Blockchain Security: A Survey of Techniques and Research Directions

Jiewu Leng, Man Zhou, J. Leon Zhao, Yongfeng Huang, and Yiyang Bian

Abstract—Blockchain, an emerging paradigm of secure and shareable computing, is a systematic integration of 1) chain structure for data verification and storage, 2) distributed consensus algorithms for generating and updating data, 3) cryptographic techniques for guaranteeing data transmission and access security, and 4) automated smart contracts for data programming and operations. However, the progress and promotion of Blockchain have been seriously impeded by various security issues in blockchain-based applications. Furthermore, previous research on blockchain security has been mostly technical, overlooking considerable business, organizational, and operational issues. To address this research gap from the perspective of information systems, we review blockchain security research in three levels, namely, the process level, the data level, and the infrastructure level, which we refer to as the PDI model of blockchain security. In this survey, we examine the state of blockchain security in the literature. Based on the insights obtained from this initial analysis, we then suggest future directions of research in blockchain security, shedding light on urgent business and industrial concerns in related computing disciplines.

Index Terms—Blockchain security, process security, data security, consensus algorithm, smart contract

1 Introduction

Blockchain is a chain structure that links data blocks sequentially according to the chronological order and thereby ensures that this distributed ledger cannot be tampered with or forged cryptographically. With the rapid and widespread adoption of Bitcoin [1], [2], blockchain is hailed as a disruptive innovation in the computing paradigm [3]. Blockchain has become a new secure and shareable computing paradigm for supporting business innovation [4].

As shown in Fig. 1, blockchain integrates multiple techniques, such as cryptography, P2P network protocol, and consensus algorithm, to achieve more security system functions. Blockchain can be integrated with smart contracts, ensuring that the program can execute the preset logic through self-limitation and security encryption credibly and automatically [5]. The P2P network protocol allows blockchain to handle unpredictable patterns so that the business logic will not be outdated [6], [7]. However, security issues in blockchain continue to impose a significant challenge [8]. Blockchain systems have suffered many outside attacks around the Internet [9]. Blockchain security is the protection of transactions in a block against internal, malevolent, peripheral, and unintentional threats. This protection depends on detection, prevention, and appropriate response to threats from different levels using security policies and tools [10]. However, a wide range of current blockchain security research is technical in nature, suffering from limited considerations of organizational and operational issues [11].

There have been several blockchain security-related surveys conducted from different perspectives, highlighting some research gaps and future exploratory directions for academics and practitioners. Lin and Liao [12] reviewed several blockchain security issues, such as majority attack, data scale, and cost problems. However, the analysis is not comprehensive enough for guiding future research. Li *et al.* [13] conducted a study on security threats to blockchain and surveyed the corresponding real attacks by examining popular blockchain systems such as Ethereum. This survey neglects security issues when utilizing these systems to build business applications. Joshi *et al.* [10] surveyed consensus protocols from the data security and privacy protection aspect. However, there exist security issues from other aspects, such as smart contracts. Hossain [14] overviewed blockchain research from the perspective of digital business transformation with its future evolutions. However, it lacks a discussion on the security issues of blockchain. Park *et al.* [15] adopted the blockchain security solution to cloud computing. However, other application scenarios were left out of the survey scope. Fran *et al.* [16] presented a systematic literature review of blockchain-based applications across multiple domains. However, the study lacks a discussion on the blockchain

- *Jiewu Leng is with the State Key Laboratory of Precision Electronic Manufacturing Technology and Equipment, Guangdong University of Technology, Guangzhou, Guangdong 510006, China, and also with the Department of Information Systems, City University of Hong Kong, Hong Kong. E-mail: jwleng@gdut.edu.cn.*
- *Man Zhou is with the State Key Laboratory of Precision Electronic Manufacturing Technology and Equipment, Guangdong University of Technology, Guangzhou, Guangdong 510006, China. E-mail: theoggw@163.com.*
- *J. Leon Zhao is with the School of Management and Economics, Chinese University of Hong Kong, Shenzhen 518172, China. E-mail: leonzhao@cuhk.edu.cn.*
- *Yongfeng Huang is with the School of Computer Engineering, Jiangsu University of Technology, Changzhou, Jiangsu 213000, China. E-mail: cz_hyf@163.com.*
- *Yiyang Bian is with the Department of Information Management and Information Systems, Nanjing University, Nanjing, Jiangsu 210000, China, and also with the Department of Information Systems, City University of Hong Kong, Hong Kong. E-mail: bianyiyang@nju.edu.cn.*

Manuscript received 11 May 2020; revised 5 October 2020; accepted 13 November 2020. Date of publication 25 November 2020; date of current version 8 August 2022.
(Corresponding author: J. Leon Zhao.)
Digital Object Identifier no. 10.1109/TSC.2020.3038641

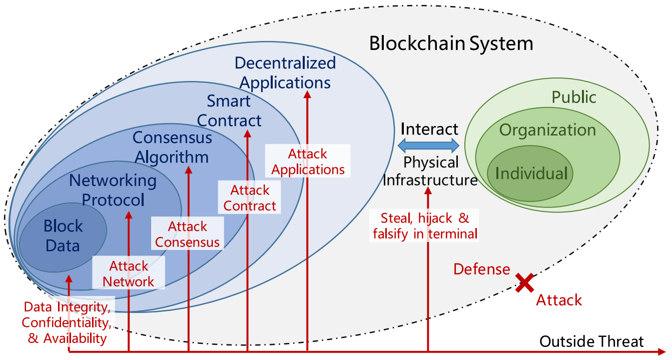


Fig. 1. Major elements and threats in blockchain security.

security issues. Moreover, many new blockchain security studies have been conducted in the last two years and therefore need to be included.

Blockchain security is a serious issue that must be considered as an element embedded in the whole lifecycle of system, from requirement analysis to coding and maintenance [17]. However, scholars from the information systems area seem to have limited awareness of the contributions achieved by the researchers of other disciplines. It is therefore necessary to study how and to what extent blockchain security issues have been addressed and then summarize relevant research methodologies in previous research. This study adopts the information systems perspective and presents a systematic survey of blockchain security from different layers shown in Fig. 1. The blockchain security is organized at the process level, the data level, and the infrastructure level, which we refer to as the PDI framework.

The rest of this survey is outlined as follows. Section II proposes a PDI framework for surveying blockchain security. The following three sections examine to what extent these security issues have been addressed. New directions for studying blockchain security are presented in section VI, followed by concluding remarks.

2 A Framework of Blockchain Security

There exist different architectures and models of system security, which can be used as a guideline to build a framework of blockchain security (see Table 1). Firstly, the first three architectures in Table 1 emphasize the implementation and organization security of a system. The Information Security Architecture (ISA) proposed by Tudor focuses on five essential components: 1) organization & infrastructure, 2) policies & procedures, 3) baselines & risk assessments, 4) awareness & training programs, and 5) compliance [18]. The Policy Framework for Interpreting Risk in E-Business Security (PFIRES) proposed by Rees *et al.* [19] addressed the policy rules to manage information among multiple organizations. Another security architecture proposed by Eloff *et al.* [20] identified the demands for an integrated architecture to achieve maximum security. Secondly, the next two architectures offered by two IT companies (i.e., IBM and Oracle) explain concepts in security. IBM [21] provided a view of security, including business, technology, service delivery, and fusion of domains tend to share common elements. Oracle [22] proposed a reference architecture for achieving security, namely, data security, fraud prevention, and compliance enablement.

TABLE 1
Typical Security Architectures, Frameworks, and Standards

Models	Author	Metrics
ISA	Tudor [18]	Guide the implementor to develop an effective security architecture.
PFIRES Life Cycle Model	Rees et al. [19]	Guidance to facilitate coordination between managers and engineers.
Security architecture	Eloff et al. [20]	An integrated architecture to achieving the maximum-security.
Security Reference Architecture	IBM [21]	A broad view of security, including business, technology, and service.
Security Conceptual View	Oracle [22]	Illustrate how security could be realized.
ISO/IEC 27000	ISO/IEC [23]	Summarize the best-practice of security management.
Critical Security Controls	CIS [24]	Help organizations focus their security resources and expertise.
Security Reference Architecture	Tripwire [25]	Automate security and IT operations.
Cybersecurity Framework	NIST [26]	A framework for enabling organization improving ability.

The last three references in Table 1 concentrate on the uniform information security standards summarized from practice. The ISO/IEC 27000 family of standards summarize a globally acknowledged best-practice of security management [23]. The Center for Internet Security (CIS) helps organizations focus their security resources and expertise on defending against cyber-attacks [24]. The Security Reference Architecture proposed by Tripwire integrates high-fidelity asset visibility and deep endpoint intelligence with business context, automate security, and IT operations [25]. The Cybersecurity Framework proposed by the National Institute of Standards & Technology (NIST) presents a security framework for enabling enterprises, improving their ability to defending against cyber-attacks [26]. In terms of the blockchain-based applications, the primary security problems include the generation and protection of private keys, vulnerabilities of the signature algorithm, the centralization of the consensus process, vulnerabilities of smart contracts, and vulnerabilities of decentralized application.

To integrate existing security architectures with blockchain-based applications regarding technological and organizational issues in a balanced way [27], this paper builds a Process-Data-Infrastructure model (PDI model) by mapping the security architectures proposed by IBM and Oracle to the five-level blockchain architecture. The PDI framework (shown in Fig. 2) includes three aspects: process level, data level, and infrastructure level. Data-level security is flanked by process and infrastructure. The blockchain security in the data level is composed of authentication, encryption, consensus algorithms, access control, and key management. The process level includes smart contracts, implementation security, operation standards, and fraud detection. The infrastructure level includes the terminal devices, network, and super-node server (if any). Security contributions can be mapped according to these three levels. The next three sections examine to what extent these issues have been studied.

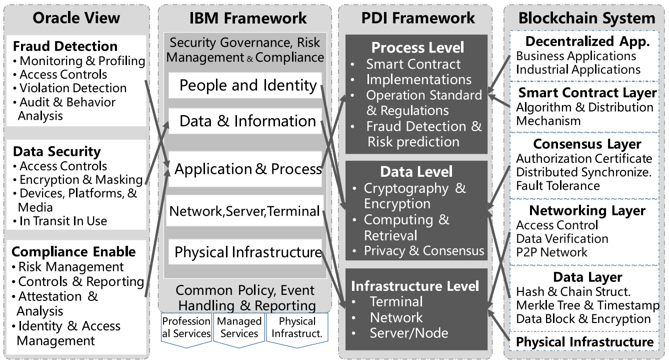


Fig. 2. A framework of blockchain security issues.

3 Process Security in Blockchain

A business process requires the practitioner to decide on and adhere to policies that participants can enact to keep their systems secure. Blockchain security on the process level is complicated due to multiple tasks and governance [28]. Four aspects of process security in blockchain are reviewed in this section (Fig. 3).

3.1 Security of Smart Contract

Smart contracts refer to scripts automatically executed on a distributed network consisting of mutually distrusting nodes without an external trusted third-party [29]. To complete the data transactions between the participants, smart contracts expose the transaction data to the risk of compromising sensitive information. Otherwise, difficulties of supervision arise [30]. Since smart contracts transfer value, both their correct execution and secure implementation against attacks/tampering are crucial [31]. Criminals can leverage Criminal Smart Contracts (CSC), a new critical cyber-weaponry, to produce 0-day vulnerability data transactions [32]. The smart contract is one of the most significant sources of security issues on the process level of blockchain. Besides, a smart contract (e.g., blockchain wallet, crowdfunding fund) can hardly be revised once having been issued. A recorded fault or malicious data transaction cannot be deleted from the blockchain application [33]. The method to roll back a recorded data transaction is to make a hard fork on the blockchain, and it calls for new consensus among the participated members and thereby damages the trustworthiness of the system [34]. Therefore, the security of the smart contract often determines the security of

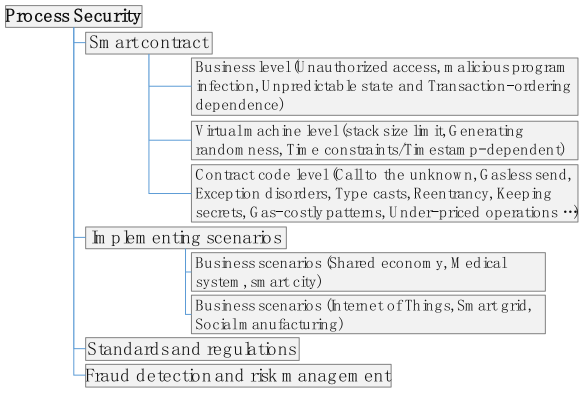
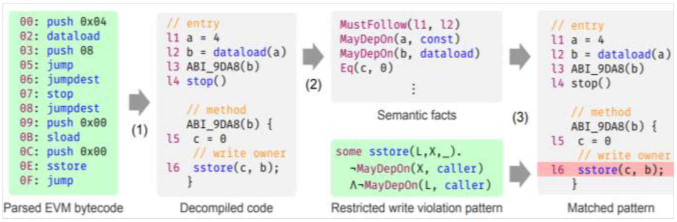

Process Security
Smart contract
Business level (Unauthorized access, malicious program infection, Unpredictable state and Transaction-ordering dependence)
Virtual machine level (stack size limit, Generating randomness, Time constraints/Timestamp-dependent)
Contract code level (Call to the unknown, Gasless send, Exception disorders, Type casts, Reentrancy, Keeping secrets, Gas-costly patterns, Under-priced operations …)
Implementing scenarios
Business scenarios (Shared economy, Medical system, smart city)
Business scenarios (Internet of Things, Smart grid, Social manufacturing)
Standards and regulations
Fraud detection and risk management

Fig. 3. Major aspects of the process security in the blockchain.


```
00: push 0x04
02: dataload
03: push 08
05: jump
06: jumpdest
07: stop
08: jumpdest
09: push 0x00
0B: sload
0C: push 0x00
0E: sstore
0F: jump
```

```
// entry
l1 a = 4
l2 b = dataload(a)
l3 ABI_9DA8(b)
l4 stop()

   // method
   ABI_9DA8(b) {
l5  c = 0
    // write owner
l6  sstore(c, b);
   }
```

```
MustFollow(l1, l2)
MayDepOn(a, const)
MayDepOn(b, dataload)
Eq(c, 0)
        ⋮
```

```
some sstore(L,X,_).
  ¬MayDepOn(X, caller)
  ∧¬MayDepOn(L, caller)
```

```
// entry
l1 a = 4
l2 b = dataload(a)
l3 ABI_9DA8(b)
l4 stop()

   // method
   ABI_9DA8(b) {
l5  c = 0
    // write owner
l6  sstore(c, b);
   }
```

Fig. 4. High-level flow illustrating how securify finds a security issue [39].

blockchain [35]. Many smart contracts are vulnerable to attacking [36].

The security issues of smart contracts occur in three levels, namely, business level, virtual machine level, and contract code level. Specifically, *business level* security issues include unauthorized access, malicious program infection, unpredictable state, and transaction-ordering dependence. The *virtual machine level* security issues include stack size limit, generating randomness, and time constraints. The *code level* security of smart contracts is a critical issue to blockchain applications [31]. The volume of transactions involved in smart contracts in the blockchain is enormous, and more practical scenarios are needed to test the system stability to find potential code vulnerabilities. Smart contracts could be applied in more complex situations, and the complexity and technical difficulty of the contract code may also increase accordingly. Typical code level security issues of contracts include *call to the unknown*, *gasless send*, and *deadlocked state* [33]. Reentrancy and unpredictable states are also common code vulnerabilities [37]. Failure to encode a correct state machine (e.g., neglecting to check the current state and omitting specific transitions) is the most frequently observed issue.

To verify the security of contracts, it is of considerable significance to capture their semantics and the security properties from the bytecode being executed [38]. Tsankov *et al.* [39] proposed a security analyzer named Securify, which automatically extracts precise semantic information of the security of smart contracts from the code. Fig. 4 illustrates how Securify finds a security problem. The input factors, namely, Environment Virtual Machine (EVM) bytecode and security patterns, are marked with the green color, the output (a violated instruction) is marked with the red color, and the gray color refers to intermediate analysis results. Securify works via 1) decompiling the bytecode into a single static assignment, 2) inferring semantics of the smart contract, and 3) matching the violated patterns of the restricted write property. Securify can check whether contract behaviors are safe or not concerning a given property via compliance and violation patterns. To inspect the logic security of smart contract code, some scholars have proposed the automated security process method [40] and the symbolic execution system [41] to identify security bugs.

In the stage of contract design, a semantic framework is needed for aiding the design of contracts and generating smart contracts via a set of design patterns automatically [33]. New security-related design patterns (e.g., Rate Limit and Balance Limit) could be designed to code smart contracts [35]. Also, more security properties (e.g., call integrity and atomicity) for smart contracts could be defined to model the semantics of bytecode for obtaining executable code [38]. However, complex semantics still impose

TABLE 2
An Overview of the Implementation Security of Typical Blockchain

Scenarios	Goals	Issues	Countermeasures	References
Internet of Things	Full autonomy capability on processing and exchanging data without human intervention	TPS, privacy violation, denial-of-service, network disruption	Robust identification and authentication of devices	[45], [46], [47], [48], [49], [50]
Shared economy	Enforce the agreement between demanders and suppliers of services without any trusted party	Leak privacy of involved parties	Zero-knowledge proof	[51], [52]
Electronic medical system	Deal with the security and privacy issue in the electronic medical system	Interconnect multi-organization	Sidechain	[53]
Smart city	Provide better services to its citizens while ensuring optimal utilization of resources	Digital disruption	Fraud detection & robust consensus	[54], [55]
Smart grid	Solve the trustworthiness issue of decentralized energy exchanges	Privacy protection of trading data	Zero-knowledge proof	[56], [68]
Social manufacturing	Endow intelligence to every entity on the manufacturing network	Bonding the physical and cyber world	Digital twin technology	[58], [67]

significant challenges to automate the coding of smart contracts in a blockchain securely.

3.2 Implementation Security of Blockchain

Apart from the well-known cryptocurrency and blockchain-based Fintech applications, there are broad sectors of blockchain implementations in the contexts of Internet of Things (IoT), shared economy, healthcare systems, smart city, smart grid, social manufacturing, and supply chain management [42] in social services [43]. Consequently, much risk exists when implementing blockchain. The integration of blockchain with existing systems may impose substantial challenges in real businesses. Implementation of high security requires rigorous testing of critical code. Original business models may not fit in blockchain-enabled business logic [44] as the realization of security is strongly correlated with deployment environments. Table 2 provides an overview of the implementation security of typical blockchain-based systems.

