Accepted Manuscript

A survey on the security of blockchain systems

Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, Qiaoyan Wen

PII: S0167-739X(17)31833-2
DOI: http://dx.doi.org/10.1016/j.future.2017.08.020
Reference: FUTURE 3615

To appear in: *Future Generation Computer Systems*

Received date : 21 January 2017
Revised date : 11 July 2017
Accepted date : 14 August 2017

Please cite this article as: X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, *Future Generation Computer Systems* (2017), http://dx.doi.org/10.1016/j.future.2017.08.020

Research Highlights

Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, Qiaoyan Wen

The main contributions of this paper are as follows:

- To the best of our knowledge, we conduct the ***first*** investigation on security risks for popular blockchain systems systematically and classify them.
- We enumerate the attack cases suffered by blockchain from 2009 to the present (May of 2017) and analyze the vulnerabilities exploited in these cases.
- We summarize practical academic achievements for the enhancement of blockchain, which will provide reference and guidance to developers and users.
- Based on the systematic examination on the security of current blockchain systems, we list a few future directions to stir up research efforts into this area.

A Survey on the Security of Blockchain Systems

Xiaoqi Li[a], Peng Jiang[b], Ting Chen[c], Xiapu Luo[a], Qiaoyan Wen[b]

[a]*Department of Computing, The Hong Kong Polytechnic University*
[b]*Beijing University of Posts and Telecommunications*
[c]*Center for Cybersecurity, University of Electronic Science and Technology of China*

Abstract

Since its inception, blockchain technology has shown attractive development prospects. From the initial cryptocurrency to the current smart contract, blockchain is applied to many fields. Although there are some studies on the security and privacy issues of blockchain, there lacks of a systematic examination on the security of blockchain systems. In this paper, based on several of the most popular blockchain systems (e.g., Ethereum, Bitcoin, Monero, etc.), we make a systematic investigation of the security threats for blockchain and enumerate the relevant real attack cases. Then we summarize security enhancement solutions for the blockchain, which will provide guidance for the healthy development of blockchain.

Keywords: blockchain, security, cryptocurrency, smart contract

1. Introduction

In 2009, Satoshi Nakamoto released the Bitcoin open source user group node and hash function system. Since then, peer decentralized system, that is blockchain technology began to enter our vision. Bitcoin as the first cryptocurrency, develops extremely rapidly. Bitcoin was the world's strongest currency from 2010 to 2013, even outperforming the gold [1]. Bitcoin was rated top performing currency in 2015 [2] and best performing commodity in 2016 [3]. According to statistics [4], the number of daily confirmed Bitcoin transactions is 333,161 on 1st of May, 2017, which is increased by 305,552% than that of 17th of January, 2009 (the number is 109).

As the fundamental technology of Bitcoin, blockchain begins to be applied to various fields, including medicine [5, 6, 7], economics [8, 9, 10], Internet of things [11, 12, 13], software engineering [14, 15, 16] and so on. In recent years, a variety of blockchain systems have been proposed, and blockchain 2.0 era has arrived. Now, blockchain systems have released their unique Turing-complete programming languages, allowing anyone to develop their own smart contracts and run them on the blockchain. With the decentralized consensus

Email addresses: csxqli@comp.polyu.edu.hk (Xiaoqi Li), pennyjiang0301@gmail.com (Peng Jiang), brokendragon@uestc.edu.cn (Ting Chen), csxluo@comp.polyu.edu.hk (Xiapu Luo), wqy@bupt.edu.cn (Qiaoyan Wen)

mechanism of blockchain, smart contracts can let mutually distrusted users complete data exchange or transaction, which do not rely on any third-party trusted authority. Ethereum is now the most widely used blockchain for smart contracts, on which there are already 317,506 smart contracts and more than 75,000 transactions daily happen [17].

With the rapid development of blockchain technology, there are some studies on the security and privacy issues of blockchain. However, there lacks of a systematic examination on the security of current blockchain systems. As smart contracts once deployed to the blockchain, they will not be able to be modified. Loi et al. discovered that among the 19,366 existing Ethereum contracts, 8,833 of them are vulnerable [18]. Because many contracts involve property transactions, contracts with security vulnerabilities will often bring great harm and losses. In June 2016, the smart contract `DAO` was attacked [19]. The attacker exploited a recursive calling vulnerability, resulting in `DAO` losing about 60 million dollars. In March 2014, hackers exploited transaction mutability in Bitcoin to attack `MtGox`, the largest Bitcoin trading platform. It caused the collapse of `MtGox`, with a value of 450 million dollars Bitcoin stolen [20]. Furthermore, Bitcoin became the world's worst performing currency in 2014 [1]. From the above indications, we can conclude that it is necessary to conduct a systematic examination of blockchain systems.

There already have some research aiming at blockchain security. But none of them investigates risks and attack cases for blockchain systems systematically, and at the same time proposes security enhancements. The most similar research work to ours is [21]. But [21] only focuses on Ethereum smart contracts, rather than all blockchain systems. From security programming perspective, they analyze the security vulnerabilities of Ethereum smart contracts, providing a taxonomy of common programming pitfalls which may lead to vulnerabilities. They show a series of related attacks for smart contracts, but they haven't proposed any security enhancement. This paper focuses on the security of blockchain from more comprehensive perspectives. The main contributions of this paper are as follows:

(1). To the best of our knowledge, we conduct the *first* investigation on security risks for popular blockchain systems systematically and classify them.

(2). We enumerate the attack cases suffered by blockchain from 2009 to the present (May of 2017) and analyze the vulnerabilities exploited in these cases.

(3). We summarize practical academic achievements for the enhancement of blockchain, which will provide reference and guidance to developers and users.

(4). Based on the systematic examination on the security of current blockchain systems, we list a few future directions to stir up research efforts into this area.

This paper is organized as follows. In Section 2, we introduce the main technologies leveraged in blockchain systems. Based on typical blockchain systems, Section 3 systematically investigate and classify security risks faced by blockchain. For each risk or vulnerability, we analyze its causes and possible harm. Then, Section 4 enumerates real attack cases suffered by the blockchain. At the same time, we analyze the types and principles of vulnerabilities exploited in these attack cases. Section 5 summarizes blockchain security enhancements to guide the healthy development of the blockchain technology. In Section 6, we list a few future directions to stir up research efforts into this area. Finally, we conclude the paper in Section 7.

2. Overview of Blockchain Technologies

In this section, we introduce the main technologies leveraged in blockchain. Firstly, we introduce the fundamental trust mechanism used in blockchain, that is consensus mechanism. Then, we explain the synchronisation process between nodes. Finally, we introduce the two development stages of blockchain and its application achievements.

2.1. Consensus Mechanism

Compared to other technologies, the main characteristic of blockchain is decentralization. In a blockchain system, there is no third-party trusted authority. In order to guarantee the reliability and consistency of the data, and transactions of each distributed node, the blockchain adopts the consensus mechanism. In the existing blockchain systems, there are two main consensus mechanisms: PoW (Proof of Work) and PoS (Proof of Stake). In addition, there are other widely used consensus mechanisms, such as PBFT [22], Proof of Bandwidth [23], Proof of Elapsed Time [24] and so on. The two most popular blockchain systems, Bitcoin and Ethereum, all use the PoW mechanism. Ethereum also incorporates the PoS mechanism. In addition, there are some cryptocurrencies using the PoS mechanism, such as PeerCoin, ShadowCash and so on.

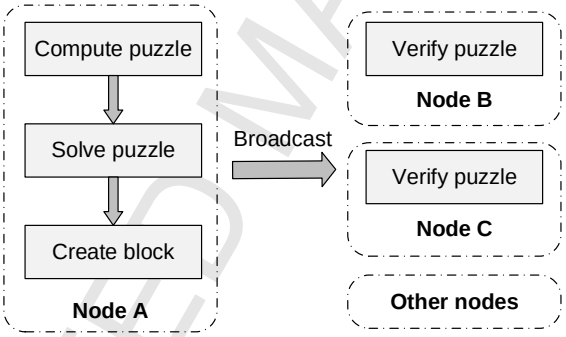


Figure 1: PoW consensus mechanism

PoW mechanism refers to that proving the credibility of the data by getting the solution of puzzles. The puzzle is computationally hard but easily verifiable problem. When a node creates a block or transaction, it must also resolve a PoW puzzle. After the PoW puzzle is resolved, it will be broadcasted to other nodes, so as to achieve the purpose of consensus, as shown in Fig.1.

Each block contains `PrevHash`, `nonce` and `T`. `PrevHash` is the characteristic information of the block. `nonce` is obtained by solving PoW puzzle and `T` is transaction. The goal of solving the PoW puzzle is to find a `nonce`, then the miner will make the `nonce` and some other information hashed together, the result of which must be less than a target value, as shown in the Equation.1. Furthermore, the difficulty of PoW puzzle can be adjusted by changing the target value.

$$H(prevHash \,||\, Tx1 \,||\, ... \,||\, Nonce) < Target \tag{1}$$

PoS mechanism refers to that proving the credibility of the data by the proof of ownership of cryptocurrency. In Ethereum, in the process of creating block or transaction, miners are required to pay a certain amount of Ethers (cryptocurrency in Ethereum). If the block or transaction created can eventually be validated, the Ether will be returned to the original node as a bonus, otherwise, it will be fined. In the PoW mechanism, it needs a lot of calculation, resulting in a waste of computing power. PoS mechanism can greatly reduce the amount of computation, thereby increasing the throughput of the entire blockchain system.

