Essentials of Blockchain Technology

Contents

List of Figures

List of Tables

CHAPTER 1

Towards Preserving Privacy and Security in Blockchain

Mohammad Mustafa Helal

The University of Auckland, New Zealand

Muhammad Rizwan Asghar

The University of Auckland, New Zealand

CONTENTS

BLOCKCHAIN is at present one of the most disruptive technologies that have the potential to radically change today's business models. Blockchain is a decentralised database distributed across several systems. One of the key aspects of the blockchain is it does not require any dependence on a central trusted authority. Besides, no entity can tamper with data stored in a blockchain without the agreement among the majority, if not all, of the participating nodes. Blockchain is also used for smart contracts. A smart contract is a self-executed contract used to automate the verification process, execute a transaction, or exchange anything of value as per a predefined set of rules and conditions. Smart contracts do not rely on a central trusted authority.

Unfortunately, the protection of private information in the blockchain framework is still an open challenge. On the one hand, building applications on top of blockchain is growing, and expected to be used across different sectors, such as finance, government, and healthcare domains. On the other hand, protecting sensitive information is becoming very imperative as revealing such information could lead to revealing confidential business information or privacy loss. Moreover, storing smart contracts on blockchain nodes can be at risk of Man-at-the-End (MatE) attacks because the implementation of a smart contract is accessible to the curious blockchain nodes.

Bitcoin is the first blockchain application that uses the anonymity princi-

ple to tackle privacy issues. Many applications were developed afterwards for ensuring privacy in blockchain using different techniques, such as Zerocash, Coinparty, Mixcoin, and Monero. However, none of the available solutions addresses the privacy in smart contracts.

In this chapter, we aim at presenting a privacy-preserving model for ensuring the privacy in blockchain. Our proposed approach is based on White-Box Cryptography (WBC) to ensure the privacy in smart contracts. We propose to transform the smart contract into an obfuscated smart contract, shipped to the blockchain node along with the private assets hidden within the contract implementation. In this way, we introduce a system that can protect sensitive data. First, the new system is resistant to the most serious attacks including MatE and the white-box attacks, which enable the attacker to gain full control of the execution environment. Furthermore, storing sensitive data in an encrypted form within the obfuscated smart contract prevents information leakage.

1.1 INTRODUCTION

Blockchain is currently one of the most emerging technologies that have a great potential to significantly impact industry and business models. Blockchain technology is expected to be used by different sectors, such as finance, government, and healthcare domains. Blockchain is basically a peer-to-peer cryptographic-based mechanism where each peer holds a digital database known as a ledger in some applications. Blockchain transactions are stored chronologically with timestamps in blocks. Each block is chained with the previous block. Once the blocks are created in the blockchain, the transactions cannot be tampered or removed. This provides a tamper-proof data storage that makes it computationally impossible to reverse the transactions.

A smart contract is used in the blockchain framework in order to execute some actions when certain predefined conditions in the contract are met. A smart contract is a piece of programme code stored in blockchain network. A blockchain network includes all the participants' systems, and since a smart contract is managed by those systems, it is important to find a way to hide its implementation from any observer who can have access to a system in the blockchain network.

Problem Statement. The decentralised nature of blockchain can ensure availability; however, it also raises privacy concerns. Bitcoin is the first application built on top of the blockchain where the transactions are stored in plaintext by Bitcoin nodes. In terms of privacy, Bitcoin uses anonymous public addresses in order to hide the real identity involved in the transaction. However, the transaction itself is made publicly available to all the participants in the Bitcoin network so that they can do the verification. Unfortunately, linking these transactions could reveal real identities.

Some applications deal with personal information, such as medical data. In

general, protecting sensitive information is quite important as revealing such information could lead to serious consequences, such as privacy loss. Therefore, using traditional blockchain is not an option due to potential privacy issues. A smart contract has a vital role in such applications, as it is executed and running automatically on a blockchain node based on predefined rules and instructions. However, storing smart contracts on blockchain nodes at the peer's level can be at risk of a Man-at-the-End (MatE) attack because the implementation of a smart contract is accessible to the curious blockchain nodes. To address this issue, a naive approach could be to encrypt before storing the contract in the blockchain network. This simple approach introduces some new challenges when it comes to storing and executing the contract. There are several other problems, such as losing cryptographic keys could cost the users their personal information, or even worst, stolen keys could be misused. For data sharing, public keys could be used, which are typically managed by a Certificate Authority (CA). Nevertheless, public keys also come with its issues such as the single point of failure, where the Public Key Infrastructure (PKI) is highly dependent on the CA. Moreover, ineffective revocation mechanisms in the current PKI open doors for Man-in-the-Middle (MitM) attacks.

There are different techniques implemented by different blockchain applications all of which aim to improve the privacy in a blockchain network. For example, Bitcoin is one of the first applications in blockchain [30] using the anonymity idea to hide the real identity of the sender and the receiver. This is achieved by letting a Bitcoin user generate a new anonymous public address on each Bitcoin transaction. Another application [34] creates a separate anonymous currency called Zerocoin [28] on top of a non-anonymous currency referred to as basecoin (let us say Bitcoin). Users then can start to deal with the new anonymous currency. It also uses the Zero-Knowledge Proof (ZKP) concept in the verification process. Coinparty [40] enables users to transfer funds that are controlled by multiple mixing peers not only one peer, and this is achieved by using the multi-signature technique. Moreover, some applications combined more than one methodology to improve the privacy such as Monero [31], which uses ring signature, ring confidential transaction, kovri, and stealth addresses to obfuscate the transactions details.

Solution Statement. We propose a preserving-privacy framework for blockchain technology. Our solution is to use WBC to obfuscate the implementation of a digital smart contract. Furthermore, we hide the most valuable asset, which is the private key within the smart contract itself. In this way, blockchain nodes will be able to process the smart contract without learning sensitive information. Moreover, if a MatE attacker gets full access to the smart contract implementation, she will not be able to recover its implementation since the smart contract is obfuscated.

To the best of our knowledge, we are the first to leverage WBC to secure private data in a blockchain system. We aim at proposing a privacy-preserving framework for blockchain using the concept of WBC techniques to ensure

the privacy of smart contracts, where our solution is inspired by [12], which discusses obfuscating smart contracts in blockchain.