Internet of Things (IoT) is a critical enabling technology for the interactions of devices to exchange data for smart decision-making [45]. However, lacking security countermeasures makes the IoT network vulnerable to cyber threats and attacks [46], which thereby has an impact on the privacy protection of the involved stakeholders [47]. It is cumbersome to create a centralized authentication system because of the massive data scale and high costs of server maintenance costs for IoT [48]. The capabilities of blockchain, including tamper-resisting, transparency, auditability, and network resilience, can amend the architectural shortcomings of centralized IoT and industrial networks [49]. With heterogeneous devices ranging from sensors to servers, a dynamically adaptable multi-layer security architecture could be designed with intelligent standardization of the networked IoT devices [50]. The deployed protocols, together with conversion mechanisms, need to interoperate at different layers of the IoT network.

With the increase of the shared economy scenarios such as digital copyright management and asset heritage, few users are eager to adopt blockchain to enable decentralized applications due to security issues arising from unsuitable business models. In a conventional centralized operation architecture, security risk of the trusted center itself becomes the core problem in system security [51]. Trust-free systems are usually developed to support the shared economy by enabling complex interactions among participants who require high trust [52]. While blockchain-based applications have many desirable metrics, they may leak private information of involved participants due to its openness to the public network. Nevertheless, challenges remain in engineering trust-free systems in the shared economy because the intricacies of trust vary with the context of blockchain. Meanwhile, it requires different interfaces to build blockchain ecosystems.

Electronic medical systems are playing a critical role in improving the healthcare of people. Patient health conditions can now be monitored continuously, processed remotely, and transferred to a cloud medical data center. The growing volume of data managed by the medical system implies higher exposing of sensitive and private data. The privacy protection requirements in the transmission and storage process are of significant concern [53], which needs to satisfy specific demands on data availability and system resilience. The primary difficulty of adopting blockchain in the electronic medical system is how to build secure interconnections across multiple organizations and hospitals. Sidechain technologies may resolve this challenge.

Smart city is a vision of using new generations of information technology to manage cyber, physical, and social infrastructures to provide better services to its citizens while maintaining the optimal utilization of resources. Biswas *et al.* [54] presented a blockchain-enabled security model integrally engaging with resources to provide a secure environment for the smart city. Dorri *et al.* [55] showed a blockchain-enabled security system for privacy protection of the vehicular ecosystem. However, digital disruption imposes great challenges related to implementing security and information privacy. Challenges such as fraud detection and robust consensus mechanism may surface in the blockchain-based smart city system.

Smart grid is envisaged to offer sophisticated consumption monitoring and energy trading via bi-directional communication flow. Blockchain can help to resolve the trustworthiness issue of decentralized energy exchanges, enabling automated and auditable multi-party transactions based on contracts between decentralized energy providers and consumers. Smart contracts could be utilized to facilitate the monetization of energy transaction flows and financial transactions without the support of a trusted third-party, and thereby to reduce the operation costs and to increase the resilience of the energy integration system [56]. Internet of Energy is a potential practical networking approach for the smart grid. Huang *et al.* [57] proposed the lightning network and smart contract (LNSC) model to improve the security capability of trading in the authentication and scheduling among electric vehicles and distributed charging piles. However, the privacy protection of consumption trading data remains a critical issue.

The sustainability goal of the government calls for investigation on how blockchain can make manufacturing more sustainable. Social manufacturing is a vision of open-sourced product manufacturing processes. An increased requirement for prosumers leads to verifying the quality and authenticity of individualized products. Leng *et al.* [58] proposed an anti-counterfeiting system by integrating chemical signature with blockchain. Specific chemical signature data is physically bonded with the parts and then linked to a securitized blockchain, making transactions in the manufacturing network trustworthy. Smart contracts can be introduced for crowdsourcing of machining tasks in the collaborative manufacturing paradigm [59], [60], [61] and supporting the decentralized scheduling in the workshop level [62], [63]. Digital twin [64], [65], [66] is a key enabling technology of authenticating the subjects and ensuring security in synchronizing blockchain with the physical system.

Although blockchain promises to facilitate the sharing of information across industrial partners, it is of great importance for decision-makers to ascertain their specific situations for their particular business contexts [13]. The scalability of blockchain-based system is another practical issue in its implementation process [67]. Blockchain could be implemented only if it is applicable and offers better securities for achieving more business benefits.

3.3 Operation Standards and Regulations

Blockchain is immature in the areas of scalability, performance, and interoperability with other systems. In addition to technical challenges, enterprises face management challenges since blockchain must be assimilated within complex institutional, regulatory, social, economic, and physical systems [69]. The open-source blockchain platforms create anomalies in attaining a boarder sense of unified approaches with its own standards and code [15]. Standardization of terminology and technologies is critical for optimizing the interoperability of different models [70]. Governance is crucial for successfully implementing blockchain while protecting participants and enhancing the system resilience against cybersecurity attacks. Despite the self-governing nature of blockchain, the regulation of a decentralized system remains to be resolved in the actual implementation [71]. The lack of common standards and clear regulations severely limit the ability of blockchain to scale. The lack of standardization and regulation means that it is hard for practitioners to benefit from others' exploration and mistakes. Besides, when users incorporate blockchain into the business context, they need to identify which blockchain models fit their specific requirements. The standardization for evaluating the performance of security needs to be conducted previously (Table 3). For instance, Standards Australia is exploring global standards for blockchain technology with ISO. Meanwhile, the R3 Blockchain Consortium has started to develop industry standards for interbank applications. The General Data Protection Regulation (GDPR) from the European Union identifies privacy protection demands on the processors and controllers [72]. Standardization in blockchain includes basic data models, consensus algorithms, and smart contracts [70]. Standards play critical roles in certifying the interoperability among multiple blockchain implementations. The regulatory works promote future innovations of blockchain [73].

TABLE 3
Typical Standardization Work Related to Blockchain

Type	Group/Content
World Wide Web Consortium (W3C)	Credentials Community Group, Digital Verification Community Group, Blockchain Community Group, Verifiable Claims Working Group, Interledger Community Group, Web Ledger Protocol
ISO TC 307	Joint ISO/TC 307 - ISO/IEC JTC 1/SC 27
Standards Australia	Roadmap for Blockchain Standards
International Telecommunications Union	Focus Group on DLT (FG DLT)
IEEE	Blockchain and DLT
SWIFT	Blockchain and DLT
European Union	General Data Protection Regulation (GDPR)
China Electronics Standardization Institute	Reference architecture of blockchain

However, rules and regulations will affect more new security risk complexity rather than a technical issue. Guo and Liang [74] proposed a regulatory sandbox of industry implementation to give a more flexible space for innovations while imposing simplified access standards and procedures. A maturity model helps guide organizations to make decisions more systematically in the blockchain-implementation process [75]. A benchmarking system is also practical for evaluating the data processing performance of blockchains [76]. Particularly, operation standards and regulations are necessary for the redactable blockchain and controllable blockchain [77], or editable blockchain [78] to remove and rewrite inappropriate data. To create trust, it is necessary to establish standards to address concerns of security, resilience, privacy protection, and governance in blockchain implementations [79]. Enforcing restrictions on the automatic execution of transactions of blockchain directly could reduce the regulatory compliance and auditing costs [80]. However, any regulation of the technology might hamper its development until it has been adequately tested.

3.4 Fraud Detection and Risk Prediction

Fraud detection is a critical security aspect in the running process of information systems, such as the tax administration system [81]. Fraud detection usually relies on rules and schemes to recognize usual from suspicious and disruptive behavior. The rules and schemes may be summarized or learned from the data mining and analysis of behavior patterns. Scholars have utilized the blockchain to solve some fraud detection issues in information systems [82]. Meng *et al.* [83] reviewed the intersection of intrusion detection and blockchains. Blockchain not only improves the capability of fraud detection but also cut down illegitimate manipulation [84].

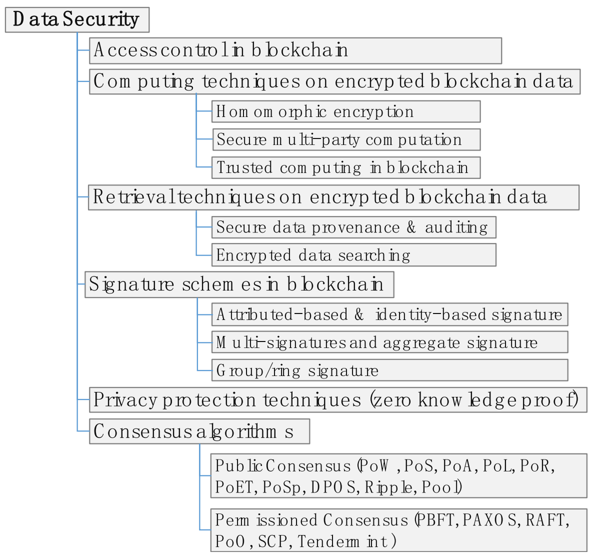

Data Security
Access control in blockchain
Computing techniques on encrypted blockchain data
Homomorphic encryption
Secure multi-party computation
Trusted computing in blockchain
Retrieval techniques on encrypted blockchain data
Secure data provenance & auditing
Encrypted data searching
Signature schemes in blockchain
Attributed-based & identity-based signature
Multi-signatures and aggregate signature
Group/ring signature
Privacy protection techniques (zero knowledge proof)
Consensus algorithms
Public Consensus (PoW, PoS, PoA, PoL, PoR, PoET, PoSp, DPOS, Ripple, Pool)
Permissioned Consensus (PBFT, PAXOS, RAFT, PoO, SCP, Tendermint)

Fig. 5. Major aspects of data security in the blockchain.

However, the blockchain-based application suffers from objective fraud, subjective fraud, and rating fraud [85]. Thus, fraud detection is needed to enhance its robustness. Feng *et al.* [86] presented cyber-insurance and cyber-risk management approaches to neutralize cyber-attacks on the blockchain service network. Fraud detection in blockchain could be deployed via two aspects, namely, preventative control and detective control. Preventative control can monitor the activity of users proactively, predict the potential risk/fraud, and exert countermeasures. Detective control can perform audits and what-if analysis via dedicated criteria in blockchain [87]. There is a pressing need to predict and prevent potential threats from detecting existing attacks. The key is how to identify potential risks automatically and effectively with minimal human intervention.

4 Data Security in Blockchain

Data security in the blockchain can be characterized into integrity, confidentiality, and availability. This section surveys data security issues in blockchain from the aspects of cryptography, signature schemes, encryption techniques (including computing and retrieval), privacy protections, and consensus algorithms (Fig. 5).

4.1 Access Control in Blockchain

Access control has become more complicated in evolving information systems from integrated systems to distributed, cloud-based, and blockchain-based applications [22]. In blockchain research [88], [89], [90], [92], fine-grained access control of decentralized data is usually maintained by combining smart contract and Attribute-based Encryption (ABE) schemes. A smart contract can store the updated ciphertext of ABE, express the logic semantics of authorization, and flexibly define access control policies for removing and editing the data. Recently, scholars tried to combine a smart contract with machine learning models to provide automated, dynamic, optimized, and self-adjusted access control (authorization) [93].

Although smart contract-based access control has many advantages, scholars need to deal with security issues for improving the availability of blockchain. Access control strategies in blockchain are Public Key Infrastructures rather than a trusted Certificate Authority and can lead to significant security improvements [94]. The Stealth Address (SA) technique has been adopted in blockchain to make it unlikable among multiple payments made to the same payee [95]. However, some variants of SA techniques lack enough security assurance [96]. A robust SA method is needed for preventing data leakage and attacks.

In the ABE schema, private key is associated with the ciphertext data after encryption to a specific set of properties. Only when the property of the private key matches the property of the ciphertext can the user decrypt it [97]. Lewko and Waters [98] proposed a distributed ABE algorithm to allow participants to publish properties and distribute keys as administrators. The algorithm supports combinations of features to set the access control policy when encrypting data in the blockchain. However, the immutability of blockchain prevents the updating of attributes of ABE. Thus, its adoption has been severely hindered by the incompatibility between the immutability of blockchain and the essential need for revocations attribute of ABE. How to enable secure attribute updating in ABE-based access control for blockchain is challenging.

A redactable blockchain to secure and control access to the data is a potential solution to allow access policies. Adams proposed a hybrid fine-grained access control model in blockchain to store participant data postings in long-term–isolated environments [99]. Fig. 6 shows the process of one encrypts his postings in a block and gives the keys to another for the specific shared postings. In detail, Alice's biometric is input to the fuzzy extractor component to generates a uniformly random key K, from which the derivation function derives n symmetric encryption keys, along with a private signing key S. These symmetric keys are used to encrypt Alice's n postings, which are assembled into a block. The key S is used to digitally sign the block appended to the blockchain. Alice can share key k_2 with Bob if she wishes to share posting #2 with him. Bob can thereby get the ciphertext c_2 from the blockchain, decrypt it with k_2, and obtain the plaintext posting m_2. It is a hybrid use of the randomness of fuzzy extractor-generated binary strings, the illegibility of AES encryption, the unforgeability of ECDSA digital signature operations, and the collision-resistance of SHA hashing.

Many access control schemes in blockchain [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99] are complex to implement or use. Designing an efficient and easy-to-use model that enables both privacy and fine-grained access control in a blockchain is in practical need. How to efficiently and securely achieve revocation in access control schemes (i.e., to remove access to previously-shared posts) is challenging.

4.2 Computing Techniques on Encrypted Blockchain Data

4.2.1 *Homomorphic Encryption in Blockchain*

The attacks, such as collision attack, primage attack, and attacks on user wallets, motivated homomorphic encryption [100]. Homomorphic encryption supports algebraic calculations executed directly on the ciphertext instead of plaintext [101], [102], resulting in the confidentiality of data [103].

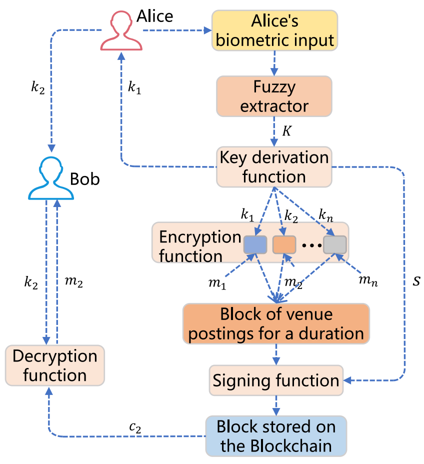


Fig. 6. A hybrid fine-grained access control model in blockchain [99].

Enabling the homomorphic encryption in blockchain could support users' data without any information about themselves [104]. Hsiao *et al.* [105] combined the blockchain with homomorphic encryption to perform the e-voting without any trusted third-party. The full-homomorphic encryption mechanism is usually inefficient, and thus partial homomorphic encryption attracts more attention from researchers [106] (e.g., the Pedersen commitment [107] in Enigma).

The homomorphic signature is proposed as an essential primitive to perform data authentication and to assure the information correctness [108]. There exist different kinds of homomorphic signatures, including the linear homomorphic signature scheme, polynomial homomorphic scheme, and leveled-fully homomorphic scheme. For instance, Lin *et al.* [109] designed a linear homomorphic signature scheme in identity-based cryptography. The proposed identity-enabled linearly homomorphic signature avoids the shortcomings of using public-key certificates. Li *et al.* [110] proposed a homomorphic signature scheme for networking and inter-operation of multiple devices in IoT. When integrating the homomorphic signature into the blockchain model, the computation and analysis could be executed while realizing the data authentication. However, in practical blockchain applications such as the federated learning model, other secure computing techniques such as pseudo-random noise generation and differential privacy are utilized as an alternative solution, due to the inefficiency and immaturity of full-homomorphic encryption in practice. Multiplicative depth of circuits is the primary practical limitation in the performance of the majority of homomorphic encryption algorithms. In terms of malleability, homomorphic encryption schemes have weaker security properties than non-homomorphic schemes. Still, they lack practical and efficient full-homomorphic signature schemes for blockchain in addition to the Gentry algorithm, especially for enabling the security and deep learning of decentralized blockchain data efficiently.

4.2.2 Secure Multi-Party Computation in Blockchain

Extending blockchain to support private data opens the door to many exciting new applications in healthcare, insurance, finance, etc. [111]. It is critical for multiple parties to compute a specific task while maintaining their input privacy and fairness collaboratively [112]. Secure multi-party computation (SMPC) [113] is cryptography for enabling multi-parties to cooperatively compute on their joint inputs while making those inputs private [114]. Unlike the conventional cryptographic methods that the adversaries are outside of the network to ensure data security and communication integrity, the adversaries in SMPC are also participants [115]. The decentralization characteristic of blockchain makes SMPC an ideal model in applications that needs privacy. To analyze a massive amount of data on the public blockchain, Enigma [115] used SMPC to offload intensive computations to off-chain. PlatON [116] delegated all control and incentive mechanisms to the smart contract, in which computation tasks and data can be kept confidential.

Many issues exist in integrating SMPC within blockchains, such as computation cost [117], poor scalability, and computation latency for the large-scale networks [118]. To obtain fairness against malicious behavior, full nodes providing computation/storage resources are required to submit a security deposit to a smart contract. Most SMPC methods are not verified practically in terms of communication bandwidth and computation costs. Some ongoing works try to make scalable solutions via hardware design or multi-processor architecture that supports the efficient simultaneously-computing on the joint data.

4.2.3 Trusted Computing in Blockchain

Some blockchain applications, such as sealed-bid auction, need verifiable computations [119]. Trusted computing (also termed as verifiable computation) improves data security by introducing an incentive mechanism [120] and usually offloads the computation of some functions to other untrusted clients while maintaining verifiable results [121], [122]. In a blockchain, the smart contract can only be verified by the miner's repeated computation to get consensus, which is not verifiable and results in the inefficient computation [120]. A malicious node may return forged computations to other parties in Enigma [115] and PlatON [116]. Therefore, verifiable computation (e.g., xjSnark framework [123]) is introduced to guarantee the integrity of the computation and the correctness of the computation result [118]. Incorporating blockchain into trusted execution environments (TEEs) helps the privacy protection in the executing implementations [124].