2.2. Block Propagation and Synchronisation

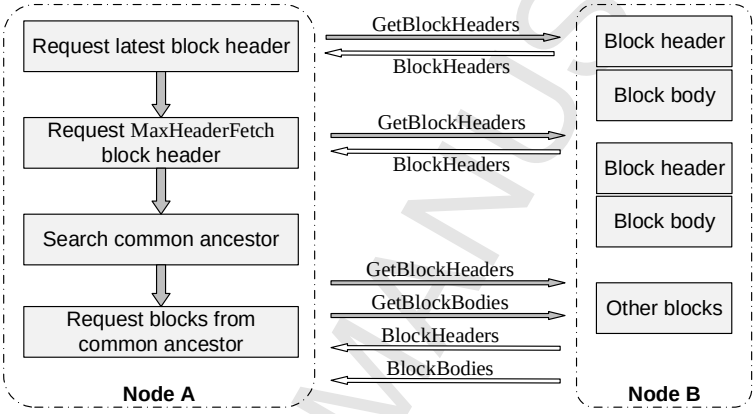


Figure 2: Block synchronisation process between nodes

In the blockchain, each node stores all blocks' information. Block propagation mechanism is the foundation to build consensus and trust for blockchain. Block propagation mechanisms are divided into the following categories [25, 26, 27]:

(1). Advertisement-based propagation. This propagation mechanism is originated from Bitcoin. When node `A` receives information of a block, `A` will send an `inv` message (a particular message type originated from Bitcoin) to all its connected peers. When node `B` receives the `inv` message from `A`, it will do the following: If node `B` already has the information of this block, it will not do anything. If node `B` does not have the information, it will reply to node `A`. When node `A` receives the reply message from node `B`, node `A` will send the complete information of this block to node `B`.

(2). Sendheaders propagation. This propagation mechanism is an improvement to the advertisement-based propagation mechanism. In the sendheaders propagation mechanism, node `B` will send a `sendheaders` message (a particular message type originated from Bitcoin) to node `A`. When node `A` receives information of a block, it will send the block header information directly to node `B`. Compared to the advertisement-based propagation mechanism, node `A` does not need to send `inv` messages, which speeds up block propagation.

(3). Unsolicited push propagation. In the unsolicited push mechanism, after one block is mined, the miner will directly broadcast the block to other nodes. In this propagation mechanism, there is no `inv` message and `sendheaders` message. Compared with the previous

two propagation mechanisms, unsolicited push mechanism can further improve the speed of block propagation.

(4). Relay network propagation. This propagation mechanism is an improvement to the unsolicited push mechanism. In this mechanism, all the miners share a transaction pool. Each transaction is replaced by a global ID, which will greatly reduce the broadcasted block size, thereby further reducing the network load and improve the propagation speed.

(5). Push/Advertisement hybrid propagation. This hybrid propagation mechanism is used in Ethereum. We assume that node `A` has n connected peers. In this mechanism, node `A` will push block to $\sqrt{n}$ peers directly. For the other $n - \sqrt{n}$ connected peers, node `A` will advertise the block hash to them.

The block synchronization process may be different in various blockchain systems. In Ethereum, node `A` can request block synchronization from node `B` with more total difficulty. The specific process is as follows (shown in Fig.2) [25, 26, 27].

(1). Node `A` requests the header of latest block from node `B`. This action is implemented by sending a `GetBlockHeaders` message. Node `B` will reply to node `A` a `BlockHeaders` message that contains the block header requested by `A`.

(2). Node `A` requests `MaxHeaderFetch` blocks to find common ancestor from node `B`. The value of `MaxHeaderFetch` defaults to 256, but the number of block headers sent by node `B` to `A` can be less than this value.

(3). If `A` has not found common ancestor after the above two steps. Node `A` will continue to send `GetBlockHeaders` message, requesting a block header one time. Also, node `A` repeats in binary search to find the common ancestor in its own blockchain.

(4). After node `A` discovers a common ancestor, `A` will request block synchronization from the common ancestor. In this process, `A` requests `MaxHeaderFetch` blocks per request, but the actual number of nodes that `B` sends to `A` can be less than this value.

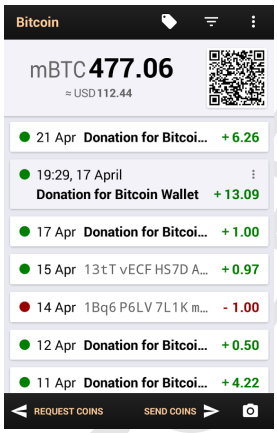


Figure 3: Query Bitcoin transaction history

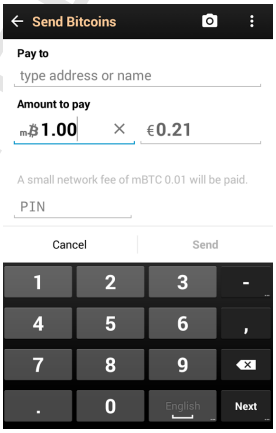


Figure 4: Pay with Bitcoin

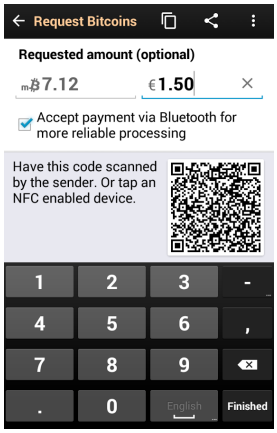


Figure 5: Collect payments with Bitcoin

2.3. Technology Development and Achievements

From the birth of the first blockchain system Bitcoin, blockchain technology has experienced two stages of development: blockchain 1.0 and blockchain 2.0.

In the blockchain 1.0 stage, blockchain technology is mainly used to cryptocurrency. In addition to Bitcoin, there are many other types of cryptocurrencies, such as Litecoin, Dogecoin and so on. There are currently over 700 types of cryptocurrencies, and the total market capitalizations of them are over 26 billion US$ [28]. The technology stack of cryptocurrency is divided into two layers: the underlying decentralized ledger layer and protocol layer [29]. Cryptocurrency client, such as Bitcoin Wallet, runs in the protocol layer to conduct transactions, as shown in Fig.3 to Fig.5. Compared to traditional currency, cryptocurrency has the following characteristics and advantages [30]:

(1). Irreversible and traceable. Transfer and payment operations are irreversible using cryptocurrency. Once the behavior is completed, it is impossible to withdraw. In addition, all users behaviors are traceable, and these behaviors are permanently saved in the blockchain.

(2). Decentralized and anonymous. There is no third-party organization involved in the entire structure of cryptocurrency, nor does it has central management like banks. In addition, all users behaviors are anonymous. So according to the transaction information we can not obtain the user's real identity.

(3). Secure and permissionless. The security of the cryptocurrency is ensured by the public key cryptography and the blockchain consensus mechanism, which are hard to be attacked by the criminal. In addition, there is no need to apply for any authority or permission to use cryptocurrency. Users only need to download the relevant client, which is very convenient.

(4). Fast and global. Transactions can be completed in only several minutes using cryptocurrency. In addition, cryptocurrencies are mostly based on public chain, so anyone in the world can use them. And the user's geographical location has little impact on the transaction speed.

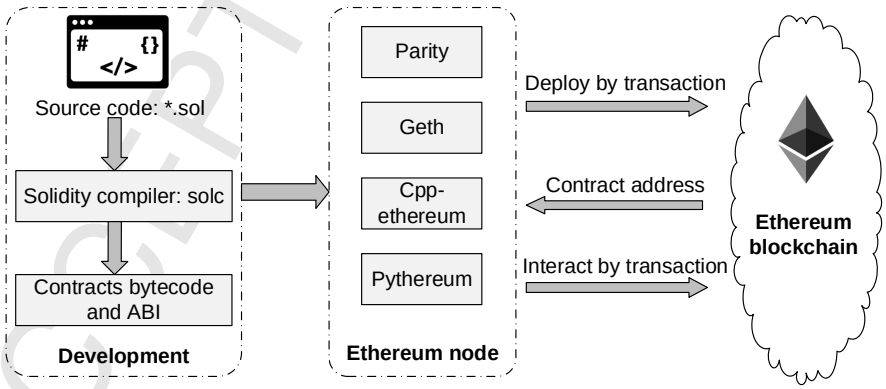


Figure 6: The process of smart contract's development, deployment and interaction

In blockchain 2.0 stage, blockchain technology is mainly used to develop various applications. Ethereum is a typical system of blockchain 2.0. In Ethereum, each node is running

Table 1: Statistics of blockchain systems supporting smart contracts (until 6th of May, 2017)

System	Contract language	Total TXs	Market Capitalization /M US$
Ethereum	EVM bytecode	23,102,544	8,468
Bitcoin	Bitcoin scripts	219,577,617	25,012
Counterparty	EVM bytecode	12,170,386	15
Stellar	Transaction chains	Unknown	139
Monax	EVM bytecode	Unknown	N/A
Lisk	JavaScript	Unknown	71

an EVM (Ethereum Virtual Machine). Users can develop smart contract to run in EVM. Developers can use a variety of programming languages to develop smart contracts, such as Solidity (the recommended language), Serpent, and LLL. These languages are Turing complete, so smart contract can achieve rich functions. The process of smart contracts' development, deployment and interaction are shown in Fig.6. Each deployed smart contract corresponds to a unique address. Therefore, users can use this address to interact with the smart contract, and the interaction is achieved in the form of transactions. Different smart contracts can also call each other through messages. Smart contract can be considered lightweight dAPP (decentralized application). Besides Ethereum, there are several blockchain systems supporting smart contracts now, and the statistics of them are shown in Table.1 [31]. Based on smart contract, developers can further integrate and develop more feature-rich dAPPs. Compared to the traditional application, dAPP has the following characteristics and advantages [32]:

(1). Autonomy. DAPP is developed on the basis of smart contract, and smart contracts are deployed and run in the blockchain. So dAPP run autonomically without any third partys assistance and participation, and there is no central server.

