IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 8, 2018, accepted July 12, 2018, date of publication July 17, 2018, date of current version August 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2856864

SBLWT: A Secure Blockchain Lightweight Wallet Based on Trustzone

WEIQI DAI[1,2], JUN DENG[1], QINYUAN WANG[1], CHANGZE CUI[1], DEQING ZOU[1,2], AND HAI JIN[1]

[1]Services Computing Technology and System Lab, Big Data Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China

Corresponding author: Deqing Zou (deqingzou@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602200, in part by the Shenzhen Fundamental Research Program under Grant JCYJ20170413114215614, and in part by the Fundamental Research Funds for the Central Universities under Grant HUST2016YXMS087.

ABSTRACT Due to the increasing total value of the digital currency, the security of encryption wallet is becoming more and more important. The hardware-based wallet is safe, but it is inconvenient because users need to carry an additional physical device, the software-based wallet is convenient, but the safety cannot be guaranteed. All these wallets need to synchronize the blockchain, while most current mobile devices do not have the capability to store all blocks. To solve these problems, mobile devices can use simplified payment verification (SPV). Nevertheless, in existing methods, there is no good way to protect the verification process of the transaction. In this paper, we design a secure blockchain lightweight wallet based on Trustzone to protect SPV. It is more portable compared with the hardware wallet, and safer than the software wallet. Through the isolation, it can also protect the private key and the wallet's address from being stolen by the attackers no matter whether the Rich OS is malicious or not. Meanwhile, it can protect the verification process by verifying transactions in the secure execution environment (SEE), and keep the local block headers unreadable directly from the Rich OS through encryption. We deploy it on the RASPBERRY PI 3 MODEL B development board. The result of the experiment shows that it has little impact on the system.

INDEX TERMS Bitcoin, SPV, wallet, Trustzone, blockchain.

I. INTRODUCTION

Bitcoin is a decentralized digital currency invented by Satoshi Nakamoto [1], which can do everything that traditional currency can do. After many years' development, bitcoin has become more and more popular in the financial area of science and technology. Currently, the highest total market value of all digital currencies has reached nearly 800 billion USD (bitcoin accounts for more than one third of it) on January 18, 2018 [2]. Compared with traditional currency, digital currency has a significant advantage because it does not depend on any country or central banks. Instead, it uses all the nodes in the blockchain network to record each transaction. Meanwhile, the transaction is secure because it is irreversible in the blockchain. The transaction of bitcoin is completed through blockchain, and its security is ensured by cryptography and the whole network, which has been safe for nine years.

According to the Cambridge University's study in 2017 [3], there are more than 5.8 million active users using encryption wallets, bitcoin taking up the majority of it. For encryption wallets, the private keys are the core components. The wallets' private keys are used to prove the ownership of bitcoins. Once leaked, the thief can steal the user's bitcoin through the private key [4]. And if the private key is lost, the bitcoin will not be available and stay in the blockchain forever unless you get the private key back. According to the estimate of Chainalysis [5], there are about 17% to 23% of bitcoins cannot be used because of losing or forgetting the private key by the end of 2017. In general, unlike the traditional centralization online payment system based on the login (username + password), bitcoin depends more on the private key. Hence, the security of the private keys in bitcoin network is extremely important.

The bitcoin wallet is used to view, store and trade the bitcoin which the user holds. And it has a wide variety of formats. A cold wallet is achieved when the private key generated and stored in a secure offline environment [6]. Compared with the online environment, the offline environment is much

safer because hackers cannot get the private key through the Internet. Besides, hardware wallets such as Ledger Nano, KeepKey and TREZOR, have become popular because users can make a transaction securely whenever and wherever. Similar to cold wallets, hardware wallets generate private keys offline and private keys never leave the wallet. A hardware wallet communicates with a device, like the user's computer, which is connected to the Bitcoin network to send transactions. When a transaction happens, the hardware wallet signs it and sends it back to the computer, then the signed transaction is broadcast to the Bitcoin network. However, users have to keep an additional device to make transactions. Even the hardware wallets, attackers can also divert the cryptocurrency to a fraudulent address through Man-in-the-Middle attack when generating and displaying the wallet's address [7].

What's more, there are also many popular software wallets. Bitcoin Core and Bitcoin Knots are full-node wallets. They are more secure when trading in the bitcoin network and do not need a trusted third party. But they require a lot of storage to store the blocks (more than 145G). ArcBit and BitGo do not store the large amounts of the block information in the local storage, but they need a trusted third party which is deviated from the decentralized idea of the blockchain. Electrum and breadwallet belong to the lightweight wallet category. Although the wallets are isolated from other applications, they can not effectively protect their private keys, transactions and local block headers. The compromised Rich OS can easily steal their information privacy.

At present, most mobile devices do not have the ability to maintain the copy of the blockchain. For the devices with limited hardware (such as smartphones, tablets, embedded systems, etc.) [8], they can securely finish the transaction through Simplified Payment Verification (SPV) without storing all blocks in local storage. With the upsurge of digital currency, SPV wallets have become the commonest form of the digital wallets, including bitcoin wallets. The SPV wallet only needs to connect with the blockchain network's full nodes, and stores the block headers instead of all blocks in local storage, which is about one out of three thousand of the full nodes. During the transaction, it will request the corresponding transaction information from the connected remote node, including the copy of the transaction and the merkle tree. And then, it will calculate and verify the validity of the transaction through the information stored in the merkle tree structure. However, the existing SPV wallet only protects its private key cryptographically. Attackers can still steal the private key when using it. Besides, if the target address of a transaction is tampered, the money will directly go to the attacker's wallet. Meanwhile, if the block headers stored in the local database is tampered, we cannot guarantee the result of the verification is accurate. To sum up, the security of the private key and the address is extremely important for the encryption wallet, and there is no good way to protect the block headers stored in the local database and the verifications of transactions' processes [9].