Combining the hardware design (e.g., the Intel SGX technology) with an execute-order-validate smart contract execution system could prevent rollback attacks on the enterprise-level TEE implementations [125]. Fu and Fang [126] employed an encryption algorithm for enhancing data privacy and then utilized the Proof-of-Credibility to build trusted computing in the social network. The integrity of computing can be ensured only via verifiable results (Fig. 7). Malicious nodes cannot obtain permission to the data computing in the TEE. Other trusted computing techniques include 1) building off-chain channels to resolve exchanges bilaterally without incurring a blockchain transaction, which boosts the overall throughput of the system; 2) addressing state continuity for memoryless secure processors that have access to a blockchain; 3) leveraging trusted execution to enhance the resilience of consensus protocols.

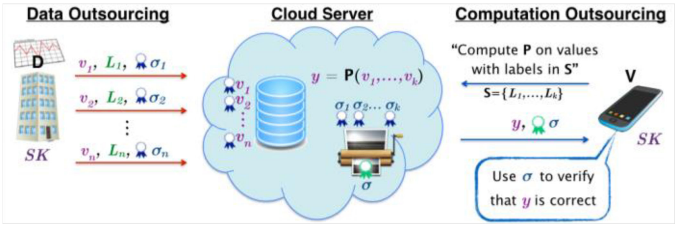

Data Outsourcing
Cloud Server
Computation Outsourcing
D
SK
v_1, L_1, σ_1
v_2, L_2, σ_2
v_n, L_n, σ_n
$y = \mathbf{P}(v_1, \ldots, v_k)$
$\sigma_1\,\sigma_2 \ldots \sigma_k$
σ
"Compute P on values with labels in S"
$S = \{L_1, \ldots, L_k\}$
y, σ
V
SK
Use σ to verify that y is correct

Fig. 7. Rational of trusted computing on outsourced data [127].

Distinguishing failed nodes from compromised nodes may send incorrect information to other nodes. Thus it is critical to achieving trusted computing by identifying the information transmissions [128]. Although the amount of verifiable computation is negligible compared with mining in consensus algorithms such as Proof-of-Work, trusted computing demands that device suppliers obey the technical specifications and community consensus for enabling interoperability among various computing stacks. Also, their technical specifications are still changing due to the evolution of blockchain. How to avoid the constraint that the blockchain must run the verifiable computation in an EVM is of great significance.

4.3 Retrieval Techniques on Encrypted Blockchain Data

4.3.1 Secure Data Provenance and Auditing in Blockchain

Holding complete transparency and control over digital identity will become far more common. Data provenance is based on metadata that captures the creation, change, and operation event that happened on the data. Provenance monitors observe nodes externally and record the evolution of data securely. Secure data provenance is critical to achieving data privacy and accountability. Assembling an accurate provenance record across the distributed environment can be realized by incorporating blockchain into the system. The blockchain transactions could anchor the provenance records to track data operations while preserves data privacy as well [129]. Incorporating smart contracts into blockchain could prevent malicious alteration data [130]. For instance, Liang *et al.* [131] proposed a decentralized data provenance architecture named ProvChain to enabling tamper-proof provenance, privacy protection, and low-cost cloud storage reliability. Tosh *et al.* [132] provided an assured data provenance method and modeled the block withholding attack while considering the incentive reward factor. However, new emerging regulations impose more data provenance protection demands on the processors and controllers.

Secure data auditing maintains an accurate audit history tracking of data operations/interactions, which is the precondition to distribute encrypted user-sensitive data securely. Blockchain could enable data usage to audit with high availability and privacy [133]. Deploying publicly auditable smart contracts on the blockchain could provide data accountability and provenance tracking ability, which increases data transparency [72]. Amir [134] customized a data authentication protocol for integrating blockchain with local storage to decentralized sensitive data. The protocol abstracts encrypted interactions to form a dashboard for secure auditing, testing, and evaluation data. Xia *et al.*

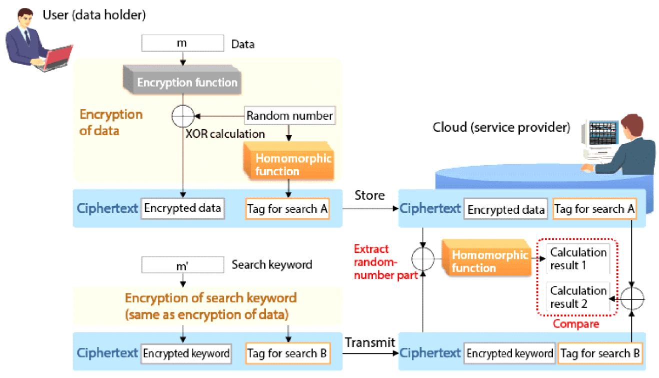


Fig. 8. Mechanism of searching on encrypted data [146].

[135] proposed a blockchain-based data auditing system named MeDShare to secure control for shared cloud medical datasets. However, an empirical analysis of metadata utilization suggests that users need to improve the auditability of blockchain transactions and smart-contract protocols [136]. Meanwhile, there are still some other crucial challenges, such as data assurance and resilience issues.

4.3.2 Encrypted Data Searching in Blockchain

A secured searchable data service is essential in a blockchain-type storage system for the data owner to upload their data in an encrypted form and enable others to search it [137]. However, encryption in blockchain limits the computability of the data block. It is complex to search the keywords in encrypted transaction data. On the other hand, keeping confidentiality in searching encrypted data on an untrusted server is tough [138]. Conventionally, the cloud storage adopted in a searchable symmetric encryption (SSE) model is usually implemented privately. In a public blockchain system, there exist two concerns in an SSE. The first concern is the security of encrypted data searching, which is critical for privacy protection and trustless computation [139]. The second concern is how to increase search efficiency [140]. Zhang *et al.* [141] introduced TKSE (i.e., Trustworthy Keyword Search Encryption) to realize server-side verifiability in cloud computing. It preserves servers and data from being attacked by malicious users in a storage step (Fig. 8). However, keyword-based search methods suffer from the limited capability of reflecting all search intentions of the user.

Moreover, preventing the service provider from forging the search result is difficult in a trustless context [142], [143]. It is necessary to put the data on a public blockchain to resist malicious adversary. Utilizing smart contracts could provide an encrypted data searching service [144]. The smart contract-based SSE mechanisms could ensure that the participants get the expected results [145]. Nevertheless, with the growth of data amount, the search issue is more intractable. Various efficient searchable encryption models should be developed according to implementation scenarios.

4.4 Signature Schemes in Blockchain

Conventional signature algorithms in blockchain include lattice-based signature [147], blind signature [148], ring signature [149], aggregation signature [150], multi-signature

[151], and threshold signature [152], [153]. This paper reviews the signature schemes in blockchain from the following three aspects.

4.4.1 Attribute-Based and Identity-Based Signature for the Blockchain

There are two kinds of cryptosystems, namely, attribute-based signature (ABS) and identity-based signature. The first kind refers to that the public key could be reasoned from the user's identifiers. The second kind can simplify certificate-based public key models to realize authentication in the blockchain. The attribute-based scheme is a useful tool to form cryptographic primitives [154]. ABS enables the signer to obtain a fine-grained control ability to identify information when endorsing a message. The signature provides a reliable guarantee for signers on privacy protection and a security guarantee for verifiers on the unforgeability. Many derived models have been proposed [155]. Sun *et al.* [156] proposed a decentralized ABS model for preserving data privacy in healthcare blockchain. Guo *et al.* [157] presented an ABS model with multi-authorities that do not need a third party to generate the public and private key. It prevents the escrow issue from happening in blockchain. Nevertheless, both two kinds of cryptosystems are still insufficient to satisfy the distributed demands.

4.4.2 Multi-Signatures and Aggregate Signature in Blockchain

The most widely used method of creating the contract is a multiple-signature technique called multisig [15]. Multi-signature enables multiple users to sign the same document. For instance, Boneh *et al.* [158] proposed the accountable-subgroup multi-signature model for compressing signature and aggregating the public key. The multi-signature-based authorization model could improve the security, extensibility, and programmability of Internet resources [159]. Hybrid use of blockchain, multi-signature, and anonymous encrypted data streams enable peers to negotiate and secure transactions [151].

An aggregate signature scheme supports shorten the size of the signatures on multiple messages. Yuan *et al.* [150] proposed an aggregate signature model for blockchain-based on the elliptic curve discrete logarithm and bilinear maps, where the size of the signature on the blockchain remains unchanged under a different number of inputs. Li *et al.* [160] used an anonymous vehicular announcement aggregation model for privacy-preserving incentive announcement. However, enabling non-deterministic users to generate and send signatures anonymously is still insufficient in the untrusted context.

4.4.3 Group/Ring Signature in Blockchain

Blockchain has obvious limitations for its uncertainty on the transactions to be confirmed. Dash uses a technique known as CoinJoin, which provides different variants of anonymity for multiple transactions [161]. However, Dash still suffers from the disclosure privacy of the users when malicious users abuse the server. Further, Monero [149] was proposed as a hybrid cryptographic model independent from the central nodes. There exist two kinds of models in Monroe: stealth address [96] and ring signature. Stealth addresses enhance the privacy of participants [162]. Ring signatures enable any participant to generate a signature representing the group without exposing its identity [163]. A linkable ring signature [164] provides linkable anonymity in a public blockchain. Scholars have improved the linkable ring signature from the following aspects: 1) storage space-saving capability [149], 2) multi-layered structure to enable verifiable hidden key details of transactions [165], 3) interactive incontestable scheme by which transactions can be written into blockchain in a non-repudiation manner [166], and 4) Lattice-based one-time Linkable Ring Signature (L2RS) scheme for absolute anonymity and privacy-preserving in the post-quantum stage [167].

The ring signature could be integrated with other techniques to enhance anonymity. Heilman *et al.* [148] combined blind signatures with Bitcoin transaction contracts to solve the anonymity and fairness problem for both transactions on and off the blockchain. Wang *et al.* [168] proposed a self-management anonymous electronic voting scheme while integrating homomorphic encryption and ring signature. A CryptoNote is designed to provide complete anonymity associated with key re-usage, and tracing [169] is (Fig. 9). In detail, a user (e.g., B in Fig. 9) could hide the link to his output among the foreign keys. To prevent double-spending, the user can derive the Key image from his One-time private key. Then, the user can sign the transaction and appends the resulting Ring Signature to the end of the transaction. In general, the incontestability of dealers and the unforgeability of owners are two ultimate goals of securing a signature scheme.

4.5 Privacy Protection Techniques in Blockchain

Although the blockchain relies on asymmetric encryption technology, it lacks identifying both the sensitivity and security level of data provided. The privacy protection is the primary goal in the data security level of blockchain systems [170]. Sensitive information is still easy to be obtained by cluster analysis and correlation analysis on an open distributed ledger. Zero-knowledge proof (ZKP) is a model with which the user can prove to another user that the statement is correct with high probability without revealing any information except the statement [171]. A zero-knowledge proof satisfies properties of completeness, soundness, and zero-knowledge. Combined with robust encryption methods and techniques [172], zero-knowledge proofs establish a much more secure way of storing and accessing data, enhancing the ability of data managers to protect critical information [173]. Zero-knowledge proofs can guarantee that transactions are correct even though the details remain hidden (e.g., Zerocoin [174], Zerocash [175], zkLedger [176], and Solidus [177]). Thomas *et al.* [178] proposed the ChainAchor framework for protecting privacy in a permissioned blockchain. Kosba *et al.* [179] proposed a ZKP-based privacy blockchain system named Hawk to generate a cryptographic protocol for enabling contractual parties to interact with the blockchain. The advantage of ZKP lies in that auditors can efficiently audit transactions [180]. However, ZKP suffers from many security issues on scalability [181], computation cost [182], as well as implementing complexity

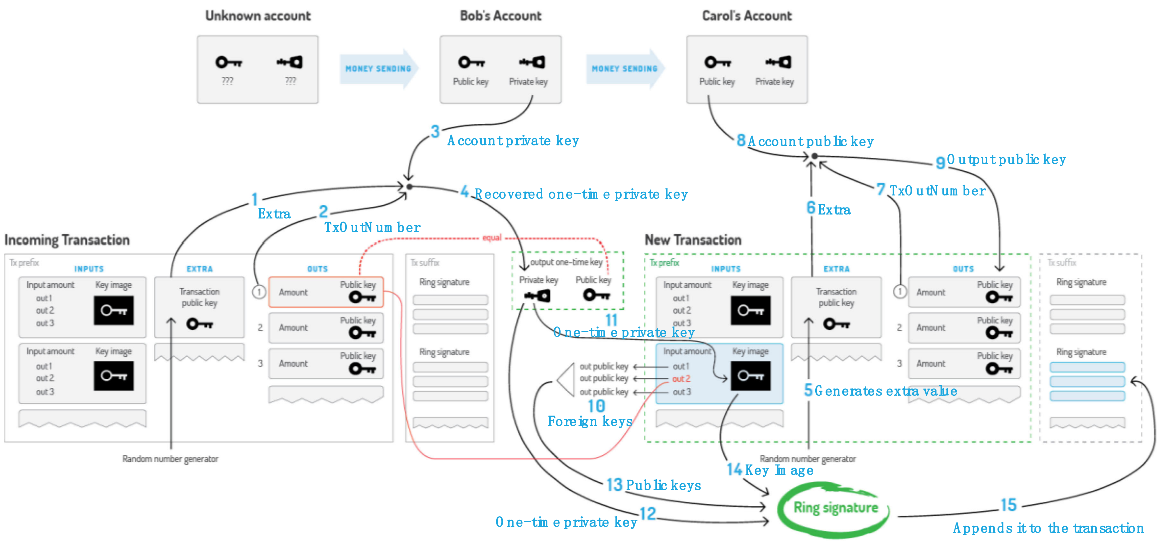


Fig. 9. A standard CryptoNote transaction [169].

TABLE 4
Signature Schemes and Privacy Protection Techniques in the Blockchain

Techniques	Advantages	Security issues	References
Homomorphic Encryption	Allowing arbitrary computing calculations on the encrypted dataset without the decryption key	Lack of efficient homomorphic encryption implementation	[102], [104], [105], [106], [107], [109]
Secure Multi-party Computing	Enabling distrusted nodes to compute over mutual inputs while ensuring confidentiality jointly	Computationally intensive impractical, inefficiency and not scalable	[111], [112], [115], [116], [117], [118]
Trusted computing	Allowing servers to provide improved security on that which is currently available	How to enable more ubiquitous computing tasks verifiable efficiently	[119], [120], [121], [122], [123]
Secure data provenance	Recording the history of the creation, change, and operations performed on the dataset	The privacy and availability of the provenance data	[129], [130], [131], [132]
Secure data auditing	Keeping a trust-less audit tracking for data operations and controlling access to the dataset	Data assurance and resilience issues are challenges	[39], [72], [133], [134], [135], [136]
Encrypted data searching	Enabling the encrypted data on an untrusted server searchable without leaking information	The fairness of searching service providers and users	[139], [140], [141], [142], [143], [144], [145]
Multi-signatures	Requiring multiple keys to finish a transaction	Original signer forgery attack	[15], [150], [151], [158], [159], [160]
Group/Ring Signature	Providing a characteristic of linkable anonymity	Privacy-preserving in any post-quantum secure applications	[148], [149], [157], [167], [168]
Zero-Knowledge Proof	User can prove to another that the statement is correct via revealing the veracity of the statement	Poor scalability, expensive proof generation, and engineering complexity	[122], [173], [174], [175], [176], [177], [178], [179], [180], [181], [182]

[177]. For instance, the security of zkSNARKs (provides zero-knowledge proof using elliptic curves [181]) used in Zerocash has not yet been proved in theory.

Table 4 provides an overview of data security techniques. The integration of blockchain with these cryptographic techniques can perform more secured functions that are difficult to be realized by the blockchain itself or other technologies. Still, security issues in the integration process need to be further addressed.

4.6 Consensus Algorithms for Distributed Data Synchronization

Ethereum blockchain was launched in July 2015 [31], followed by Hyperledger [111], Eris [117], Stellar [118], Ripple [119], and Tendermint. Blockchain needs to motivate participants via sophisticated consensus algorithms in the absence of trust under the Internet environment. It is necessary to propose robust consensus algorithms with higher fault tolerance and anti-fraud ability. The consensus algorithm enables nodes in the distributed system to reach an agreement given faulty processes or deceptive behaviors [183]. It is the indicator from which we can judge the blockchain, whether public or permissioned is. The consensus is the data synchronization used in decentralized blockchain. Any user in the public blockchain can participate in verifications of data. Permissioned consensus-based blockchain has strict access control for the dataset while enabling the privacy protection of data transactions.

4.6.1 Consensus for Public Blockchain

Well-known consensus algorithms for public blockchain includes Proof of Work (PoW), Proof of Stake (PoS), Proof of

TABLE 5
Comparative Analysis of Public Blockchain Consensus Algorithms

Algorithms	Ref.	Advantages	Issues
POW	[185]	Complete decentralization, ad free entry, and exit of nodes	51% Vulnerability, selfish mining attack, and double-spending
POS	[188]	Shorten the time to reach consensus	Requires mining
DPOS	[189]	Second-level consensus validation	Relies on tokens, while many applications do not require tokens
PoA	[190]	Integrates POW with POS while producing less delay	Demands a large number of computations for mining
PoL	[191]	Low-latency transaction validation and equitably distributed mining	It is expensive to violate the security requirements of the TEE for all participants
PoR	[192]	Provide quality-of-service guarantees	An efficient set of parameters, choices, protocol variants, and tradeoffs in the implementation of POR remains a problem
PoET	[194]	Use trusted computing to regulate random waiting times for the generation of data block	Lack of theoretical security analysis
PoSp	[195]	Replace the computation in PoW as a disk space.	An efficient implementation remains a problem.
Ripple	[196]	Circumvent the network synchronization demand by using collectively-trusted subnetworks in the super network	Lack of clearly-defined incentive verification mechanisms

Activity (PoA), Proof of Luck (PoL), Proof of Retrievability (PoR), Proof of Elapsed Time (PoET), Proof of Space (PoSp), Delegated Proof of Stake (DPOS), Ripple, and Pool [184]. As information is visible across the entire public blockchain network, transactions based on the exposed data could be tracked by users, and thereby, individuals may not be able to obtain privacy protection under this mechanism.