Chapter Organisation. The rest of this chapter is organised as follows. Section 1.2 briefly discusses an overview of the privacy concept in blockchain, and then we shed lights on different types of attacks on blockchain applications. In Section 1.3, we summerise general solutions that use the blockchain framework, then review certain solutions for improving privacy in blockchain, in addition to existing frameworks including Hyperledger and Ethereum. In Section 1.4, we describe our contributions, and our proposed framework to overcome the privacy issue in blockchain. In Section 1.5, we conclude the chapter and discuss the blockchain challenges and future directions.

1.2 OVERVIEW AND ATTACKS ON BLOCKCHAIN

This section consists of two sub sections. In Section 1.2.1, we provide an overview of different concepts of blockchain privacy and explain different types of blockchain. In Section 1.2.2, we discuss various attacks on systems built on top of the blockchain framework.

1.2.1 Overview of Privacy in Blockchain

In this sub section, we briefly discuss several types of blockchain and illustrate different methods and concepts used in current applications to preserve privacy.

1.2.1.1 Blockchain Types

There are multiple types of blockchain: public, private, and permission-based blockchains. Public blockchain means that everyone can join and contribute to the network. All the transaction data is recorded in a shared ledger. Bitcoin and Ethereum are examples of this type of blockchain. Public blockchain comes with few disadvantages, the main disadvantage that we focus in this chapter is it does not address privacy issues. The second type is called private blockchain; it allows only selected entry of verified participants. The main difference between public and private blockchain is that the private blockchain controls who is allowed to join and who can be part of the network. The owner has the right to override or amend the necessary entries as required. Finally, the third type is permission-based blockchain, which allows a mixture of both public and private blockchains with a customisation of features. This permission-based blockchain is built to grant special permissions to each participant on specific functions, for example, read, write, and access operations.

1.2.1.2 Anonymity

Anonymity is the idea of performing an action without revealing who has done the action. To address the privacy issue, *Bitcoin* [30] uses the anonymity concept by which the sender commits a transaction with a new and anonymous public address, which is not used in previous transactions by the user. This way, it becomes harder for an attacker to link an anonymous address to a given user. However, Meiklejohn *et al.* [27] were successful in identifying addresses belonging to online wallets, merchants, and other service providers by interacting with them and learning at least one address associated with such entities.

1.2.1.3 Mixing Protocol

In mixing protocols, the main idea is to build anonymity set for a specific transaction. For example, in a cryptocurrency transaction, a set of cryptocurrency holders can create a series of transactions, hence, making each participant anonymous within this set. This process may be repeated between different users to increase the anonymity set. CoinJoin [10] and CoinShuffle [32] have implemented this kind of protocol in their cryptocurrencies' models.

1.2.1.4 Altcoins

Altcoins are using a base currency, such as Bitcoin, to derive a new anonymous currency. Transactions are made through the new derived currency instead of the base currency to anonymise the transactions. Zerocoin and Zerocash [34] are examples of this type of currencies. Users can do a transaction in the base currency. However, users can cycle the base currency into and out of the anonymous derived currency to make the transaction anonymously.

1.2.2 Attacks on Blockchain

In this sub section, we describe major attacks on blockchain and provide a brief overview of each attack. This motivates us to explore our options towards a privacy-preserving framework.

1.2.2.1 Double-Spending Attack

In general terms, if a single transaction is executed twice in a system using the same asset, then it is considered a double-spending attack. For example, in Bitocin, an attacker uses the same Bitcoin in (at least) two different transactions triggered simultaneously with an aim to deceive the system to spend the same Bitcoin twice [10].

Finney attack [22] is a variation of double-spending attack where a dishonest miner broadcasts a pre-mined block for double-spending as soon as it receives product from a merchant.

1.2.2.2 Sybil Attack

Sybil attack [17] occurs when an entity tries to control multiple nodes in a network. At the same time, the network does not know that these nodes are controlled by the attacker. When the adversary maximises the control over a network, then there are chances that a victim might be connected to a node under the attacker's control.

Typically, decentralised applications are more subject to Sybil attacks than centralised applications. The existence of a trusted central authority eliminates Sybil attacks in centralised applications because the central authority is responsible for users registration and activities. In a decentralised application, Sybil attacks can be avoided in different ways. For example, Bitcoin application avoids Sybil attacks through the proof-of-work mechanism. Bitcoin acquires the miner to consume computational power to generate a Bitcoin block. Hence, the attacker is limited on how to control more nodes in the Bitcoin blockchain.

1.2.2.3 Denial-of-Service (DoS)

An attacker aims at flooding the system with the data more than the system can handle, thus resulting in unavailability of the system. By exploiting this opportunity, an attacker can perform malicious operations. In blockchain, an attacker may try to send fake data to the nodes. For example, the Bitcoin Satoshi client version 7.0 [7] has built a system that would prevent such attacks. The signature verification process is one of the most computationally heavy processes run by the client that could lead to DoS attacks. Bitcoin Satoshi client version 7.0 introduced a signature-caching as a new feature to mitigate DoS attacks. Using this feature, developers create a cache allowing peers to store previously validated signatures and avoid redundant work. Furthermore, it does not allow transaction duplication to prevent unwanted overloading the system.

1.2.2.4 Eclipse Attack

The target of this attack is the peer-to-peer network [19]. The Eclipse attack grants the attacker a huge advantage to take several IP addresses and manipulate the connections from/to the victim's node. Thus, the attacker controls the information flow, isolates the victim in the network from its peers, and leads the victim to communicate with malicious nodes instead of legitimate peers in the blockchain network.

1.2.2.5 Identity Lost

Users can easily lose their private assets, even without potential theft. However, losing a private key would compromise the valuable users' data in any system including blockchain applications. For example, in a cryptocurrency

application, a user owns wallets to manage her assets. One possible way to secure the access to the wallet is through a user-chosen password. If the user loses the password, then the entire assets that a user owns would vanish.

1.2.2.6 Identity Theft

Since the attacker knows that guessing any user's keys is practically a very complicated process and time-consuming, she may shift her focus towards stealing them instead of guessing or cracking. The attacker can increase the chances of getting the keys by attacking the weakest point in the system, which could be the users' mobile devices or their personal computers.