In this paper, we design a Secure Blockchain Lightweight Wallet based on Trustzone (SBLWT), which offers a comprehensive protection for the private key, the address of the wallet, the block headers stored in the local database and the verifications of transactions' processes. Our contributions are as follows:

- We design a secure and lightweight bitcoin wallet based on Trustzone. It is more portable than the hardware-based wallet, and safer than the software-based wallet. It can protect the private key from being stolen by attackers no matter whether the Rich OS is malicious or not.
- We have achieved the protection of the sync and transaction verification processes, and keep the local block headers invisible from the Rich OS. Thus, the result of the verification and the transaction will be reliable.
- We also implemented and evaluated SBLWT on RASPBERRY PI 3 MODEL B boards. SBLWT brings very little changes to Rich OS and does not increase the trusted computing base of the secure section. Meanwhile, it has little influence on the performance of the wallet.

The paper is organized as follows: Section 2 introduces the background works, including bitcoin, blockchain, merkle tree and bloom filter used in SPV, and the Trustzone technology which is ubiquitous in mobile devices. Section 3 introduces the possible attacks and some assumptions of our lightweight wallet. Section 4 details the design of SBLWT. We implement and evaluate our design in section 5, and we also analyze the security of SBLWT. Section 6 shows the limitations of SBLWT and discusses the future works and section 7 introduces the related works. Finally, we make a conclusion and acknowledgment.

II. BACKGROUND

A. BITCOIN WALLET

Bitcoin is a decentralized digital currency, invented by Satoshi Nakamoto [1], which can do everything that traditional currency can do. In the bitcoin network, the wallet's address is derived from the corresponding public key through a series of irreversible hashes. And during the transaction of bitcoin, the private key is used for Digital Signature. Since there is no additional third party involved, bitcoin avoids high commissions and complicated transaction processes. The bitcoin wallet is a medium to store private keys. It has some main functions: creating and storing private keys and addresses; displaying the balance of bitcoin; verifying and signing the transactions. According to the data maintenance methods of blockchain, the wallets can be divided into three types: full node wallets, SPV wallets and central wallets. However, using the lightweight SPV wallets has become increasing popular, because it neither relys on a trusted third party nor needs to store a large amounts of blockchain data in the local database.

1) BLOCKCHAIN

Blockchain [10] is a decentralized distributed ledger for recording transaction information of bitcoin among multiple nodes. It uses cryptography to ensure the existing data not be tampered by anyone, and makes a consensus with new data through a series of consensus algorithms, such as Proof-of-Work (POW) [1] and Proof-of-Stake (POS) [11]. The block is a container data structure that stores the transaction information in the blockchain. It consists of a block header and a long list of transactions. From figure 1, we can see the block header is an 81 byte data structure, which includes the hash of the parent block, the hash of the merkle root, time-stamp and nonce. Moreover, the size of block headers is only one out of three thousand of total blocks' size which contains all transactions.

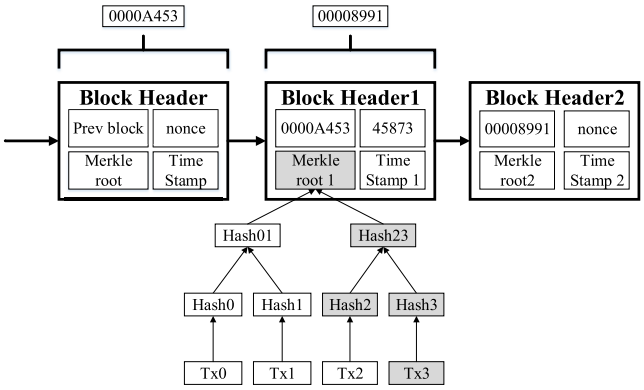

FIGURE 1. The struct of block header and merkle tree.

2) MERKLE TREE

Merkle tree [12], also named hash tree, is essentially a binary tree which stores the hash values. All the leaf nodes store small data blocks and the corresponding hash values. As the figure 1 shows, from the leaf nodes to the root node, it will constantly combine two adjacent hash values into a string and calculate the hash of the string until getting the hash of the root node. If the number of the leaf nodes is odd, it will directly hash the odd leaf nodes.

3) SIMPLIFIED PAYMENT VERIFICATION (SPV)

The SPV [1] nodes only need to download the block headers and store them in the local database instead of detail information of blocks. Thus, its size is only about one out of three thousand of total blocks. Due to the lack of the detail transaction information in SPV nodes, the verification of the transaction needs the full nodes in the blockchain to provide the detail information and the merkle route. Then, it will calculate the hash of the merkle root based on the result obtained and compare it with the real hash of the merkle root. To confirm the validity of transactions, wallets usually wait for several blocks to be generated over the block containing the transaction to avoid rollback. There is a risk of privacy leaks when the SPV nodes request the information about specific transaction from the full nodes. Therefore, to keep the confidentiality of the transaction addresses, SBLWT uses the Bloom Filter to probabilistic search and filter according to the keywords.

4) BLOOM FILTER

The Bloom Filter [13], a space-efficient probabilistic data structure, is used for data searching according to keywords instead of precise descriptions. It can realize effective searching without leaking out users' privacy. The core components of the Bloom Filter are a very long binary array and several hash functions. The length of the binary is m and the number of hash functions is k. The output values of hash functions and every bit of the binary array are one to one correspondence, and all the functions are deterministic functions. Anyone who uses the same Bloom Filter and the same function will get the consistent result. The accuracy and secrecy depend on the length of the binary array and the number of hash functions. The Bloom Filter will initialize every bit of the array to 0. When new data needs to be added, all the hash functions will be calculated one by one and the corresponding bits of the array will be set to 1 according to the calculation results. The process of searching is same as adding data. Bloom Filter will map to the k bits according to hash functions. If any one of the k bits' original value is not 1, the data searched must not exist.

B. TRUSTZONE

Trustzone [14], [15] is a mechanism providing a hardware-based isolation, proposed by ARM, which can build a secure and reliable environment for the code requiring high security. It separates the hardware and software resources of the system into two parts. One is secure, and the other is insecure. All the sensitive operations should be protected in the secure execution environment, such as Fingerprint, Secure Authentication, Digital Rights Management, etc. And the rest operations execute in the normal execution environment, such as the Rich OS and most applications. The status of the core is distinguished by the NS bit in Secure Configuration Register (SCR), and the NS bit can only be modified by the secure core. Through the Secure Monitor Call (SMC), it can enter the monitor mode and switch to another environment no matter in what kind of environment.