Although the Bitcoin has been thoroughly analyzed, the security provisions of variant PoW-powered blockchains have not obtained enough attention [185]. The 51 percent vulnerability [186], selfish mining attack [187], and double-spending are still significant security issues in PoW-powered blockchains. PoS-powered blockchains stake with rigorous security guarantees. It shortens the time to reach consensus. Gervais *et al.* [188] presented a reward mechanism for incentivizing PoS for preventing selfish mining. However, it still requires mining, which has not solved the pain points of commercial application.

DPOS-powered blockchain is a candidate for the inefficient PoW. Andreina *et al.* [189] proposed two DPOS consensus protocols to solve the issue of nothing-at-stake and stop malicious users from launching long-range attacks. However, it relies on tokens, while many commercial applications do not require tokens. PoA is designed to ensure that the data transactions are genuine. It is a mixed model that integrates POW with POS while producing less networking delay [190]. PoA still needs massive computations for mining. PoL-powered blockchains use a trusted execution environment (TEE) system's random number generator to select a consensus leader with the advantages of highly-efficient validation and deterministic confirmation [191]. However, it is expensive to violate the PoL assumption that all participants should satisfy the security demands of the TEE. PoR allows an archive to reliably produce and transmit a concise proof that a verifier can check and recover without downloading all the files [192]. PoR is a type of cryptographic proof of knowledge (POK) [193] to manipulate large-scale data. Searching an efficient set of modeling parameters, protocol variants, and points of equilibrium in the practical implementation of POR remains a problem under rigorous quality-of-service guarantees. PoET uses trusted computing to regulate random waiting times for the generation of the data block. However, the PoET system lacks a thorough theoretical security analysis for evaluating its strength in front of failures and malicious attacks [194]. PoSp is a method that a service demander needs to spend a large amount of storage space instead of the computing complexity in PoW [195]. However, searching for an efficient set of modeling parameters, protocol variants, and tradeoffs in the implementation is costly. Ripple circumvents the demand that all participants in the system communicate synchronously by utilizing collectively-trusted subnetworks within the more extensive network. It is a low-latency consensus model with high robustness for Byzantine failures [196]. However, the decentralization character is doubtful in Ripple since it lacks a clearly-defined incentive mechanism

The consensus mechanism of public blockchain may suffer from a coordinated attack since it is assumed that most of the participants are honest to run the blockchain application [30]. The existing consensus mechanisms have the characteristics of decentralization and trust-free, leading to the difficulty in balancing equality, efficiency, and energy consumption [197]. The POW has proved to be the most mature consensus mechanism at present, which is entirely decentralized, but requires many resources to "mine" and takes a long time to reach consensus. POS takes less time to reach consensus than the POW mechanism, but mining is still needed. DPOS can achieve second level consensus verification. However, it also depends on the token mechanism. Pool verification is developed based on the integration of traditional distributed consistency and data verification mechanism. It has been widely used in the industry blockchain. Table 5 summarized the advantages and disadvantages of consensus mechanisms. One challenge faced by blockchain is that consensus on the attack had to be reached in a disseminated environment because it is decentralized with no central authority or node.

The energy consumption of the public blockchain is prohibitive to enterprise scenarios. New attempts (e.g., Fabric) were made to deal with these issues. However, the above solutions are dedicated solely to satisfy the individualized requirements of the specific business context. The existing

TABLE 6
Comparative Analysis of Permissioned Blockchain Consensus Algorithms

Algorithms	Ref.	Advantages	Issues
PBFT	[201]	Reach a consensus despite malicious nodes propagating incorrect information	The additional cost from protocol complexity
PAXOS	[206]	Guarantee safety (consistency) that can prevent it from making progress	Hard to understand and challenging to implement
RAFT	[207]	Easy to understand and implement	Searching an efficient set of parameters, modeling choices, protocol variants, and tradeoffs in the implementation of these algorithms remains a problem
PoO	[208]	Enable offline ownership proofs	
SCP	[209]	Make no assumptions about the rational behavior of attackers	
Tendermint	[210]	Do not need costly mining	

enterprise security systems compromise on performance when adopting complex protocols. Therefore, one potential research agenda in public blockchain platforms is to study the optimization of the consensus algorithm. Some platforms (e.g., Coco [198]) are designed to cut down energy consumption.

4.6.2 *Consensus for Permissioned Blockchain*

The popular consensus algorithm for permissioned blockchain includes Practical Byzantine Fault Tolerance (PBFT), PAXOS, RAFT, Proof of Ownership (PoO), Stellar Consensus Protocol (SCP), and Tendermint [184] (shown in Table 6). Permissioned blockchain is composed of prespecified nodes with the data access authority in the distributed network. Its participants do not fully trust each other, and thus should select their consensus nodes. Meanwhile, private blockchain allows a central authority to be present with higher efficiency in verification and validation of transactions. For instance, a Fabric project hosted by the Linux Foundation enables the blockchain network to be configured to particular scenarios and trust models for running distributed applications.

The primary deficiency of private blockchain is that it does not ensure decentralized security provided by the public blockchain [10]. Conflicts happen among nodes may become malicious [199]. Secondly, unsynchronized and isolated nodes always exist in consensus schemes [200]. It is not conducive to efficient and robust-time synchronization in industrial applications, which is crucial for achieving high accuracy. Thirdly, compared with crash fault tolerance (CFT), the Byzantine fault-tolerant (BFT) system has not been implemented practically because of the substantial cost relative to crash fault-tolerance (CFT) in algorithm complexity [201]. Scholars have proposed many methods to avoid these deficiencies. For instance, Fan [202] proposed a short-lived signature-based PBFT variant utilizing blockchain-based distribution methods to update keys regularly. Liu *et al.* [203] introduced cross fault tolerance to provide the reliability guarantees of both CFT and BFT protocols asynchronously. Sankar *et al.* [204] proposed the concept of quorum slices and federated byzantine fault tolerance. Pîrlea and Sergey [205] presented the formalization of a consensus algorithm with a proof of its consistency mechanized in an interactive proof assistant.

PAXOS is usually adopted in a condition that the system durability and consistency is required, in which the amount of stable state could be considerable. Many variants such as Cheap PAXOS, Fast PAXOS, Multi-PAXOS, Generalized PAXOS, Fast Byzantine Multi-PAXOS have been proposed [206]. RAFT [207] provides a generic method to distribute a state machine over a cluster of nodes while assuring that each participant agrees upon the same series of state transitions, which is more understandable and reasonable than PAXOS due to its separation of logic. PoO [208] enables offline ownership proofs in which copyright holders can prove their ownership without any other third-party. SCP does not need any assumption about the rational behavior of malicious attackers. Compared to decentralized PoW, SCP [209] has lower computational requirements, making it easier to entry and consequently opening up an economic system to newly-join participants. Tendermint [210] does not need costly mining to the Byzantine Generals Problem.

In general, prior research mainly analyzed the security properties of the consensus algorithm by probabilistic reasoning based on a composition of distributions. Engineering blockchain consensus is similar to developing a cryptographic system, and designers should refer to the best-practice on cryptography and distributed security for developing trustworthy systems [211]. Searching an efficient set of parameters, modeling choices, protocol variants, and tradeoffs in the implementation of these algorithms remain problems. An executable semantic model can be developed to prove correctness conditions and the eventual consistency of the system.

5 Infrastructure Security in Blockchain

Blockchain infrastructure becomes more exposed to vulnerabilities than ever before [151]. This section reviews Infrastructure Security in Fig. 10. Generally, infrastructure security contains three aspects: private key management, terminal and networking vulnerability, and compliance enablement.

5.1 Private Key Management in the User Side

The blockchain generates a dual encryption system of the public key and private key employing encryption. However, there is still a lack of a mechanism for key security storage and recovery [212]. Private keys and digital wallets are at risk of being stolen. Once the private key is lost, nothing can

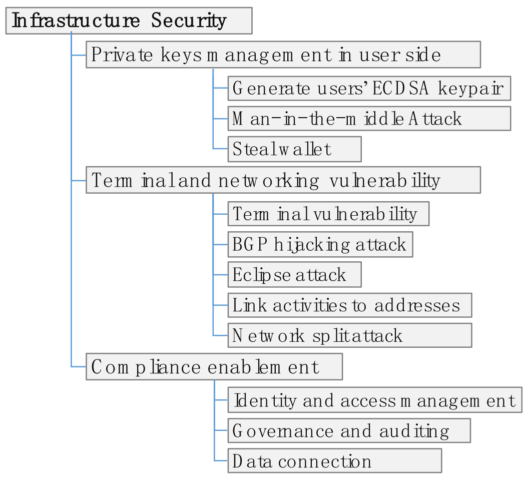


Fig. 10. Major aspects of the infrastructure security in the blockchain.

be done with the account [213]. Secure key management models are recognized as analytical technology for achieving infrastructure security.

Unlike traditional public-key cryptography, private keys in blockchain are managed by users themselves instead of a third-party. Since there is no mechanism for establishing an association between the key and the user's identity in the blockchain, a private key not only protects the privacy of user identity but also leads to high-cost in key recovering. Blockchain applications confirm a user's identity via private keys. The precondition of falsifying transaction information is to get the private key. For instance, Mayer [214] discovered a vulnerability in Elliptic Curve Digital Signature Algorithm (ECDSA): An implementation that does not guarantee enough randomness in the signature allows a malicious attacker generate/recover the private key pair of a user. Therefore, under a decentralized security framework, it is vital to figure out how to establish a more secure private key management mechanism in order to achieve overall security. It is, therefore, critical to design a practical scheme for directly obtaining user keys, even if the number of malicious hosting agents is greater than or equal to the threshold value (i.e., a Man-in-the-middle attack). Thus, a dynamic lifecycle management mechanism needs to be established to replace the current permanent key management methods.

5.2 Terminal and Networking Vulnerability

The terminal linked to the blockchain (including a computer, mobile phone, tablet, and other terminal devices) is a security vulnerability outside of the blockchain itself [215]. Watanabe and Fan [216] proposed a chip-level blockchain security system for the IoT network. Suankaewmanee *et al.* [217] introduced a new blockchain model named MobiChain to secure transactions in mobile commerce without the need for costly mining processes on mobile devices. In general, enabling terminal security needs intensive studies on designing the hardware and securing customized core operating-system-level modules.

The infrastructure security issues at the networking level mainly include the conventional DDoS attacks, Border Gateway Protocol (BGP) hijacking attack, Eclipse attack, link activities-to-address, and network split attack. To disrupt the networking traffic of the blockchain system, malicious users usually tamper the BGP routing, which is a gateway protocol for managing the reachability information across the autonomous system. The BGP guides routing decisions by analyzing the prespecified network paths, policies, and rules. BGP hijacking is the process of rerouting network traffic to a mining pool managed by the attackers and thus to steal digital assets from the users. Apostolaki *et al.* [218] analyzed the impacts of node-level and network-level routing attacks. Because the BGP security extensions have not been widely implemented, the operators of the blockchain network have to rely on a monitoring system. Moreover, solving a BGP hijacking is time-consuming since it is a complicated reconfiguration process [13]. Besides, attackers could launch the eclipse attack to isolate users from the other part of the blockchain network and block their incoming and outgoing interactions [219]. The eclipse attack may lead to other attacks such as selfish mining and splitting mining power.

There are many anti-malware filters to identify the suspected actions in terminal devices through attack pattern matching managed in a central server, vulnerable to malicious uses [10]. Noyes [220] proposed an anti-malware system to manage the learned attack patterns through the blockchain. To establish secure connections light client to each full node, as well as cut-down the client-side complexity and speed-up verification, Costa *et al.* [221] proposed a secure verification protocol for the blockchain named Distributed Lightweight Client Protocol, which requires clients to encrypt a request once, allowing a prespecified set of full nodes to manage it. Lee *et al.* [222] presented a blockchain-enabled firmware update model to check a firmware version securely and to validate the correctness of firmware. Using the private and public keys in conjunction with data encryption may provide the blockchain with a higher level of infrastructure security.

A distributed data storage mechanism creates a border attack range of blockchain. It allows an attacker to has more alternative nodes and terminal devices to access data [79]. Perpetrators could be either insider (authorized users) or outsiders who clandestinely access a system [223]. Wu and Monticelli [224] presented a survey of various network modeling models for security analysis. Protective security countermeasures [225] call for substantial managerial vigilance and an adequate level of awareness for defining the responsibilities of critical roles in the organizations.

5.3 Compliance Enablement

Organizations in blockchain applications are required to demonstrate compliance with various regulations. Compliance enablement is to demonstrate data security in access management and privileges [22]. A valid user awareness program could be developed and implemented to communicate the policies and procedures to all participants and make them aware of the consequences of abuse of information. Also, a periodic compliance evaluation model could be established to track the effectiveness of the security algorithm [18].

Besides, the data connection between blockchain and external data resources is vulnerable. Blockchain and smart contracts often need to interact with the off-chain dataset. Because the contract deployed on the network cannot access

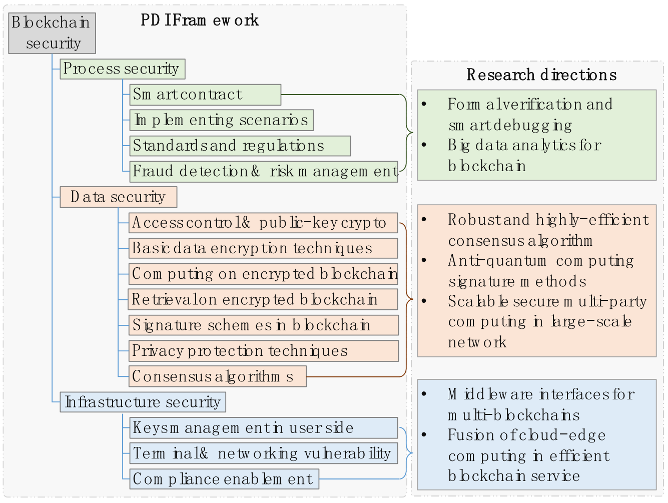

Blockchain security
PDI Framework
Process security
Smart contract
Implementing scenarios
Standards and regulations
Fraud detection & risk management
Data security
Access control & public-key crypto
Basic data encryption techniques
Computing on encrypted blockchain
Retrieval on encrypted blockchain
Signature schemes in blockchain
Privacy protection techniques
Consensus algorithms
Infrastructure security
Keys management in user side
Terminal & networking vulnerability
Compliance enablement
Research directions
Formal verification and smart debugging
Big data analytics for blockchain
Robust and highly-efficient consensus algorithm
Anti-quantum computing signature methods
Scalable secure multi-party computing in large-scale network
Middleware interfaces for multi-blockchains
Fusion of cloud-edge computing in efficient blockchain service

Fig. 11. Research directions from the PDI view of blockchain security.

external data directly through HTTPS, Zhang *et al.* [226] proposed an authenticated data feed system named Town Crier for aiding the robust and secure interaction between HTTPS data and smart contracts. Also, in the Web era, the compliance enablement of blockchain-based applications may be facilitated by integrating with service computing methods.

6 Research Directions

Blockchain needs an integrated technological, organizational [227], policy-based intelligence, and security informatics solution [228]. Based on the above analysis, future directions are gathered (Fig. 11).

6.1 Directions for Enhancing Process Security of Blockchain

6.1.1 Formal Verification and Smart Debugging Mechanism

Extensive adoption of cryptographic algorithms in blockchain may introduce unpredictable vulnerabilities in the implementation processes [30]. The issues of carrying out transactions in a secure manner consistently are vital challenges to blockchain [79]. Current research usually analyzes the security properties of the consensus algorithm by probabilistic reasoning as a composition of distributions. An executable semantics model can be developed to prove correctness conditions and the eventual consistency of the system. From the contract security aspect, the problems in modeling semantics of the contract are troublesome. It is of considerable significance to identify their semantics and the security properties in the bytecode being executed [38]. An automated analyzer to mine semantic information of the security of contracts helps check behaviors as safe or unsafe concerning a prespecified property via compliance and violation patterns. From the implementation security aspect, a regulatory sandbox of implementation could give a more flexible space for innovations while imposing simplified standards and procedures. A verification protocol is a requisite to harness the infrastructure security.

This dimension includes three future sub-directions. In terms of the contract security, new vulnerability scanning methods that could discover unknown vulnerabilities (infer whether there are vulnerabilities that have identical or similar principles to the known vulnerabilities) and optimize vulnerability detection capability is of great significance. Secondly, in terms of the correctness of smart contracts, formulating/designing the standards for the programming and development processes of smart contracts is also critical. Also, visualizing the contract execution process using formal modeling tools such as Petri Nets is a promising direction. Finally, the mathematical model is more rigorous and reliable for formal verification. Combining formal verification algorithms with vulnerability analysis algorithms to complement each other is also helpful.

6.1.2 Big Data Analytics for Blockchain

Since participant behavior in the blockchain network is traceable, many blockchains take countermeasures such as one-time accounts and private keys for protecting the transaction privacy of users, which are not robust enough as required [13]. To pursue a more secure way to share data, blockchain needs to provide significant high-quantity data [79]. Although data in the blockchain is tamper-resisting, attackers can conduct data analysis to extract valuable information [30]. Various statistical characteristics regarding transactions (e.g., sum, skew, variance, and outlier) can be audited correctly without missing any transactions. Techniques such as context discovery, classifiers [229], link prediction [230], graph summarization [231], and flow analysis [232] could be utilized. The public transaction graph could be annotated and analyzed to find and summarize the activities of all participants statistically [233], [234]. Threats to data integrity and tampering with data may affect crucial decisions [235].

When the credibility of data can be assured in blockchain, more trustworthy outcomes would be achieved with artificial intelligence. Three aspects of implementing big data analytics in blockchain could be further studied. Firstly, in the process level, deep learning [236], [237] methods can be introduced to analyze the interaction contexts in the blockchain network for risk and attack prediction. Secondly, most public blockchains adopted incentive mechanisms whose security performance is untested. Deep reinforcement learning could be introduced to analyze attacks on incentive mechanisms [238]. Thirdly, decentralized management of data has substantial impacts on big data processing [239], [240]. Due to throughput, latency, and stability issues [241], there is a severe problem with leveraging big data analytics to process high-quality blockchain data. Federated learning algorithm is a promising direction to unfolding the high-quality data in the blockchain system to achieve group decision intelligence while preserving data privacy.

