IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON RESEARCH CHALLENGES AND OPPORTUNITIES IN SECURITY AND PRIVACY OF BLOCKCHAIN TECHNOLOGIES

Received November 30, 2017, accepted January 8, 2018, date of publication January 29, 2018, date of current version March 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2799385

A Social-Network-Based Cryptocurrency Wallet-Management Scheme

SHUANGYU HE[1], QIANHONG WU[1], (Member, IEEE), XIZHAO LUO[2], ZHI LIANG[1], DAWEI LI[1], HANWEN FENG[1], HAIBIN ZHENG[1] AND YANAN LI[1]

[1]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[2]School of Computer Science Technology, Soochow University, Suzhou 215006, China

Corresponding author: Xizhao Luo (qhwu@xidian.edu.cn)

This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under Project 2015AA017205, in part by the Natural Science Foundation of China under Project 61772538, Project 61672083, and Project 61370190, and in part by the National Cryptography Development Fund under Project MMJJ20170106.

ABSTRACT Effective cryptocurrency key management has become an urgent requirement for modern cryptocurrency. Although a large body of cryptocurrency wallet-management schemes has been proposed, they are mostly constructed for specific application scenarios and often suffer from weak security. In this paper, we propose a more effective, usable, and secure cryptocurrency wallet-management system based on semi-trusted social networks, therein allowing users to collaborate with involved parties to achieve some powerful functions and recovery under certain circumstances. Furthermore, we employ an identity-based hierarchical key-insulated encryption scheme to achieve time-sharing authorization and present a semi-trusted portable social-network-based wallet-management scheme that provides the features of security-enhanced storage, portable login on different devices, no-password authentication, flexible key delegation, and so on. The performance analysis shows that our proposed schemes require minimal additional overhead and have low time delays, making them sufficiently efficient for real-world deployment.

INDEX TERMS Cryptographic protocols, cryptocurrency, wallet-management protocols.

I. INTRODUCTION

As of November 30, 2017, the value of a bitcoin had gone past $10,000, and the total value of all bitcoins has now surpassed $177 billion. There is no doubt that bitcoin is becoming the most successful cryptocurrency [1]. Bitcoin, proposed by Nakamoto [2], represents a new concept of decentralized currency, in which security is guaranteed by cryptography and consensus mechanisms. For example, the unforgeability of transactions is protected by public-key cryptography. However, modern public-key cryptography is not easy to use by the public because the private key is too long and complex to be managed. Some researchers [3]–[5] have shown that public-key cryptography cannot be deployed for practical usage for numerous reasons, especially for complicated key management.

Furthermore, each bitcoin is associated with a pair of keys, and if someone wants to deal with transactions on different devices, they have to keep the copies of the private keys for each bitcoin address on each device. Compared with traditional centralized economic systems, the pseudonym mechanism cuts off the relationship between actual identities and ownership of bitcoin, and the blockchain technology leads to irreversible transactions. When the private keys become poorly managed, the bitcoin might be stolen or lost forever [6], [7]. As the report in [8] mentioned, by the end of November 2017, bitcoins valuing more than $31 million have been stolen.

Thus, managing cryptocurrency keys effectively and securely is one of the biggest security challenges restricting the widespread usage of modern cryptocurrency. Recently, some enthusiastic cryptocurrency developers have proposed a variety of technologies such as password-derived keys [9], physical storage, secret-key QR codes, password-protect wallets, air-gapped storage, and online wallets. However, these proposed schemes also have various disadvantages. Password-derived key management relies on the security of the password, which is problematic when the password is too weak and easily guessed or is reusable when stolen. Although the physical storage methods fill the gap presented by human memory, they face numerous threats, e.g., they are not secure against theft. Moreover, the security of online wallets relies on the assumption that the centralized online server is trusted,

which is an idealization in realistic scenarios. Password-protected wallets adopt the idea of traditional U-key models used in online banking; if passwords are forgotten, the user loses all of their bitcoins in their wallet.

Overall, although bitcoin and other cryptocurrencies have attracted significant attention and influenced the whole world, there has not been a secure and efficient key management mechanism to faultlessly bridge the gap between the limited memory of humans and the complex key structures of cryptocurrencies. Therefore, proposing an effective and secure cryptocurrency wallet management system currently is the key aspect in making cryptocurrency more applicable to daily payments by individuals.

A. OUR CONTRIBUTION

In this paper, we propose a more effective, usable, and secure cryptocurrency wallet management system based on semi-trusted social networks. In this system, a userаs wallet cannot be recovered by anyone but the user, and the user can collaborate with some involved parties to perform some powerful functions. Our main contributions are as follows:

We propose a semi-trusted portable social-network-based wallet management scheme, which provides the benefits of security-enhanced storage, portable login on different devices, no-password authentication, flexible key delegation, blind wallet recovery, etc. The proposed scheme is also proved to be secure.

We incorporate a key management evaluation framework to evaluate various aspects of our scheme. We conduct comprehensive comparisons among popular cryptocurrency key management schemes following the evaluation framework. The comparisons confirm the advantages of our scheme over other solutions.

Finally, we give a performance analysis of our proposed schemes, and the results show that using such security-enhanced and full-function wallet requires minimal overhead and that the time delay is only a couple of milliseconds, which is sufficiently efficient for real-world utilization. The proposed wallet management mechanism provides a new vision for innovation in making public-key cryptography deployment practical, thereby moving cryptocurrency closer to practical application.

B. PAPER ORGANIZATION

The remainder of this paper is organized as follows. We review related work in Section 2. We formalize the model of our key management scheme and the system security model in Section 3. We introduce some underlying cryptographic algorithms in Section 4. Section 5 discusses our proposed social-network-based cryptocurrency wallet management scheme. We demonstrate the system's security in Section 6 and evaluate the performance of our protocols in Section 7. Finally, Section 8 provides concluding remarks.

II. RELATED WORK

To make cryptocurrency convenient to use, energetic cryptocurrency community developers have invested significant amounts of time and energy in implementing numerous automatic wallet management tools. In this section, we will classify these tools by deployment issues introduced by Eskandari *et al.* [10] and introduce basic design principles and potential risks.