Trustzone can intercept the interrupt requests of the device through the Trustzone Aware Interrupt Controller (TZIC). The interrupt in the secure environment can be fast interrupt request (FIQ) or interrupt request (IRQ), while the interrupt in the normal environment can only be IRQ. When we start a transaction, there will be an action that clicks the touchscreen. So, we can leverage Direct Memory Access (DMA) to force the system to switch to make sure it can safely change into the secure execution environment.

III. THREAT MODEL AND ASSUMPTIONS

Based on the problems mentioned in Section 1, the design objectives of the lightweight blockchain wallet (SBLWT) are as follows: Firstly, SBLWT must keep the confidentiality of

the private key. That is to say, only the authorized users can use the key to generate the public key or sign the transactions. In order to meet this requirement, we should isolate and protect the private key. At the same time, we need to make sure the code of generating private keys and the keys' storage are safe and reliable. Secondly, SBLWT should ensure the address provided is real and valid, so that we can make sure the bitcoin will not be hacked by the attackers. Thus, we should isolate the address generating process and securely display the address to users. Finally, in order to ensure the effectiveness of transactions, the verification processes of the transaction must be protected. So, we need to keep the verification process away from the Rich OS and guarantee the secrecy of block headers stored in the local database.

The threat model is that the Rich OS and the running environment may be malicious. There are a lot of normal applications and the Rich OS in the Normal Execution Environment (NEE), so the security of the environment is hard to guarantee.

1) *Stealing and tampering sensitive resources.* Attackers can easily read and steal sensitive resources (such as private key, public key, transactions, etc.) directly from the permanent storage and memory.
2) *Tampering the static image of SBLWT.* Attackers may attempt to manipulate either the image or the control flow of SBLWT to steal sensitive resources.
3) *Denial-of-Service Attack.* Attackers may launch denial of service attacks to prevent the realiable Human-Machine Interaction and the realiable switching through the Rich OS vulnerabilities. The Rich OS may also prevent SBLWT from being either executed or displayed correctly.

We also make some assumptions as follows:

- The information obtained from the full nodes in the blockchain are reliable. The consensus mechanism of the blockchain guarantees that information stored in the full nodes are the longest chain. That is to say, we always download the latest information from the longest chain in the blockchain. So the information we obtained from the full nodes is credible.
- The Trustzone technology is secure and reliable. There has been a lot of work to enhance the security of Trustzone.
- Both sides of the transaction have a trusted execution environment with Trustzone. This assumption is reasonable, because most of the mobile devices being used today are equipped with Trustzone.
- The related cryptographic algorithms and their implementations are safe [16]. This is also a commonly accepted assumption.

IV. DESIGN

A. SYSTEM OVERVIEW

We propose a Secure Blockchain Lightweight Wallet based on Trustzone (SBLWT). SBLWT achieves the protection for the private keys, the addresses of wallets, the sync and transaction verification processes based on ARM's Trustzone technology. The structure of SBLWT is shown in Figure 2. A Rich OS and most applications are running in the NEE, and their storage is vulnerable. The sensitive lightweight wallet SBLWT is running in the SEE, allowing it to provide a real-time and systematical protection against a compromised Rich OS. We implemented the Secure Display and Touchscreen driver to protect the Human-Machine Interaction aiming to secure the input and output of SBLWT. Privkey Generator in the SEE is used to generate the private key. As it is located in the SEE, private keys' generation is safe. Besides, the private keys are stored in the secure permanent storage, where they cannot be accessed or modified by adversary. Through Secp256k1 we can get the public key. And then, in the SEE, the public key will be hashed twice to get the corresponding hash of the public key, and SBLWT uses the hash value to generate the wallet's address by Base58Check encoding. When in local mobile devices, the address will not leave the SEE.

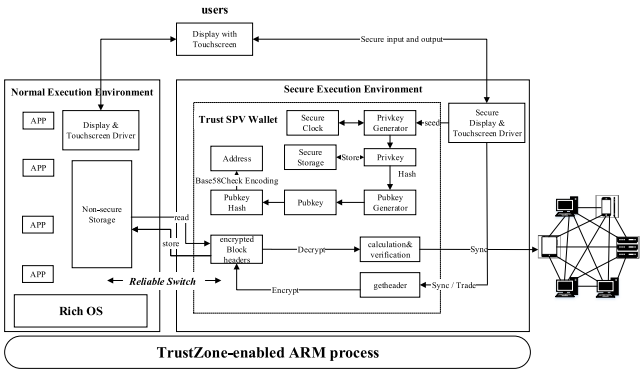

FIGURE 2. The framework of SBLWT.

After getting the wallet's address, we can make transactions. When there is a transaction, the wallet in Secure World will download all the related block headers from the full nodes in the blockchain (via *getheaders*) firstly, then encrypt the block headers and store the ciphertext in the non-secure permanent storage in the NEE. Next, the system will find the merkle blocks and calculate the merkle_root_hash, then compare it with the merkle_root_hash stored in the local block headers. If the result is matching and there are enough blocks confirm the transaction, the transaction will be signed with the private key and broadcast to the blockchain. Since private keys and the authentication processes are isolated in the SEE, SBLWT is safe even if the Rich OS and other normal applications are malicious.

This section will show the detail of SBLWT. Secure Booting is the foundation of the security of the wallet code and information stored in the local database. Reliable Switching will switch to the secure environment safely whenever we start SBLWT for transaction or synchronization. Next we will describe how to protect private keys and the addresses of wallets. Finally, we show the protection of the block headers

stored in the local database, the sync and verification processes of the transaction.