(2). Stable. The bytecode of smart contract is stored in every block in of blockchain. Each full node save all blocks information, including the bytecode of smart contract. So some of nodes shutdown will not affect its operation. This mechanism ensures that dAPP can run stably.

(3). Traceable. All the invocation information of smart contracts is stored in the blockchain as transactions. So all key behaviors of dAPP are recorded and traceable.

(4). Secure. The public key cryptography and the blockchain consensus mechanism can ensure the security and correct operation of smart contract, so as to maximize the security of dAPP.

3. Blockchain's Risks

In this section, we divide the common blockchain risks into six categories, as shown in Table 1. We will detailedly introduce the causes and possible harm of each risk in this section. Risks 3.1.* exist in blockchain 1.0 and 2.0, and their causes are mostly related to the blockchain operation mechanism. Risks 3.2.* are unique in blockchain 2.0, and their causes are mostly related to the development, deployment, and execution of smart contracts.

Table 2: Taxonomy of blockchain's risks

Number	Risk	Cause	Range of Influence
3.1.1	51% vulnerability	Consensus mechanism	Blockchain1.0, 2.0
3.1.2	Private key security	Public-key encryption scheme	
3.1.3	Criminal activity	Cryptocurrency application	
3.1.4	Double spending	Transaction verification mechanism	
3.1.5	Transaction privacy leakage	Transaction design flaw	
3.2.1	Criminal smart contracts	Smart contract application	Blockchain2.0
3.2.2	Vulnerabilities in smart contract	Program design flaw	
3.2.3	Under-optimized smart contract	Program writing flaw	
3.2.4	Under-priced operations	EVM design flaw	

3.1. General Risks

3.1.1. 51% Vulnerability

The blockchain relies on the distributed consensus mechanism to establish mutual trust. However, the consensus mechanism itself has 51% vulnerability, exploiting which attackers can essentially host the entire blockchain. In PoW-based blockchains, if a single miner's hashing power accounts for more than 50% of the total hashing power of the entire blockchain, then 51% attack may occur. The mining power concentrating in a few mining pools results in fears of an inadvertent situation, where a single pool controls more than half of all computing power and corresponding actions. On January of 2014, after the mining pool `ghash.io` reached 42% of the total Bitcoin computing power, a number of miners voluntarily dropped out of the pool and `ghash.io` issued a press statement to reassure the Bitcoin community that it would avoid reaching the 51% threshold [33]. In PoS-based blockchains, 51% attack may also occur if the number of coins owned by a single miner are more than 50% of the total blockchain. When 51% attack occurs, the attacker can arbitrarily manipulate and modify the blockchain information. Specifically, an attacker can exploit this vulnerability to initiate the following attacks [34]:

(1). Reverse transactions and initiate double spending attack (the same coins are spent multiple times).

(2). Exclude and modify the ordering of transactions.

(3). Prevent normal mining operations of other miners.

(4). Prevent normal transactions' confirmation operation.

3.1.2. Private Key Security

When we use blockchain, the private key is regarded as the identity and security credential. Private key is the basis to ensure blockchain security. Private key is generated and maintained by the user, no other third party agencies. For example, when you create a cold storage wallet in Bitcoin blockchain, you must import your private key. Hartwig et al. [35] discovered a vulnerability of ECDSA (Elliptic Curve Digital Signature Algorithm) scheme. In this vulnerability, it does not guarantee enough randomness during the signature process, which allows an attacker to recover the private key of a user. Once the user's private key is lost, it will not be able to be recovered. Once the private key is stolen by criminals, the user's blockchain account will face the risk of being tampered by others. As the blockchain is not dependent on any centralized third-party trusted institutions. So once the user's pri-

vate key is stolen, the criminals behavior will be hard to track and the modified blockchain information will be difficult to recover.

3.1.3. Criminal Activity

Table 3: Top 10 categories of items available in `Silk Road`

Number	Category	Items	Percentage
1	Weed	3338	13.7%
2	Drugs	2194	9.0%
3	Prescription	1784	7.3%
4	Benzos	1193	4.9%
5	Books	955	3.9%
6	Cannabis	877	3.6%
7	Hash	820	3.4%
8	Cocaine	630	2.6%
9	Pills	473	1.9%
10	Blotter (LSD)	440	1.8%

Bitcoin users can have multiple Bitcoin addresses, and the address has no relationship with their real life identity. Thus Bitcoin is anonymous in some way. Therefore, Bitcoin is widely used in various illegal activities. Through some third-party trading platforms that support Bitcoin, users can buy or sell any product. And this process is anonymous, so user behaviors are difficult to be tracked, let alone subject to legal sanctions. Some frequent criminal activities with Bitcoin are the following:

(1). Ransomware virus. There are hackers leveraging ransomware for money extortion, and Bitcoin is often used as trading currency. In July 2014, a ransomware named `CTB-Locker` [36] began to spread around the world. `CTB-Locker` was disguised as mail attachment to spread. After the user clicked the attachment, the virus will run in the background of the system. After the user's computer was infected with `CTB-Locker`, about 110 types of files will be encrypted. The user must pay to the attacker a certain amount of Bitcoin within 96 hours, otherwise the encrypted files will not be restored. In May 2017, another ransomware `WannaCry` (also named as `WannaCrypt`) [37] began to spread widely, which attacked about 230,000 victims across 150 countries in 2 days. `WannaCry` exploited a Windows system vulnerability to spread, and it also maliciously encrypted users' files to ask for Bitcoin ransom.

(2). Black market. Bitcoin is often used as the currency in black market. For example, `Silk Road` is an anonymous, international online marketplace that operates as a Tor hidden service and uses Bitcoin as its exchange currency [38]. And top 10 categories of items available in `Silk Road` are shown in Table.3 [38]. Most of the items sold in `Silk Road` are drugs, or some other controlled items in real world. And international transactions account for a significant proportion in `Silk Road`. Bitcoin makes the transaction in the black market more convenient, which will bring great harm to the social security.

(3). Money laundering. Bitcoin has features of anonymity and network virtual payment, and Bitcoin has been recognized by many countries. So compared with other currencies, Bitcoin carries the lowest risk of being used for money laundering [39]. Cody et al. proposed `Dark Wallet` [40], which is a Bitcoin application that can make Bitcoin transaction completely stealth and private. `Dark Wallet` can encrypt transaction information and mix

the users' valid coins with chaff coins. Dark Wallet can make money laundering much more easier.

3.1.4. Double Spending

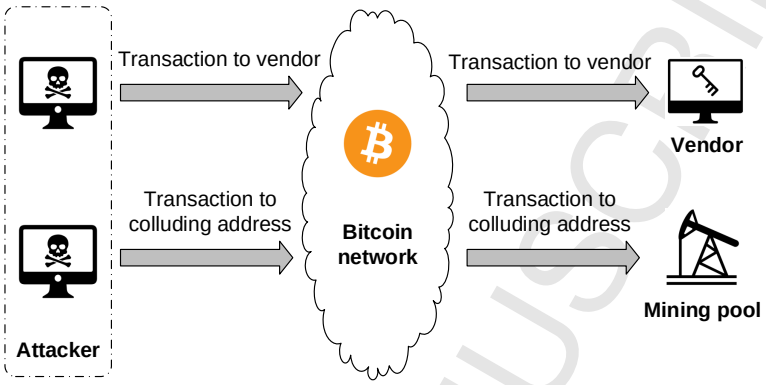


Figure 7: Double spending attack model against fast payment in Bitcoin

Although the consensus mechanism of blockchain can validate transactions, it is still impossible to avoid double spending. Double spending refers to that consumer uses the same cryptocurrency multiple times for transactions. For example, an attacker could leverage race attack for double spending. This kind of attack is relatively easy to implement in PoW-based blockchains. The attacker leverages the intermediate time between two transactions' initiation and confirmation to quickly make an attack. Before the second transaction is mined to be invalid, the attacker has already get the first transaction's output, resulting in double spending.