A. KEY STORAGE IN LOCAL STORAGE

Storing keys in local storage is an obvious solution. Typically, the keys are stored in a file system or a pre-assigned directory. When a cryptocurrency client wants to extract the key to execute a transaction, the local storage server can read the pre-configure file system to obtain the key files. Once the client creates a new key pair, it can also update the key files. Although this method is easy to implement, any malware can also read and write the key files without access control. Litke and Stewart [11] found that the primary targets of specialized cryptocurrency-stealing malware are wallets. If malware can tamper with key files, wallet hijacking becomes possible, therein silently stealing miners' earnings.

B. PASSWORD-PROTECTED WALLET

To guarantee the confidentiality of key files, the Multi-Bit Team proposed a password-protected wallet [12]. The password-protected wallet can prevent physical theft, and the security of this system is based on the strength of the password. Although this method can recover the wallets when the user keeps the corresponding password, no effective mechanism can be used to bypass the encryption system once someone forgets the password. The owner must brute force the system by guessing a strong password.

C. OFFLINE STORAGE WALLET

When a wallet is stored on portable media, enhanced security against malware and storage theft can be achieved. However, offline storage wallets have a very conspicuous disadvantage. Compared with software wallets, wallets stored offline take more time to spend funds because the offline storage device is not always readily available. A very interesting case of this wallet is the paper wallet, which is always in the form of a QR code printed on paper. However, offline storage wallets have the drawback in that they are easily lost. Once the USB device becomes missing or the paper with the QR code is blown away, the owner may lose the corresponding wallet forever. Furthermore, the media must be regenerated or updated when the key pool changes.

D. PASSWORD-DERIVED WALLET

In the BIP32 protocol [13], the bitcoin core development group proposed a key generation algorithm called hierarchical deterministic wallets. This type of wallet contains a master key, decided by a single user, for delegating other derived secret keys. The advantage is obvious. With this

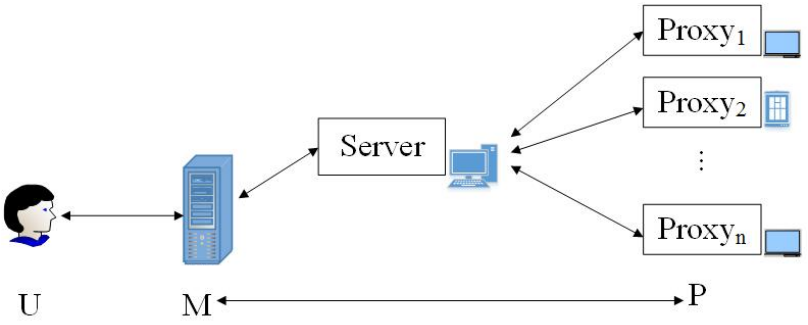

FIGURE 1. The social-network-based cryptocurrency key management framework.

wallet, the users only need to remember a chosen memorable secret such as their favorite food, birthday, or best friends. However, once a derived key is revealed, according to the public algorithms, any malicious actor can obtain the sub-key derived from the derived key. A very popular case of this wallet is called brain wallet. However, Vasek *et al*. [14] recently analyzed the security of the brain wallet. The analysis results show that more than 800 brain wallets were using weak passwords and were worth more than $ 100K.

E. HOSTED WALLETS

A final method of key management is to host the users' keys on a web service. The host service provider provides the servers to store the users accounts and assist users in key management and transactions. The users on these platforms can use the cryptocurrency system even when they have little knowledge about cryptocurrency. However, the hosted wallets are based on a very strong security assumption that the third-party service provider is trusted. However, accidents whereby hosted wallets are stolen by hackers or are lost by the host service provider closing down occur frequently. The statistical data show that there have been over 40 events involving losses of greater than 1000 bitcoin [15].

III. MODEL AND DEFINITION

In this subsection, we give an overview of our social-network-based cryptocurrency wallet management scheme. First, we introduce some important components of our proposed scheme. Then, we give some formalized definitions. The system architecture is depicted in Figure 1.

A. SYSTEM MODEL

- ***User***. A user, denoted as $\mathcal{U}$ in our system, is the owner of the cryptocurrency wallets. Generally, a cryptocurrency wallet is a key pool that consists of a variety of long disordered strings; it is difficult for $\mathcal{U}$ to remember all of these keys by his memory. Therefore, $\mathcal{U}$ always needs a more efficient method to memorize this sensitive information. In our system, $\mathcal{U}$ only needs to link his identity to a special device called the ''Management device'', which acts as a representative to help $\mathcal{U}$ generate and record his wallet.
- ***Management device***. The management device, denoted as $\mathcal{M}$, acts as a plenipotentiary for managing $\mathcal{U}$'s wallets. $\mathcal{M}$ initializes and stores $\mathcal{U}$'s wallet with some extra information generated by a remote server. When $\mathcal{U}$ wants to check the wallet on a proxy device, $\mathcal{M}$ checks the validity of the proxy and gives it the authority to decrypt the pre-encrypted wallet data stored on a remote sever.
- ***Proxy***. The proxy, denoted as $\mathcal{P}$, can be installed on any smart electronic device, e.g., a computer, iPad, and smart phone, on which any $\mathcal{U}$ wants to obtain his wallet to perform a transaction. When receiving the request from some $\mathcal{U}$, $\mathcal{P}$ recognizes the type of wallet and both retrieves and fills the corresponding accounts.
- ***Central Server***. The central server, denoted as $\mathcal{C}$, acts as both centralized data storage and an information transfer and processing platform. $\mathcal{U}$ registers $\mathcal{U}$s and their representative and provides functions such as identity authentication, wallet generation assistance, wallet back-up assistance, wallet restore assistance, and message forwarding. We assume that the servers are honest but curious, which means that $\mathcal{C}$ will execute the protocol by default but also wants to extract some information from the stored data.

1) SYSTEM SETUP

The three parties generate necessary parameters, authenticate each other's identity and establish a secure communication channel among them.

2) REGISTER PHASE

The registers a new $\mathcal{M}$ account to the central server and generates available login token.

3) MASTER KEY BACKUP

The user splits the master password into shares to the central server and other users (friends) without letting the latter know any information about the exact shares.