6.2 Directions for Enhancing Data Security in Blockchain

6.2.1 Anti-Quantum Computing Signature Methods

An indispensable part of the transaction data is the signature of transaction. With the rapid development of quantum computing research, the traditional public-key algorithms, such as RSA (Rivest, Shamir, Adleman), ECDSA (Elliptic Curve Digital Signature Algorithm)], ECDH (Elliptic Curve Diffie-Hellman), and DSA (Digital Signature Algorithm)

[242], face enormous challenges, since the quantum computing makes the attacks possible based on Grover's and Shor's algorithms [243]. Quantum computing can crack 1024-bit keys in seconds theoretically. If an anti-quantum digital signature algorithm with a short signature can be constructed, the signature security of blockchain can be much guaranteed. For instance, a quantum-safe transaction authentication scheme based on lattice-based cryptography is presented to provide a standard transaction model to prevent quantum attacks [244]. Some scholars have suggested the hardware design of quantum-safe blockchains. For instance, the IOTA (www.iota.org) program used ternary hardware (instead of traditional binary hardware), which supports a new hash function called CURL-P, to resist quantum attacks. Other scholars suggested using quantum cryptography to implement smart contracts. More research is necessary for the physics-based methods that are known as Quantum-Key Distribution. The transition from pre-quantum to post-quantum blockchains requires recognizing relevant security demands to ensure an accurate grasp of risks and countermeasures before implementing quantum-secured blockchain [245].

6.2.2 *Robust and Highly-Efficient Consensus Algorithm*

Consensus algorithms are critical to achieving data consistency among all distributed participants in a system. In general, prior research mainly analyzed the security properties of consensus algorithms by probabilistic reasoning based on a composition of distributed components. Searching an efficient set of parameters, modeling choices, protocol variants, and tradeoffs in the implementation of these algorithms remain to be open problems. An executable semantic model can be developed to prove correctness conditions and the eventual consistency of the system. Moreover, a light-weight and highly-efficient consensus on the public blockchain is desired. One potential research agenda in public blockchain platforms is to study the optimization of the consensus algorithm to cut down energy consumption.

On the other hand, the hybrid consensus protocol is a promising approach to cherry-pick individual protocol components to fulfill specific application needs. The flexibility and fast switchover of plugging among different consensus algorithms could achieve better performance under dynamic and asynchronous network conditions.

6.2.3 *Scalable Secure Multi-Party Computing in Large-Scale Network*

Dealer incontestability and owner unforgeability are the ultimate goals of a secure signature scheme. It is favorable for non-deterministic users to generate and send signatures anonymously in the untrusted context. There are many issues in integrating SMPC with blockchains, such as computation cost, poor scalability, and computation latency for large-scale networks. Improving the scalable hardware design or multi-processor architecture to support efficient simultaneous computing on the joint data is one promising research direction. More practical leveled fully homomorphic encryption schemes can be developed for enabling the secure and efficient transfer learning of decentralized blockchain data.

6.3 Directions for Enhancing Infrastructure Security of Blockchain

6.3.1 *Middleware Interfaces for Multi-Blockchains*

The lack of middleware interfaces makes it hard for practitioners to benefit from known exploration and mistakes. Middleware interfaces play a critical role in ensuring interoperability among multiple blockchains. Establishing middleware interfaces to address the security, resilience, privacy protection, and governance concerns in blockchain could create trust [79]. Interfaces should be deployed to support open standards among security components [22]. Cross-chain middleware [246] plays a similar role in limiting the scope of security crisis in cross-chain cooperations. Heterogeneous information fusion could be helpful for credibility assessments in border security [247]. A unified middleware interface for upper applications is convenient for security monitoring. Also, sidechain technologies could be investigated to build secure interconnections across blockchains.

6.3.2 *A Fusion of Cloud-Edge Computing for Efficient Blockchain Service*

Blockchain applications face various scalability issues as adoption increases [15]. Besides effort on optimizing the consensus algorithm and infrastructure (such as SBFT [248], SMChain [249], RapidChain [250], and LinBFT [251]), one trend in blockchain research is the shift of data services to centralized clouds, because of convenience and cost-saving reasons [252]. With the integration of blockchain into cloud computing, blockchain could be improved into an efficient service with more robust security [253]. A distributed secure cloud architecture based on blockchain could provide on-demand services under computing infrastructures [254]. Granular computing [255], [256] can be introduced for searching the optimal granularity of node selection in the blockchain application. However, using conventional cryptographic models to address privacy issues in a blockchain-based cloud context is questionable [257]. Identifying the cause of security violations in the cloud context would result in heavyweight collection tasks of logs from a massive number of sources. The dynamic networking topology between the cloud server and the edge devices is a bottleneck for latency-sensitive disaster recovery [258]. Fog computing is a potential architecture to resolve these issues [259], allowing provisioning services between the cloud and the end machines, and closer to edge devices. Incorporating fog computing into blockchain is not a replacement for cloud computing but an excellent complement.

7 Concluding Remarks

This paper surveys the landscape of blockchain security issues and outline research opportunities in information systems and services. The security of the blockchain is categorized into three levels, namely, the process level, the data level, and the infrastructure level, which we refer to as the PDI model of blockchain security. Our study also examines to what extent these security aspects have been addressed. Based on insights obtained from the analysis of research issues, promising research directions for blockchain security are outlined. We believe that our study reflects significant conceptual and technical advances in this junction of dramatic development, and we hope that our effort lays a strong foundation for making blockchain security a new venue of service research and engineering.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No. 2018AAA0101700; the National Natural Science Foundation of China under Grant No. 52075107 and No. 71932002; the Shenzhen Special Fund for the Development of Strategic Emerging Industries under Grant No. JCYJ20170818100156260; the S&T Planning Project of Guangdong Province under Grant No. 2019A050503010, and the S&T Planning Project of Sichuan Province under Grant No. 2019-YF05-02050-GX and 2020YFSY0061.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, [Online]. Available: https://git.dhimmel.com/bitcoin-whitepaper/.

[2] S. Underwood, "Blockchain beyond bitcoin," *Commun. ACM*, vol. 59, pp. 15–17, 2016.

[3] Y. Yuan and F. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[4] J. L. Zhao, S. Fan, and J. Yan, "Overview of business innovations and research opportunities in blockchain and introduction to the special issue," *Financial Innov.*, vol. 2, pp. 1–7, 2016.

[5] M. Crosby, P. P. Nachiappan, S. Verma, and V. Kalyanaraman, "BlockChain technology: Beyond bitcoin," *Appl. Innov. Rev.*, vol. 6, pp. 1–16, 2016.

[6] D. Efanov and P. Roschin, "The all-pervasiveness of the blockchain technology," *Procedia Comput. Sci.*, vol. 123, pp. 116–121, 2018.

[7] I. O. Pappas, P. Mikalef, M. N. Giannakos, J. Krogstie, and G. Lekakos, "Big data and business analytics ecosystems: Paving the way towards digital transformation and sustainable societies," *Inf. Syst. e-Bus. Manage.*, vol. 16, pp. 479–491, 2018.

[8] H. Ngo Higgins, "Corporate system security: Towards an integrated management approach," *Inf. Manage. Comput. Secur.*, vol. 7, pp. 217–222, 1999.

[9] M. T. Siponen and H. Oinas-Kukkonen, "A review of information security issues and respective research contributions," *Database Adv. Inf. Syst.*, vol. 38, pp. 60–80, 2007.

[10] A. P. Joshi, M. Han, and Y. Wang, "A survey on security and privacy issues of blockchain technology," *Math. Foundations Comput.*, vol. 1, pp. 121–147, 2018.

[11] G. Dhillon and G. Torkzadeh, "Value-focused assessment of information system security in organizations," *Inf. Syst. J.*, vol. 16, pp. 293–314, 2006.

[12] I. Lin and T. Liao, "A survey of blockchain security issues and challenges," *Int. J. Netw. Secur.*, vol. 19, pp. 653–659, 2017.

[13] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comp. Syst.*, vol. 107, pp. 841–853, 2020.

[14] S. A. Hossain, "Blockchain computing: Prospects and challenges for digital transformation," in *Proc. 6th Int. Conf. Rel. Infocom Technol. Optim.*, 2017, pp. 61–65.

[15] J. Park and J. Park, "Blockchain security in cloud computing: Use cases, challenges, and solutions," *Symmetry*, vol. 9, 2017, Art. no. 164.

[16] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Inform.*, vol. 36, pp. 55–81, 2019.

[17] C. Blanco, J. Lasheras, R. Valencia-Garcia, E. Fernandez-Medina, A. Toval, and M. Piattini, "A systematic review and comparison of security ontologies," in *Proc. 3rd Int. Conf. Avail. Rel. Secur.*, 2008, pp. 813–820.

[18] J. Tudor, *Information Security Architecture*. New York, NY, USA: Auerbach Publishers, 2000.

[19] J. Rees, S. Bandyopadhy, and E. Spafford, "PFIRES: A policy framework for information security," *Commun. ACM*, vol. 46, pp. 101–106, 2003.

[20] J. H. P. Eloff and M. M. Eloff, "Information security architecture," *Comput. Fraud Secur.*, vol. 2005, pp. 10–16, 2005.

[21] J. J. Darwin, "Security reference architecture," 2010. [Online]. Available: ftp://public.dhe.ibm.com/software/au/downloads/pulse_ANZ/srmc/Session2.pdf.

[22] D. Chappelle, "Security in depth reference architecture," Oracle enterprise transformation solutions series, an oracle white paper, 2013.

[23] ISO/IEC, "ISO/IEC 27000 family - Information security management systems," 2013. [Online]. Available: https://www.iso.org/isoiec-27001-information-security.html.

[24] *CIS*, "CIS critical security controls," 2017. [Online]. Available: https://www.cisecurity.org/controls/.

[25] D. Meltzer, "Security reference architecture: A practical guide to implementing foundational controls," 2017. [Online]. Available: https://www.tripwire.com/.

[26] NIST, "Cybersecurity framework," 2018. [Online]. Available: https://www.nist.gov/cyberframework.

[27] D. Trèek, "An integral framework for information systems security management," *Comput. Secur.*, vol. 22, pp. 337–360, 2003.

[28] L. Kiely and T. V. Benzel, "Systemic security management," *IEEE Secur. Privacy*, vol. 4, no. 6, pp. 74–77, Nov./Dec. 2006.

[29] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: Platforms, applications, and design patterns," in *Financial Cryptography and Data Security*. Cham, Switzerland: Springer, 2017.

[30] F. Dai, Y. Shi, N. Meng, L. Wei, and Z. Ye, "From bitcoin to cybersecurity: A comparative study of blockchain application and security issues," in *Proc. IEEE 4th Int. Conf. Syst. Inform.*, 2017, pp. 975–979.

[31] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (SoK)," in *Proc. Int. Conf. Princ. Secur. Trust*, 2017, pp. 164–186.

[32] A. Juels, A. Kosba, and E. Shi, "The ring of gyges: Investigating the future of criminal smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 283–295.

[33] A. Mavridou and A. Laszka, "Designing secure ethereum smart contracts: A finite state machine based approach," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2017, pp. 523–540.

[34] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Financial Cryptography and Data Security*. Cham, Switzerland: Springer, 2017.

[35] M. Wohrer and U. Zdun, "Smart contracts: Security patterns in the ethereum ecosystem and solidity," in *Proc. Int. Workshop Blockchain Oriented Softw. Eng.*, 2018, pp. 2–8.

[36] M. Giancaspro, "Is a smart contract really a smart idea? Insights from a legal perspective," *Comput. Law Secur. Rev.: Int. J. Technol. Law Practice*, vol. 33, pp. 825–835, 2017.

[37] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2016, pp. 79–94.

[38] I. Grishchenko, M. Maffei, and C. Schneidewind, "A semantic framework for the security analysis of ethereum smart contracts," in *Principles of Security and Trust*. Cham, Switzerland: Springer, 2018.

[39] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 67–82.

[40] K. Bhargavan *et al.*, "Formal verification of smart contracts," in *Proc. ACM Workshop Program. Lang. Anal. Secur.*, 2016, pp. 91–96.

[41] L. Luu, D. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2016, pp. 254–269.

[42] M. M. Queiroz, R. Telles, and S. Bonilla, "Blockchain and supply chain management integration: A systematic review of the literature," *Supply Chain Manage., An Int. J. Online*, vol. 25 No. 2, pp. 241–254, 2019. [Online]. Available: https://doi.org/10.1108/SCM-03-2018-0143

[43] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, pp. 352–375, 2018.

[44] Y. Zhang and J. Wen, "The IoT electric business model: Using blockchain technology for the Internet of Things," *Peer Peer Netw. Appl.*, vol. 10, pp. 983–994, 2017.

[45] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 277–293, Feb. 2013.

[46] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, pp. 533–546, 2016.

[47] R. H. Weber, "Internet of Things – New security and privacy challenges," *Comput. Law Secur. Rev.*, vol. 26, pp. 23–30, 2010.

[48] N. Kshetri, "Can blockchain strengthen the Internet of Things?" *IT Professional*, vol. 19, pp. 68–72, 2017.

[49] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT integration: A systematic survey," *Sensors*, vol. 18, 2018, Art. no. 2575.

[50] J. Granjal, E. Monteiro, and J. Sa Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 3, pp. 1294–1312, Thirdquarter 2015.

[51] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Commun. Surv. Tuts.*, vol. 21, no. 1, pp. 858–880, First Quarter 2019.

[52] F. Hawlitschek, B. Notheisen, and T. Teubner, "The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy," *Electron. Commerce. Res. Appl.*, vol. 29, pp. 50–63, 2018.

[53] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Security and privacy in the medical internet of things: A review," *Secur. Commun. Netw.*, vol. 2018, 2018, Art. no. 5978636.

[54] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun.; IEEE 14th Int. Conf. Smart City; IEEE 2nd Int. Conf. Data Sci. Syst.*, 2016, pp. 1392–1393.

[55] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "BlockChain: A distributed solution to automotive security and privacy," *IEEE Commun. Magazine*, vol. 55, no. 12, pp. 119–125, Dec. 2017.

[56] M. Mylrea and S. N. G. Gourisetti, "Blockchain: A path to grid modernization and cyber resiliency," in *Proc. North Amer. Power Symp.*, 2017, pp. 1–5.

[57] X. Huang, C. Xu, P. Wang, and H. Liu, "LNSC: A security model for electric vehicle and charging pile management based on blockchain ecosystem," *IEEE Access*, vol. 6, pp. 13565–13574, 2018.

[58] J. Leng *et al.*, "Makerchain: A blockchain with chemical signature for self-organizing process in social manufacturing," *J. Cleaner Prod.*, vol. 234, pp. 767–778, 2019.

[59] J. Leng *et al.*, "Digital twin-driven joint optimisation of packing and storage assignment in large-scale automated high-rise warehouse product-service system," *Int. J. Comput. Integr. Manuf.*, 2019. doi: 10.1080/0951192X.2019.1667032

[60] J. Leng *et al.*, "Blockchain-empowered sustainable manufacturing and product lifecycle management in industry 4.0: A survey," *Renew. Sustain. Energ. Rev.*, vol. 132, 2020, Art. no. 110112.

[61] J. Leng *et al.*, "Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model," *Robot. Comput.-Integr. Manuf.*, vol. 63, 2020, Art. no. 101895.

[62] J. Leng, P. Jiang, C. Liu, and C. Wang, "Contextual self-organizing of mass individualization process under social manufacturing paradigm: A cyber-physical-social system approach," *Enterprise Inf. Syst.*, vol. 14, no. 8, pp. 1124–1149, 2020.

[63] J. Leng and P. Jiang, "Dynamic scheduling in RFID-driven discrete manufacturing system by using multi-layer network metrics as heuristic information," *J. Intell. Manuf.*, vol. 30, pp. 979–994, 2019.

[64] Q. Liu *et al.*, "Digital twin-based designing of the configuration, motion, control, and optimization model of a flow-type smart manufacturing system," *J. Manuf. Syst.*, 2020, doi: 10.1016/j.jmsy.2020.04.012

[65] Q. Liu, H. Zhang, J. Leng, and X. Chen, "Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system," *Int. J. Prod. Res.*, vol. 57 no. 12, pp. 3903–3919, 2019.

[66] J. Leng, H. Zhang, D. Yan, Q. Liu, X. Chen, and D. Zhang, "Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop," *J. Ambient Intell. Humanized Comput.*, vol. 10, pp. 1155–1166, 2019.

[67] J. Leng *et al.*, "ManuChain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 182–192, Jan. 2020.

[68] H. Liu, Y. Zhang, and T. Yang, "Blockchain-Enabled security in electric vehicles cloud and edge computing," *IEEE Netw.*, vol. 32, no. 3, pp. 78–83, May/Jun. 2018.

[69] M. C. Lacity, "Addressing key challenges to making enterprise blockchain applications a reality," *MIS Quart. Executive*, vol. 17, pp. 201–222, 2018.

[70] A. Anjum, M. Sporny, and A. Sill, "Blockchain standards for compliance and trust," *IEEE Cloud Comput.*, vol. 4, no. 4, pp. 84–90, Jul./Aug. 2017.

[71] G. W. Peters, E. Panayi, and A. Chapelle, "Trends in cryptocurrencies and blockchain technologies: A monetary theory and regulation perspective," *J. Financial Perspectives*, vol. 3, pp. 92–113, 2015.

[72] R. Neisse, G. S. European, and I. Nai-Fovino, "A Blockchain-based approach for data accountability and provenance tracking," in *Proc. 12th Int. Conf. Avail. Rel. Secur.*, 2017, pp. 1–10.

[73] P. Yeoh, "Regulatory issues in blockchain technology," *J. Financial Regulation Compliance*, vol. 25, pp. 196–208, 2017.

[74] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innov.*, vol. 2, 2016, Art. no. 24.

[75] H. Wang, K. Chen, and D. Xu, "A maturity model for blockchain adoption," *Financial Innov.*, vol. 2, pp. 1–5, 2016.

[76] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.

