk}[k_2^{-1} \times m] + Enc_{pk}[k_2^{-1} \times r.x_1.x_2]$. Note that this is only possible because of homomorphism in the Enc_{pk}, which is an additively homomorphic encryption. The $c_1 = Enc_{pk}[\rho.q + (k_2^{-1} \times m \; mod \; q)]$ is equal to the same part of the summations with a little difference of including $\rho.q$. However, this added random number will be wiped-out during the modulation process (modulo q) in the *gateway*. The random number $\rho.q$ is added to $(k_2^{-1} \times m \, mod \, q)$ before encryption in order to prevent *gateway* from guessing k_1^{-1}. On the other hand, $C_2 = (k_2^{-1} \times r.x_2) \times C_{key}$ is equal to $Enc_{pk}[(k_2^{-1} \times r.x_2 \times x_1)]$ since $k_2^{-1} \times r.x_2$ is a scalar and can be multiplied trough $Enc_{pk}(x_1)$. After calculating C_1 and C_2, the *core* sends $C_3 = C_1 + C_2$ and R_2 to the *gateway*.

The *gateway* simply decrypts C_3 with its Pailliar private key sk. The result only requires multiplication of k_1^{-1} to calculate the final s. Same as normal ECDSA signature generation, the final signature must have absolute value less than $q/2$ and therefore, s_{final} will be $min\{s'', q - s''\}$. Moreover, *gateway* calculates $r = [k1 \times R_2].x()$ and can use r and s values as final signature pair of the transaction for broadcast.

4.2.2 Schnorr

In contrast to ECDSA, constructing multi-party protocols over Schnorr signature is straightforward. The process starts with the *gateway* picking a random scalar k_1 and calculating public point related to it $R_1 = k_1 \times G$. Moreover, It send R_1 along with the raw transaction data to the *core*. The *core* selects its own random scalar k_2 and calculates $R_2 = k_2 \times G$. Now the final $e = H((R_1 + R_2)|P|m)$ can be calculated and the *core* provides its share of the final signature $s_2 = (k_2 + x_2.e)\ mod\ l$. Finally, the *core* sends s_2 and R_2 to the *gateway*. Now by having R_2, the *gateway* can also calculate e and its own share of the signature $s_1 = (k_1 + x_1.e)\ mod\ l$. The final multi-signature is simply the sum of s_1 and s_2 and their corresponding R values as follows:

$$s = s_1 + s_2 = (k_1 - x_1.e) + (k_2 - x_2.e) = (k_1 + k_2) - (x_1 + x_2).e = k - x.e$$

$$R = R_1 + R_2 = k_1 \times G + k_2 \times G = (k_1 + k_2) \times G$$

Note that nor R_1, R_2 neither s_2 reveal any information regarding secret values k_1, k_2 or x_2, respectively.

5 Evaluation

This section provides security analysis of the proposed method, while assuming the underlying PKI is secure. Moreover, we compare the proposed method against the state-of-the-art in terms of communication and computation complexity.

5.1 Security Analysis

The security of the employed 2PC-ECDSA/Schnorr in the proposed method are extensively analyzed in details in [8, 10, 11] and proven to be as hard as underlying ECDSA, Paillier, and Schnorr schemes themselves. In the following, we discuss the security of the proposed method with respect to [8, 14] and [10, 11].

1. **Wallet-creation:** During the wallet creation process, according to the hardness of underlying scheme [9, 19, 10, 11], it is considered to be computationally impossible for any of the parties (core or gateway) to extract private keys (x_1 and x_2) from public keys (P_1 and P_2). This also holds the same for any eavesdropper in the protocol, because the only shared information are public keys.
   Another important issue is the resistance to side-channel attacks in protocol-level. Although the side-channel attacks are applied to the implementation and require countermeasures in implementation-level, the proposed protocol prevents side-channel analysis since in no scenario and in no device, the final private key ($x = x_1.x_2\ mod\ q$ for ECDSA and $x = x_1 + x_2\ mod\ l$ for Schnorr) are available. Therefore, no side-channel analysis, such as timing [18], SPA/DPA [15] or cache attacks [20, 21], can be employed to directly extract final private key x.
2. **Signature:** We analyze the security of the signature creation process in three folds: (a) privacy of each party, (b) message (raw transaction) integrity, and (c) confidentiality of the entire system, while an attacker is present.