1.2.2.7 System Hacking

One of the key advantages of blockchain technology is it is hard to revert, amend, or alter the stored data in the blockchain network. Particularly, the attacker must have control of more than half of the nodes in order to manipulate data, and this is quite hard to achieve, if not impossible. However, programming codes, scripts, and systems that are used to implement the blockchain can be more vulnerable. For example, in 2014, some outdated codes gave attackers the ability to double spend Bitcoin transactions worth of 700 million dollars [33]. A similar incident happened in 2016, where the attacker exploited a code vulnerability and was able to steal 50 million of Ethereum [39].

1.3 LITERATURE REVIEW AND EXISTING FRAMEWORKS

In this section, we mention general solutions that use blockchain to run their businesses. Then, we focus on certain solutions designed to ensure the privacy in the blockchain network. Last but not least, we explain some of blockchain frameworks including Hyperledger and Ethereum.

1.3.1 General Solutions

We summerise general solutions that use the blockchain framework. These solutions use blockchain as part of their business model. We can see different sectors that blockchain can be utilised.

1.3.1.1 Central Bank Digital Currency

Sun *et al.* [36] propose a model for central bank digital currency called MBDC, which is based on the permissioned-based blockchain technology. MBDC utilises the multi-blockchain to fulfil the bank's business prerequisite. The permissioned-based blockchain is utilised to guarantee that each unit of the currency is made by the central bank. The central bank maintains a blockchain with all of the business banks and different agencies. Blockchain holds the to-

tal value of daily exchanges. The central bank can examine this enormous information that is stored in the blockchain. Business banks put their nodes in the blockchain with the goal that the banks could transfer the daily exchanges. Each bank is responsible for approving the client's identity when the client is enrolled, at the same time, the client's public and private keys are created by the client's data. Clients save their particular private key and the bank keeps a record of their public key.

1.3.1.2 Energy Trading through Multi-signatures

Aitzhan *et al.* [2] address the issue of providing transaction's security in decentralised smart grid energy trading. The proposed solution does not depend on any trusted third party. It uses multi-signatures and anonymous encrypted messaging to secure nodes communication. Multi-signature provides a way to form contracts without trusting any other party in the blockchain. Anonymous messaging streams provide two types of communication. First is sending a private peer-to-peer and second is message broadcasting. The system secures the participants through hiding the content, for example, masking identities by assigning unique strings of 36 alphanumeric characters.

1.3.1.3 Personal Data Protection

Zyskind *et al.* [41] introduce a convention that transforms a blockchain into an automated access control management that does not require trust in an external entity. This model aims to protect personal data on a blockchain. The framework consists of three elements. The first element is the users who are inspired by downloading and utilising applications. The second element is services to handle user's information and perform business operations. The third element is nodes that are substances depended on keeping up the blockchain and a disseminated private key-value data store. The blockchain acknowledges two new kinds of exchanges: Taccess, utilised to control access; and Tdata, for information storage and retrieval. A user introduces an application that uses the framework for safeguarding her security. As the user agrees to accept the first run through, a shared identity is produced and sent, alongside the associated permissions, to the blockchain in a Taccess exchange. Information gathered from the phone (*e.g.,* information from sensors, such as location) is encrypted using a shared encryption key and then is sent to the blockchain in a Tdata exchange. Tdata exchange sends the shared key to a key-value store, and holds it as a link to the data on the public ledger. The link is used by the users and services to retrieve the data.

1.3.1.4 MedRec

MedRec [5] is a prototype, which gives users the ability to access their electronic medical records across multiple providers. It addresses privacy con-

cerns in a blockchain where medical records are considered sensitive data that should not be publicly available. MedRec utilises smart contracts on an Ethereum blockchain. Contracts are used to store data pointers instead of the data itself. Data pointers are references to where the actual medical records are stored outside the blockchain. The blockchain stores the contracts data structures, references to the medical records, and permissions for ownership and viewership of the records. However, the raw data is stored separately in providers data storage.

MedRec incentivises medical researchers and healthcare stakeholders to be part of the blockchain network by giving them the ability to access the data in a single and common interface where patients can grant the permissions to share their data. Moreover, it provides immutable audit logs, data sharing authorisation, and custom Application Programming Interfaces (APIs), which are used, for instance, for posting to the Ethereum blockchain.

1.3.1.5 Model Chain

Model Chain [25] is a decentralised framework to preserve the privacy of the Protected Healthcare Information (PHI) in a private blockchain network. The system is cross-institutional healthcare to generate and exchange models instead of exchanging the private data of patients. The exchange happens among the connected healthcare sites. These models are partially trained by machine learning algorithms. Model Chain applies machine proof-of-information to decide the order of learning in the process of generating the model to be transferred. The site that contains fewer patients' data implies to have less accurate models; hence, contains more information to improve the model, the protocol will choose it as the next model to update the site. The process is repeated to update the model until a site cannot find any other site with higher error to update the model.

1.3.1.6 File Storage

Kopp *et al.* [23] designed a decentralised file storage system. It addresses the problem of a privacy-preserving payment mechanism based on ring signatures and one-time addresses. Instead of simply referencing the recipient by its public key, the sender obtains a new temporary public key using both a random nonce and the recipient's public key. The derived one-time public key, called destination key, and the original long-term public key of the recipient are unlinkable without knowledge of the recipient's private key. Ring signatures are used to prove membership in a group without explicitly revealing the identity. The signer needs its private key, as well as the set of public keys of the other members in the group to create a ring signature. The user can store their files in a storage provider by creating a contract. Storage providers publish the proof of retrieving the file using the ring signature to prove their compliance with storage contracts.

1.3.1.7 CryptoNote

CryptoNote [37] provides mainly untraceable transaction. CryptoNote scheme is based on a cryptography primitive called a group signature. It implements the ring signature technology, which allows the user to sign a message on behalf of a group. The signature is used to prove that the transaction is created by someone from the group such that all the signers are indistinguishable from each other. This protocol has better performance but weaker anonymity compared to Zerocoin or Zerocash [10].

CryptoNote solution enables a user to distribute a single address and receive unlinkable transactions. By default, the destination of each CryptoNote output is a public key, derived from the recipient's address and sender's random data. The main advantage over Bitcoin is that every destination key is unique by default. Thus, there is no external party can link two addresses together. This is based on the assumption that a sender does not use the same random data for the transactions delivered to the same recipient.