Ghassan et al. [41] conducted analysis of double spending against fast payment in Bitcoin. They proposed an attack model, as shown in Fig.7. Assuming that an attacker knows the vendor's address before the attack. In order to complete double spending, the attacker will send two transactions, TX_v and TX_a. And the attacker choose the same BTCs (cryptocurrency in Bitcoin) as inputs for TX_v and TX_a. TX_v's recipient address is set to the targeted vendor's address, and TX_a's recipient address is set to the colluding address controlled by the attacker. If the following three conditions are met, double spending will be successful: (1) TX_v is added to the wallet of the targeted vendor; (2) TX_a is mined valid into the blockchain; (3) The attacker gets TX_v's output before the vendor detects misbehavior. When the above conditions are met, TX_v will eventually be verified as invalid transaction, and BTCs are really spent by TX_a. The attacker has received TX_v's output, which is the vendor's normal service. Since TX_a's recipient address is controlled by the attacker, these BTCs are still owned by herself.

3.1.5. Transaction Privacy Leakage

As mentioned earlier, the users' behavior in the blockchain is traceable and anonymous. The blockchain takes measures to protect the transaction privacy of users. In the Bitcoin and Zcash, they use one-time accounts to store the received cryptocurrency. Also, the user

needs to assign a private key to each transaction. In this way, the attacker cannot infer whether cryptocurrency in different transactions are received by the same user. In Monero, users can include some chaff coins (called "mixins") when they initiate a transaction. In this way, the attacker cannot infer the linkage of actual coins really spent by the transaction.

Table 4: Linkability analysis of Monera transaction inputs with mixins

	Not deducible	**Deducible**	**In total**
Using newest TXO	15.07%	4.60%	19.67%
Not using newest TXO	22.61%	57.72%	80.33%
In total	37.68%	62.32%	100%

The privacy protection measures are not very robust in blockchain. Andrews et al. [42] empirically evaluate two linkability weaknesses in Monero's mixin sampling strategy. They discover that 66.09% of all transactions does not contain any mixins. 0-mixin transaction will not only lead to the privacy leakage of its sender, and it will bring harm to other users. Because users may use the outputs of 0-mixin transaction as mixins, then these mixins will be deducible. In addition, they study the sampling method of mixins and find that the selection of mixins is not really random. Newer transaction outputs (TXOs) tend to be used more frequently. They further discover that 62.32% of transaction inputs with mixins are deducible, as shown in Table.4 [42]. By exploiting these weaknesses in Monero, they can infer the actual transaction inputs with 80% accuracy, which will bring great harm to users' transaction privacy.

3.2. Unique Risks for Blockchain 2.0

3.2.1. Criminal Smart Contracts

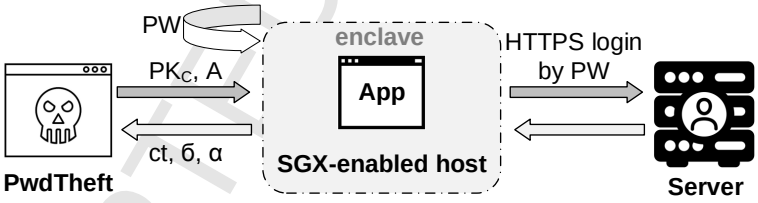


Figure 8: Execution procedure of `PwdTheft` using SGX-enabled platform

Smart contract technology is now in rapid development. Smart contracts can be developed by a variety of Turing-complete programming languages. Smart contracts can be used as many feature-rich applications. Now people pay more attention to the convenience smart contracts bring to people. However, there are also critical risks hidden in smart contracts. CSCs (Criminal Smart Contracts) can facilitate leakage of confidential information, theft of cryptographic keys, and various realworld crimes (murder, arson, terrorism) [43]. Juels et al. proposed an example of password theft CSC `PwdTheft`, the process of which is shown in Fig.8 [43]. `PwdTheft` can be leveraged for a fair exchange between contractor `C` and perpetrator `P`. `C` will pay a reward to `P` if and only if `P` gives a valid password to `C`. The entire transaction process can be done without any third party trusted agencies involved, which

will be subject to the risk of legal sanctions. In the actual work process of `PwdTheft`, it will be combined with trusted hardware technology, which can help to prove the validity of the password.

In this example, criminals can leverage Intel SGX (Software Guard eXtension) to do this job. SGX will create a trusted execution environment named `enclave`, which can protect the app from being attacked by others. Any privileged or unprivileged software can not access `enclave`. In addition, SGX can produce `quote`, a digitally signed attestation. `Quote` can get the hash value of the application run in `enclave` environment. Meanwhile, `quote` can access the relevant data during runtime of the application. The whole password exchange process is divided into three steps:

(1). `PwdTheft` provides (pk_C, A), pk_C is the public key of `C`, and `A` is the target account for stealing.

(2). Using the `PW` provided by `P`, log on to the server account `A` by establishing an HTTPS connection.

(3). If the preceding steps are successful, the data `ct`, σ and α will be transmitted to `PwdTheft`. `ct` $= enc_{pk_C}[PW]$ and $\sigma = Sig_{sk_{app}}[ct]$. sk_{app} is the signature private key of the app. α is a `quote` that runs on `P`'s SGX-enabled host.

After `PwdTheft` receive `ct`, σ and α, `C` can decrypt them to verify whether the data is valid, and then decide whether to pay reward to `P`. In this process, in order to prevent `P` from changing the password maliciously after the data transmission to `PwdTheft`, we can add a timestamp in the data. In addition, `PwdTheft` can also be improved to be used as other malicious CSCs. For example, criminals can leverage CSCs to make 0-days vulnerability transactions, which are critical cyber-weaponry. Overall, CSCs will bring the ecological environment of blockchain 2.0 huge threat and challenge.

3.2.2. Vulnerabilities in Smart Contract

Table 5: Taxonomy of vulnerabilities in smart contract

Number	Vulnerability	Cause	Level
1	Call to the unknown	The called function doesn't exist	Contract source code
2	Out-of-gas send	Fallback of the callee is executed	
3	Exception disorder	Irregularity in exception handling	
4	Type casts	Type-check error in contract execution	
5	Reentrancy vulnerability	Function is re-entered before termination	
6	Field disclosure	Private value is published by the miner	
7	Immutable bug	Alter a contract after deployment	EVM bytecode
8	Ether lost	Send ether to an orphan address	
9	Stack overflow	The number of values in stack exceeds 1024	
10	Unpredictable state	State of the contract is changed before invoking	Blockchain mechanism
11	Randomness bug	Seed is biased by malicious miner	
12	Timestamp dependence	Timestamp of block is changed by malicious miner	

As programs running in the blockchain, smart contract also has security vulnerabilities caused by program writing defects. Nicola et al. [21] conducted a systematic investigation and analysis of 12 vulnerabilities in smart contract, as shown in Table.5. Loi et al. [18] proposed a symbolic execution tool called Oyente to find 4 kinds of potential security bugs. They discovered that among the 19,366 existing Ethereum contracts, 8,833 of them are vulnerable. The details of these 4 bugs are as follows:

(1). Transaction-ordering dependence. Valid transactions can change the parameter of blockchain state σ to σ': $\sigma \xrightarrow{T} \sigma'$. In every epoch, each miner proposes their own block to update the blockchain. And a block may contain multiple transactions. So blockchain state σ may change within an epoch multiple times. When a new block contains multiple transactions, which call the same smart contract, it may trigger this vulnerability. Because the execution of smart contract is associated with state σ, the order of transactions' execution affects the results. The order of transactions' execution depends entirely on miners. So in this case, TOD (Transaction-Ordering Dependent) contracts are vulnerable.

(2). Timestamp dependence. In the blockchain, every block has a `timestamp`. Some smart contracts trigger conditions depend on `timestamp`, which is set by the miner. The value of `timestamp` depends on the miner nodes local system time, which can be changed by an attacker. So in this case, timestamp-dependent contracts are vulnerable.

(3). Mishandled exceptions. This category of vulnerability may occur when different smart contracts are called from each other. When contract `A` calls contract `B`, if `B` runs abnormally, `B` will stop running and return `false`. In some invocations, contract `A` must explicitly check the return value to verify if the call has been executed properly. If `A` does not correctly check the exception information, it may be vulnerable.

(4). Reentrancy vulnerability. During the invocation of the smart contract, the actual state of the contract account is changed after the call is completed. An attacker can use the intermediate state to make repeated calls to the smart contract. If the invoked contract involves Ether transaction, it may result in illegal Ether stealing.

3.2.3. Under-Optimized Smart Contract

Table 6: Taxonomy of under-optimized patterns in smart contract

Number	Under-optimized pattern	Category
1	Dead code	Useless-code related patterns
2	Opaque predicate	
3	Expensive operations	Loop-related patterns
4	Constant outcome	
5	Loop fusion	
6	Repeated computations	
7	Comparison with unilateral outcome	

When a user interact with a smart contract deployed in Ethereum, a certain amount of gas is charged. Gas can be exchanged with ether, which is the cryptocurrency in Ethereum. However, some smart contracts' development and deployment are not adequately optimized, resulting in the presence of additional operations that need to run, then the user need to pay additional money. Ting et al. [44] identify 7 related gas-costly patterns and group them into 2 categories (as shown in Table.6): useless-code related patterns, and loop-related patterns. They proposed a tool Gasper, which can automatically discover 3 gas-costly patterns in smart contracts: dead code, opaque predicate, and expensive operations in a loop. Leveraging Gasper, they find that more than 80% smart contracts deployed in Ethereum (4,240 real smart contracts) have at least one of these 3 patterns, the details of which are as follows:

(1). Dead code. It means that some of the operations in the bytecode of smart contract will never be executed, but they will still be deployed into the blockchain. In the smart contract deployment process, the consumption of gas is related to bytecode size, so dead code will cause additional gas consumption.