   a. During the process of calculating the multi-signature, it is vital to ensure no private information is exposed to other parties. The security/randomness of R_1 and R_2 are the same as P_1 and P_2 (which are reduced to the security of the underlying scheme, i.e. ECDSA [9] and Schnorr [10, 11]) and do not reveal any information regarding k_1 and k_2, respectively. In 2PC-ECDSA scenario, $C_{key} = Enc_{pk}(x_1)$ does not reveal any information regarding x_1 as long as the underlying Paillier scheme is hard to break. Moreover, in order to prevent *gateway* from extracting any information about k_2^{-1} or x_2, Lindell suggests [8] adding $\rho.q$ to $k_2^{-1} \times m$, which results in a randomness that is only removed my reducing entire phrase by modulo q. Therefore, even if the *gateway* tries to provide corrupted inputs for *core* (such as $C_{key} = Enc_{pk}(0\ or\ 1)$), it cannot redeem any useful information. Thus, none of the parties (*core* or *gateway*) can extract critical information from shared contents of the other one.

In 2PC-Schnorr scenario, two parties share nothing but pure public values, such as s, P, and R, which do not reveal any information regarding private values as long as the underlying Schnorr security claims hold.

b. Both parties require to ensure the integrity of message m. To this end, the *core* verifies the given m by recomputing tx hash. Moreover, it verifies the destination address of the received transaction since the destination addresses of the cold system are predefined *hot_wallet* addresses. Note that it is not possible for the *core* to verify other parts of the transaction due to the fact that it is not connected to the internet. It is worth mentioning that there is not need to ensure validity of the entire transaction in *core* because the output data C_3 and R_2 reveal nothing about x_2 and k_2, respectively. Thus, even if a corrupted transaction is given to the *core*, the output does not compromise the security of cold wallets as long as ECDSA and Paillier remain hard to break.

c. As discussed in previous scenarios, according to [8], even a malicious party (who has access to one share of the secret data) cannot achieve any information regarding other party's secret shares. The same statement also holds for an eavesdropper who does not have access to any secret shares. Moreover, in no state of the signature preparation, the main private data, such as $x = x_1.x_2$ or $k^{-1} = k_1^{-1}.k_2^{-1}$ are present in non-encrypted manner. Therefore, it is impossible to reach main private keys with any kind of side-channel analysis on only one device.

5.2 Complexity Analysis

The proposed method imposes computational overhead on both *core* and *gateway* systems. Moreover, it increases communication complexity between both systems. However, the communication between the two systems is *air-gapped* and therefore, no charges apply to the communication overhead (usually the air-gapped communications are based on transferring information via a storage device, i.e USB flash driver). It is important to note that the proposed method does not effect the size of the final message for broadcast on the blockchain and the signature does not differ from normal single signatures.

5.2.1 Communication Complexity

Table 3 provides detailed analysis regarding the imposed overhead during communications between *gateway* and *core* by underlying algorithm parameters. The 2PC-ECDSA variant imposes higher communication overhead because of employing additional Paillier ciphertexts. On the other hand, the 2-PC Schnorr variant has almost negligible overhead (only during step one arround 32 Bytes). Note that in both scenarios, communication overhead of the proposed method during step two is negligible (in 2PC-Schnorr there is no overhead).

We also include exact overhead size in our implementations of the proposed method before and after applying standard compression techniqies on the *multisig* part of the communication messages.

5.2.2 Computation Complexity

In terms of computational complexity, the overhead of the proposed method highly depends on the underlying signature scheme and its method of implementation. To this end, we

Table 3 Communication Complexity of the Proposed Method Compared to the State-of-the-art. The q_{ec} and n_p values stand for configuration parameters of *elliptic curve* and *Paillier* cryptosystems, respectively.

Cold wallet system			Step one (gateway-to-core)	Step two (core-to-gateway)
State-of-the-art			tx	tx + sig≈tx+64B
Proposed method	2-PC ECDSA	Theory	tx+R_1+C_{key}+pk =tx+$log\ q_{ec}$+$log\ n_p^2$+$log\ n_p$	tx + R_2+c_3 =tx + $log\ q_{ec}$+$log\ n_p^2$
		Imp.	=tx+32B+512B+256B =tx+800B After Comp.≈tx+600B	=tx+32B+512B=tx+534B After Comp.≈tx+420B
	2-PC Schnorr	Theory	tx+R_1 =tx+$log\ q_{ec}$	tx+R_2+s_2 =tx+2$log\ q_{ec}$
		Impl.	=tx+32B After Comp.≈tx+26B	=tx+2×32B=tx+64B After Comp.≈tx+50B

Table 4 Computation Complexity of the Proposed Method in Comparison with the State-of-the-art. E_m, M_s, M_{ec}, and I_m stand for modular exponentiation, modular scalar multiplication, elliptic curve multiplication, and modular inversion operations, respectively.