B. SECURE BOOTING AND RELIABLE SWITCHING

1) SECURE BOOTING

The whole lightweight wallet SBLWT has always been in the SEE and isolated from the Rich OS since the smartphone booted up. Figure 3 shows the booting sequence. First, ROM-based bootloader and Flash Device bootloader are executed, initializing the critical sections (such as memory controllers, etc.), then the secure boot items are loaded from the secure permanent storage into the secure memory in order to control the whole SEE. After that, secure operating system starts and SBLWT will be loaded into the secure memory. To make sure SBLWT is not malicious, we need to check its integrity. So we can calculate SBLWT's hash and verify it through signature in SEE. Finally, the system will switch into the NEE, load the normal Bootloader and start the Rich OS. At the same time, the status of CPU changes from secure into non-secure. As the whole SBLWT is booted before the Rich OS and always in the secure memory, which is isolated from the NEE, it is safe no matter whether the Rich OS is malicious or not.

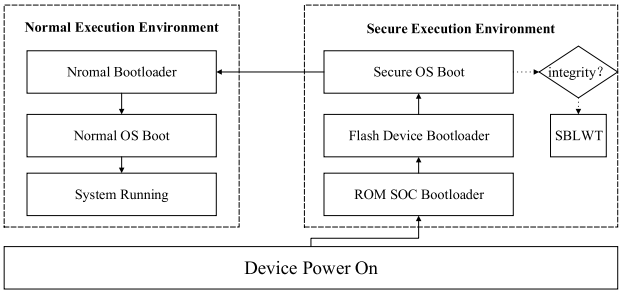

FIGURE 3. The booting sequence of SBLWT.

2) RELIABLE SWITCHING

From Secure Booting, we can make sure that the Secure OS and SBLWT are always in SEE even when the Rich OS is running. In general, the lightweight SPV wallets in the blockchain are for trading. Therefore, when a transaction occurs, the Rich OS in the NEE needs to be suspended, the system switches to the SEE and restores the secure operating system and SBLWT which are stored in the secure storage. The safe and reliable forcing environment switch is the guarantee for safe transactions. Software interrupts don't apply to environment switch, because if the Rich OS is malicious, the compromised OS may tamper the software interrupts, and safe switch to SEE are disabled. So we use the Non-Maskable Interrupt (NMI), a hardware interrupt, to achieve safe switch. As long as there is a transaction, the switch between the two environments will be triggered. The system won't switch back to NEE until the transactions and verifications have completed. Before the system switches back to NEE, the context of the SBLWT, including the wallet's private key and address, will be stored in the secure storage, and the block headers will be encrypted in the non-secure storage. Meanwhile, all footprints in SEE should be cleaned up.

C. SECURE KEYS AND RELIABLE ADDRESSES

In Figure 4, we can see the threats of the private key and the address of the wallet. The private key is used to generate the public key. Attackers can easily get the private key by reading from memory directly, or steal the seed for generating the private key. What's more, attackers can intercept and modify the code that is used to generate the private key and users may be tricked into using malicious private keys generated by attackers. Once the private key is leaked, the user's wallet is no longer safe, which causes extremely serious loss. Similarly, attackers can also tamper the control flow of generating public key and address or directly modify them. Meanwhile, attackers can also modify the address when the address is displayed to users or tamper the address when users enter or copy it to clipboard. The modified address may cause bitcoins of the transaction flow to the attackers' wallet. To solve the problems above, we safeguard the bitcoin transaction and its cash flow through the following points.

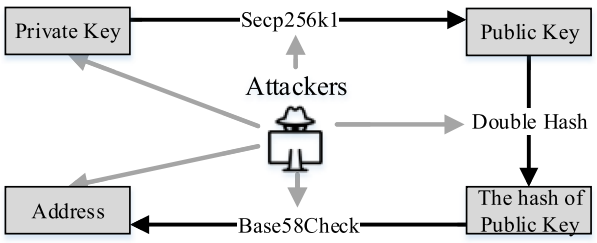

FIGURE 4. The threats of the private key and address.

1) SECURE PRIVATE AND PUBLIC KEY

The private key is generated from the seed, so the safety of the seed is very important. To protect the seed, we store the seed in the secure persistent storage which cannot be accessed by the compromised Rich OS. On the premise of the seed's safety, the integrity of the code and process for generating private keys is also a significant concern that our system need to consider, because the private key will no longer be safe once the code is modified or the process is hijacked. So in SBLWT, the code and the process of generating private keys are isolated from the Rich OS. In detail, the code is stored in the secure persistent storage which can only be accessed by SBLWT. The operation of generating private keys can only be executed in SEE. When the system boots up, it launches a series of loaders in the SEE, including SBLWT. Then the system switches to the NEE and loads the Rich OS. Thus, there is no chance for the Rich OS to manipulate or tamper the code of generating private keys. The Rich OS cannot get close to the private key and steal it, because the private key is stored in the secure persistent storage after being generated. As with the private key, the process of generating the public

key is also in SEE, and the public key will never leave SEE until getting the address.

2) SECURE ADDRESS

A reliable address is crucial for bitcoin to arrive at the right wallet. But attackers can tamper the process of addresses generation or replace the user's address with his own one. So, it is necessary to protect address before it is sent to the blockchain network. The whole process of generating address is finished in SEE which is isolated from the Rich OS, so the local Rich OS and other applications have no chance to get close to the address. In this way, our system ensures that nobody can trick users to send bitcoins to the attacker's wallet by a counterfeit address.

3) SECURE DISPLAY AND TOUCHSCREEN

A reliable graphical user interface (GUI) is the guarantee for the secure Human-Machine Interaction. So we implemented reliable Display Driver and Touchscreen Driver in SEE. The users can safely input or copy the addresses of wallets and confirm the transaction through Touchscreen Driver. And they can also get real and credible detailed information of the transaction from the screen through Display Driver. Besides, before switching back to the NEE, SBLWT cleans up all the related information.

D. SECURE SYNC AND VERIFICATION

In our design, we implement two key functions: the sync process which is responsible for keeping the information in local database as fresh as on the full nodes, and the verification process which is used to verify whether the transaction is authorized. We put the source code of original SPV wallet, which supports synchronizing block headers and verifying transactions, in SEE.