(2). Opaque predicate. It refers to that in the process of running smart contract, the results of some statements are always the same, and they will not affect other statements. The presence of the opaque predicate causes the EVM to execute useless operations, thereby consuming additional gas.

(3). Expensive operations in a loop. It refers to that there are some calls of expensive operations in a loop. These expensive operations can be moved outside the loop, thus saving some gas consumption.

3.2.4. Under-Priced Operations

As mentioned earlier, each operation is set with a specific gas value in Ethereum, which can be queried in the yellow paper [45]. Ethereum sets the gas value based on the execution time, bandwidth, memory occupancy and other parameters. In general, the gas value is proportional to the computing resources consumed by the operation. However, the consumption of computing resources is difficult to measure accurately, resulting in that some gas values are set unreasonable. Among them, some IO-heavy operations' gas values are set too low, so these operations can be executed in quantity in one transaction. In this way, an attacker can initiate a DoS attack on Ethereum.

Table 7: Gas table modification in EIP150

Number	Operation	Old value	EIP150 value
1	EXTCODESIZE	20	700
2	EXTCODECOPY	20	700
3	BALANCE	20	400
4	SLOAD	50	200
5	CALL	40	700
6	SUICIDE (doesn't create account)	0	5000
7	SUICIDE (creates an account)	0	25000

The attacker exploited the under-priced operation `EXTCODESIZE` to attack Ethereum [46]. When `EXTCODESIZE` is executed it needs to read state information, then the node will read hard disk. As the gas value of `EXTCODESIZE` is only 20, the attacker can call it more than 50,000 times in one transaction. This will cause the user to consume a lot of computing resources, and block synchronization will be significantly slower compared to the normal situation. In addition, some attackers exploited another under-priced operation `SUICIDE` to launch an attack [47]. They leveraged `SUICIDE` to create about 19 million empty accounts, which need to be stored in the state tree. This attack caused a waste of hard disk resources. At the same time, the node information synchronization and transaction processing speed are significantly decreased.

In order to solve the security problem caused by under-priced operations, the gas values of 7 IO-heavy operations are modified in EIP (Ethereum Improvement Proposal) 150 [48], as shown in the Table.7. EIP150 has already been implemented in the Ethereum public chain by hard fork. The new gas table parameters are used after No.2463000 block.

4. Attack Cases

In this section, we enumerate real attack cases suffered by current blockchain systems, and we analyze the types and principles of vulnerabilities exploited in these attack cases.

4.1. Selfish Mining Attack

The selfish mining attack is conducted by attackers (i.e., selfish miners) for the purpose of obtaining undue rewards or wasting the computing power of honest miners [49]. The attacker holds discovered blocks privately and then attempts to fork a private chain [50]. Afterwards the attacker would mine on the private branch and try to develop a longer chain as it privately mining more blocks. In the meantime, honest miners continue mining on the public chain. New blocks mined by the attacker would be revealed when the public branch approaches the length of private branch, such that the honest miners end up wasting computing power and gaining no reward. As a result, the attacker gains a competitive advantage and rational miners would be incentivized to join its branch of the blockchain. Through a further consolidation of mining power into the attacker's favor, this attack undermines the decentralized nature of blockchain.

Ittay et al. [50] proposed a Selfish-Mine strategy, which can force the honest miners into performing wasted computations on the stale public branch. In the initial circumstance of Selfish-Mine, the length of the public chain and private chain are the same. The Selfish-Mine involves the following three scenarios:

(1). Public chain is longer than the private chain. Because the computing power of selfish miners are less than that of the private miners. In this scenario, selfish miners will update the private chain according to the public chain, so selfish miners can not gain any reward.

(2). Selfish miners and honest miners almost simultaneously find the first new block. In this scenario, selfish miners will publish the newly discovered block. At this point, there will be two concurrently forks of the same length. Honest miners will mine in either of the two branches. And selfish miners will continue to mine on the private chain. If selfish miners firstly find the second new block, they will publish this block immediately. At this point selfish miners will gain two blocks' rewards at the same time. Because the private chain is longer than the public chain, the private chain will be the ultimate valid branch. If honest miners firstly find the second new block: If this block is written to the private chain, selfish miners will gain the first new block' rewards, and honest miners will gain the second new block' rewards. If this block is written to the public block, honest miners will gain these two new blocks' rewards, and selfish miners will not gain any reward.

(3). After selfish miners find the first new block, they also find the second new block. In this scenario, selfish miners will hold these two new blocks privately, and they continue to mine new blocks on the private chain. When the first new block is found by honest miners, selfish miners will publish its own first new block. When honest miners find the second new block, the selfish miners will immediately publish its own second new block. Then selfish miners will follow this response in turn, until the length of the public chain is only 1 greater than the private chain, after which the selfish miners will publish its last new block before

honest miners finds this block. At this point, the private chain will be considered valid, so selfish miners will gain rewards of all new blocks.

4.2. DAO Attack

Table 8: Some other attacks that exploit smart contracts' vulnerabilities

Number	Attack case	Related vulnerabilities
1	King of the ether throne	Out-of-gas send Exception disorder
2	Multi-player games	Field disclosure
3	Rubixi attack	Immutable bug
4	GovernMental attack	Immutable bug Stack overflow Unpredictable state Timestamp dependence
5	Dynamic libraries attack	Unpredictable state

The `DAO` is a smart contract deployed in Ethereum on 28th May of 2016, which implements a crowd-funding platform. The `DAO` contract was attacked only after it has been deployed for 20 days. Before the attack happened, `DAO` has already raised 150 million US$, which is the biggest crowdfund ever. The attacker successfully stole about 60 million US$, and it is regarded the most important attack case in cryptocurrency since the birth of Bitcoin.

The attacker exploited the reentrancy vulnerability in this case. Firstly, the attacker publishes a malicious contract, in which there is a `withdraw()` function `call` to `DAO` in its callback function. The `withdraw()` will send ether to the callee, which is also in the form of `call`. Therefore, it will invoke the callback function of the malicious again. In this way, the attacker is able to steal all the ether from `DAO`. There are some other cases [21] that exploit smart contracts' vulnerabilities (described in Section 3.2.2), some of which are shown in the Table.8.

4.3. BGP Hijacking Attack

BGP is a de-facto routing protocol and regulates how IP packets are forwarded to their destination. To intercept Bitcoin traffic at scale, attackers either leverage or manipulate BGP routing. BGP hijacking typically requires the control of network operators while independent from Bitcoin, which could potentially be exploited to delay network messages. Maria et al. [51] comprehensively analyzed the impact of network attacks on Bitcoin from terms of both node-level and network-wide attacks and showed that the number of the successfully to-be-hijacked Internet prefixes depends on the distribution of mining power. Internet-level BGP hijacking is executed by configuring an edge router to non-assigned announce prefixes. If the malicious announcement is more specific than the valid one or claims to offer a shorter path, the traffic is possibly directed to the attacker. To avoid attention from the valid owner, attackers frequently target unused prefixes for hijacking. The compromised router may poison the RIB (Routing Information Base) of its peers by broadcasting false prefix announcements. After poisoning one peer, the malicious routing information could propagate to other peers, autonomous systems, and the broader Internet [52].

BGP hijacking was leveraged to intercept Bitcoin miners' connections to a mining pool server, as analyzed by Dell SecureWorks in 2014 [53]. By rerouting traffic to a mining pool controlled by the attacker, it was possible to steal cryptocurrency from the victim's mining. This attack collected an estimated US$83,000 of cryptocurrency over a two month period. BGP hijacking is actually within the reach of anyone who can access a BGP-enabled network and hijack less than 900 prefixes. BGP security extensions are seldom deployed, such that operators just rely on monitoring systems. These systems would report rogue announcements, such as BGPMon [54]. Even if detected, solving a hijacking still cost hours as it is a human-driven process consisting of altering configuration or disconnecting the attacker. For example, YouTube ever took about three hours to resolve a hijacking of its prefixes by a Pakistani ISP [55].

4.4. Eclipse Attack

Table 9: Some other attacks that may be caused by the eclipse attack

Number	Attack	Harm
1	Engineering block races	Wasting mining power on orphan blocks
2	Splitting mining power	51% vulnerability may be triggered
3	Selfish mining	Attacker can gain more than normal mining rewards
4	0-confirmation double spend	The vendor wouldn't get rewards for its service
5	N-confirmation double spend	

The eclipse attack allows an attacker to monopolize all of the victim's incoming and outgoing connections, which isolates the victim from the other peers in the network [56]. Then the attacker is able to filter the victim's view of the blockchain or let the victim cost unnecessary computing power on obsolete views of the blockchain. Also, it coopts the victim's computing power due to its own nefarious targets. Ethan et al. [57] considers two types of eclipse attack on Bitcoin's peer-to-peer network, namely botnet attack and infrastructure attack. The botnet attack is launched by bots with diverse IP address ranges. The infrastructure attack models the threat from an ISP (Internet Service Provider), company or nation-state that has contiguous IP addresses. The Bitcoin network might suffer from disruption and a victim's view of the blockchain will be filtered due to the eclipse attack. Additionally, the eclipse attack is a useful basis for other attacks, as shown in Table.9.