[77] L. Zhu, Y. Wu, K. Gai, and K. R. Choo, "Controllable and trustworthy blockchain-based cloud data management," *Future Gener. Comp. Syst.*, vol. 91, pp. 527–535, 2019.

[78] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain – or – Rewriting history in bitcoin and friends," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2017, pp. 111–126.

[79] A. Deshpande, K. Stewart, L. Lepetit, and S. Gunashekar, "Distributed ledger technologies/blockchain: Challenges, opportunities and the prospects for standards," 2017. [Online]. Available: https://www.bsigroup.com/LocalFiles/zh-tw/InfoSec-newsletter/No201706/download/BSI_Blockchain_DLT_Web.pdf

[80] H. Kakavand, N. Kost De Sevres, and B. Chilton, "The blockchain revolution: An analysis of regulation and technology related to distributed ledger technologies," 2017. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2849251

[81] R. T. Ainsworth and A. Shact, "Blockchain (Distributed ledger technology) solves VAT fraud," 2016. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2853428

[82] J. Dai and M. A. Vasarhelyi, "Toward blockchain-based accounting and assurance," *J. Inf. Syst.*, vol. 31, pp. 5–21, 2017.

[83] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10179–10188, 2018.

[84] J. Dai, Y. Wang, and M. A. Vasarhelyi, "Blockchain: An emerging solution for fraud prevention," *CPA J.*, vol. 87, 2017, Art. no. 12.

[85] Y. Cai and D. Zhu, "Fraud detections for online businesses: A perspective from blockchain technology," *Financial Innov.*, vol. 2, pp. 1–10, 2016.

[86] S. Feng, W. Wang, Z. Xiong, D. Niyato, P. Wang, and S. S. Wang, "On cyber risk management of blockchain networks: A game theoretic approach," to be published, doi: 10.1109/TSC.2018.2876846.

[87] M. Muzammal, Q. Qu, and B. Nasrulin, "Renovating blockchain with distributed databases: An open source system," *Future Gener. Comp. Syst.*, vol. 90, pp. 105–117, 2019.

[88] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *Lecture Notes in Computer Science*. L. Y. Chen and H. P. Reiser Eds., 2017, vol. 10320, pp. 206–220.

[89] T. Le and M. W. Mutka, "CapChain: A privacy preserving access control framework based on blockchain for pervasive environments," in *Proc. IEEE Int. Conf. Smart Comput.*, 2018, pp. 57–64.

[90] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, 2015, pp. 180–184.

[91] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "FairAccess: A new Blockchain-based access control framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, pp. 5943–5964, 2016.

[92] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Proc. Europe MENA Cooperation Adv. Inf. Commun. Technologies*, 2016, pp. 523–533.

[93] A. Outchakoucht, H. Es-Samaali, and J. Philippe, "Dynamic access control policy based on blockchain and machine learning for the Internet of Things," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, pp. 417–424, 2017.

[94] J. Lee, "BIDaaS: Blockchain based ID as a service," *IEEE Access*, vol. 6, pp. 2274–2278, 2018.

[95] X. Fan, "Faster dual-key stealth address for blockchain-based Internet of Things systems," in *Proc. Int. Conf. Blockchain*, 2018, pp. 127–138.

[96] N. T. Courtois and R. Mercer, "Stealth address and key management techniques in blockchain systems," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, 2017, pp. 559–566.

[97] Y. Rahulamathavan, R. C. W. Phan, M. Rajarajan, S. Misra, and A. Kondoz, "Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption," in *Proc. IEEE Adv. Netw. Telecommun. Syst.*, 2017, pp. 1–6.

[98] A. Lewko and B. Waters, "New proof methods for attribute-based encryption: Achieving full security through selective techniques," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 180–198.

[99] C. Adams, "A privacy-preserving blockchain with fine-grained access control," *Secur. Privacy*, vol. 3, no. 2, 2020, Art. no. e97.

[100] S. Yaji, K. Bangera, and B. Neelima, "Privacy preserving in blockchain based on partial homomorphic encryption system for AI applications," in *Proc. IEEE 25th Int. Conf. High Perform. Comput. Workshops*, 2018, pp. 81–85.

[101] Iamtrask, "Building safe A.I.: A tutorial for encrypted deep learning," 2017, [Online]. Available: https://iamtrask.github.io/2017/03/17/safe-ai/.

[102] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. Thesis: Stanford: Stanford University, Dept. Comput. Sci., 2009.

[103] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 1978.

[104] L. Zhou, L. Wang, Y. Sun, and P. Lv, "BeeKeeper: A blockchain-based IoT system with secure storage and homomorphic computation," *IEEE Access*, vol. 6, pp. 43472–43488, 2018.

[105] J. Hsiao, R. Tso, C. Chen, and M. Wu, "Decentralized e-voting systems based on the blockchain technology," in *Proc. Int. Conf. Comput. Sci. Ubiquitous Comput.*, 2017, pp. 305–309.

[106] P. Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Comput. Surv.*, vol. 50, pp. 1–33, 2018.

[107] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Annu. Cryptol. Conf.*, 2010, pp. 465–482.

[108] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Proc. Topics Cryptol.*, 2002, pp. 244–262.

[109] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen, and Y. Tang, "An ID-Based linearly homomorphic signature scheme and its application in blockchain," *IEEE Access*, vol. 6, pp. 20632–20640, 2018.

[110] T. Li, W. Chen, Y. Tang, and H. Yan, "A homomorphic network coding signature scheme for multiple sources and its application in IoT," *Secur. Commun. Netw.*, vol. 2018, pp. 1–6, 2018.

[111] F. Benhamouda, S. Halevi, and T. Halevi, "Supporting private data on hyperledger fabric with secure multiparty computation," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2018, pp. 357–363.

[112] M. Andrychowicz, S. Dziembowski, D. Malinowski, and Ł. Mazurek, "Secure multiparty computations on bitcoin," *Commun. ACM*, vol. 59, pp. 76–84, 2016.

[113] M. J. Atallah and W. Du, "Research on secure multi-party computational geometry," in *Proc. Int. Conf. Inf. Comput. Appl.*, 2011, pp. 322–329.

[114] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Foundations Comput. Sci.*, 1982, pp. 160–164.

[115] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," 2015. [Online]. Available: https://arxiv.org/abs/1506.03471.

[116] PlatON, "PlatON: A high-efficiency trustless computing network," 2018. [Online]. Available: https://platon.network/.

[117] Y. Lindell, "Secure multiparty computation for privacy preserving data mining," in *Encyclopedia of Data Warehousing and Mining*, IGI Global, 2005, pp. 1005–1009.

[118] C. Zhao *et al.*, "Secure multi-party computation: Theory, practice and applications," *Inf. Sci.*, vol. 476, pp. 357–372, 2019.

[119] H. S. Galal and A. M. Youssef, "Verifiable sealed-bid auction on the ethereum blockchain," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2018, pp. 265–278.

[120] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," 2019. [Online]. Available: https://arxiv.org/abs/1908.04756.

[121] X. Yu, Z. Yan, and A. V. Vasilakos, "A survey of verifiable computation," *Mobile Netw. Appl.*, vol. 22, pp. 438–453, 2017.

[122] G. Danezis, C. Fournet, M. Kohlweiss, and B. Parno, "Pinocchio coin: Building zerocoin from a succinct pairing-based proof system," in *Proc. 1st ACM Workshop Lang. Support Privacy-Enhancing Technologies*, 2013, pp. 27–30.

[123] A. Kosba, C. Papamanthou, and E. Shi, "xJsnark: A framework for efficient verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 944–961.

[124] T. Kerber, "Verifiable computation on ethereum," 2012. [Online]. Available: https://project-archive.inf.ed.ac.uk/ug4/20170882/ug4_proj.pdf.

[125] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," 2018. [Online]. Available: https://arxiv.org/abs/1805.08541.

[126] D. Fu and L. Fang, "Blockchain-based trusted computing in social network," in *Proc. Blockchain-Based Trusted Comput. Soc. Netw.*, 2016, pp. 19–22.

[127] D. Fiore, "Verifiable delegation of computation on outsourced data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 863–874.

[128] W. Tsai, X. Bai, and L. Yu, "Design issues in permissioned blockchains for trusted computing," in *Proc. IEEE Symp. Serv.-Oriented Syst. Eng.*, 2017, pp. 153–159.

[129] S. Shetty, V. Red, C. Kamhoua, K. Kwiat, and L. Njilla, "Data provenance assurance in the cloud using blockchain," *SPIE Defense + Secur.*, vol. 2, pp. 1–11, 2017.

[130] A. Ramachandran and M. Kantarcioglu, "Using blockchain and smart contracts for secure data provenance management," 2017. [Online]. Available: https://arxiv.org/abs/1709.10000.

[131] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A Blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2017, pp. 468–477.

[132] D. K. Tosh, S. Shetty, X. Liang, C. A. Kamhoua, K. Kwiat, and L. Njilla, "Security implications of blockchain cloud with analysis of block withholding attack," in *Proc. 17th IEEE Int. Symp. Cluster Cloud Grid Comput.*, 2017, pp. 458–467.

[133] N. Kaaniche and M. Laurent, "A blockchain-based data usage auditing architecture with enhanced privacy and availability," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl.*, 2017, pp. 1–5.

[134] A. Lazarovich, "Invisible ink: Blockchain for data privacy," *Doctoral dissertation, Massachusetts Institute of Technology*, 2015.

[135] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017.

[136] T. Faisal, N. Courtois, and A. Serguieva, "The evolution of embedding metadata in blockchain transactions," in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–9.

[137] H. G. Do and W. K. Ng, "Blockchain-Based system for secure data storage with private keyword search," in *Proc. IEEE World Congr. Serv.*, 2017, pp. 90–93.

[138] M. Abdalla *et al.*, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," *J. Cryptol.*, vol. 21, pp. 350–391, 2008.

[139] M. Bellare, A. Boldyreva, and A. O. Neill, "Deterministic and efficiently searchable encryption," in *Proc. Annu. Int. Cryptol. Conf.*, 2007, pp. 535–552.

[140] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," 2017. [Online]. Available: https://arxiv.org/abs/1711.01030.

[141] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain," *IEEE Access*, vol. 6, pp. 31077–31087, 2018.

[142] R. Bost, "$\sum o\varphi o\varsigma$: Forward secure searchable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1143–1154.

[143] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. NDSS*, 2014, pp. 1–15.

[144] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 792–800.

[145] H. Li, H. Tian, F. Zhang, and J. He, "Blockchain-based searchable symmetric encryption scheme," *Comput. Elect. Eng.*, vol. 73, pp. 32–45, 2019.

[146] S. Hisayoshi, Y. Masayuki, and N. Ken, "Searchable encryption: A technology for supporting secure application," 2013. [Online]. Available: https://www.hitachi.com/rd/sc/story/searchable_encryption/index.html

[147] R. E. Bansarkhani and J. Sturm, "An efficient lattice-based multisignature scheme with applications to bitcoins," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2016, pp. 140–155.

[148] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2016, pp. 43–60.

[149] S. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2017, pp. 456–474.

[150] C. Yuan, M. Xu, and X. Si, "Research on a new signature scheme on blockchain," *Secur. Commun. Netw.*, vol. 2017, pp. 1–10, 2017.

[151] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Dependable Secure*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.

[152] S. Goldfeder *et al.*, "Securing bitcoin wallets via a new DSA/ECDSA threshold signature scheme," 2015. [Online]. Available: http://stevengoldfeder.com/papers/threshold_sigs.pdf

[153] C. Stathakopoulous and C. Cachin, "Threshold signatures for blockchain systems," 2017. [Online]. Available: https://dominoweb.draco.res.ibm.com/reports/.

[154] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2005, pp. 457–473.

[155] A. Ge, C. Ma, and Z. Zhang, "Attribute-based signature scheme with constant size signature in the standard model," *IET Inf. Secur.*, vol. 6, pp. 47–54, 2012.

[156] Y. Sun, R. Zhang, X. Wang, K. Gao, and L. Liu, "A decentralizing attribute-based signature for healthcare blockchain," in *Proc. 27th Int. Conf. Comput. Commun. Netw.*, 2018, pp. 1–9.

[157] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676–11686, 2018.

[158] D. Boneh, M. Drijvers, and G. Neven, "Compact Multi-signatures for smaller blockchains," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2018, pp. 435–464.

[159] A. Hari and T. V. Lakshman, "The internet blockchain: A distributed, tamper-resistant transaction framework for the Internet," in *Proc. 15th ACM Workshop Hot Topics Netw.*, 2016, pp. 204–210.

[160] L. Li *et al.*, "CreditCoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Trans. Intell. Transp.*, vol. 19, no. 7, pp. 2204–2220, Jul. 2018.

[161] E. Duffield and D. Diaz, "Dash: A privacycentric cryptocurrency," 2015. [Online]. Available: https://cdn.bitturk.com/whitepaper/dash.pdf.

[162] M. Möser and R. Böhme, "Anonymous alone? Measuring bitcoin's second-generation anonymization techniques," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, 2017, pp. 32–41.

[163] R. Mercer, "Privacy on the blockchain: Unique ring signatures," 2016. [Online]. Available: https://arxiv.org/abs/1612.01188.

[164] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Proc. ACISP*, 2004, pp. 325–335.

[165] S. Noether, "Ring signature confidential transactions for monero," 2015. [Online]. Available: https://eprint.iacr.org/2015/1098.

[166] Y. Zhu, R. Guo, G. Gan, and W. Tsai, "Interactive incontestable signature for transactions confirmation in bitcoin blockchain," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf.*, 2016, pp. 443–448.

[167] W. A. A. Torres *et al.*, "Post-Quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (Lattice RingCT v1.0)," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2018, pp. 558–576.

[168] B. Wang, J. Sun, Y. He, D. Pang, and N. Lu, "Large-scale election based on blockchain," *Procedia Comput. Sci.*, vol. 129, pp. 234–237, 2018.

[169] N. van Saberhagen, "CryptoNote v 2.0," 2013. [Online]. Available: https://cryptonote.org/.

[170] L. Axon, "Privacy-awareness in Blockchain-based PKI," 2016. [Online]. Available: https://ora.ox.ac.uk/.

[171] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *J. Cryptol.*, vol. 1, pp. 77–94, 1988.

[172] T. P. Pedersen, "Noninteractive and information-theoretic secure verifiable secret sharing," in *Proc. Conf. Annual Int. Cryptol.* 1991, pp. 129–140.

[173] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *IACR Cryptol. ePrint Archive*, vol. 46, pp. 1–83, 2018.

[174] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed E-Cash from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 397–411.

[175] E. Ben-Sasson *et al.*, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 459–474.

[176] N. Narula, W. Vasquez, and M. Virza, "zkLedger: Privacy-preserving auditing for distributed ledgers," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 65–80.

[177] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi, "Solidus: Confidential distributed ledger transactions via PVORM," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 701–717.

[178] T. Hardjono and A. Pentland, "Verifiable anonymous identities and access control in permissioned blockchains," 2019. [Online]. Available: https://arxiv.org/abs/1903.04584.

[179] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 839–858.

[180] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, pp. 161–174, 1991.

[181] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Scalable zero knowledge via cycles of elliptic curves," *Algorithmica*, vol. 79, pp. 1102–1160, 2017.

[182] H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica, "DIZK: A distributed zero knowledge proof system," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 675–692.

[183] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. IEEE 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectronics*, 2018, pp. 1545–1550.

[184] N. Chalaemwongwan and W. Kurutach, "State of the art and challenges facing consensus protocols on blockchain," in *Proc. Int. Conf. Inf. Netw.*, 2018, pp. 957–962.

[185] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 3–16.

[186] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 51% attack," *Appl. Sci.*, vol. 9, 2019, Art. no. 1788.

[187] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, pp. 95–102, 2018.

[188] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Annu. Int. Cryptol. Conf.*, 2017, pp. 357–388.

[189] W. Li, S. Andreina, J. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Proc. Int. Workshop Data Privacy Manage.*, 2017, pp. 297–315.

[190] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake," in *Proc. IACR Cryptol. ePrint Archive*, 2014, Art. no. 452.

[191] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proc. 1st Workshop Syst. Softw. Trusted Execution*, 2016, pp. 1–6.

[192] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[193] U. Feige and A. Shamir, "Zero knowledge proofs of knowledge in two rounds," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 526–544.

[194] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *Proc. Int. Symp. Stabilization Safety Secur. Distrib. Syst.*, 2017, pp. 282–297.

[195] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Proc. Annu. Cryptol. Conf.*, 2015, pp. 585–605.

[196] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," 2014. [Online]. Available: https://c3.coinlore.com/pdf/ripple-white-paper.pdf.

[197] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, and J. Kishigami, "Blockchain contract: Securing a blockchain applied to smart contracts," in *Proc. IEEE Int. Conf. Consumer Electron.*, 2016, pp. 467–468.

[198] Microsoft, "The coco framework technical overview," 2017. [Online]. Available: https://www.slideshare.net/WillyDevNET/coco-framework-whitepaper.

[199] M. Du, X. Ma, Z. Zhang, X. Wang, and Q. Chen, "A review on consensus algorithm of blockchain," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2017, pp. 2567–2572.

[200] T. Qiu, Y. Zhang, D. Qiao, X. Zhang, M. L. Wymore, and A. K. Sangaiah, "A robust time synchronization scheme for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3570–3580, Aug. 2018.

[201] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, pp. 398–461, 2002.

[202] X. Fan, "Scalable practical byzantine fault tolerance with short-lived signature schemes," in *Proc. 28th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, 2018, pp. 245–256.

[203] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukoli, "XFT: Practical fault tolerance beyond crashes," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, pp. 485–500, 2016.

[204] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst.*, 2017, pp. 1–5.

[205] G. Pîrlea and I. Sergey, "Mechanising blockchain consensus," in *Proc. 7th ACM SIGPLAN Int. Conf. Certified Programs Proofs*, 2018, pp. 78–90.

[206] L. Lamport, "Paxos made simple," *ACM Sigact News*, vol. 32, pp. 18–25, 2001.

[207] H. Howard, "ARC: Analysis of raft consensus," Bachelor of Arts: University of Cambridge, 2014.

[208] A. Adelsbach and A. Sadeghi, "Zero-knowledge watermark detection and proof of ownership," in *Proc. Int. Workshop Inf. Hiding*, 2001, pp. 273–288.