Cold wallet system		*Gateway*	*Core*
State-of-the-art	ECDSA	Ver: $I_m + 2M_s + 2M_{ec}$	Sig: $M_{ec} + I_m + 2M_s$
	Schnorr	Ver: $2M_{ec}$	Sig: $M_{ec} + M_s$
Proposed method	2-PC ECDSA	Step one: $2E_m + M_s + M_{ec}$ Step three: $E_m + 2M_s$ $+I_m + M_s + M_{ec}$+ver Total: $3E_m + 6M_s + 4M_{ec} + 2I_m$	$2M_{ec} + I_m + 2M_s$ $2E_m + M_s + 2M_s + E_m + M_s$ Total: $2M_{ec} + I_m + 6M_s + 3E_m$
	2-PC Schnorr	Step one: M_{ec} Step three: M_s+ver Total: $M_s + 3M_{ec}$	Total: $M_{ec} + M_s$

provide theoretical analysis of the proposed method against the state-of-the-art as is shown in Table 4. It is worth mentioning that in 2PC-ECDSA scenario, the imposed computation overhead is relatively high (in comparison with Schnorr variant) and is dominated by Paillier homomorphic operations.

The *gateway* always has to perform a verification on the final signature before broadcasting it to the blockchain network. Therefore, in both ECDSA and Schnorr algorithms, the *gateway* perform at least two elliptic curve multiplications. However, in the proposed method, the *gateway* is also participates in signature generation process. Therefore, the imposed overhead on *gateway* is at least eaqual to a full signature (approximately four point multiplication and one inversion in ECDSA, while three point multiplication in Schnorr). On the other hand, The*core* always takes part in signature generation. Therefore, in 2PC-Shnorr scenario, the imposed overhead to the *core* is almost negligible since it is only required to perform the same signature as same as the state-of-the-art method.

6 Extended Cold Wallet System

In this section, we demonstrate how the proposed method solves shortcomings of the basic *cold* wallet management technique. Later, we discuss different applications of the employed

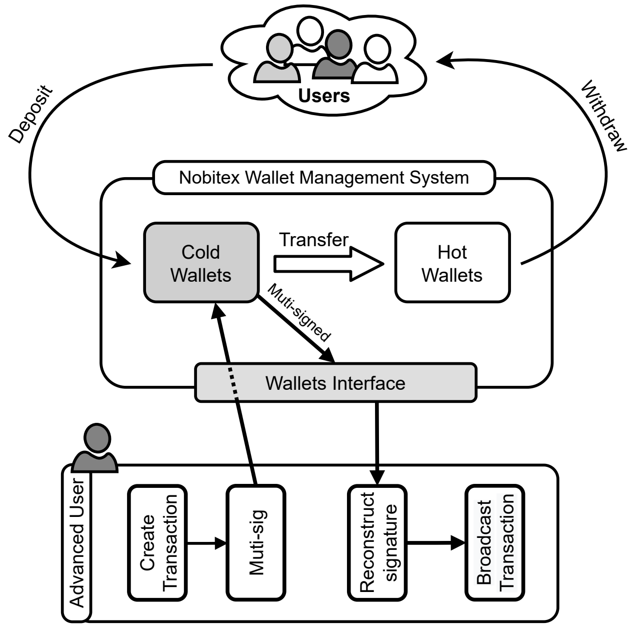

Figure 5 Proposed Multi-signature Protocol where the User is Directly Involved in Signing Process of the Transaction

native multi-signature protocol in centralized systems.

The first outcome of employing an MPC-based signature scheme in *cold* wallet, is that the final wallet private keys cannot be accessed by taking control of only one device (addresses the shortcomming 3.2.1). Therefore, the proposed method gives no individual authority the right to create valid signatures for *cold* wallets (addresses the shortcomming 3.2.3). Moreover, the protocol ensures that none of the parties (cold gateway and cold core) can gain information from other one using corrupted messages. To this end, it will not be possible for a compromised *gateway* to extract parts of private keys from *cold storage* using corrupted transaction data (addresses the shortcomming 3.2.4).