To make sure the information we get from remote nodes will not be tampered by any attackers, the communication channel with remote nodes should be trustworthy. This is easy to achieve, in our system, we rely on digital signature to enhance reliability of communication channel. Just as Figure 5 shown, to reduce the trusted computing base of the secure environment, we store block headers in NEE. But to prevent the information stored in local database from being tampered, we encrypt it in the non-secure persistent storage. And the information will be decrtpted in SEE when needed. When synchronizing, the Rich OS suspended. Thus, only SBLWT in SEE can communicate with the external blockchain network by the communication interface. To make sure the integrity of the information, the system will calculate the hash of the information and compare it with the standard hash we precalculate before encrypting in SEE. The source code of SPV wallet relies on the C standard library when calling input or output functions, but in SEE, it uses *EMSG()* and *IMSG()* for input and output. Meanwhile, according to the GlobalPlatform TEE System Architecture specification, OP-TEE supports the basic encryption algorithms, such as ***RAS***, ***AES***, ***HMAC***, ***SHA*** and ***RANDOM*** etc. So, we can call the corresponding algorithm for encryption and decryption, which will not increase the TCB of the secure environment.

When SBLWT is triggered, the system will build a session between the Rich OS and SBLWT in Secure Execution Envrionment. Thus, the Rich OS will not get any private information of the wallet. The struct of the session data is shown in Listing 1. *StorageID* is responsible for storing the storage id which is used while opening or closing the storage object, and *storageIDLen* is used to store *storageID*'s size. *write_cache* and *read_cache* are current contents of write and read cache. There is a wallet loaded when *wallet_loaded* is true, and *current_wallet* will show the cache of the loaded wallet's record. The address of the wallet is stored in *wallet_address*. And *sha256_op_internal* is the handle for SHA-256 encryption operation.

```
typedef struct session_data{
        char * storageID;
        size_t storageIDLen;
        uint8_t * write_cache;
        uint8_t * read_cache;
        WalletRecord * current_wallet;
        bool wallet_loaded;
        uint32_t wallet_address;
        TEE_OperationHandle * sha256_op_internal;
}Session_data;
```

LISTING 1. The struct of session_data.

```
static TEE_Result sync_block_headers(
        Session_data * session_data,
        uint32_t wallet_address,
        WalletRecord * remote_node,
        uint256      * merkle_block)
static TEE_Result verify_transaction(
        Session_data * session_data,
        uint32_t wallet_address,
        WalletRecord * remote_node,
        BigNum256 transaction_hash,
        uint256      * merkle_block)
```

LISTING 2. Sync and verification functions.

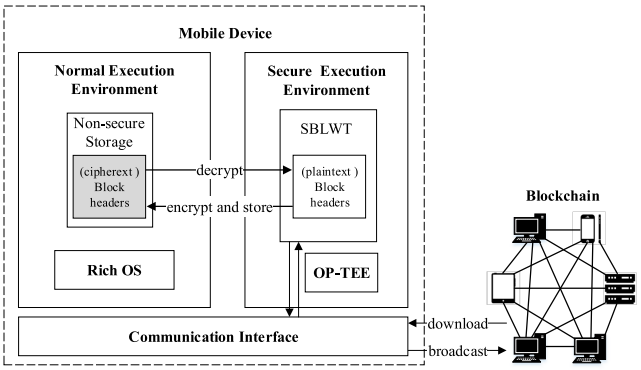

FIGURE 5. The block headers are stored in non-secure storage.

From the Listing 2, we can see the two major functions of SBLWT in the SEE: ***sync_block_headers()*** and ***verify_transaction()***.

- ***sync_block_headers()*** aims to keep the wallet up-to-date with the address and transactions. Listing 1 illustrates the detail of *session_data*. When synchronizing, the system will get the wallet's address and the details of the wallet from *wallet_address* and *remote_wallet*. When SBLWT boots up, the block headers will be decrypted from the non-secure persistent storage, and the system will also check its integrity. Then, SBLWT will get the height of blockchain, and send a *getheaders* command to quickly download block headers until the height of local block headers is no less than the one of remote block headers. Next, SBLWT will get the merkleblock message and the transaction message from the remote node, and store them in local database.
- ***verify_transaction()*** is responsible for verifying the transactions. When there is a transaction, SBLWT calculate the hash of the transaction and then store it in *transaction_hash*, a 32-bytes number which is little-endian and multi-precision. Next, SBLWT needs to add the key words, such as the wallet's address, *transaction_hash* and other related information, to the bloom filter, and requests bloom filter to provide related information by sending *getdata* commands. The bloom filter will select the matched blocks and the merkle route according to the key words. And then, detail information of corresponding transactions will respond to the *getdata* command sent by SBLWT. After that, SBLWT can calculate the merkle root's hash, and compare it with the merkle_root_hash stored in local block headers to verify the transaction.

V. EVALUATION AND SECURITY ANALYSIS

A. EVALUATION

In this section, we describe the implementation and evaluation of SBLWT in detail. We use Open Portable Trusted Execution Environment (OP-TEE 2.4) [17] as the trust OS. We deploy it on the RASPBERRY PI 3 MODEL B development board [18], which supports Trustzone and OP-TEE. This board has a quad-core 1.2GHz Broadcom BCM2837 64bit CPU and 1 GB RAM. We also provide an 8G SD card and a 7-inch Element14 display. OP-TEE, an open source system based on Trustzone, belongs to Linaro Security Working Group (SWG). It builds a secure execution environment via a lightweight secure OS isolated from the Rich OS. And the secure OS and its secure environment follow GlobalPlatform TEE System Architecture specification. We build SBLWT as a Trust Application (TA) in the SEE, and use a DMA instead of the normal interrupt to force the environment switch.

1) TCB

In our design, when a user presses the screen to start a transaction or sync, NMI will be triggered to switch to SBLWT, which is located in the SEE, and meanwhile the Rich OS will be suspended. However, the trading involved with bitcoin only happens a few times. The Prikey Generate module is only about 543 lines. The key code of Address Generate module is only about 309 lines, because OP-TEE provides a series of ECDSA functions needed. The Sync module is about 1037 lines and the verification is only about 559 lines. In general, SBLWT increases the trust base of the secure environment just a little. And the same to the normal applications, SBLWT has little impacts on the performance of the system.