CryptoNote uses Diffie-Hellman exchange method to obtain a shared secret from the user's data and half of the recipient's address. The user then computes a one-time destination key, using the shared secret and the second half of the address. Two different keys are required from the recipient for these two steps, so a standard CryptoNote address length is nearly double as of Bitcoin wallet address. Nevertheless, the receiver conducts a Diffie-Hellman exchange to recover the corresponding secret key.

1.3.1.8 HAWK

Kosba *et al.* [24] developed a programming framework called HAWK. The framework is used for building a decentralised smart contract system. HAWK is intended to compile the program - with no implementation of cryptography - into an efficient cryptography protocol. HAWK is built on top of ZKP. The main idea of ZKP is to prove statements about a particular value without exchanging any information about that value between the prover and the verifier. HAWK protocol consists of users, a manager, and the blockchain program. Users must generate ZKP parameters and store them in the blockchain in three phases. The first phase is the freeze phase in which the data is stored in the contract. The second phase is the compute phase in which users send encrypted data with the public key of the manager. There is a finalise phase in which the manager decrypts the data with their private key, runs the functions, and creates the encrypted output, which is sent to the parties based on the previously agreed policy.

HAWK provided a sealed auction example to illustrate how HAWK can be implemented. In the sealed auction, the highest bidder wins; besides, the second highest price is rewarded as well in order to incentivise a truthful auction. Most important is that bidders submit their bids without knowing the bid of others. Hawk can compile such programs into two portions. First is the private portion that determines the winner and the price. Second is the

public portion, which relies on public deposits to protect users from a quitting manager. Hawk program declares three timeouts where T1 < T2 < T3. T1 is the time of collecting the bids, no more bid can be submitted after T1. T2 is the time when all users must open their bids to the manager, if a user fails to open their bid then the bid would be dropped out the auction. T3 is to control if the manager aborts, users can reclaim their private bids.

1.3.1.9 Zerocash

Ben-Sasson *et al.* [34] constructed a decentralised cryptocurrency protocol called Zerocash. It aims not to revealing any transaction's information such as the origin address, the destination address, and the amount. Zerocash creates a separate anonymous currency called Zerocoin on top of a non-anonymous currency known as basecoin. Users then start dealing with the new anonymous currency. Zerocash's functionality involves mint transactions and pour transactions. A mint transaction is the process of transforming the basecoins into Zerocoins. It includes a hash value of a unique serial number, coin's value, and the owner's address. A pour transaction gives the ability to the user to make a private payment through a ZKP. Pour transactions consist of up to two input coins, and up to two output coins. It uses ZKP to prove three things. First, a user has the two input coins. Second, the input coins exist in a previous mint transaction. Third, the value of the input coins is equal to the value of the output coins.

1.3.2 Solutions for Improving Privacy

In this section, we review existing solutions that address the privacy issue in blockchain applications. For each solution, we describe distinct features, advantages, and disadvantages. Recall anonymity concept to address the privacy, each solution brings in new features towards a preserving-privacy framework.

1.3.2.1 MixCoin

MixCoin [11] provides anonymity to Bitcoin transactions by allowing users to send their transactions to third party mix peers and receive back the same amount of the transaction submitted by other users. In this case, mixing is done with the help of a trusted third party mixing server called the mix. Each user sends a new encrypted address and transfers the funds to the mix. Afterwards, the mix decrypts the new addresses, randomly shuffles them, and sends the funds back to each participant. Moreover, MixCoin provides an accountability mechanism to expose any theft. The mix entity issues signed warranties to participants to state that if a user sends me a certain amount of coins by a specific time T1, then I will send the same amount to the user later by T2. In this way, the user can send funds to the mix with confidence that she can publish this warrant to degrade the mix's reputation if misbehaves.

This provides anonymity across external participants. Users outside the mix cannot learn about links between users in the mix. However, the primary drawback of this model is that the participants deal with a third party and need to trust the mix. In this scenario, the mix can learn which output address belongs to which input address. Therefore, the privacy is based on the assumption of a trusted third party, which can lead to de-anonymisation or exposing user's identities.

1.3.2.2 CoinJoin

CoinJoin [10] addresses the main drawback of MixCoin, where the mix can learn and link users' input and output. Coinjoin provides anonymity using multi-signature transactions. Multi-signature requires more than one party to be involved in the transaction. In order to let the participants mix their coins, they generate one single mixed transaction. The transaction with multiple inputs is considered valid only if it has been signed with all keys related to the input addresses. Hence, each user verifies the generated mix and refuses to sign the transaction in order to stop or proceed with the exchange. CoinJoin also provides external unlinkability; to this end, a set of users contributes to each transaction such that no external party can determine which input corresponds to which user. In this way, CoinJoin hides the ownership of Bitcoins by joining them with others in a single mixed transaction.

One of the possible disadvantages of CoinJoin is that one of the involved parties can learn how to link transactions between inputs and outputs.

1.3.2.3 CoinShuffle

CoinShuffle [32] is a decentralised protocol for coordinating CoinJoin transactions using a mixing protocol. Unlike CoinJoin, CoinShuffle provides anonymity even among the involved participants. It preserves the privacy of the transaction by allowing the users to mix their coins with other interested users in the network. CoinShuffle prevents any of the involved parties to link between inputs and outputs in the transaction. The recipient addresses are not known by the senders.

There are some advantages of CoinShuffle. One of the advantages is it requires only standard cryptography primitives such as signature and public key encryption. One more advantage is it is executed only by the Bitcoin users and does not require any trusted third party. Besides, CoinShuffle does not require any change in the Bitcoin protocol; it is fully compatible with the Bitcoin network. Last but not least, it does not charge any extra fees for additional mix transactions. Despite the aforementioned advantages, CoinShuffle increases an additional overhead for the rest of the Bitcoin network.

CoinShuffle protocol consists of three phases. First is the announcement phase, where each user generates a new ephemeral encryption-decryption key pair. Second is the shuffling phase, where each user creates a new Bitcoin

address as her output address in the mixed transaction. Then, the users shuffle the newly created output addresses using the encryption keys of all users. The last is the transaction verification phase, where each user can verify if the output address belongs to her is on the list. Each user signs the transaction and sends the signature. On receiving signatures from all users, each user is then able to create a fully-signed version of the mixed transaction. Then, the transaction is considered valid and pushed to the Bitcoin network. In each phase, every user checks that all other users follow the protocol or not. If this validation fails, then the user can report this misbehaviour, refuses to sign the transaction, and prevents the funds from being stolen.