4) AUTHENTICATED

The user submits the request to the server through a proxy, and the master device authenticates the user's identity and allows the legitimate proxy to obtain the related wallet.

However, without being authorized, any other user cannot obtain any wallet information via the proxy.

5) WALLET RECOVERY

When the user loses the master device or needs to update the master device, he/her recovers the master key using the central server and other users. However, throughout the process, neither the server nor the assisting users can know any information about the master password.

B. SECURITY MODEL

Before giving the detailed security model, we must emphasize two aspects. First, we assume that the running environment is secure, which means that the program code is not tampered with and that there are no keyboard or memory loggers recording any wallet account information. Moreover, we also assume that all the underlying algorithms are implemented correctly and that all the parameters meet the security requirements.

In our system, the main security goal is to ensure the confidentiality of the wallet accounts. Considering the semi-trusted server, the adversary can have access to the server's wallet database, but it can only read certain database items. In addition, once $\mathcal{M}$ is obtained by an attacker, the attacker can obtain access to M and can read or write the encrypted database of M. This assumption holds, for example, when someone is the best friend or a relative of $\mathcal{M}$ąś owner; it is easy for them to have access to M when emergencies occur.

To define the adversary ability, we first introduce the definition of the wallet management database, which utilizes the definition of the password management database proposed by Gasti and Rasmussen [16]. Although this definition is used to analyze the security of password management databases, it is also suitable for our scheme because in a sense the cryptocurrency wallet accounts can also be seen as a variant of a password. The wallet management database can be defined by 3 probabilistic polynomial-time algorithms, *Setup*, *Create*, and *Open*, as follows.

Definition 1 (Wallet Management Database): The wallet management database can be defined by 3 probabilistic polynomial-time algorithms, *Setup*, *Create*, and *Open* as follows.

$mk \leftarrow Setup(I, \lambda)$: Given a security parameter λ, the algorithm outputs a key pair mk corresponding to the identity I.

$DB \leftarrow Create(mk, RC)$: Given a system key pair mk and a set of records $RC = \{record_1, record_2, \ldots, record_n\}$, the algorithm outputs a database DB, where $DB = \{db_i\}_{i=1}^{n}$.

$RC \leftarrow Open(mk, DB)$: Given a system key pair mk and a database DB, the algorithm outputs a database RC.

Then, we defined the adversary model as follows:

$KG(\cdot)$: For a query with identity I, the oracle runs $SetupI$, λ and returns mk.

$PG(\cdot)$: For a query with identity I and record set RC, the oracle runs $Create(mk, RC)$ and returns DB.

$CG(\cdot)$: For a query with identity I and DB, the oracle runs $Open(mk, DB)$ and returns RC.

$Test(\cdot)$: This oracle runs only once during the whole game with adversary execution. The adversary first chooses 2 RS'_0, RS'_1 to the oracles. The oracle randomly selects a bit $b \in \{0, 1\}$ and runs $Create(mk, RS'_b)$; then, it outputs DB'_b.

Finally, the adversary guesses a random bit b'. If $b' = b$, we claim that the adversary wins the game.

If the adversary only queries KG, PG, CG, and $Test$ oracles, we claim that this adversary is $\mathcal{A}_1$; the advantage to attack the database is defined as $Adv_{\mathcal{A}_1}(\lambda) = | Pr[b = b'] - \frac{1}{2} |$.

Definition 2 (Indistinguishability of Chosen Database (IND-CDBA)): A wallet management database is said to be IND-CDBA secure if, for all PPT adversaries $\mathcal{A}_1$, $Adv_{\mathcal{A}_1}$ is negligible.

IV. BUILDING BLOCK

Definition 3 (Identity-Based Hierarchical Key-Insulated Encryption [17]): An ℓ's level key insulated encryption consists of six algorithms (*PGen*, *KGen*, *UpGen*, *DkUp*, *Enc*, and *Dec*), which are defined as follows [].

$(pp, mk) \leftarrow PGen(\lambda, \ell)$: The *PGen* algorithm, a probabilistic polynomial-time algorithm, is used to initialize the system parameters. The algorithm takes the system security parameter λ and the depth of system ℓ as input, and then, it initializes the system's public parameter pp and master key mk.

$(dk_{I,0}, hk_{I,0}^{(1)}, \ldots, hk_{I,0}^{(\ell)}) \leftarrow KGen(mk, I)$: The *KGen* algorithm, also a probabilistic polynomial-time algorithm, takes the system master key mk and the user's identity $I \in \mathbb{I}$ as input; then, it outputs the initial decryption key $dk_{I,0}$ and the helper keys $hk^{\langle i \rangle}{}_{I,0}$ for the ith level device, where $i \in \{1, \ldots, \ell\}$.

$(\delta_{T_{i-1}}^{i-1}) \leftarrow UpGen(hk_{I,t_i^{(j)}}^{(i)}, time)$: The *UpGen* algorithm takes the ith-level helper key at $t_i^{(j)}$ and the current $time$ as input; then, it outputs the decryption update message $\delta_{T_{i+1}}^{i+1}$. We remark that there is a time period map function $T_i(time)$ introduced by Watanable and Shikata [17], in which a long time period is separated into shorter periods. We set $\mathcal{T}_i = \{t_i^{(0)}, \ldots, t_i^{(j)}\}$ for the ith level. The ith time period is always longer than the time period of its lower levels. For the ith level, when the current $time$ matches the jth period, we have $T_i(time) = t_i^{(j)}$.

$(hk_{I,T_i}^{i}) \leftarrow DkUp(hk_{I,t_i^{(j)}}^{(i)}, \delta_{T_i}^{i})$: The *DkUp* algorithm takes the ith-level helper key $hk^{\langle i \rangle}{}_{I,t_i^{(j)}}$ at $t_i^{(j)}$ and the decryption update $\delta_{T_i}^{i}$ as input; then, it updates the current time's decryption key as hk_{I,T_i}^{i}.

$(C, time) \leftarrow Enc(I, time, M)$: The *Enc* algorithm takes the user's identity I, a message M and a fixed time $time$ as input; then, it outputs the corresponding ciphertext C and the $time$.