TABLE 1. The detail TCB of SBLWT.

	Module	lines
1	prikey generate	543
2	address generate	309
3	sync	1037
4	verification	559

2) OVERHEAD

We use the NMI instruction for switching, the result shows that it takes only 1.7 us for environment switching. This means that the switching time is extremely negligible. In our experiment, when the screen is pressed to sync, it will enter the SEE immediately. Then the system will check the integrity of SBLWT. After that, SBLWT will read the block headers stored in local database and synchronize with the remote node to get the newest information. Finally, the system will clean up all the related information and return back to the Rich OS. From table 2 we can see, the main time is spent on read and write. Thus, the system has no impact on the users' experience.

TABLE 2. Some overheads of SBLWT.

	Operations	Times
1	environment switching	1.7 us
2	read(1KB)	2 ms
3	write(1KB)	12 ms
4	SBLWT check	1.54 ms
5	information cleanup	0.521 ms

To find out the performance on SBLWT, we did 1000 experiments to test SHA-256 on different sizes of data: 1 Byte, 81 Bytes, 1 KB, 4 KB and 1 MB. The experiment results are shown in figure 6. The result shows that, the time of SHA-256 in normal environment is less than it in SBLWT just

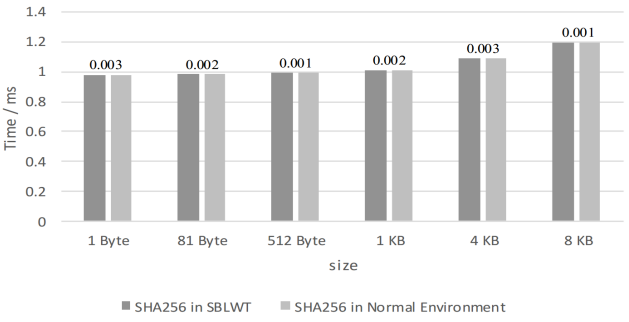

FIGURE 6. SHA-256 speed comparision.

a little, the increased overhead is less than 3 us. This means the secure environment will not increase the running time of SHA-256. Meanwhile, because the size of block headers is very small, the system will not spent too much time on SHA-256 operation.

We also have performed some experiments on the synchronous process to compare the performance in different environments. In this experiment, we synchronize different numbers of the block headers. The result is shown in Figure 7. Due to the encryption, decryption and environment switching operations, the overhead of synchronous in SBLWT is a little larger than it in normal SPV wallet. The difference of the synchronous time between the two environment is always about the same no matter how much the block headers' number is. And as the number of block headers increases rapidly, the difference proportion of the synchronous time is becoming smaller and smaller.

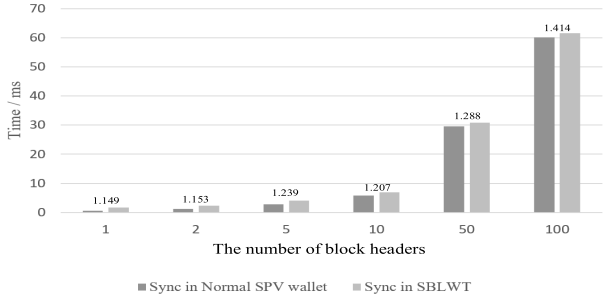

FIGURE 7. The sync time comparision.

TABLE 3. SBLWT's start and verify overhead comparision.

	Normal SPV wallet	SBLWT
start	1.387 s	1.443 s
verification	2.283 ms	3.137 ms

Similar to the synchronous process, we also did 1000 experiments to test the overhead of the booting time and the verification time of SBLWT and compare the performance with the normal SPV wallet running in Rich OS. The result is shown in Table 3. As the system will switch to the SEE and restore the secure OS and SBLWT, the average booting overhead of the SPV wallet in normal world is smaller than SBLWT. For one transaction's verification, the average verification overhead in SBLWT is a little larger than it in normal SPV wallet because SBLWT needs to decrypt the information stored in the non-secure storage. However, as all the costs are small, it will not impact us when using SBLWT.

We can see the overhead of SBLWT is mainly spent on the operation of read, write and communicating with the network. The time spent on encryption and decryption is very small. Meanwhile, the original SPV wallet also has the verification process. So, we will not increase the overhead of the wallet too much. From the results we can draw the conclusion that our design has no impact on the users' experience.

B. SECURITY ANALYSIS

Our design aims to ensure the security, confidentiality and integrity of the blockchain lightweight wallet. As the threat models proposed in section 3, the Rich OS and normal applications may be compromised by adversaries using the Rich OS vulnerabilities. Thus, we must keep the Rich OS and normal applications away from the code and data of SBLWT.

1) SECURE BOOTING

The booting sequence ensures a secure route when starting SBLWT and switching between the two environment. To ensure SBLWT's image is not malicious before loaded on the processor chips, we need to check its authenticity and integrity. Through validating the signature, the system can make sure the code and data of SBLWT is original and realiable. Therefore, if the compromised Rich OS or other normal applications tamper with the code or data of SBLWT, the system can easily find it and reject to load them into the secure memory.

2) INFORMATION LEAKAGE

Attackers can directly read the information from the permanent storage or memory. When the system boots up, it launches a series of loaders in SEE, including SBLWT. Thus the compromised Rich OS and other normal applications cannot get close to any sensitive data of SBLWT from either the permanent storage or the RAM memory. As all the process of SBLWT are finished in SEE, the compromised Rich OS and other normal applications have no chance accessing any resources of SBLWT. What's more, all the temporary information, including CPU registers that may contain sensitive resources, will be cleaned up before switching back to NEE and restoring the Rich OS.