1.3.2.4 Monero

Monero [3, 1] is a version of CryptoNote. It hides the sender, amount, transaction, and receiver using ring signatures, Ring Confidential Transactions (RingCT), kovri, and stealth addresses, respectively. Monero provides two features unlinkability and untraceability. Unlinkability means that an inability to find a relation between two transactions sent to the same user. Untraceability means that no one can identify where the transaction is originated from.

Unlike Bitcoin, the funds are not associated with the public address. When users send funds, they actually send funds to a random newly created one-time destination address. Hence, neither public records of the sender nor the receiver will appear in a public record. Instead, Monero uses a stealth address concept to hide the recipient address. To generate a stealth address, a Monero user is associated with two key pairs. One is a secret key pair (secret viewing key as skv and secret spending key as sks) known only by the user and a second public key pair is publicly shared (public viewing key as pkv and public spending key as pks). A stealth address is a new address derived from a one-time public key generated by the sender on behalf of their intended receiver. Hence, any transaction is always marked by a unique destination address. A sender generates a stealth address by two species of information: first is a random number used to generate a shared secret known only by both parties, while second is the public key pair of the receiver. The shared secret is generated through a Diffie-Hellman exchange. On the receiver end, Monero user actively scans the network to listen to every transaction, detects if the transaction is intended for their recipient's address, and then recovers the private key associated with this one-time public key in order to spend the funds.

Monero uses ring signatures to hide the sender address and provide the untraceability feature. A user receives several inputs linked together as a ring. Any input is linked to more than one transaction, thus making it hard to track the origin of a transaction. In this way, Monero hides where the transaction is coming from because it is linked with several random other transactions and signed using the ring signature. A digital signature contains more than one element. One element is a key image created from all these selected transac-

tions. The network scans for this key image. If the key image is found in the blockchain in any prior transaction, then the system will refuse the transaction to prevent the double spending issue.

Pedersen commitment is used to hide the actual amount that is being spent, so a user commits to spending a certain amount defined in this commitment. However, other users never know the exact amount to be spent. This Pedersen commitment is part of RingCT. It obfuscates the transaction amount by adding a random number. The commitment is then calculated using a certain formula for the set of inputs and outputs of the transaction, and then it is broadcast to the network. Hence, the actual amount is never published in the network in the plain.

Finally, Monero adopts kovri project to obfuscate the Internet traffic in a way such that any passive traffic monitoring can neither reveal the sender's geographical location nor the IP addresses. The Kovri project is based on the Invisible Internet Project (I2P) routing service. All the traffic is encrypted and then routed through the I2P nodes. Passive listeners can detect that one is using the I2P service. However, they cannot determine what are you using it for, nor the destinations set up by users.

Despite all of the aforementioned advantages, Monero transaction is significantly larger than other cryptocurrencies such as Bitcoin. For example, to construct a stealth address, the generated one-time destination address size is at least twice as Bitcoin recipient public address.

Miller *et al.* [29] mention the impact of few weaknesses of Monero. For example, many Monero transaction inputs contain deducible mixins and can be linked to prior transactions via chain-reaction analysis. However, Monero addressed this weakness by setting a minimum limit of mixins. This is one of the reasons why Monero had some implementation issues. However, the past discovered issues were addressed by Monero team, but there is no guarantee of having uncovered issues.

1.3.2.5 AEON

Anonymous Electronic On-line Coin (AEON) [31] is a fork of Monero. It is also a privacy-focused coin. AEON is meant to be simple enough to be used by anyone. AEON has started as an experiment but then found its vendors and now AEON is fully functional CryptoNote currency.

There are several advantages of AEON. First, AEON is considered to be mobile-friendly. It performs well on mobile devices as well as regular laptops and desktops. Second, AEON has a different proof-of-work known as CryptoNote-Lite, which is a lightweight version from CryptoNote protocol to speed up the verification process of the blockchain. Third, blockchain scalability, AEON allows the blockchain not to grow fast, it meant to be a good match in devices with limited storage. Last but not least, AEON gives users the ability to have a traceable transfer for non-sensitive transactions, it reduces the cost of operation and improves the performance of viewable transactions. De-

spite all of its advantages, having a lighter version of cryptography to run on any device can limit the usage of advanced cryptographic algorithms.

Table 1.1 provides a comparative analysis of existing solutions for improving privacy in blockchain. We present the main features besides disadvantages of each solution.

TABLE 1.1 Comparative analysis of existing solutions for improving privacy in blockchain.

Solution	**Main Features**	**Disadvantages**
Mixcoin (Section 1.3.2.1)	Increase anonymity and provide accountability	Require to trust a third party mixing server
Coinjoin (Section 1.3.2.2)	Utilising anonymity using multi-signature transactions	One of the involved parties can learn to map transactions between inputs and outputs
Coinshuffle (Section 1.3.2.3)	Expand anonymity even among the involved participants	Add an additional overhead for the rest of the Bitcoin network
Monero (Section 1.3.2.4)	Hide the sender, amount, transaction, and receiver	Transaction size is doubled and experienced some implementation issues
AEON (Section 1.3.2.5)	A lighter version of Monero, suits storage-constraint devices	Limiting advanced security features because of having a lighter version of cryptography

1.3.3 Existing Frameworks

We survey existing frameworks including Hyperledger [4] and Ethereum [38].

1.3.3.1 Hyperledger

Hyperledger is a set of open-source blockchain frameworks and platforms, created to improve blockchain technologies [18]. It is a global collaboration established as a project of the Linux Foundation in early 2016. Hyperledger has a wide list of well-known industry members. The list includes huge corporates such as Airbus, IT companies like IBM, Fujitsu, SAP, Huawei, Nokia, Intel, and Samsung, besides financial companies like American Express.

Hyperledger Fabric Hyperledger Fabric [20], [13] is an implementation of Hyperledger project. It was developed by IBM corporate. The primary consideration was to develop a blockchain framework that runs a real-world business scenario.

Hyperledger Fabric supports distributed ledger on permissioned networks for a wide range of industries. It is designed in a way to maximise the confidentiality, resilience, and flexibility of blockchain solutions.

Permissioned Membership. In a permissioned network, all participants must be known and can be identified by their unique identifiers. This kind of network is the best use in a business case where the business needs to fulfil certain data regulations. For example, financial and healthcare industries are subject to data protection laws.