$(M) \leftarrow Dec(C, time, dk_{I,T_\ell})$: The *KGen* algorithm, a probabilistic polynomial-time algorithm, takes the decryption

key dk_{I,T_ℓ}) in *time*, the fixed *time* and a valid ciphertext C as input; then, it outputs a plaintext M.

Definition 4 (Digital Signature Scheme): A digital signature scheme consists of three polynomial-time algorithms, (*KeyGen*, *Sign*, *Verify*), denoted as $(\mathcal{K}, \mathcal{S}, \mathcal{V})$.

$(PK, SK) \leftarrow KeyGen(\lambda)$. A probabilistic key generation algorithm that, given a security parameter λ, outputs the key pair (PK, SK).

$\sigma \leftarrow Sign(SK, m)$. A probabilistic signature algorithm that takes the secret key SK and the message m as input and then outputs the signature σ.

1 or $\perp \leftarrow Verify(PK, \lambda)$. A probabilistic verification algorithm that takes the public key PK and the signatureλ as input and then outputs 1 if the signature is valid or outputs $\perp$.

Definition 5 (Secret Sharing): A secret sharing scheme allows one to divide a secret into$n$ shares; if any t or more shares are combined, they can reconstruct the origin secret with their shares. The first secret sharing scheme was proposed by Shamir [18] as follows: To divide the secret a_0 into pieces, pick a random t-1-degree polynomial $q(x) = a_0 + a_1x + \ldots + a_{t-1}x^{t+1}$; let $q(i)$ be the ith share. Given any subset of t and these $q(i)$ values, the coefficients of $q(x)$ can be recovered by Lagrange interpolation. Finally, let $x = 0$, and the value $q(0) = a_0$ is the reconstructed secret. We defined the Shamir secret sharing scheme $\prod_{SS} = (f(n,t), \mathcal{R})$,, where $f(n,t)$ denotes the splitting algorithm of the secret sharing scheme and $\mathcal{R}$ is the secret reconstruction algorithm.

Definition 6 (Digital Signature Scheme): A digital signature scheme consists of the polynomial-time algorithms (*KeyGen*, *Sign*, *Verify*), denoted as $(\mathcal{K}, \mathcal{S}, \mathcal{V})$.

$(PK, SK) \leftarrow KeyGen(\lambda)$. A probabilistic key generation algorithm that, given a security parameter λ, outputs the key pair (PK, SK).

$\sigma \leftarrow Sign(SK, m)$. A probabilistic signature algorithm takes the secret key SK and the message m as input and outputs the signature σ.

1 or $\perp \leftarrow Verify(PK, \lambda)$. A probabilistic verification algorithm that takes the public key PK and the signatureλ as input and either outputs 1 if the signature is valid or outputs $\perp$.

Definition 7 (Secret Sharing): A secret sharing scheme allows one to divide a secret into n shares whereby if any t or more shares are combined, they can reconstruct the origin secret with their shares. The first secret sharing scheme was proposed by Shamir [18] as follows: To divide the secret a_0 into pieces, pick a random t-1-degree polynomial $q(x) = a_0 + a_1x + \ldots + a_{t-1}x^{t+1}$; let $q(i)$ be the ith share. Given any subset of t of these $q(i)$ values, the coefficients of $q(x)$ can be recovered by Lagrange interpolation. Finally, let $x = 0$, and the value $q(0) = a_0$ is the reconstructed secret. We define the Shamir secret sharing scheme $\prod_{SS} = (f(n,t), \mathcal{R})$, where $f(n,t)$ denotes the splitting algorithm of the secret sharing scheme and $\mathcal{R}$ is the secret reconstruction algorithm.

V. SOCIAL-NETWORK-BASED CRYPTOCURRENCY WALLET MANAGEMENT SCHEME

We propose a portable semi-trusted social-network-based cryptocurrency wallet management scheme. The utilized cryptographic building blocks include a Shamir secret sharing scheme $\prod_{SS} = (f(n,t), \mathcal{R})$, a digital signature scheme $\prod_{sig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$, an identity key-insulated encryption $\prod_{IKE} = (PGen, KGen, UpGen, DkUp, Enc, Dec)$, and symmetric encryption $(\mathcal{E}, \mathcal{D})$.

Protocol 1 (System Setup): When the system is set up, it initializes several system parameters. The $\mathcal{S}$ first generates an RSA key pair (e, N, d), where e, N is the public key and d is the private key. Then, it chooses a symmetric encryption $(\mathcal{E}, \mathcal{D})$ and a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ and generate the parameters for $\prod_{SS}$ and $\prod_{Sig}$ with the security parameter λ. To establish a secure channel and authentication between each other, the system generates he initial certificates for $\mathcal{M}$, $\mathcal{C}$ and $\mathcal{P}$.

Protocol 2 (Registration Phase): When a $\mathcal{U}$ wants to be a new user of the system. He/she should first bind their identity with an $\mathcal{M}$ and generate a valid login credential on $\mathcal{M}$; then, they should submit registration information to $\mathcal{C}$:

1) $\mathcal{U}$ registers a system account, and $\mathcal{M}$ generates the necessary parameters and secure channel to $\mathcal{C}$. The authentication between $\mathcal{U}$ and $\mathcal{M}$ can use the popular bio authentication method.
   - $\mathcal{U}$ first chooses a system account $ID_\mathcal{U}$ and then generates his/her personal authentication information Per. Let $A = (per, ID_\mathcal{U})$.
   - $\mathcal{M}$ chooses a random number r, computes $A' = r^e H(A) \bmod N$, and submits $(ID_\mathcal{U}, A')$ to $\mathcal{C}$.
   - After $\mathcal{C}$ receives the message from $\mathcal{M}$, it computes $\bar{A} = A'^d \bmod N$, and then, it returns $\bar{A}$ and a challenge to $\mathcal{M}$
   - After receiving the response message, $\mathcal{M}$ computes $\hat{A} = \bar{A}r^{-1} \bmod N$ and checks whether $\hat{A}^e = A \bmod N$. If so, $\mathcal{M}$ generates a secret key $SK_\mathcal{M} = H(A \| \hat{A})$ and then computes the related public key $PK_\mathcal{M}$ using $\mathcal{K}(\lambda)$.
   - Then, $\mathcal{M}$ detects its individual device code and running environment, and then, it obtains distinguishable information $Dinfo$. Next, it computes its $SK_D = H(SK_\mathcal{M} \| Dinfo)$ and corresponding public key $PK_\mathcal{D}$ using $\mathcal{K}(\lambda)$.
   - $\mathcal{M}$ establishes $(H(challenge \| PK_\mathcal{M} \| PK_D \| ID_\mathcal{U}), ID_\mathcal{U}, PK_\mathcal{M}, PK_D)$ as registration message and generates $\sigma_0 = \mathcal{S}(reg, SK_{cert_\mathcal{M}})$; then, it sends (reg, σ_0) to $\mathcal{C}$.
   - $\mathcal{C}$ runs $\mathcal{V}(PK_{cert_\mathcal{M}}, \sigma_0)$ to check whether the signature is valid. If so, it records the message reg in its registration database.
2) $\mathcal{M}$ initializes local wallet management. $\mathcal{M}$ randomly chooses a long-bit string mp as a delegated key seed.