4.5. Liveness Attack

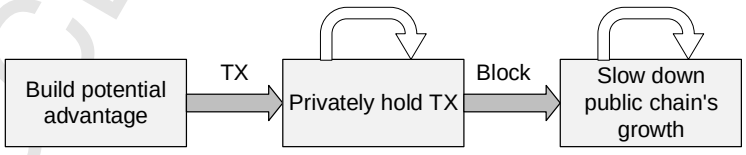


Figure 9: Overview of the liveness attack process

Aggelos et al. [58] proposed the liveness attack, which is able to delay as much as possible the confirmation time of a target transaction. They also present two instantiations of the

this attack in Bitcoin and Ethereum. Liveness attack consists of three phases, namely attack preparation phase, transaction denial phase, and blockchain retarder phase (shown in Fig.9):

(1). Attack preparation phase. Just like selfish mining attack, an attacker builds advantage over honest miners in some way before the target transaction TX is broadcasted to public chain. The attacker builds the private chain, which is longer than the public chain.

(2). Transaction denial phase. The attacker privately holds the block that contains TX, in order to prevent TX from being written into public chain.

(3). Blockchain retarder phase. In the process of public chain's growth, TX will no longer be able to be privately hold in a certain time. In this case, the attacker will publish the block that contains TX. In some blockchain systems, when the depth of the block that contains TX is greater than a constant, TX will be regarded valid. Therefore, the attacker will continue building private chain in order to build advantage over public chain. Then the attacker will publish the blocks into private chain in proper time to slow down the rate at which public chain grows. The liveness attack will end when TX is verified valid in public chain.

4.6. Balance Attack

Christopher et al. [59] proposed the balance attack against PoW-based blockchain, which allows a low-mining-power attacker to momently disrupt communications between subgroups with similar mining power. They abstracted blockchain into a DAG (Directed Acyclic Graph) tree, in which DAG $= < B, P >$. After introducing a delay between correct subgroups of equivalent mining power, the attacker issues transactions in one subgroup (called "transaction subgroup") and mines blocks in another subgroup (called "block subgroup"), to guarantee that the tree of block subgroup outweighs the tree of transaction subgroup. Even though the transactions are committed, the attacker is able to outweigh the tree containing this transaction and rewrite blocks with high probability.

The balance attack inherently violates the persistence of the main branch prefix and sufficiently allows to conduct double spending. The attacker needs to identify the merchant-involved subgroup and create transactions to purchase goods from those merchants. Thereafter the attacker issues transactions to this subgroup and propagates the mined blocks to the rest nodes of the group. As long as the merchant ships goods, the attacker stops delaying messages. With a high probability that the DAG tree seen by the merchant is outweighed by another tree, so the attacker would successfully reissue another transaction using the exactly same coins. Balance attack proves that PoW-based blockchain is block oblivious: When writing a transaction into the main chain, there is a certain probability that we can override or delete the block containing this transaction. In the related experiment, the authors configured an Ethereum private chain with equivalent R3 torortium parameters. They can successfully carry out the balance attack, which only needs to control about 5% total computation power.

5. Security Enhancement Achievements

In this section, we summarize blockchain security enhancement achievements for blockchain systems, which can guide the healthy development of the blockchain technology.

5.1. SmartPool

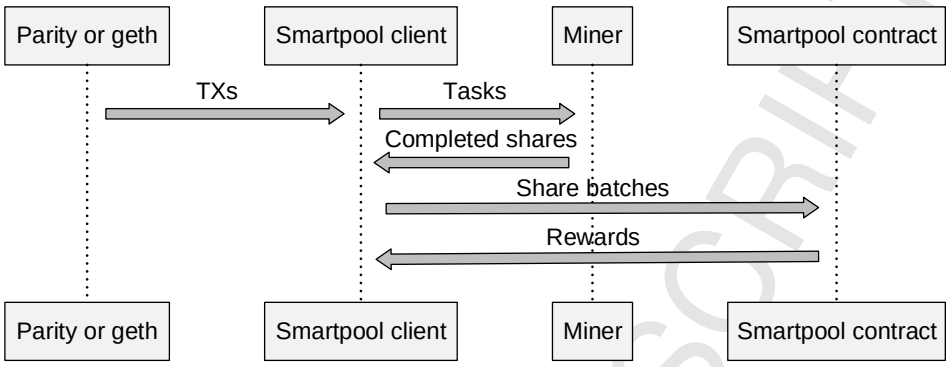


Figure 10: Overview of SmartPool's execution process

As described in Section 3.1.1, there already has mining pool with more than 40% of total computing power of blockchain. This poses a serious threat to the decentralization nature, making blockchain vulnerable to several kinds of attacks. Loi et al. [60] proposed a novel mining pool system named SmartPool, the workflow of which is shown in Fig.10. SmartPool gets transactions from parity or geth, which contain mining tasks information. Then the miner conducts hashing computation based on the tasks, and it returns completed shares to the smartpool client. When the completed has accumulated to a certain amount, they will be committed to smartpool contract, which is deployed in Ethereum. The smartpool contract will verify the shares and deliver rewards to the client. Compared to traditional P2P pool, SmartPool system has the following advantages:

(1). Decentralized. The core of the SmartPool is implemented in the form of smart contract, which is deployed in blockchain. Miners need firstly connect to Ethereum to mine through the client. Mining pool can rely on Ethereum's consensus mechanism to run. In this way, it ensures decentralization nature of pool miners. The mining pool state is maintained by Ethereum and no longer requires a pool operator.

(2). Efficiency. Miners can send the completed shares to the smartpool contract in batches. Furthermore, miners only need to send part of shares to be verified, not all shares. This form is more efficient than the P2P pool.

(3). Secure. SmartPool leverages a novel data structure, which can prevent attacker from resubmitting shares in different batches. Furthermore, the verification method of SmartPool can guarantee that honest miners will gain expected reward even there exist malicious miners in the pool.

5.2. Quantitative Framework

There exist tradeoffs between blockchain's performance and security. Arthur et al. [61] proposed a quantitative framework, which is leveraged to analyze PoW-based blockchain's execution performance and security provisions. The operating principle of the framework is shown in Fig.11. The framework is divided into two components: blockchain stimulator and security model. The stimulator mimics blockchain's execution, the inputs of which

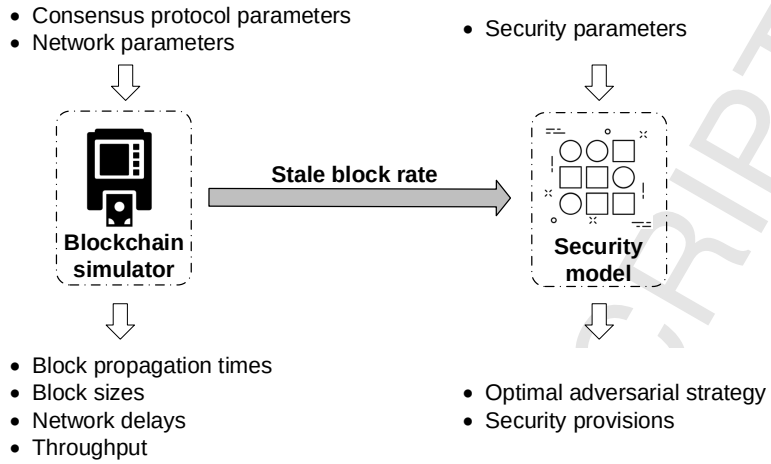


Figure 11: Components of quantitative framework

are parameters of consensus protocol and network. Through the simulator's analysis, it can gain performance statistics of the target blockchain, including block propagation times, block sizes, network delays, stale block rate, throughput, etc. Stale block refers to block that is mined but not written to the public main chain. Throughput mainly refers to the number of transactions that the blockchain can handle per second. Stale block rate will be passed as a parameter to the security model component, which is based on MDP (Markov Decision Processes) for double spending and selfish mining attacks. The framework eventually outputs optimal adversarial strategy against attacks, and helps to build security provisions for the blockchain.

5.3. Oyente

Loi et al. [62] proposed Oyente, which can be used to detect bugs in Ethereum smart contracts. Oyente is a symbolic execution tool exclusively designed to analyze the bytecode smart contracts in Ethereum. It is the first security analyzer for smart contracts. If users directly detect the source code (written by Solidity) of smart contract, Oyente will use solc to firstly compile the contract into bytecode. It follows the execution model of EVM, and it can directly works with EVM bytecode without access to the high level representation of contracts. Ethereum blockchain stores the EVM bytecode of contracts, so Oyente can be directly leveraged to detect bugs in deployed contracts.