Performance. The Hyperledger Fabric architecture separates transaction processing into three phases. The first phase is distributed logic processing and agreement known as chaincode. The second phase is transaction ordering. The last one is transaction validation and commitment phase. This separation assists to optimise the Hyperledger performance and reduce the number of levels of trusts and verification.

We describe the transaction flow in Hyeperledger Fabric. 1) An application submits a proposal to an endorsing peer. 2) Endorsement policy determines how many endorsing peers are needed to sign the proposal and then the endorsing peers execute chaincode such as a smart contract. 3) Then, the endorsing peers send the signed proposal back to the application. 4) The application then sends the transactions and signatures to the ordering service. 5) The ordering service generates a block of transactions and delivers them to committing peers. 6) Finally, the committing peer receives the blocks of transactions and validates that the endorsement policy was met or not. Then, a block is committed to the ledger. The performance is optimised as a result that only the signatures are sent around the network.

Data on a Need-to-know Basis. Hyperledger Fabric allows for data to go only to the parties that need to know. This is similar to the principle of least privilege, where each part must have access only to the data that it needs to know.

Protection of Digital Keys and Sensitive Data. Hyperledger Fabric supports the use of Hardware Security Module (HSM). This helps in safeguarding digital keys and managing them for strong authentication.

1.3.3.2 Ethereum

Ethereum [26] is an open-source blockchain platform. It allows developers to build decentralised applications and create many different services using the smart contracts concept. Ethereum's most innovative part is the Ethereum Virtual Machine (EVM). It allows running any program written in any language. EVM facilitates the creation of blockchain applications. For example, instead of creating a different blockchain for each new application, new applications can be created and managed on one platform. Contracts written in a smart contract are compiled into bytecode, each node in the blockchain executes this contract using its EVM. A smart contract gets executed when rules and conditions the developer initially programmed are met.

The Ethereum blockchain is a transaction-based state system. The system accepts a series of inputs and, based on those inputs, will be transitioned to a new state. In Ethereum's state machine, it starts with a blank state called a genesis state; this is before any transactions are on the blockchain. When transactions are executed, the state transits into a final state. A state has millions of transactions grouped into blocks. Each block is chained with its previous block to form the blockchain.

Ethereum applications have several advantages inherited from all the blockchain properties. First is immutability *i.e.,* data is considered immutable and no third party can make any changes to it. Second, there is zero downtime.

1.4 OUR PROPOSED FRAMEWORK

In this section, we first define the system and threat models, which explain the new entities in the proposed framework, the relation between them, and the adversary we consider. Then, we present our proposed approach for a privacy-preserving framework in blockchain. Finally, we discuss some benefits and limitations of our approach.

1.4.1 System Model

In this section, we define the entities of the proposed framework and their relations with each other. The system model consists of three entities: User Transaction, WBC-Smart-Contract, and Storage.

User Transaction. User transaction is the actual operation a user needs to perform on the blockchain. A transaction can be either send a request to the blockchain or get data request to receive the user's related data on the blockchain. Each user has a pair of cryptographically-linked keys: private and public keys. A user uses the public key to encrypt requests before submitting a transaction to the blockchain network.

WBC-Smart-Contract. WBC-Smart-Contract is a digital contract designed by a WBC implementation. The contract hides a private key within its implementation and accepts instructions encrypted with the corresponding public key. The contract stores the user data in storage in an encrypted form. When the contract needs to read the storage, it decrypts it internally and when the contract needs to write to storage, it encrypts the desired result before writing it back to the storage. The WBC-Smart-Contract code checks the signature on the transaction sent by the user to see if that user is entitled to read the data, and only if they are entitled to read, it returns the data. If the used signature is invalid, then the contract code returns an error, and the user will not be able to extract the requested information.

Storage. Storage is an internal element in the WBC-Smart-Contract, it is used to store the data in an encrypted form. It implements two interfaces, read and write, both of which are used by the contract. The contract encrypts the data before writing it to the storage and decrypts the data after reading it before the actual processing.

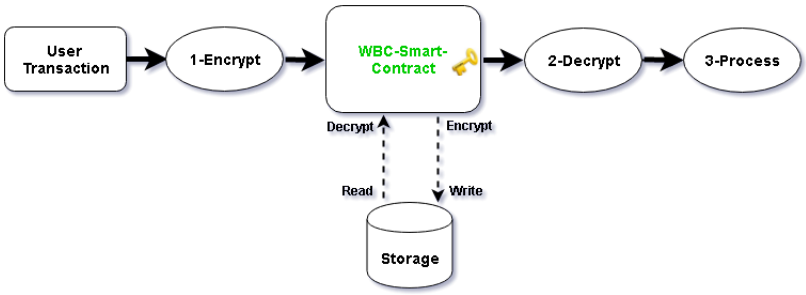

FIGURE 1.1 Flow diagram of a single user transaction in the proposed framework.

Figure 1.1 is a flow diagram to illustrate the entities and operations involved in the proposed framework. First, a user encrypts the request before sending it to the WBC-Smart-Contract. The WBC-Smart-Contract validates the user request to check if the given user is entitled to the given request or not. A request can be any user-related operation data, for example, getting user balance or medical records. While processing the user request, the WBC-

Smart-Contract decrypts data when reads it from the storage, processes it, and then encrypts the data back to the storage.

1.4.2 Threat Model

In the proposed framework, we define our threat model with the assumption that a Man-at-the-End (MatE) attacker has full access to the execution environment. Thinking of what will happen if the environment where the smart contract resides is untrusted and can be controlled by an attacker. In the context of this threat model, if an attacker managed to control a normal smart contract, they will be able to decompose the implementation to find a more compact representation that can be used in a way to control how the smart contract works effectively. Unlike the WBC-Smart-Contract, using a WBC implementation obfuscates the implementation of the smart control in a way that even if adversaries gain access to the environment, they will not be able to recover the smart contract in plaintext.

1.4.3 Proposed Approach

We propose a novel framework to address security and privacy issues in blockchain. The framework is based on WBC concept and smart contract. A smart contract is normally pushed into a blockchain node to be automatically executed as per pre-defined rules and conditions. Imagine that we have a MatE attacker who has limitless privileges and authorised access to the blockchain node. In order to minimise the loss that can be occurred, our approach is to obfuscate the contract before pushing it into a blockchain node. Moreover, we propose to embed cryptographic keys within the smart contract implementation through WBC techniques. In this way, we aim to exchange only encrypted data between a user and a blockchain node. This enables the contract to decrypt the transaction while processing it, then to encrypt the transaction to maintain it in the storage.