Meanwhile, attackers may tamper with the data flow of SBLWT. However, they will never have the rights to modify SBLWT's code because the code of SBLWT always runs in the SEE. Moreover, as the interruption for switching is NMI which is handled in SEE, the compromised Rich OS will never tamper the execution of SBLWT through any interrupts. Although SBLWT shares the same graphical user interface (GUI) device with the Rich OS, SBLWT has its own independent Display Driver and Touchscreen Driver in SEE which cannot be access by the Rich OS. The dedicated Display and Touchscreen Driver keep a secure context switching when inputting and outputting, thus the Rich OS's states will be suspended and saved when switching to SEE and the SBLWT's states will be cleaned up before returning back to NEE.

3) DENIAL-OF-SERVICE ATTACK

SBLWT is designed for mobile devices (such as smartphones, tablets, embedded systems, etc.), and the security of mobile OS can't be guaranteed. Thus, the DOS attack is the most serious threat for our design.

First, when starting a transaction or synchronization, the system needs to switch to SBLWT. But attackers can

intercept the requests through the Rich OS vulnerabilities. But in our design, the Reliable Switching can keep SBLWT away from these DOS attacks through the NMI interrupt, a hardware-based interrupt which cannot be bypassed by the compromised Rich OS. Meanwhile, the handler of NMI interrupt is in SEE, so no one can manipulate the handler. Thus, we can make sure that the switch between the two environments will be triggered as long as there is a transaction.

Second, attackers can also delete the binary code of SBLWT from permanent storage by the Rich OS vulnerabilities. To deal with that, we store the SBLWT code in the secure permanent storage which can only be accessed by SBLWT. Similarly, all the code of SBLWT in memory is protected by Trustzone, and attackers have no chance to tamper with it. Note that it is unrealistic to store the whole block headers in secure storage, which will make the trusted computing base of SEE too big. However, the Rich OS may tamper information in block headers if they are stored directly in insecure storage. To address this problem, we encrypt all the block headers before put them into the non-secure storage. The process of encryption and decryption is done in SEE.

Third, attackers may also prevent the realiable results of SBLWT from being displayed. To deal with this problem, we build a secure display and touchscreen controller to keep the Rich OS isolated so that the Rich OS has no rights to intercept and tamper with the control flow or data flow of the I/O.

VI. DISCUSSION AND FUTURE WORKS

We focus on the protection of the lightweight bitcoin wallet based on Trustzone provided by ARM processors, but there is one exception. Although IOS smartphones use ARM processors, it does not support Trustzone. But in IOS smartphones, there is an extra processor called Secure Enclave to deal with the related security operations. In the future, we will extend our system to all the mobile devices, including IOS. And then, we will enlarge to all the devices with hardware-based isolation mechanism. Thus, we can make a transaction anytime and anywhere.

VII. RELATED WORKS

The security of Bitcoin is closely bound up with the private keys. Bitcoins worth millions of dollars have been stolen for the reason that hackers get the private keys stored on the users' vulnerable computers. To protect private keys, and also Bitcoins, much work has been investigated recently. Some researchers have proposed several practical Bitcoins signature schemes. ECDSA (Elliptic Curve Digital Signature Algorithm) has been applied in Bitcoin for encryption, and improved in [19] to better protect private keys in practical. Goldfeder and Gennaro *et al*. proposed the first threshold signature scheme compatible with Bitcoin's ECDSA signatures and then use it to realize threshold wallets. Also, the team presented an efficient and optimal threshold DSA scheme [20] in 2016. These works aim to provide private keys storage security, while our work considers private keys security of both storage and process. Another line of research focuses on private keys protection by creating and storing keys offline, such as cold wallets and hardware wallets. Cold wallets [6] usually need two computers. One of the computers is offline and responsible for signing transactions, and the other one is connected to the Bitcoin network and broadcasts the signed transactions. But this method isn't suitable for mobile payment. BlueWallet [21], a hardware token for the authorization of transactions, communicates with computers using Bluetooth Low Energy and stores private keys offline. However, users have to keep the hardware device while using BlueWallet, which is not convenient sometimes.

Although SPV has many advantages, it also brings some privacy concerns [22]. Reference [23] proposes a solution to deal with the serious addresses leakage offered by Bloom filters in existing SPV clients. Reference [24] introduces a design to prevent bitcoin addresses from being leaked based on the privacy metric γ-Deniability. All these methods manage to solve the problems of bitcoin address leakage, but they do not protect the processes of sync and transaction verification.

Teechain [25] implemented an offline blockchain trading system based on Trustzone. The author proposes a new chain protocol and asynchronous channel for blockchain trading, which can accelerate the confirmation time of bitcoin transaction. Miraje *et al.* [26] proposed a bitcoin wallet based on Trustzone, which can protect the private key. However, both the two methods protect the full nodes' wallets and need to store all blocks in local database. They are not applicable to the devices with limited hardware (such as smartphone, tablets, embedded systems, etc.) widely used in our daily life.

Trustzone can separate the hardware and software resources of the system into two parts. Recently, there has been much work [27], [28] on enhancing the security of Trustzone. Also, many researchers have taken advantage of Trustzone to address some security issues. Azab *et al*. [29] achieve a real-time protection by building a system integrity checking and a malicious software selection in the secure environment. Sun *et al.* [30] build a One-Time Password Tokens in the secure environment. And W Li design a system to prevent the ad from being tampered or faking click rate.

VIII. CONCLUSION

We design a Secure Blockchain Lightweight Wallet called SBLWT for securing private keys, addresses of wallets, block headers stored in the local database, the sync and the verification processes of transactions no matter whether the Rich OS is malicious or not. It does not need to modify the Rich OS, and has little impacts on the system.