In addition, the proposed method removes any linear or non-linear relation between the generated keys. More precisely, summation (in 2-PC Schnorr: $x = x_1 + x_2$) or multiplication (in 2-PC ECDSA: $x = x_1.x_2$) of child private keys in one device by another series of keys from other device, completely removes any relation between the final private key and inner master keys in *core* because the other private share acts as a complete random value added/multiplied to the key (addresses the shortcomming 3.2.2).

The protocol presented in Figure 3 and Figure 4 can be altered in a way that a customer replaces the *gateway*. This scenario is shown in Figure 5, where the user is responsible for transaction creation and broadcast (addresses the shortcomming 3.2.5). Therefore, the exchange has no control over user's transactions. However, the security of user's wallet is backed-up by the exchange. Thus, on a security breach in the exchange or a successful attack on user's local wallet, the assets of user are secure.

The key generation process in this scenario can be implemented in different ways depending on the user expertise and suitable policies for the corresponding account. The private key share in user side can be generated locally by user itself, which results in complete implementation of the original proposed protocol without compromising the user privacy. However, upon a destructive attack on user's local wallet or loss of key information in user-side, it will not be possible to withdraw wallet funds. In order to remove such responsibility from user, the user's shared key can be initiated from a *master_key* in exchange, which does not fully preserve user's privacy but can be recovered upon certain conditions.

7 Conclusion

This paper proposes an enhanced cold wallet system based on the native multi-signature schemes in blockchain. The proposed method solves fundamental shortcomings of the-state-of-the-art cold wallet system. The proposed protocol has strong security claims reducible to the underlying signature/encryption schemes, such as ECDSA, Paillier, and Schnorr. Moreover, we evaluated the proposed method against the state-of-the-art in terms of communication and computation complexity. Finally, we extend the application of the enhanced cold wallet system to a scenario where users can have direct involvement in transaction signature generation process.

References

1. Will Warren and Amir Bandeali. 0x: An open protocol for decentralized exchange on the ethereum blockchain. *URl: https://github. com/0xProject/whitepaper*, pages 04–18, 2017.
2. Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core. Technical report, Tech. rep., Uniswap, 2021.
3. Coinbase. A behind the scenes look at the biggest (and quietest) crypto transfer on record [online]. available: https://blog.coinbase.com/a-behind-the-scenes-look-at-the-biggest-and-quietest-crypto-transfer-on-record-682ff4a6d9e4, last accessed: August 10, 2021, aug.
4. Nicolas T Courtois, Filippo Valsorda, and Pinar Emirdag. Private key recovery combination attacks: On extreme fragility of popular bitcoin key management, wallet and cold storage solutions in presence of poor rng events. 2014.
5. Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 651–668, 2019.
6. Mordechai 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)*, pages 1308–1316. IEEE, 2018.
7. Farah Maath Jasem, Ali Makki Sagheer, and Abdullah M Awad. Enhancement of digital signature algorithm in bitcoin wallet. *Bulletin of Electrical Engineering and Informatics*, 10(1):449–457, 2021.
8. Yehuda Lindell. Fast secure two-party ecdsa signing. In *Annual International Cryptology Conference*, pages 613–644. Springer, 2017.
9. Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.
10. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer, 1989.
11. Yannick Seurin. On the exact security of schnorr-type signatures in the random oracle model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 554–571. Springer, 2012.
12. Mike Hamburg. Decaf: Eliminating cofactors through point compression. In *Annual Cryptology Conference*, pages 705–723. Springer, 2015.
13. Henry de Valence, Isis Lovecruft, and Tony Arcieri. The ristretto group [online]. available: https://ristretto.group/ristretto.html, last accessed: August 10, 2021.

14 Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

15 Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

16 Pieter Wuille. Bip32: Hierarchical deterministic wallets [online]. available: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, last accessed: August 10, 2021.

17 TronZ. Shielded transaction protocol [online]. available: https://www.tronz.io/, last accessed: August 10, 2021.

18 Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

19 Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

20 Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+ flush: a fast and stealthy cache attack. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 279–299. Springer, 2016.

21 Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE, 2019.