3) $\mathcal{M}$ creates a local wallet database and inserts the cryptocurrency wallet record RS_i for currency $(type_j, count_k)$:
   - For $M_i = (mp, type_j, count_k)$, $\mathcal{M}$ chooses a random number r_i and computes $M_i' = r_i^e H(M_i) \bmod N$; then, $\mathcal{M}$ submits (M_i') to $\mathcal{C}$.
   - $\mathcal{C}$ computes $\bar{M}_i = (M_i')^d \bmod N$ and then returns $\bar{M}_i$ to $\mathcal{M}$.
   - $\mathcal{M}$ computes $\hat{M}_i = \bar{M}_i r_i^{-1} \bmod N$ and checks whether $\hat{M}_i^e = M_i \bmod N$. If so, $\mathcal{M}$ generates a private key for this cryptocurrency as $prik_i = H(mp \| \hat{M}_i)$, and then, it generates the corresponding address $Addr_i$ by running the related address generation algorithm. The w_i is in the form of $(Addr_i, prik_i)$
   - $\mathcal{M}$ calls $\prod_{IKE}$ to generate the record's encryption key and encrypts the record as $C^{(i)} \leftarrow Enc(I, time, w_i)$. We should note that in our system, the HIKE system is reduced to a 2-level HIKE, which means that $\mathcal{M}$ only needs to update the decryption key for $\mathcal{P}$.
   - $\mathcal{M}$ then generates a unique tag_i and a hash value of this record as $MAC_i = H(tag_i \| C^{(i)} \| row_i)$. Finally, the record RS_i is submitted to $\mathcal{C}$ and stored in the form of

$$(row_i, tag_i, C^{(i)}, mac_i).$$

Protocol 3 (Master Key Backup): The masker key is backed up as follows.

1) $\mathcal{M}$ divides mp into several parts and then calls the $\prod_{SS}$ algorithm to perform the secret sharing with friends in a social network.
   - $\mathcal{M}$ randomly selects 2 random numbers s_0, s_1 such that $mp = s_0 \bigoplus s_1$.
   - $\mathcal{M}$ chooses a set of friend identities $\{ID_0, ID_1, \ldots, ID_n\}$ and then computes $s_{1,i} = f(ID_i, s_1)$ for $i = 1, \ldots, n$.
2) $\mathcal{M}$ processes and delivers the shared secret.
   - $\mathcal{U}$ submits his/her personal authentication information $Pinfo$ to $\mathcal{M}$.
   - $\mathcal{M}$ chooses a random number $\tilde{r}$ and computes $Pinfo' = \tilde{r}^e H(Pinfo) \bmod N$; then, $\mathcal{M}$ submits $(Pinfo')$ to $\mathcal{C}$.
   - After $\mathcal{C}$ receives the message, it computes $\bar{Pinfo} = (Pinfo')^d \bmod N$ and then returns $\bar{Pinfo}$ to $\mathcal{M}$.
   - After receiving the above messages, $\mathcal{M}$ computes $\hat{Pinfo} = \bar{Pinfo}\tilde{r}^{-1} \bmod N$ and checks whether $\hat{Pinfo}^e = Pinfo \bmod N$. If so, it encrypts $S_0 = \mathcal{E}(s_0, \hat{Pinfo})$, $S_{1,i} = \mathcal{E}(s_{1,i}, \hat{Pinfo})$, where $i = 1, \ldots, n$.
   - Finally, $\mathcal{M}$ sends S_0 to $\mathcal{C}$ and sends $S_{1,i}$ to ID_i, where $i = 1, \ldots, n$.

Protocol 4 (Authenticated): When a user decides to perform a transaction in an untrusted environment through a proxy software and obtain an assigned wallet account, he/she has to do as follows:

1) $\mathcal{U}$ launches $\mathcal{P}$ and initializes the communication parameters.
   - According to the system parameters, $\mathcal{P}$ chooses a random number $a \in Z_p^*$ and computes $Q = aP$, where P is the generator of an elliptic curve used in $\prod_{sig}$. Then, $\mathcal{P}$ runs $\mathcal{S}(\{Q_a, \tau_0\}, SK_{cert_{\mathcal{P}}})$ to generate the signature σ_1 of the random number Q and timestamp τ_0.
   - Finally, $\mathcal{P}$ transmits the (Q, τ_0, σ_1) to $\mathcal{U}$ and transforms (Q_a, τ_0, σ_1) into a QR code on the device screen.
2) $\mathcal{U}$ authenticates his/her identity to $\mathcal{M}$. $\mathcal{M}$ proves to $\mathcal{C}$ that it has received the request from $\mathcal{P}$.
   - $\mathcal{U}$ should authenticate his identity to $\mathcal{M}$ by bio authentication (e.g., fingerprints). Then, $\mathcal{U}$ can use $\mathcal{M}$ to scan the QR code provided by $\mathcal{P}$, which means that $\mathcal{U}$ needs some wallet information on $\mathcal{P}$.
   - Next, $\mathcal{M}$ can obtain (Q_a, τ_0, σ_1) from the QR code. Then, it will run $\mathcal{V}(\sigma_1, PK_{cert_{\mathcal{P}}})$ to check the validity of this signature. If it is valid, $\mathcal{M}$ chooses a random number $b \in Z_p^*$ and computes $Q_b = bP$.
   - Finally, $\mathcal{M}$ outputs a signature σ_2 by running $\mathcal{S}(\{H(Q_b \| Q_a \| \tau_0 \| \tau_1)\}, SK_D)$ and then sends (Q_b, τ_1, σ_2) to $\mathcal{C}$.
3) Using $\mathcal{C}$, $\mathcal{P}$ and $\mathcal{M}$ establish a secure communication channel.
   - $\mathcal{C}$ first runs $\mathcal{V}$ to verify the validity of both σ_1 and σ_2. If they are valid, $\mathcal{C}$ considers that $\mathcal{M}$ allows $\mathcal{P}$ to query $\mathcal{M}$'s wallet data stored in its database.
   - Then, $\mathcal{C}$ forwards (Q_b, τ_1, σ_2) to $\mathcal{P}$.
   - After verifying σ_2, $\mathcal{P}$ and $\mathcal{M}$ can negotiate a session key TK by computing $TK = aQ_b = bQ_a = abP$.
4) $\mathcal{P}$ obtains the assigned cryptocurrency address and private key from record RS_i.
   - $\mathcal{M}$ takes the current time and his own IKE helper key as input, runs the $UpGen(\cdot)$ algorithm, and obtains the HIKE decryption key update information $\delta_{I,T_i}^{(0)}$.
   - $\mathcal{M}$ encrypts $\delta_{I,T_i}^{(0)}$ by running $\mathcal{E}(\delta_{I,T_i}^{(0)}, TK)$ and then transmits it to $\mathcal{P}$ over the secure channel.
   - After receiving the messages, $\mathcal{P}$ runs $DkUp(hk_{T_i}^0, \delta_{I,T_i}^{(0)})$ to update its decryption key.
   - Finally, $\mathcal{P}$ queries the record database of $\mathcal{C}$ with the record tag tag_i, and then, it uses $Dec(dk_{I,T_0}, C, time)$ to obtain the related wallet record.

Protocol 5 (Wallet Recovery): When a $\mathcal{U}$ loses or changes his/her $\mathcal{M}$, our system can support him/her in recovering their wallet without any extra information.

1) $\mathcal{U}$ initializes a new $\mathcal{M}$ and authenticates him or herself to $\mathcal{S}$.

- $\mathcal{U}$ first chooses a system account $ID_{\mathcal{U}}$ and then generates his/her personal authentication information Per. Let $A = (per, ID_{\mathcal{U}})$.
- $\mathcal{M}$ chooses a random number r and computes $A' = r^e H(A) \bmod N$; then, $\mathcal{M}$ submits $(ID_{\mathcal{U}}, A')$ to $\mathcal{C}$.
- When $\mathcal{C}$ receives the message from $\mathcal{M}$, it computes $\bar{A} = A'^d \bmod N$; then, it returns $\bar{A}$ and a challenge to $\mathcal{M}$.
- After receiving the response message, $\mathcal{M}$ computes $\hat{A} = \bar{A}r^{-1} \bmod N$ and checks whether $\hat{A}^e = A \bmod N$. If so, $\mathcal{M}$ generates a secret key $SK_{\mathcal{M}} = H(A\|\hat{A})$ and then computes the related public key $PK_{\mathcal{M}}$ using $\mathcal{K}(\lambda)$.
- For a real-time τ_3, $\mathcal{M}$ generates the signature $\sigma_3 = \mathcal{S}(\tau_3, SK_{\mathcal{M}})$ and then submits (τ, σ_3) to $\mathcal{S}$.
- $\mathcal{S}$ executes $\mathcal{V}(\sigma_3, \tau_3, PK_{\mathcal{M}})$, where the $PK_{\mathcal{M}}$ generated by $\mathcal{U}$ in the Registration Phase is stored by $\mathcal{C}$. If it outputs 1, the $\mathcal{C}$ remarks this $\mathcal{U}$ and related $\mathcal{M}$.
- Then, $\mathcal{M}$ detects its individual device code and running environment and obtains distinguishing information $Dinfo'$; then, it computes its $SK_{D'} = H(SK_{\mathcal{M}}\|Dinfo')$ and corresponding public key $PK_{D'} \leftarrow \mathcal{K}(\lambda)$. Next, $\mathcal{M}$ submits $PK_{D'}$ to $\mathcal{S}$.

- $\mathcal{S}$ replaces the device public key with $PK_{D'}$ and updates the master device information associated with $\mathcal{U}$.

2) $\mathcal{M}$ submits its bound secret information to trigger the wallet recovery mechanism with $\mathcal{C}$.
   - $\mathcal{U}$ submits his/her personal authentication information $Pinfo$ to $\mathcal{M}$.
   - Next, $\mathcal{M}$ chooses a random number $\tilde{r'}$ and computes $Pinfo' = \tilde{r'}^e H(Pinfo)$. mod N. Then, $\mathcal{M}$ submits $(Pinfo')$ to $\mathcal{C}$.
   - After $\mathcal{C}$ receives the message, it computes $\bar{Pinfo} = (Pinfo')^d \bmod N$ and then returns $\bar{Pinfo}$ to $\mathcal{M}$.
   - After receiving the above message, $\mathcal{M}$ computes $\hat{Pinfo} = (\bar{Pinfo})\tilde{r'}^{-1} \bmod N$ and checks whether $\hat{Pinfo}^e = Pinfo \bmod N$.
3) $\mathcal{M}$ collects the shared secret from other social network participants.
   - $\mathcal{S}$ asks the other social network participants with $ID_1, ID_2, \ldots, ID_n$ to return the shared secret deposited by them.
   - $\mathcal{M}$ should receive at least $t + 1$ secret sharing parts from different participants and S_0 from $\mathcal{C}$. The received part is denoted as $S_{1,i}, S_{1,i+1}, \ldots, S_{1,i+t}$.
   - $\mathcal{M}$ decrypts $S'_0 = \mathcal{D}(S_0, \hat{Pinfo})$ and $S'_{1,i} = \mathcal{D}(S_{1,i}, \hat{Pinfo})$, where $i \in [1, \ldots, n]$. Next, $\mathcal{M}$ recovers s'_1 by executing $\mathcal{R}(s_{1,i}, \ldots, s_{1,i+t})$. $\mathcal{M}$'s master key can be recovered by combining $s'_1 \bigoplus S'_0$.
4) Using the master key of $\mathcal{M}$, $\mathcal{M}$ can easily recover all the wallet accounts according to **protocol 2** and **protocol 3.**

VI. SECURITY ANALYSIS

Theorem 1: Assuming that an encryption algorithm is an IND-CPA-secure encryption scheme and that the hash function H is a collision-resistant hash function and also a secure pseudorandom function, the proposed system is $IND - CDBA_{\mathcal{A}} - secure$.