[209] D. Mazières, "The stellar consensus protocol: A federated model for internet-level consensus," 2015. [Online]. Available: https://www.stellar.org/.

[210] J. Kwon, "Tendermint: Consensus without mining," 2014. [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf.

[211] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," 2017. [Online]. Available: https://arxiv.org/abs/1707.01873.

[212] R. Henry, A. Herzberg, and A. Kate, "Blockchain access privacy: Challenges and directions," *IEEE Security Privacy*, vol. 16, no. 4, pp. 38–45, Jul./Aug. 2018.

[213] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-Based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1832–1843, Dec. 2017.

[214] H. Mayer, "ECDSA security in bitcoin and ethereum: A research survey,", 2016. [Online]. Available: https://www.semanticscholar.org/paper.

[215] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin P2P network," in *Proc. 2014 ACM SIGSAC Conf. Comput. Commun. Secur*, 2014, pp. 15–29.

[216] H. Watanabe and H. Fan, "A novel chip-level blockchain security solution for the Internet of Things networks," *Technologies*, vol. 7, 2019, Art. no. 28.

[217] K. Suankaewmanee, D. T. Hoang, D. Niyato, S. Sawadsitang, P. Wang, and Z. Han, "Performance analysis and application of mobile blockchain," in *Proc. Int. Conf. Comput. Netw. Commun.: Mobile Comput. Veh. Commun.*, 2018, pp. 642–646.

[218] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 375–392.

[219] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 129–144.

[220] C. Noyes, "Fast anti-malware by distributed blockchain consensus and feedforward scanning," 2016. [Online]. Available: https://arxiv.org/abs/1601.01405.

[221] L. D. Costa, A. Neto, B. Pinheiro, R. Araujo, A. Abelem, and W. Cordeiro, "DLCP: A protocol for securing light client operation in blockchains," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, 2018, pp. 1–6.

[222] B. Lee and J. Lee, "Blockchain-based secure firmware update for embedded devices in an internet of things environment," *J. Supercomputing*, vol. 73, pp. 1152–1167, 2017.

[223] J. Hochberg, K. Jackson, C. Stallings, J. F. McClary, D. DuBois, and J. Ford, "NADIR: An automated system for detecting network intrusion and misuse," *Comput. Secur.*, vol. 12, pp. 235–248, 1993.

[224] F. F. Wu and A. Monticelli, "Critical review of external network modelling for online security analysis," *Elect. Power Energy Syst.*, vol. 5, pp. 222–235, 1983.

[225] D. L. Goodhue and D. W. Straub, "Security concerns of system users: A study of perceptions of the adequacy of security," *Inf. Manage.*, vol. 20, pp. 13–27, 1991.

[226] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*, 2016, pp. 270–282.

[227] J. Wahab, "Privacy in blockchain systems," 2018. [Online]. Available: https://arxiv.org/abs/1809.10642.

[228] H. Chen, F. Wang, and D. Zeng, "Intelligence and security informatics for homeland security: Information, communication, and transportation," *IEEE Trans. Intell. Transp.*, vol. 5, no. 4, pp. 329–341, Dec. 2004.

[229] M. Bartoletti, B. Pes, and S. Serusi, "Data mining for detecting bitcoin ponzi schemes," in *Proc. Crypto Valley Conf. Blockchain Technol.*, 2018, pp. 75–84.

[230] V. Martínez, F. Berzal, and J. Cubero, "A survey of link prediction in complex networks," *ACM Comput. Surv.*, vol. 49, pp. 1–33, 2017.

[231] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM Comput. Surv.*, vol. 51, pp. 1–34, 2018.

[232] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Proc. Secur. Privacy Soc. Netw.*, 2012, pp. 197–223.

[233] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," 2015, [Online]. Available: https://arxiv.org/abs/1502.01657.

[234] T. Chen *et al.*, "Understanding ethereum via graph analysis," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1484–1492.

[235] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," in *Proc. Italian Conf. Cybersecurity*, 2017, pp. 1–10.

[236] J. Leng *et al.*, "A loosely-coupled deep reinforcement learning approach for order acceptance decision of mass-individualized printed circuit board manufacturing in industry 4.0," *J. Cleaner Prod.*, vol. 280, 2021, Art. no. 124405.

[237] J. Leng and P. Jiang, "A deep learning approach for relationship extraction from interaction context in social manufacturing paradigm," *Knowl.-Based Syst.*, vol. 100, pp. 188–199, 2016.

[238] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[239] P. J. Taylor, T. Dargahi, A. Dehghantanha, R. M. Parizi, and K. R. Choo, "A systematic literature review of blockchain cyber security," *Digit. Commun. Netw.*, vol. 6, pp. 147–156, 2020.

[240] E. Karafiloski and A. Mishev, "Blockchain solutions for big data challenges: A literature review," in *Proc. IEEE 17th Int. Conf. Smart Technol.*, 2017, pp. 763–768.

[241] P. Mikalef, I. O. Pappas, J. Krogstie, and M. Giannakos, "Big data analytics capabilities: A systematic literature review and research agenda," *Inf. Syst. E-Bus. Manage.*, vol. 16, pp. 547–578, 2018.

[242] P. Dikshit and K. Singh, "Efficient weighted threshold ECDSA for securing bitcoin wallet," in *Proc. ISEA Asia Secur. Privacy*, 2017, pp. 1–9.

[243] E. Gerjuoy, "Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers," *Amer. J. Phys.*, vol. 73, pp. 521–540, 2005.

[244] W. Yin, Q. Wen, W. Li, H. Zhang, and Z. Jin, "An anti-quantum transaction authentication approach in blockchain," *IEEE Access*, vol. 6, pp. 5393–5401, 2018.

[245] E. O. Kiktenko *et al.*, "Quantum-secured blockchain," *Quantum Sci. Technol.*, vol. 3, 2018, Art. no. 035004.

[246] P. Fremantle and P. Scott, "A survey of secure middleware for the internet of things," *PeerJ Comput. Sci.*, vol. 3, 2017, Art. no. e114.

[247] D. C. Derrick, A. C. Elkins, J. K. Burgoon, J. F. Nunamaker, and D. D. Zeng, “Border security credibility assessments via heterogeneous sensor fusion,” *IEEE Intell. Syst.*, vol. 25, no. 3, pp. 41–49, May/Jun. 2010.

[248] G. G. Gueta *et al.*, “SBFT: A scalable and decentralized trust infrastructure,” in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2019, pp. 568–580.

[249] G. Wang, Z. J. Shi, M. Nixon, and S. Han, “SMChain: A scalable blockchain protocol for secure metering systems in distributed industrial plants,” in *Proc. Int. Conf. Internet Things Des. Implementation*, 2019, pp. 249–254.

[250] M. Zamani, M. Movahedi, and M. Raykova, “RapidChain: Scaling blockchain via full sharding,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.

[251] Y. Yang, “LinBFT: Linear-communication byzantine fault tolerance for public blockchains,” 2018. [Online]. Available: https://arxiv.org/abs/1807.01829.

[252] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, “Blockchain based efficient and robust fair payment for outsourcing services in cloud computing,” *Inf. Sci.*, vol. 462, pp. 262–277, 2018.

[253] J. Li, J. Wu, and L. Chen, “Block-secure: Blockchain based scheme for secure P2P cloud storage,” *Inf. Sci*, vol. 465, pp. 219–231, 2018.

[254] P. K. Sharma, M. Chen, and J. H. Park, “A software defined fog node based distributed blockchain cloud architecture for ioT,” *IEEE Access*, vol. 6, pp. 115–124, 2018.

[255] J. Leng and P. Jiang, “Granular computing based development of service process reference models in social manufacturing contexts,” *Concurrent Eng., Res. Appl.*, vol. 25, pp. 95–107, 2017.

[256] J. Leng, Q. Chen, N. Mao, and P. Jiang, “Combining granular computing technique with deep learning for service planning under social manufacturing contexts,” *Knowl.-Based Syst.*, vol. 143, pp. 295–306, 2018.

[257] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K. R. Choo, “Blockchain: A panacea for healthcare cloud-based data security and privacy?,” *IEEE Cloud Comput.*, vol. 5, no. 1, pp. 31–37, Jan./Feb. 2018.

[258] I. García-Magariño, R. Lacuesta, M. Rajarajan, and J. Lloret, “Security in networks of unmanned aerial vehicles for surveillance with an agent-based approach inspired by the principles of blockchain,” *Ad Hoc Netw.*, vol. 86, pp. 72–82, 2019.

[259] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, “A comprehensive survey on fog computing: State-of-the-Art and research challenges,” *IEEE Commun. Surv. Tuts.*, vol. 20, no. 1, pp. 416–464, First Quarter 2018.

Jiewu Leng received the PhD degree from Xi’an Jiaotong University, in 2016. He is currently an associate professor with the Guangdong University of Technology, China. He has been a postdoctoral fellow at the City University of Hong Kong, under the support of “Hong Kong Scholars” program since 2018. His current research interests include blockchain and cyber-physical system. He has published more than 30 papers on the *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, etc.

Man Zhou received the BS degree from Guangxi University and the postgraduate degree from the Guangdong University of Technology, China, in 2020. His research interests include blockchain and cyber-physical systems.

J. Leon Zhao received the PhD degree from the Haas School of Business, UC Berkeley. He is currently a presidential chair professor of information systems with the School of Management and Economics, Chinese University of Hong Kong (Shenzhen). He was a chair professor during 2009-2020 and former head (during 2009-2015) of the Department of Information Systems at City University of Hong Kong. Before then, he was interim head and eller professor in MIS, University of Arizona. He has been associate editors of *ACM Transactions on MIS*, *Information Systems Research*, *IEEE Transactions on Services Computing, Decision Support Systems*, etc. He has co-edited more than 20 special issues in various academic journals. He received IBM Faculty Award, in 2005 and Changjiang Scholar Award, in 2009. His research is on information technology and management, including blockchain technology and applications, FinTech, and workflow technology.

Yongfeng Huang received the PhD degree in computer applications from Southeast University, in 2016. He joined the School of Computer Engineering of Jiangsu University of Technology, in China since 2016. Currently he is a visiting scholar with the City University of Hong Kong. His research interests include mobile opportunistic networks, blockchain, big data analysis, and big data privacy protection. His work has appeared in *Computer Communications*, *Transaction on Emerging Telecommunication Technologies*, and *Journal on Communications*.

Yiyang Bian received the PhD degrees from the City University of Hong Kong, and the University of Science and Technology of China. He is currently an assistant professor with the School of Information Management at Nanjing University. His research interests include IT switching, cloud computing, and data analytics. His work has appeared in *Transportation Research Part A*, *Industrial Management & Data Systems*, and several conferences of information systems.

A Secure and Flexible Blockchain-Based Offline Payment Protocol

Wanqing Jie, Wangjie Qiu, *Member, IEEE*, Arthur Sandor Voundi Koe, *Member, IEEE*, Jianhong Li, Yin Wang, Yaqi Wu, Jin Li, *Senior Member, IEEE*, and Zhiming Zheng

***Abstract*—Off-chain transactions seek to address the low on-chain scalability and enable blockchain-based payments over unreliable on-chain networks. The key problem with existing works is that they fail to balance security and flexibility in their designs. These studies would have been more useful if they could provide a sense of security without compromising their flexibility. We hypothesize that two offline parties having loosely synchronized clocks and channels with known bounded latency can conduct off-chain transactions while maintaining a high level of security and flexibility: we introduce a novel blockchain-based offline payment protocol that supports our hypothesis. Our work leverages on-chain smart contracts and offline wallet interactions to build resilience against intermittent on-chain connectivity. Our protocol achieves flexible and trusted computations with the use of platform-agnostic Trusted Execution Environments (TEEs) and open transactions. We empirically evaluate our design over the mainstream Intel Software Guard Extensions (SGX) and compare our protocol with state-of-the-art solutions. We found that our protocol attains high efficiency and exhibits an advanced level of security and flexibility in functionality. We evaluate our construction against several real-world attacks. We prove the security and robustness of our scheme based on a practical universally composable framework with synchronous settings. This work contributes to the existing knowledge of safe and user-friendly offline payment solutions for the blockchain technology.**

***Index Terms*—Blockchain, offline payment, smart contract, security, flexible, protocol.**

Manuscript received 12 January 2023; revised 12 July 2023; accepted 5 November 2023. Date of publication 13 November 2023; date of current version 15 January 2024. This work was supported by the National Key Research and Development Program of China under Grant 2021YFB2700300 and Beijing Natural Science Foundation Z230001. Recommended for acceptance by C. Li. *(Corresponding author: Wangjie Qiu.)*

Wanqing Jie, Wangjie Qiu and Zhiming Zheng are with the Institute of Artificial Intelligence, Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, Beijing 100191, China, also with Zhongguancun Laboratory, Beijing, China, also with the Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangdong 510006, China, and also with Health Blockchain Research Center, Binzhou Medical University, Yantai, Shandong 264003, China (e-mail: wanqing-jie@qq.com; wangjieqiu@buaa.edu.cn; zzheng@pku.edu.cn).

Arthur Sandor Voundi Koe is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China, and also with the Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangdong 510006, China (e-mail: 2517482859@qq.com).

Jianhong Li, Yin Wang, Yaqi Wu, and Jin Li are with the Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangdong 510006, China (e-mail: 1791269020@qq.com; 867042655@qq.com; 1550322596@qq.com; lijin@gzhu.edu.cn).

Digital Object Identifier 10.1109/TC.2023.3331823

I. Introduction

In the new global economy, the blockchain technology has become a key instrument in removing trusted intermediaries. The first proof of concept towards its practical use was put forward by the pseudonym Satoshi Nakamoto: the Bitcoin [1]. Ethereum [2], which supports the second most valuable cryptocurrency after Bitcoin, was developed with an intriguing property in mind: the ability to run smart contracts that satisfy Turing-completeness. What is striking is the growing amount of blockchain-based cryptocurrency projects [3] that leverage smart contracts to promote diverse use cases out of the sole financial realm. Governments, academicians and practitioners increasingly adopt the blockchain technology to address daily real-world problems in finance [4], e-health [5], cloud services [6], unmanned vehicle network [7], [8], etc.

Research in the field of blockchain technology considers security, scalability, and decentralization to be vital concepts. According to Vitalik's trilemma [9], existing blockchain implementations have yet to achieve these three properties simultaneously. To date, much study has concentrated on the application of blockchain technology in fully synchronous networks, rather than the robustness of blockchain solutions in adversarial networks featuring intermittent connectivity. Offline payments were introduced to promote transactions that take place offline and are reconciled with network connectivity. Off-chain transactions play a vital role in bringing about offline payments to blockchain-based scenarios.

Payment channels [10], [11], [12] stand as a significant contributory factor to the development of solutions towards layer-2 blockchain scalability issues. They leverage an off-chain network of channels to support offline payments. One criticism of much of the literature on payment channels is that the channels need to be pre-established and sufficient funds need to be locked in advance via a pay-to-multisig scheme (P2MS). Such requirements impede the flexibility of existing offline payment scenarios, introduce many security issues, and limit the wide adoption of blockchain technology in offline-first design for real-world applications. Payment channels would have been more relevant if they did not require to open on-chain settlement transactions (by locking up a chosen amount of cryptocurrency as a security deposit) and were not vulnerable to random failures and targeted attacks [13], [14]: channel exhaustion and node isolation attacks.

Previous published studies on offline payment solutions for blockchain lack standardized threat model and security model: such threat model should cover the various security threats in offline payment solutions for blockchain; such security model should guide the security analysis of blockchain-based offline payment schemes and assess the basic security requirements that are needed. To date, there has been no detailed investigation that simultaneously supports security and flexibility in existing blockchain-based offline payment constructions.

In this paper, we hypothesize that two offline parties, which experience intermittent on-chain connectivity, can engage in blockchain-based off-chain transactions without sacrificing security and flexibility: we introduce a secure and flexible protocol that realizes offline payments in blockchain-based solutions.

Our design builds upon the interactions between on-chain smart contracts and secure offline accounts. We transfer the on-chain trust from the blockchain anchor to offline accounts via the use of secure off-chain states, which are modified state channels that follow our protocol workflow. To support flexibility, we use open transactions that support coin redistribution together with instant settlement. To improve the security of our protocol, we leverage platform-agnostic Trusted Execution Environments (TEEs) that achieve trusted computations and secure offline wallet interactions in distributed settings. We provide a strong theoretical security analysis of our design over a universally composable framework that captures weakly synchronous off-chain interactions. We investigate several real-life attacks against our protocol: coin forgery, transaction data forgery, double-spend and double-deposit through transaction replay, man-in-the-middle (MITM) attacks and TEE related attacks. We evaluate the performances of our proposed construction over the Intel Software Guard Extensions (SGX).

Our contributions are as follows:

- **Effective trade-off between security and flexibility.** Different from previous studies, our scheme achieves a significant trade-off between security and flexibility in blockchain-based offline payment solutions.
- **Characterization of a comprehensive threat model.** We characterize a threat model that accounts for the various security issues in blockchain-based offline payment solutions, and prove the robustness of our construction under universal composability (UC) with synchronous settings.
- **Source code provisioning and evaluation methodology.** We analyze the source code lexicalities of our protocol and evaluate the performance based on specific metrics of real-world applications.

II. Related Work

This section revisits the literature on offline blockchain-based payments. Currently, research on the subject has been focusing on either security or flexibility: existing constructions fail to simultaneously meet the desired requirements of security and flexibility.

A. Offline Payment Constructions Based on Payment Channels

The concept of payment channel originates from the Lightning Network [15]: an off-chain expansion scheme of Bitcoin, which later evolved into more general off-chain channel schemes that exhibit Turing completeness. Two prominent examples of off-chain channel constructions are Raiden Network [16] and Counterfactual [17], which are based on the Ethereum blockchain.

Payment channels have become a de facto offline payment solution. In their latest work xLumi [11], Ying et al. described the specific workflow of offline wallets in payment channel based schemes: the payer and the payee must first create a channel online through a 2-of-2 multi-signature wallet, recharge the amount in the multi-signature wallet as offline spending and mortgage, then both parties can leverage the channel metadata to conduct offline transactions. Such metadata are broadcast on the chain for attestation, since the blockchain explicitly serves as the trust anchor. The two parties can check their respective account states to ensure the consistency of their interactions, both on and off the chain. They can opt to close the established channel.