REFERENCES

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf
[2] K. Wu, S. Wheatley, and D. Sornette. (2018). "Classification of cryptocurrency coins and tokens by the dynamics of their market capitalisations." [Online]. Available: https://arxiv.org/abs/1803.03088

[3] G. Hileman and M. Rauchs, "Global cryptocurrency benchmarking study," Cambridge Centre Alternative Finance, Tech. Rep., 2017.
[4] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, Jun. 2017, pp. 557–564.
[5] M. Conti, S. Kumar, C. Lal, and S. Ruj. (2017). "A survey on security and privacy issues of bitcoin." [Online]. Available: https://arxiv.org/abs/1706.00916
[6] Y. David, "Is bitcoin a real currency? An economic appraisal," in *Handbook of Digital Currency*. Amsterdam, The Netherlands: Elsevier, 2015, pp. 31–43.
[7] Ledger. (2018). *Ledger Receive Address Attack*. [Online]. Available: https://www.docdroid.net/Jug5LX3/ledger-receive-address-attack.pdf/
[8] X. Xu, J. Liu, W. Chen, Y. Hou, and X. Tao, "Storage and computing resource enabled joint virtual resource allocation with QoS guarantee in mobile networks," *Sci. China Inf. Sci.*, vol. 60, no. 4, p. 040304, 2017.
[9] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 279–296.
[10] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 839–858.
[11] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 3, pp. 34–37, 2014.
[12] S. Michael, "Merkle tree traversal in log space and time," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2004, pp. 541–554.
[13] H. Cai, P. Ge, and J. Wang, "Applications of bloom filters in peer-to-peer systems: Issues and questions," in *Proc. Int. Conf. Netw., Archit., Storage*, Jun. 2008, pp. 97–103.
[14] W. Johannes, "Trusted computing building blocks for embedded linux-based ARM trustzone platforms," in *Proc. 3rd ACM Workshop Scalable Trusted Comput.*, 2008, pp. 21–30.
[15] "ARM security technology building a secure system using trustzone technology," ARM, Cambridge, U.K., White Paper, 2009.
[16] H. Yu, Y. Hao, and D. Bai, "Evaluate the security margins of SHA-512, SHA-256 and DHA-256 against the boomerang attack," *Sci. China Inf. Sci.*, vol. 59, no. 5, p. 052110, 2017.
[17] R. S. Mahadev, A. Seshadri, S. Rajamani, and V. Kumar, "Using trusted execution environments to enable integrity of offline test taking," *Applications of Cognitive Computing Systems and IBM Watson*. Singapore: Springer, 2017, pp. 9–19.
[18] (Apr. 8, 2017). *Raspberry Pi 3 Model B*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[19] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, "Elliptic curve cryptography in practice," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2014, pp. 157–175.
[20] R. Gennaro, S. Goldfeder, and A. Narayanan, "Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2016, pp. 156–174.
[21] T. Bamert, C. Decker, R. Wattenhofer, and S. Welten, "Bluewallet: The secure bitcoin wallet," in *Proc. Int. Workshop Secur. Trust Manage.*, 2014, pp. 65–80.
[22] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 104–121.
[23] A. Gervais, S. Capkun, G. O. Karame, and D. Gruber, "On the privacy provisions of bloom filters in lightweight bitcoin clients," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, 2014, pp. 326–335.
[24] K. Kanemura, K. Toyoda, and T. Ohtsuki, "Design of privacy-preserving mobile bitcoin client based on γ-deniability enabled bloom filter," in *Proc. 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.
[25] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, and E. G. Sirer. (2017). "Teechain: Scalable blockchain payments using trusted execution environments." [Online]. Available: https://arxiv.org/abs/1707.05454
[26] M. Gentilal, P. Martins, and L. Sousa, "TrustZone-backed bitcoin wallet," in *Proc. 4th Workshop Cryptogr. Secur. Comput. Syst.*, 2017, pp. 25–28.
[27] N. Zhang, K. Sun, W. Lou, and Y. Hou, "CaSE: Cache-assisted secure execution on ARM processors," in *Proc. IEEE Symp. Secur. Privacy*, May 2016, pp. 72–90.
[28] J. S. Jang, S. E. Kong, M. Kim, D. Kim, and B. B. Kang, "SeCReT: Secure channel between rich execution environment and trusted execution environment," in *Proc. 21st Annu. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–15.
[29] A. M. Azab *et al.*, "Hypervision across worlds: Real-time kernel protection from the ARM TrustZone secure world," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 90–102.
[30] H. Sun, K. Sun, Y. Wang, and J. Jing, "TrustOTP: Transforming smartphones into secure one-time password tokens," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 976–988.

WEIQI DAI received the Ph.D. degree in computer science and technology from the Huazhong University of Science and Technology (HUST). He is currently an Assistant Professor in computer science and technology with HUST. His expertise and research interests include blockchain, cloud computing security, trusted computing, virtualization technology, and trusted SDN.

JUN DENG received the B.S. degree in computer science and technology from Hunan University, China, in 2015. He is currently pursuing the master's degree in cyberspace security with the Huazhong University of Science and Technology. His research interests include mobile security and blockchain.

QINYUAN WANG received the B.E. degree from the School of Optical and Electronic Information, Huazhong University of Science and Technology (HUST), China, in 2017, where he is currently pursuing the master's degree in cyberspace security. His research interest includes blockchain.

CHANGZE CUI is currently pursuing the degree in information security from the Huazhong University of Science and Technology. His research interests include blockchain, smart contract, and SGX.

DEQING ZOU received the B.E. degree in computer science and technology from Fuzhou University, China, in 1997, and the Ph.D. degree from the Huazhong University of Science and Technology (HUST) in 2004. He is currently a Professor of computer science with HUST. He has applied almost 20 patents, published two books, one is the*Xen virtualization Technologies* and the other is *Trusted Computing Technologies and Principles*, and published over 50 high-quality papers, including the papers published by the IEEE TRANSACTION ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON CLOUD COMPUTING. His main research interests include system security, trusted computing, virtualization, and cloud security. He has served as the PC member/PC chair of over 40 international conferences.

HAI JIN received the Ph.D. degree in computer engineering from the Huazhong University of Science and Technology (HUST), China, in 1994. He was with The University of Hong Kong from 1998 to 2000 and a Visiting Scholar with the University of Southern California from 1999 to 2000. He is currently the Cheung Kung Scholars Chair Professor of computer science and engineering with HUST. He is the also Chief Scientist of the National 973 Basic Research Program Project of Virtualization Technology of Computing System. He has co-authored 22 books and published over 700 research papers. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security. He is a fellow of CCF and a member of the ACM. In 1996, he was awarded the German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz, Germany. He was awarded the Excellent Youth Award from the National Science Foundation of China in 2001.