The main security concern of our system is that the attacker might obtain access to the wallet management database and learn some sensitive information. To formalize the security demand against such an attacker, we introduce the definition of the wallet management database and IND-CDBA security definition. Our system can achieve IND-CDBA security since each address and private key record is encrypted with the WS16 HIKE scheme, which has been proven to be IND-KE-CCA secure under the SXDH assumption. By standard arguments, it is shown that IND-CPA security implies IND-CDBA security. Thus, the wallet management database that we built is IND-CDBA secure, while the underlying HIKE scheme can provide a much stronger security guarantee than standard IND-CPA security. As such, the attacker cannot extract any information from our wallet management database.

A. EVALUATION BASED ON KEY MANAGEMENT FRAMEWORK

In this section, we analyze the usability and security using a evaluation framework proposed by Eskandari *et al.* [10]. This evaluation framework is used to evaluate the security and usability of bitcoin key management approaches. The security evaluation considers the attacks that occur in practice such as malware, physical theft, and password loss. In addition, the usability evaluation focuses on novice users. The evaluation defines 14 core tasks involving cryptocurrency key management, including client launch time, transaction execution time, cross-device capabilities, and main wallet recovery. The evaluation of our scheme is shown in Table 1; we compare our scheme with Bitcoin core [19], MultiBit [12], Bitaddress [20], Bitcoin Blockchain.info and Brainwallet [21]. These chosen schemes represent approaches based on local storage methods, password-protected schemes, offline storage password-derived key methods, and hosted wallets, respectively. A black dot means that the system can satisfy this property. The black circle means that the system can partially satisfy the corresponding property. Empty means that the system cannot satisfy this property. In the security category, our system achieves almost all the security goals. First, the chosen encryption is identity-based key-insulated encryption, with which the proxy's decryption key is available in a fixed number of periods; beyond this number of periods, the proxy loses the ability to obtain the wallet information. Therefore, although we cannot guarantee that the run environment of the proxy is trusted, we can mitigate losses in that, even when malware can obtain a decryption key, it can only obtain a part of a wallet and not all of it. Second, our system does not suffer from offline issues; the full

TABLE 1. Evaluation framework for security.

Category	Example	Malware Resistant	Key Kept Offline	No Trusted Third Party	Resistant to Physical Theft	Resistant to Physical Observation	Resilient to Password Loss	Resilient to Key Churn	Immediate Access to Funds	No New User Software	Cross-device Portability
mix	Our scheme	○	○		●	●	●	●	●		●
Local Storage	Bitcoin Core			●		●	●	●	●		
Password-protected	MultiBit		○	●	○	●		●	●		
Offline Storage	Bitaddress	○	●	●			●				●
Password-derived	Brainwallet		●	●	○			●	●	●	●
Hosted Wallet (Hybrid)	Blockchain	○	○			●	●	●	●	●	
cash		●	●	●		●	●	●	●	●	●

TABLE 2. Identity-based key-insulated encryption algorithm execution time cost.

ℓ	$PGen$	$KGen$	$UpGen$	DkUp	Enc	Dec
1	162.2 ms	102.1 ms	80.3 ms	<1 ms	71.9 ms	60.7 ms
2	196.7 ms	148.2 ms	103.2 ms	<1 ms	83.4 ms	60.3 ms
3	231.2 ms	238.3 ms	125.7 ms	<1 ms	95.1 ms	60.5 ms
4	265.4 ms	283.7 ms	148.2 ms	< 1 ms	106.8 ms	60.8 ms
5	301.5 ms	328.9 ms	169.5 ms	< 1 ms	118.6 ms	61 ms

sensitive massage is encoded before it is delivered. Therefore, according to the evaluation standard for the key being kept offline, we can say that our scheme can partially achieve these goals. The master device is bound to a user in protocol 1, and it should authenticate the user's identity by his/her bio information, as described in protocol 4. The account text and proxy authorization are automatically executed by the master device and proxy program, which avoid physical observation attacks. By executing protocol 6, our scheme can recovery the master device secret key, which can be used to regenerate the wallet key with protocols 2 and 3. Therefore, we claim that our scheme is resilient to password loss.

VII. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of the main algorithm in terms of running time. The underlying cryptographic schemes are quite efficient and take little time, except the identity-based key-insulated scheme. Hence, in this section, we will show that using such security-enhanced and full-function wallet requires minimal additional overhead and that the time delay is only a couple of milliseconds, making it sufficient efficient for real-world deployment. We implement our identity-based key-insulated scheme with the Java Pairing-Based Cryptography (JPBC) library to evaluate the time complexity of the required cryptographic operations. To generate the prime-order bilinear groups, we use the Type A elliptic curve, which is expressed as $y^2 = x^3 + x$. All the primes chosen in our experiment are 512 bits and generated randomly by the system. The experiment was executed on a computer with a 3.3 GHz Intel Pentium G3260 CPU, 4 GB of RAM and a 64-bit Ubuntu 15.04 operating system.

Using the JPBC Library in the aforementioned PC environment, the time required to perform a multiplication in a bilinear group G_1 is approximately 0.06 ms, and in G_2, it is approximately 0.01 ms. The time to perform an exponentiation in a bilinear group G_1 is approximately 11.5 ms, and in G_2, it is approximately 11.3 ms. The time to compute a pairing in a bilinear group is approximately 12.0 ms.