Oyente's architecture design and execution process is shown in Fig.12. The tool takes the smart contract's bytecode and Ethereum global state as inputs. Firstly, based on bytecode, `CFG BUILDER` will statically build CFG (Control Flow Graph) of smart contract. Then, according to Ethereum state and CFG information, `EXPLORER` conducts simulative execution of smart contract leveraging static symbol execution technology. In this process, CFG will be further enriched and improved. The `CORE ANALYSIS` module uses the related analysis algorithms to detect four different vulnerabilities (described in Section 3.2.2). The `VALIDATOR` module validates the detected vulnerabilities and vulnerable paths. Confirmed vulnerability and CFG information will finally be output to the `VISUALIZER` module, which can be used

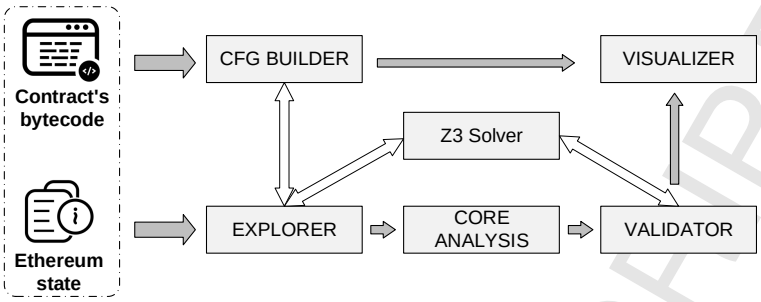


Figure 12: Overview of Oyente's architecture design and execution process

by users to carry out debugging and program analysis. Currently, Oyente is availably open source for public use [63].

5.4. Hawk

As described in Section 3.1.5, privacy leakage is a serious threat faced by blockchain. In fact, at the beginning of the blockchain's creation, the designer only focused on the correctness and availability, and they did not put users' privacy in high priority. In the era of blockchain 2.0, not only transaction privacy, but also a lot of contract-related privacy is also public, such as contract's bytecode, execution trace, invoking parameters, etc.

Ahmed et al. [64] proposed Hawk, a novel framework for developing privacy-preserving smart contracts. Leveraging Hawk, developers can write private smart contract, and it is not necessary for them to use any code encryption or obfuscation technology. Furthermore, the financial transaction's information will not be explicitly stored in blockchain. When programmers develop Hawk contract, the contract can be divided into two parts: private portion, and public portion. The private data and financial function related codes can be written into the private portion, and codes that do not involve privacy can be written into the public portion. The Hawk contract is compiled into three pieces: (1). Program that will be executed in all virtual machines of nodes, just like smart contracts in Ethereum. (2). Program that will only be executed be contract users. (3). Program that will be executed by the manager, which is a special trustworthy party in Hawk. The Hawk manager is executed in Intel SGX `enclave` (described in Section 3.1.3), and it can see the privacy information of the contract but will not disclose it. Hawk can not only protect privacy against the public, and it can protect the privacy between different Hawk contracts. If the manager aborts the protocol of Hawk, it will be automatically financially penalized, and the users will gain compensation. Overall, Hawk can largely protect the privacy of users when they are using blockchain technology.

5.5. Town Crier

Smart contract often needs to interact with off-chain data source. Fan et al. [65] proposed TC (Town Crier), which is an authenticated data feed system for this data interaction process. The smart contract deployed in blockchain does not have the ability of network connection, so contract can not get data through HTTPS. TC exactly acts as a bridge

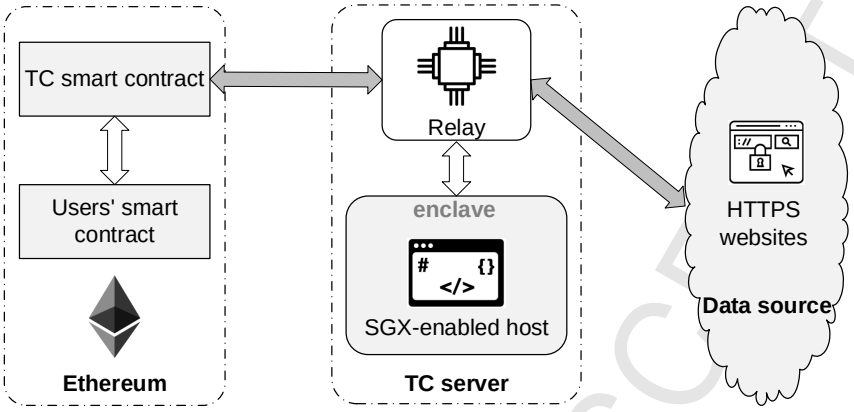


Figure 13: Basic architecture of Town Crier system

between HTTPS-enabled data source and Ethereum smart contract. The basic architecture of TC is shown in Fig.13. TC contract is the front end of the TC system, which acts as API between users' contracts and TC server. The core program of TC is running in Intel SGX `enclave` (described in Section 3.1.3). The main function of the TC server is to obtain the data request from users' contract, and obtain the data from target HTTPS-enabled website. Finally, the TC server will return a datagram to the users' contract in the form of digitally signed blockchain message.

TC can largely protect the security in the process of data requesting. The core modules of TC are respectively running on decentralized Ethereum, SGX-enabled `enclave`, and HTTPS-enabled website. Furthermore, the `enclave` disables the function of network connection to maximize its security. `Relay` module is designed as a network communication hub for smart contracts, `Relay` module, and data source websites. Even if that the `Relay` module is attacked, and the network communication packets are tampered, it will not change the normal function of TC. TC system provides a robust security model for the smart contracts' off-chain data interaction, and it has already been launched online as a public service [66].

6. Future Directions

Based on the above systematic examination on the security of current blockchain systems, we list a few future directions to stir up research efforts into this area. First, now the most popular consensus mechanism leveraged in blockchain is PoW. However, PoW has a prominent disadvantage, that is a waste of computing resources. To solve this problem, Ethereum is trying to develop a hybrid consensus mechanism of PoW and PoS. Conduct research and develop more efficient consensus mechanisms will make a significant contribution to the development of blockchain. Second, with the growth of the number of feature-rich dAPPs, the privacy leakage risk of blockchain will be more serious. DAPP itself, as well as the process of communication between dAPP and the Internet, are both faced with privacy leakage risks. There are some interesting technologies that can be applied in in this problem: code obfuscation, application hardening, execution trusted computing (eg., Intel SGX), etc.

Third, with the operation of the blockchain system, it will produce a lot of data, including block information, transaction data, contract bytecode, etc. However, not all of the data stored in blockchain is valid. For example, a smart contract can erase its code by `SUICIDE` or `SELFDESTRUCT`, but the address of the contract will not be erased. In addition, there are a lot of smart contracts containing no code or totally the same code in Ethereum, and many smart contracts are never be executed after its deployment. To achieve efficient data cleanup and detection mechanism, will improve the execution efficiency of blockchain systems.

7. Conclusion

In this paper, we focus on the security issues of blockchain technology. Based on some typical blockchain systems (e.g., Ethereum, Bitcoin, Monero, etc.), we systematically investigate and classify security risks faced by blockchain according to its characteristics. For each risk or vulnerability, we analyze its causes and possible harm. Furthermore, we enumerate real attack cases suffered by the blockchain, and analyze the types and principles of vulnerabilities exploited in these attacks. Finally, we summarize blockchain security enhancements, in order to guide the healthy development of the blockchain.

References

[1] J. Tellez, Bitcoin and the cashless future (2016).
URL `http://www.forbes.com/sites/realspin/2016/12/20/bitcoin-and-the-cashless-future/#521d83bb4df6`

[2] J. DESJARDINS, It's official: Bitcoin was the top performing currency of 2015 (2016).
URL `http://money.visualcapitalist.com/its-official-bitcoin-was-the-top-performing-currency-of-2015/`

[3] J. Adinolfi, And 2016's best-performing commodity is ... bitcoin? (2016).
URL `http://www.marketwatch.com/story/and-2016s-best-performing-commodity-is-bitcoin-2016-12-22`

[4] blockchain.info, Confirmed transactions per day (2017).
URL `https://blockchain.info/charts/n-transactions?timespan=all#`

[5] A. Ekblaw, A. Azaria, J. D. Halamka, A. Lippman, A case study for blockchain in healthcare:medrec prototype for electronic health records and medical research data (2016).
URL `https://www.media.mit.edu/publications/medrec-whitepaper/`

[6] A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, Medrec: Using blockchain for medical data access and permission management, in: International Conference on Open and Big Data (OBD), 2016, pp. 25–30.

[7] X. Yue, H. Wang, D. Jin, M. Li, W. Jiang, Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control, in: Journal of medical systems, Vol. 40, Springer, 2016, p. 218.

[8] S. Huckle, R. Bhattacharya, M. White, N. Beloff, Internet of things, blockchain and shared economy applications, in: Procedia Computer Science, Vol. 98, Elsevier, 2016, pp. 461–466.

[9] P. Bylica, L. Gleń, P. Janiuk, A. Skrzypczak, A. Zawłocki, A probabilistic nanopayment scheme for golem, 2015.
URL `http://golemproject.net/doc/GolemNanopayments.pdf`

[10] P. Hurich, The virtual is real: An argument for characterizing bitcoins as private property, in: Banking & Finance Law Review, Vol. 31, Carswell Publishing, 2016, p. 573.

[11] A. Dorri, S. S. Kanhere, R. Jurdak, P. Gauravaram, Blockchain for iot security and privacy: The case study of a smart home, in: IEEE Percom workshop on security privacy and trust in the internet of thing, 2017.