To obfuscate the smart contract, we propose using WBC mechanisms. The transformation process can be done either using a commercial tool, which converts a given program code into a white-box implementation such as [35], or a solution called SPNbox proposed by Bogdanov *et al.* [9]. We denote the output of the transformation process as a WBC-Smart-Contract. The role of WBC-Smart-Contract is explained in details in Section 1.4.1.

Currently, there are two dedicated designs handling WBC implementations. The first one is ASASA by Birykov *et al.* [6], which suffers from key extraction and decomposition attacks. The second one is SPACE by Bogdanove and Isobe [8], where SPACE reduces the risks of ASASA but introduces new performance overhead challenges. Whilst Bogdanov *et al.* [9] propose SPNbox Ciphers to overcome the challenges of the aforementioned solutions. SPNbox is designed with consideration of software efficiency and execution time. It

also relies on the black-box cipher security for resisting against key extraction attacks.

Figure 1.2 provides an overview of the proposed framework. Here, WBC-Smart-Contract holds the private key. Note that the exchange messages between the entities are encrypted messages.

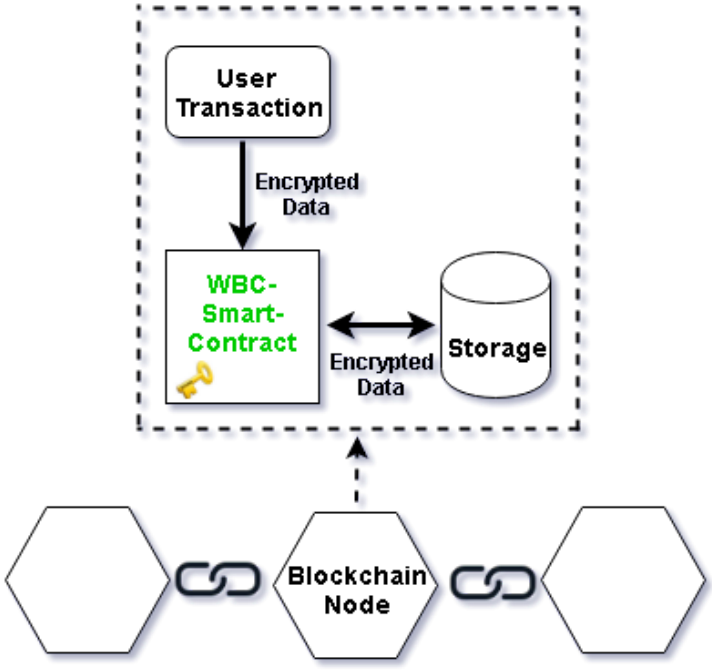

FIGURE 1.2 Overview of the proposed framework.

WBC is the concept of protecting the cryptographic keys, whilst the implementation is subject to the white-box attack model. The white-box attack model is considered the strongest attack model, based on the assumption that the attacker has full access to the source code and the environment, the attacker is able to see and manipulate the internal implementation steps and fully control the execution environment.

The first WBC implementation was introduced by Chow *et al.* in 2002 [15], which illustrated that it is possible to transform a given implementation to a white-box secure execution. They implemented a white-box Advanced Encryption Standard (AES). White-box secure execution hides the keys without exposing them in the implementation. Chow *et al.*'s WBC transformation is based on finding a representation of the algorithm as a network of lookups in randomised and key-dependent tables.

One of the primary WBC applications is Digital Right Management (DRM) systems. The end user subscribes to get a service such as Netflix

or any other on-demand videos. The digital content arrives at the user end in an encrypted form, then the software runs on the device decrypts it to stream the content to the user. The main goal in such a system is to prevent the user from being able to use her own stream for redistributing the digital content outside the DRM. WBC is used to hide the keys from the sight of the user or whoever can get access to the device.

Table 1.2 provides a comparative analysis of our approach and existing solutions mentioned earlier. We highlight the main two core key aspects of our proposed approach. The first aspect is we support smart contract privacy, and the other one is we have a model that is more resistant to white-box attacks.

TABLE 1.2 Comparative analysis of our approach and previous ones.

Approach	**Depends on Third Party**	**Cryptographic Technique**	**Supports Smart Contract**	**Resistant to White-Box Attack**
Mixcoin (Section 1.3.2.1)	Yes	Encrypts Addresses	No	No
Coinjoin (Section 1.3.2.2)	Yes	Multi-signature Transactions	No	No
Coinshuffle (Section 1.3.2.3)	No	Public Key Encryption	No	No
Monero (Section 1.3.2.4)	No	Ring Signature to hide the Sender	No	No
Our Approach	No	White-Box Cryptography	Yes	Yes

1.4.4 Discussion

There is no entire framework that is completely secure [21]. However, a framework is considered secure relatively to a security model. By defining the threat model in Section 1.4.2, and with the assumption that there is a MatE attacker. The attacker's goal can vary. For example, revealing some sensitive data on the blockchain. Another example is learning what rules and conditions are defined in the smart contract.

In the context of WBC, it is much more difficult to extract the keys from an obfuscated smart contract than revealing them from an un-obfuscated contract.

There are several advantages of using WBC within smart contract in a blockchain. The primary advantage is, the smart contract is given the ability to store encrypted instructions along with the keys. This means smart contracts can use these keys to decrypt the instructions, verifies whether the conditions are met or not, then encrypts the content back to the storage. In these operations, only the smart contract can get access to these keys when it needs to process a transaction. Another advantage of using WBC is if an attacker can get access to the blockchain node, WBC makes revealing the keys a very difficult process and time-consuming. This is with the consideration that the keys are hidden in the smart contract and the smart contract's implementation is obfuscated.

Despite the advantages of WBC, there are few disadvantages raise with WBC implementations in general. The main disadvantage is WBC acquire more resources such as memory, storage, and CPU processing [14]. Thus, WBC may not be ideal for resource-constrained platforms such as phones and tablets. Another disadvantage is that there is no known white-box solution available for asymmetric encryption. However, the known white-box solutions are currently available to the symmetric encryption.