Table 7 shows the average time cost for 50 tests with $\ell \in [1, 5]$. Our IKE scheme in our system only requires 2 levels, where $\ell = 1$. Thus, each partial time is less than 25 ms. The introduced cryptographic algorithm has almost no influence.

VIII. CONCLUSION

With the rapid increase in the value of cryptocurrencies (e.g., Bitcoin), there is an urgent need for a tool that can provide a means of using cryptocurrencies securely and efficiently. The main challenge is the gap between the limited memory of human beings and the complex key structures of cryptocurrencies.

We propose an effective, usable and secure cryptocurrency wallet management system based on a semi-trusteed social network, herein implementing a HIKE scheme, a secret-sharing scheme, a signature scheme and a symmetric encryption scheme as building blocks. We present five protocols under our system, which imply that the system enjoys the properties of security-enhanced storage, portable login on different devices, no-password authentication, flexible key delegation, blind wallet recovery, etc. In addition, we take IND-CDBA security as the security definition for the systems. Finally, we show that if the underlying HIKE scheme is IND-CPA-secure, our system is IND-CDBA-secure.

REFERENCES

[1] BBC News. (2017). *Bitcoin Crosses $10,000 Milestone*. [Online]. Available: http://www.bbc.com/news/technology-42137408
[2] S. Nakamoto. (2012). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf
[3] N. Unger *et al.*, "SoK: Secure messaging," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 232–249.
[4] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller, "How to make secure email easier to use," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2005, pp. 701–710.
[5] S. Sheng, L. Broderick, C. A. Koranda, and J. J. Hyland, "Why Johnny still can't encrypt: Evaluating the usability of email encryption software," in *Proc. Symp. Usable Privacy Secur.*, 2006, pp. 3–4.
[6] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 397–411.
[7] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2017, pp. 456–474.
[8] J. Lee. (2017). *Bitcoin Falls After $31 Million Theft of Cryptocurrency Tether*. [Online]. Available: http://www.livemint.com/Money/t6VdmdtcBq17DcR1BAcfOJ/Bitcoin-falls-after-31-million-theft-of-cryptocurrency-teth.html
[9] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, document RFC 2898, Sep. 2000.
[10] S. Eskandari, J. Clark, D. Barrera, and E. Stobert, "A first look at the usability of Bitcoin key management," in *Proc. Workshop Usable Secur.*, 2015, pp. 1–10, doi: https://doi.org/10.14722/usec.2015.23015.
[11] P. Litke and J. Stewart. (2014). *Cryptocurrency-Stealing Malware Landscape*. [Online]. Available: https://www.secureworks.com/research/cryptocurrency-stealing-malware-landscape
[12] M. Team. *Multibit*. Accessed: 2016. [Online]. Available: https://multibit.org
[13] P. Wuille. *Bip32: Hierarchical Deterministic Wallets*. Accessed: 2014. [Online]. Available: https://github.com/genjix/bips/blob/master/bip-0032.md
[14] M. Vasek, J. Bonneau, C. K. Ryan Castellucci, and T. Moore, "The Bitcoin brain drain: A short paper on the use and abuse of Bitcoin brain wallets," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science). New York, NY, USA: Springer, 2016.
[15] 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 *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 104–121.
[16] P. Gasti and K. B. Rasmussen, "On the security of password manager database formats," in *Proc. ESORICS*, 2012, pp. 770–787.
[17] Y. Watanabe and J. Shikata, "Identity-based hierarchical key-insulated encryption without random oracles," in *Public-Key Cryptography—PKC*. New York, NY, USA: Springer, 2016, pp. 255–279.
[18] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[19] B. C. Developers. *Bitcoin Core*. Accessed: 2001. [Online]. Available: https://bitcoin.org
[20] Pointbiz. *JavaScript Client-Side Bitcoin Wallet Generator*. Accessed: 2015. [Online]. Available: https://bitaddress.org
[21] *Brainwallet*. [Online]. Available: https://brainwallet.github.io/

SHUANGYU HE was born in Zhejiang, China. He received the B.S. degree from Xidian University, Xi'an, China, in 2012. He is currently pursuing the Ph.D. degree in electronic and information engineering, Beihang University, Beijing, China.

His research interests include applied cryptography and mobile security.

QIANHONG WU (M'11) was born in Sichuan, China. He received the Ph.D. degree in Cryptography from Xidian University, Xi'an, China, in 2004.

Since 2004, he has been with the University of Wollongong, Australia, as an Associate Research Fellow; with Wuhan University, China, as an Associate Professor; and with Universitat Rovira i Virgili, Catalonia, as a Research Director. He is currently with Beihang University, China, as a Professor.

Prof. Wu is a member of IACR and ACM. His current research interests include cryptography, data security and privacy, and information theory.

XIZHAO LUO received the B.S. and M.S. degrees from the Xi'an University of Technology, Xi'an, China, in 2000 and 2003, respectively, and the Ph.D. degree from Soochow University, China, in 2010.

He held a postdoctoral position with the Center of Cryptography and Code, School of Mathematical Science, Soochow University, for three years. His main fields of interest are cryptography and computational complexity.

ZHI LIANG was born in Shandong, China. He received the B.S. degree from Beihang University, Beijing, China.

He is currently pursuing the master's degree in electronic and information engineering with Beihang University. His research interests include applied cryptography and mobile security.

DAWEI LI was born in Shandong, China. He received the B.S. degree from Beihang University, Beijing, China, where he is currently pursuing the Ph.D. degree in electronic and information engineering.

His research interests include applied cryptography and blockchain.

HANWEN FENG was born in Sichuan, China. He received the B.S. degree from Beihang University, Beijing, China, where he is currently pursuing the Ph.D. degree in electronic and information engineering.

His research interests include applied cryptography and lattice cryptography.

HAIBIN ZHENG was born in Shandong, China. She received the B.S. degree from Shandong University, Jinan, China. She is currently pursuing the Ph.D. degree in electronic and information engineering with Beihang University, Beijing, China.

Her research interests include applied cryptography and mobile security.

YANAN LI was born in Shandong, China. She received the B.S. degree from Shandong University, Jinan, China. She is currently pursuing the master's degree in electronic and information engineering with Beihang University, Beijing, China.

Her research interests include applied cryptography and mobile security.