[12] Y. Zhang, J. Wen, The iot electric business model: Using blockchain technology for the internet of things, in: Peer-to-Peer Networking and Applications, Springer, 2016, pp. 1–12.
[13] J. Sun, J. Yan, K. Z. Zhang, Blockchain-based sharing services: What blockchain technology can contribute to smart cities, in: Financial Innovation, Vol. 2, Springer, 2016, p. 26.
[14] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, S. Chen, The blockchain as a software connector, in: The 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2016.
[15] E. Nordström, Personal clouds: Concedo, Master's thesis, Lulea University of Technology (2015).
[16] J. S. Czepluch, N. Z. Lollike, S. O. Malone, The use of block chain technology in different application domains, in: The IT University of Copenhagen, Copenhagen, 2015.
[17] Ethereum, Etherscan: The ethereum block explorer (2017).
URL https://www.ethereum.org/
[18] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, in: The 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 254–269.
[19] V. Buterin, Critical update re: Dao vulnerability (2016).
URL https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/
[20] J. Adelstein, Behind the biggest bitcoin heist in history: Inside the implosion of mt.gox (2016).
URL http://www.thedailybeast.com/articles/2016/05/19/behind-the-biggest-bitcoin-heist-in-history-inside-the-implosion-of-mt-gox.html
[21] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on ethereum smart contracts (sok), in: International Conference on Principles of Security and Trust, Springer, 2017, pp. 164–186.
[22] M. Castro, B. Liskov, et al., Practical byzantine fault tolerance, in: Symposium on Operating Systems Design and Implementation, Vol. 99, 1999, pp. 173–186.
[23] M. Ghosh, M. Richardson, B. Ford, R. Jansen, A torpath to torcoin, proof-of-bandwidth altcoins for compensating relays (2014).
URL https://www.smithandcrown.com/open-research/a-torpath-to-torcoin-proof-of-bandwidth-altcoins-for-compensating-relays/
[24] Intel, Proof of elapsed time (poet) (2017).
URL http://intelledger.github.io/
[25] Karl, Security of blockchain technologies, Ph.D. thesis, Swiss Federal Institute of Technology (2016).
[26] Karl, Ethereum eclipse attacks, 2016.
URL http://e-collection.library.ethz.ch/view/eth:49728
[27] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: The 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 3–16.
[28] CoinMarketCap, Cryptocurrency market capitalizations (2017).
URL https://coinmarketcap.com/
[29] M. Swan, Blockchain: Blueprint for a new economy, O'Reilly Media, 2015.
[30] BlockGeeks, What is cryptocurrency: Everything you need to know (2016).
URL https://blockgeeks.com/guides/what-is-cryptocurrency/
[31] M. Bartoletti, L. Pompianu, An empirical analysis of smart contracts: platforms, applications, and design patterns, in: arXiv preprint:1703.06322, 2017.
[32] BlockGeeks, Smart contracts: The blockchain technology that will replace lawyers (2016).
URL https://blockgeeks.com/guides/smart-contracts/
[33] N. Hajdarbegovic, Bitcoin miners ditch ghash.io pool over fears of 51% attack (2014).
URL http://www.coindesk.com/bitcoin-miners-ditch-ghash-io-pool-51-attack/
[34] Dean, 51% attack (2015).
URL http://cryptorials.io/glossary/51-attack/
[35] H. Mayer, Ecdsa security in bitcoin and ethereum: a research survey, 2016.
URL http://blog.coinfabrik.com/wp-content/uploads/2016/06/ECDSA-Security-in-Bitcoin-and-Ethereum-a-Research-Survey.pdf

[36] S. Alliance, Know your ransomware: Ctb-locker (2017).
URL https://www.secalliance.com/blog/ransomware-ctb-locker/
[37] Wikipedia, Wannacry ransomware attack (2017).
URL https://en.wikipedia.org/wiki/WannaCry_ransomware_attack
[38] N. Christin, Traveling the silk road: A measurement analysis of a large anonymous online marketplace, in: The 22nd international conference on World Wide Web, 2013, pp. 213–224.
[39] H. Treasury, Uk national risk assessment of money laundering and terrorist financing (2015).
URL https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/468210/UK_NRA_October_2015_final_web.pdf
[40] W. Cody, T. Amir, Darkwallet (2017).
URL https://darkwallet.is/
[41] G. O. Karame, E. Androulaki, S. Capkun, Double-spending fast payments in bitcoin, in: Proceedings of the ACM conference on Computer and communications security, 2012, pp. 906–917.
[42] A. Miller, M. Möser, K. Lee, A. Narayanan, An empirical analysis of linkability in the monero blockchain, in: arXiv preprint:1704.04299, 2017.
[43] A. Juels, A. Kosba, E. Shi, The ring of gyges: Investigating the future of criminal smart contracts, in: The 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 283–295.
[44] T. Chen, X. Li, X. Luo, X. Zhang, Under-optimized smart contracts devour your money, in: IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 442–446.
[45] E. community, The "yellow paper": Ethereum's formal specification (2017).
URL https://github.com/ethereum/yellowpaper
[46] Gautham, Ethereum network comes across yet another dos attack (2016).
URL http://www.newsbtc.com/2016/09/23/ethereum-dao-attack-attack-platforms-credibility/
[47] B. Rivlin, Vitalik buterin on empty accounts and the ethereum state (2016).
URL https://www.ethnews.com/vitalik-buterin-on-empty-accounts-and-the-ethereum-state
[48] E. community, Long-term gas cost changes for io-heavy operations to mitigate transaction spam attacks (2016).
URL https://github.com/ethereum/EIPs/issues/150
[49] S. Solat, M. Potop-Butucaru, Zeroblock: Preventing selfish mining in bitcoin, Ph.D. thesis, University of Paris (2016).
[50] I. Eyal, E. G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, in: Financial Cryptography and Data Security - 18th International Conference, Vol. 8437 of Lecture Notes in Computer Science, 2014, pp. 436–454.
[51] M. Apostolaki, A. Zohar, L. Vanbever, Hijacking bitcoin: Large-scale network attacks on cryptocurrencies, in: arXiv preprint:1605.07524, 2016.
[52] Z. Julian, An overview of bgp hijacking (2015).
URL https://www.bishopfox.com/blog/2015/08/an-overview-of-bgp-hijacking/
[53] D. SecureWorks, BGP hijacking for cryptocurrency profit (2014).
URL https://www.secureworks.com/research/bgp-hijacking-for-cryptocurrency-profit
[54] H. Yan, R. Oliveira, K. Burnett, D. Matthews, L. Zhang, D. Massey, BGPmon: A real-time, scalable, extensible monitoring system, in: Cybersecurity Applications Technology Conference for Homeland Security, 2009, pp. 212–223.
[55] D. Research, Pakistan hijacks youtube (2008).
URL http://research.dyn.com/2008/02/pakistan-hijacks-youtube-1/
[56] A. Singh, T. Ngan, P. Druschel, D. S. Wallach, Eclipse attacks on overlay networks: Threats and defenses, in: 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 2006.
[57] E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on bitcoin's peer-to-peer network, in:

24th USENIX Security Symposium, 2015, pp. 129–144.

[58] A. Kiayias, G. Panagiotakos, On trees, chains and fast transactions in the blockchain, 2016. URL https://eprint.iacr.org/2016/545.pdf

[59] C. Natoli, V. Gramoli, The balance attack against proof-of-work blockchains: The r3 testbed as an example, in: arXiv preprint:1612.09426, 2016.

[60] L. Luu, Y. Velner, J. Teutsch, P. Saxena, Smart pool: Practical decentralized pooled mining, in: USENIX Security Symposium, 2017.

[61] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: The ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 3–16.

[62] L. Luu, D. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, in: The 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 254–269.

[63] L. Luu, D. Chu, H. Olickel, P. Saxena, A. Hobor, Oyente: An analysis tool for smart contracts (2016). URL https://www.comp.nus.edu.sg/~loiluu/oyente.html

[64] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: IEEE Symposium on Security and Privacy, 2016, pp. 839–858.

[65] F. Zhang, E. Cecchetti, K. Croman, A. Juels, E. Shi, Town crier: An authenticated data feed for smart contracts, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 270–282.

[66] F. Zhang, E. Cecchetti, K. Croman, A. Juels, E. Shi, Town crier (2017). URL http://www.town-crier.org/

*Biographies (Text)

Xiaoqi Li — received his Bachelor's degree from Central South University and his Master's degree from Chinese Academy of Sciences. Now he is a Ph.D student in The Hong Kong Polytechnic University. His research focuses on system security and mobile security.

Peng Jiang — is currently a Ph.D student in Beijing University of Posts and Telecommunications. She is also a research assistant in The Hong Kong Polytechnic University. Her research interests include cryptography and information security.

Ting Chen — is an associate professor in the Cybersecurity Research Center, University of Electronic Science and Technology of China. His research domain includes software security, system security, and mobile security.

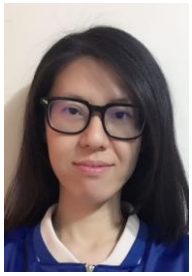
Xiapu Luo — is a research assistant professor in the Department of Computing, The Hong Kong Polytechnic University. His research focuses on mobile security, network security, and Internet measurement.

Qiaoyan Wen — is a professor in the Department of State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include coding theory, cryptography, information security, Internet security, and applied mathematics.

***Biographies (Photograph)**

Xiaoqi Li

Peng Jiang



Ting Chen

Xiapu Luo

Qiaoyan Wen