1.5 BLOCKCHAIN CHALLENGES AND FUTURE DIRECTIONS

Blockchain technology has a great potential to grow. However, there are some fundamental challenges, which could raise serious privacy concerns. Currently, this proposed solution is in an early development phase. More work is needed to expand on the idea, also to implement the WBC-Smart-Contract to identify potential pitfalls and areas for optimisation.

We explained various attacks can occur in blockchain applications in Section 1.2.2. However, We did not address them directly in the proposed solution (Section 1.4). A potential future work we recommend is to address and tackle each attack in the context of the proposed framework.

Expressing any Program. One of the known issues of the obfuscation is there is no general obfuscation solution that can obfuscate any program without limitation. A potential future work we recommend in this area is to review

different obfuscation techniques and provides distinct features for each technique. This can assist in finding a solution that is more close to the ideal generic obfuscation solution. An ideal solution is that it can obfuscate any smart contract written in any language with no issues.

Lack of Tools. The tools that are used to develop a blockchain play a vital role in providing application security. For example, using improper tools [16] can lead to compromising application's security or efficiency. The use of proper and adequate tools is essential before developing any framework or application. This includes the Integrated Development Environment (IDE) used by the developers, building tools, deployment tools, testing, logging, debugging, and security auditing tools. All of which must be secure in a way to prevent any information leakage and to prevent any vulnerability exploitation.

Performance. One of the important aspects is to test the solution efficiency. WBC consumes more resources including memory, storage, and CPU. We analysed different options to do the transformation process to transform a smart contract into an obfuscated contract with performance consideration. However, performance measurements are needed after the implementation phase to identify any possible bottleneck that can be encountered at runtime.

Bibliography

[1] A low-level explanation of the mechanics of Monero vs bitcoin in plain English. `https://www.monero.how/how-does-monero-work-details-in-plain-english`, 2017. Last accessed: November 28, 2018.

[2] Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, 2016.

[3] Kurt M Alonso. Zero to Monero.

[4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[5] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *Open and Big Data (OBD), International Conference on*, pages 25–30. IEEE, 2016.

[6] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 63–84. Springer, 2014.

[7] BitcoinCore. On-chain scaling - a review of historical performance optimization made to bitcoin's reference software.

[8] Andrey Bogdanov and Takanori Isobe. White-box cryptography revisited: Space-hard ciphers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1058–1069. ACM, 2015.

[9] Andrey Bogdanov, Takanori Isobe, and Elmar Tischhauser. Towards practical whitebox cryptography: optimizing efficiency and space hardness. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 126–158. Springer, 2016.

[10] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 104–121. IEEE, 2015.

[11] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.

[12] Vitalik Buterin. Privacy on the blockchain. `https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/`, 2016. [Online; accessed 15-Jan-2016].

[13] Christian Cachin. Architecture of the Hyperledger Blockchain Fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, volume 310, 2016.

[14] Terugu Chalapathi. How white-box cryptography is gradually eliminating the hardware security dependency. `https://medium.com/engineering-ezetap/how-the-white-box-cryptography-gradually-eliminating-the-hardware-security-dependency-40622d516e02`, 2017. [Online; accessed 03-Nov-2017].

[15] Stanley Chow, Philip Eisen, Harold Johnson, and Paul C Van Oorschot. White-box cryptography and an AES implementation. In *International Workshop on Selected Areas in Cryptography*, pages 250–270. Springer, 2002.

[16] Ardit Dika. Ethereum Smart Contracts: Security vulnerabilities and security tools. Master's thesis, NTNU, 2017.

[17] John R Douceur. The Sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[18] The Linux foundation projects. About Hyperledger. `https://www.hyperledger.org/about`, 2018. Last accessed: November 28, 2018.

[19] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium*, pages 129–144, 2015.

[20] IBM. Top 6 technical advantages of Hyperledger Fabric for blockchain networks. `https://www.ibm.com/developerworks/cloud/library/cl-top-technical-advantages-of-hyperledger-fabric-for-blockchain-networks/index.html`, 2018. Last accessed: November 28, 2018.

[21] Marc Joye. On white-box cryptography. *Security of Information and Networks*, pages 7–12, 2008.

[22] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 279–296, 2016.

[23] Henning Kopp, David Mdinger, Franz Hauck, Frank Kargl, and Christoph Bsch. Design of a privacy-preserving decentralized file storage with financial incentives. In *Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on*, pages 14–22. IEEE, 2017.

[24] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.

[25] Tsung-Ting Kuo and Lucila Ohno-Machado. Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks. *arXiv preprint arXiv:1802.01746*, 2018.

[26] Iuon-Chang Lin and Tzu-Chun Liao. A survey of blockchain security issues and challenges. *IJ Network Security*, 19(5):653–659, 2017.

[27] Sarah Meiklejohn and Claudio Orlandi. Privacy-enhancing overlays in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 127–141. Springer, 2015.

[28] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[29] Andrew Miller, Malte Mser, Kevin Lee, and Arvind Narayanan. An empirical analysis of linkability in the Monero blockchain. *arXiv preprint*, 1704, 2017.

[30] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[31] The Monero Project. AEON isn't just a currency. it's a lifestyle.

[32] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. CoinShuffle: Practical decentralized coin mixing for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014.

[33] Muhammad Saad, My T Thai, and Aziz Mohaisen. Poster: Deterring DDoS attacks on blockchain-based cryptocurrencies through mempool optimization. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 809–811. ACM, 2018.

[34] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy (SP)*, pages 459–474. IEEE, 2014.

[35] Inside Secure. Whitebox: Build, control and trust your own software crypto-security. `https://www.insidesecure.com/Products/Application-Protection/Software-Protection/WhiteBox#field-description`, 2018. Last accessed: November 28, 2018.

[36] He Sun, Hongliang Mao, Xiaomin Bai, Zhidong Chen, Kai Hu, and Wei Yu. Multi-blockchain model for central bank digital currency. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2017 18th International Conference on*, pages 360–367. IEEE, 2017.

[37] Nicolas Van Saberhagen. CryptoNote v 2.0, 2013.

[38] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[39] Jennifer J Xu. Are blockchains immune to all malicious attacks? *Financial Innovation*, 2(1):25, 2016.

[40] Jan Henrik Ziegeldorf, Fred Grossmann, Martin Henze, Nicolas Inden, and Klaus Wehrle. Coinparty: Secure multi-party mixing of bitcoins. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 75–86. ACM, 2015.

[41] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.