Many recent works have made some improvements over the Lightning Network's payment channels. For example, Teechain [18], which is based on TEE trusted security hardware, empowers payment channel users to upload asynchronously to the chain. Teechain can tolerate read, write and dispute operations sent to the blockchain within the upper limit of the time window $\triangle$, which heightens its security and its efficiency. The schemes [12], [19] use one-way functions instead of digital signature schemes to reduce offline transaction fees. Nikolay Ivanov et al. implemented the VolgaPay payment system [20] to reduce the computational overhead of the original hash-based chain. They used a multi-hash chain-based micro-payment channel and leveraged polynomial price representation. Their work achieves security and efficiency trade-off towards payment transactions processing between merchants and customers.

Although offline payment solutions centered on payment channels are relatively mature and secure, the existing accounts fail to operate offline transactions without establishing a channel in advance. Existing payment channel constructions hardly support open transactions, which limits the flexibility of current designs.

B. Other Offline Payment Schemes

Recent offline payment protocols advocate flexible interactions between random payers and payees over intermittent on-chain connectivity. Dmitrienko A. et al. [21] proposed a classic three-phase protocol design for Bitcoin offline payment. In their protocol, the payer preloads online coins, recharges the offline wallet, and carries out offline payments with the payee. The payee performs online coin redemption and double-spend cancellation whenever the on-chain communication link sparkles. Such a scheme solves the double spending problem (launched by a malicious offline payer) through a tamper-proof wallet. It achieves the revocation of double-spend offline transactions and distributed offline wallets. Besides the fact that coin preloading results from earlier successful payer verification, Dmitrienko A. et al. [21] employ a time-based verification mechanism for transaction confirmation, which prevents coin forgery attacks. The follow-up work by Takahashi et al. [22] improved the coin preloading process by requiring to verify the payee. Such a scheme gets rid of the timestamp server to ensure the security of offline coin recharge.

None of the above two solutions realizes instant settlement of the payee's offline account. Under poor on-chain connectivity, the above works fail to verify the status of emitted transactions and to timely update the balance of offline accounts. Ensuring the correct account balance at any time is crucial to maintaining the coherence of future transactions.

Igboanusi et al. [23] implemented the offline payment framework PureWallet (PW) using Ethereum smart contracts. In their work, the token manager converts cryptocurrencies into other tokens of interest, and both the sender and receiver perform offline transactions through Near Field Communication (NFC). Such a paradigm fails to suit real-world application scenarios as the system lacks a token forgery detection mechanism: resulting in offline transactions' fail to meet the basic security requirements. In addition, the tokens in Igboanusi et al. [23] cannot be re-distributed: on-chain network availability is necessary to update the state of any account. Such a requirement limits the inherent flexibility that stems from deploying offline transactions. Wang et al. designed the MOBT wallet model [24]. They generate key pairs for offline wallets through the master public key property of hierarchical deterministic (HD) wallets. In their work, the wallets only need to store and back up the master private keys (not the individual private keys generated for each offline transaction). They aimed to address storage scalability issues where the size of the offline wallet grows with the storage of individual private keys, when multiple offline transactions are performed. Besides, their work utilizes an interactive signature protocol to thwart the Kleptographic attack [25] on offline wallets.

Although recent offline payment solutions support open transactions and promote their flexibility, they still fall short of basic security requirements. In those works, the flexibility threshold to meet an acceptable level of security remains relatively low. To our knowledge, existing accounts fail to meet a high level of security and flexibility simultaneously. This is the first time a blockchain-based offline payment protocol is proposed to mitigate such a limitation in the literature.

III. Systemic Overview

A. System Model

Our protocol enables the interactions between five main entities of interest: the offline payer, the offline payee, the trusted execution environment (TEE), the rich execution environment (REE) and the on-chain smart contract. In this work, we refer to the offline payer and the offline payee as users. Fig. 1 highlights the interactions between users and the blockchain network (on-chain smart contract), while Fig. 2 captures the set of interactions between the blockchain network (on-chain smart contract), the REE, and the TEE, as well as the set of operations performed by the TEE and the REE, respectively. The following lines provide insights into the different roles each entity plays in the proposed construction.

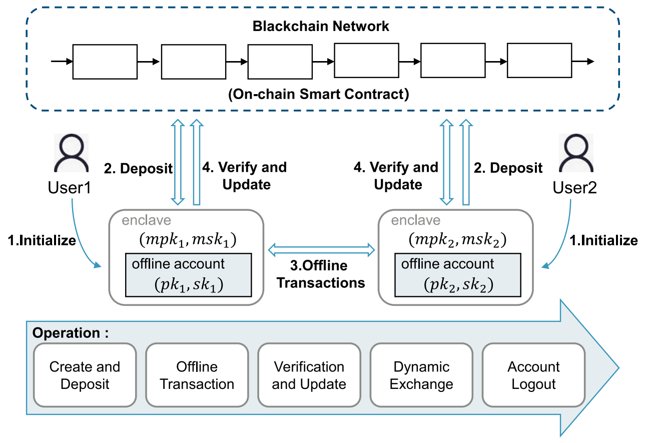


Fig. 1. Overview of our offline payment protocol.

(1) Users: Each user requires an offline account (a public/private key pair) to participate in our framework. Such an offline account possesses an address that is generated via the offline account's public key. The offline payer must perform an initial on-chain transaction that credits its offline account: there should be a confirmed on-chain transaction that deposits a strictly positive amount of cryptocurrency (for coherence) into the payer's offline account. Our protocol relaxes such a requirement on the offline payee, which is free from any mandatory initial on-chain deposit. In this work, users can deposit cryptocurrencies into offline accounts, as well as reconcile their off-chain and on-chain states during good on-chain connectivity. When the on-chain network becomes adversarial, users move off-chain. Our framework assumes that the offline payer initiates an offline transaction to the offline payee. Such an assumption is similar to many real-world payment protocols.

Our work requires the payer to embed its off-chain state metadata in every emitted offline transaction. Whenever the payee receives an offline transaction, it verifies the integrity and validity of the payer state using embedded metadata. Such metadata build upon the generation of an enclave measurement that relies on prior attestation from the on-chain smart contract. The offline payee updates its off-chain state if the verification succeeds. Our protocol empowers users to apply for account cancellation.

We require a weaker variant of synchrony between users to ensure input completeness and guaranteed termination. When the offline payer captures a receipt of transaction confirmation from the offline payee within the given lock time, it updates the balance of its off-chain state, and the offline transaction is considered successful. If no confirmation is received within the given lock time, the offline payer assumes that the transaction failed.

(2) TEE: The TEE operates in offline mode and disregards the on-chain connectivity state. It generates an offline account for the new user: a public/private key pair (pk, sk). It stores the off-chain state of the offline account and hosts the manufacturer's public key mpk. The TEE verifies the integrity of operations that aim to update the off-chain state.

(3) REE: The REE sits between the TEE and the on-chain smart contract and serves as an interface between both. It senses

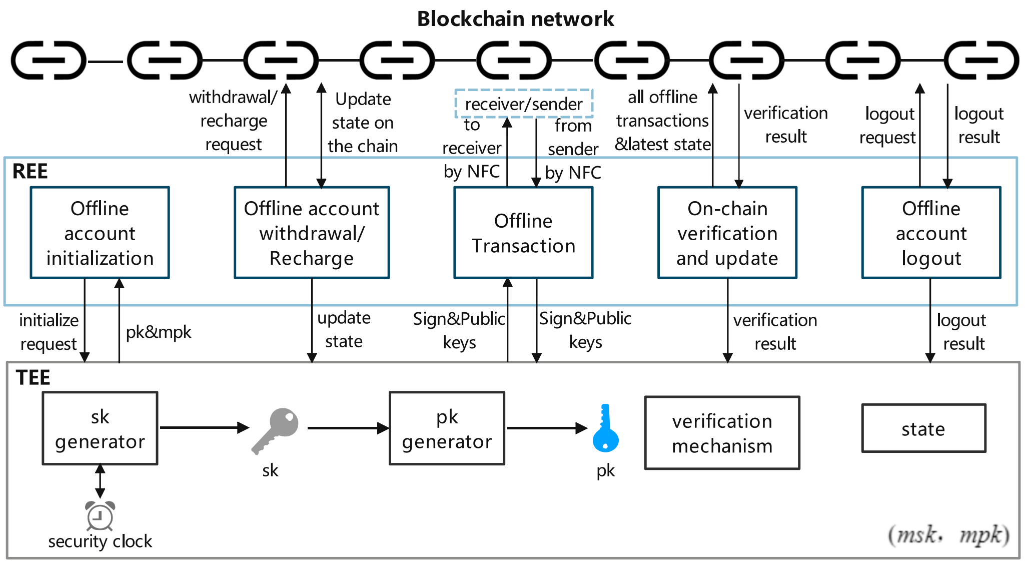


Fig. 2. Offline payment protocol workflow diagram.

the state of the on-chain connectivity; it initiates data transfer and calls to the on-chain smart contract during good on-chain network conditions. The REE provides functionalities that support the set of offline operations performed by users, such as the offline account initialization, the recharge and withdrawal of offline accounts, the processing of offline transactions, the on-chain verification of off-chain states, and the cancellation of user accounts. Section IV dives into the operations performed by REE and TEE in our construction.

(4) On-chain smart contract: Our scheme relies on an on-chain smart contract to update the on-chain state of users. In our protocol, every successful state transfer from on-chain to off-chain requires an attestation (a proof) from the on-chain smart contract: a transaction receipt. The on-chain smart contract is written in a contract-oriented language (COL) and performs verifiable calculations according to a predefined set of rules. The on-chain contract verifies the consistency and integrity of the received off-chain data during stable on-chain connectivity.

B. Threat Model and Security Assumptions

In this section, we briefly define the threat actors that participate in our protocol and state several assumptions that sustain the applicability of our novel framework.

1) Threat Model: In our proposed framework, the attacker is capable of carrying out coin forgery attacks, offline transaction data forgery, double-spend and double-deposit attacks, as well as man-in-the-middle attacks. The attacker may be a misbehaving participant, who adopts these malicious behaviors to obtain illicit gains.

Coin Forgery Attacks. Malicious users may launch coin forgery attacks during the initial on-chain deposit in their offline accounts. They may attempt to update the off-chain state without prior on-chain transaction confirmation; they may attempt to update the deposit balance in the offline wallet with a value greater than the corresponding one in the on-chain state.

Offline Transaction Data Forgery. The payer and the payee have incentives to tamper with data stored in the TEE. They might forge the amount, the generation time, or other data in offline transactions to get additional profits.

Double Spend and Double Deposit Attacks. A Malicious user may attempt sending the same offline transaction multiple times, or replay the same offline transaction to their TEE to launch double-spend and double-deposit attacks, respectively.

Man-in-the-Middle Attacks. Malicious users may try to impersonate the identities of honest users to shunt the nominal scenario in our construction.

2) Security Assumptions: Our scheme enforces the following security assumptions.

The on-chain operations exhibit correctness and soundness. The blockchain serves as trust anchor through transparent and immutable storage. This work assumes that on-chain functions are reliable.

The TEE is safe and reliable. Our protocol models the TEE as a secure abstraction that is platform-agnostic. The computations performed by the TEE are reliable.

Users operate on TEE-enabled platforms. This assumption is reasonable since recent mobile and computer devices are equipped with secure enclaves capable of trusted execution.

Cryptographic primitives are secure. This assumption is consistent with similar claims within the literature [26]. Primitives such as hash functions and digital signatures are secure and exhibit platform independence in this work.

The communication between users is reliable. Channels between users are weakly synchronous and authenticated: offline stakeholders possess loosely synchronized clocks and share authenticated channels with known bounded latency [27].

C. Design Goals

The overall goal of our manuscript is to design an efficient and secure offline payment protocol that achieves a significant trade-off between security and flexibility. The following lines depict the specific objectives we pursue in this work.

(1) Secure blockchain-based offline payment protocol. We aim to achieve a secure construction that exhibits the following properties:

- **Coins and data unforgeability:** Our protocol should prevent malicious users from launching coin forgery attacks, and users should fail to forge transaction data in their offline wallets.
- **Consistency:** The state of unconfirmed on-chain transactions should not be considered valid in offline settings. An offline payee should be able to independently verify the authenticity of an offline transaction on-premises.
- **Double-spend, double-deposit, and man-in-the-middle attacks resistance:** Our scheme should thwart double-spend and multiple deposit initiatives from the same offline transaction. Our scheme should thwart offline identity spoofing and deter man-in-the-middle attacks.

(2) Flexible blockchain-based offline payment protocol. We aim to achieve a flexible protocol that exhibits the following properties:

- **Open transactions:** Our protocol should support open transactions, which overlook prior commitment to an on-chain multi-signature account whenever the users engage into off-chain transactions.
- **Coin redistribution:** Our protocol aims to achieve coin redistribution through divisible cryptocurrency: users can transact divisible units of cryptocurrency without the need for a change address, as opposed to the Bitcoin system.
- **Instant settlement and platform-agnostic design:** The offline payee should update its offline account balance within the shortest possible delay. Our protocol should uphold deployment across different TEE platforms.

(3) Offline transaction atomicity. An offline commit should correctly update user offline accounts with non-repudiation. An offline abort should revert the user's offline state to its previous set of values with coherence.

(4) Optimistic responsiveness. To achieve a high degree of flexibility, our protocol should capitalize on-chain weaker synchrony, and strive during on-chain asynchrony.

IV. Protocol Design

We design our framework under universal composability (UC) with weak synchronous securities in the hybrid model: we express the real-world protocol as $\mathcal{P}$ and the ideal protocol as $\mathcal{F}$.

To specify our construction, we follow the interesting methodology of Camenish et al. [28], which provides a security framework for UC on top of the inexhaustible interactive Turing machine (IITM) model [29]. We specify our protocol via a set of machines. Each machine implements one or many roles (e.g. payer, payee). The running instance of a machine manages one or several entities.

We consider a player set P where each party $p_i \in P$ possesses a unique id pid_i, and supports one or many roles $role_i$. Players interact within sessions identified by a session id sid_i. An entity $(pid_i, sid_i, role_i)$ characterizes an instance of the role $role_i$ by party p_i within session id sid_i.

A. Participating Entities

Our system consists of the following types of interactive Turing machines (ITM): the environment $\mathcal{Z}$, the adversary $\mathcal{A}$, the real protocol $\mathcal{P}$, and the ideal functionality $\mathcal{F}$, which exhibits many ideal sub-functionalities.

The environment $\mathcal{Z}$. In our framework, the environment $\mathcal{Z}$ is the master meaning the first machine to run. $\mathcal{Z}$ is a mixed environment, both responsive and unresponsive. When responsive, $\mathcal{Z}$ must answer certain types of messages immediately: restricting messages [28]. The environment $\mathcal{Z}$ in this work is universally bounded: we place an upper bound on the amount of available computation power and storage capacity.$\mathcal{Z}$ provides initial parameters to all parties running the protocol under the common reference string (CRS) model [30].

The ideal functionality $\mathcal{F}$. Our work implements a single ideal functionality $\mathcal{F}$ that extends to several ideal sub-functionalities.

To achieve weak synchronous security in the UC (i.e., input-completeness and guaranteed termination), we leverage two ideal sub-functionalities depicted in [27]: $\mathcal{F}^{\delta}_{BD-SMT}$, which characterizes authenticated channels with known bounded-delay δ and eventual delivery properties, such that messages leading to an accumulated delay $T > \delta$ are ignored; $\mathcal{F}_{CLOCK}$, which establishes parties with loosely synchronized clocks enforced by a known upper bound σ on the maximum clock-drift, and ignores notifications from corrupted parties. To achieve authenticated and secure bounded-delay channel, this work assumes that $\mathcal{F}^{\delta}_{BD-SMT} = \{\mathcal{F}^{\delta}_{BD-SEC}$ [27], $\mathcal{F}^{\delta}_{BD-AUTH}$ [27]$\}$. To establish real-time delivery among players, we set $\delta = 1$.

To manage on-chain operations, we employ the ideal distributed ledger sub-functionality $\mathcal{F}_{ledger}$ [31]. $\mathcal{F}_{ledger}$ operates on chain and manages the blockchain initialization (i.e. to generate the genesis block), requests to read the global on-chain state, and updates to the on-chain state. It captures as well the full support for smart contract execution, the on-chain registration of offline parties, the consensus protocols, and other experimental on-chain operations to be set in the future.

To abstract and manage the TEEs, we employ the ideal sub-functionality $\mathcal{G}_{att}$ that captures all the platforms equipped with attested execution secure processors [32]. A secure processor embeds an internal source of randomness and stores the manufacturer-generated public/secret key pair (mpk/msk).$\mathcal{G}_{att}$ allows the installation of a new enclave using the command install, which loads the enclave program $prog_{enclave}$. To call the functions in $prog_{enclave}$, a player sends the command resume to $\mathcal{G}_{att}$.

As on-chain operations must be accessible to all participants, this work implements the ideal sub-functionality $\mathcal{F}_{ledger}$ using a single machine over a single instance that manages all entities. Such machine $\mathcal{M}_{ledger}$ implements the role ledger and concretizes the ideal sub-functionality $\mathcal{F}_{ledger}$.

We realize the ideal sub-functionality $\mathcal{G}_{att}$ through a single machine $\mathcal{M}_{install,resume}$ that spawns multiple instances. Each instance of $\mathcal{M}_{install,resume}$ manages exactly one entity. $\mathcal{G}_{att}$ supports a diverse set of operations that are detailed in [32].

Algorithm 1: Ideal Functionality $\mathcal{F}$

```
For each $p_i : (pk_i, ski) \leftarrow\$ \Sigma.genKey(1^n)$,
  $state_{offchain_i} = state^0_{offchain_i}$, $addr_{pk_i} \leftarrow G_{att}.H(pk_i)$

function: onchainDeposit (tx_deposit, withdraw: boolean):
leak ($tx_{deposit}$, ($pk_{ledger}$, $sid_{ledger}$, ledger)) to $\mathcal{A}$
await $(state_{onchain} = tx^{receipt}_{deposit} \mid \perp) from \mathcal{A}$
verify $state_{onchain}$ abort if $\perp$
verify $tx^{receipt}_{deposit} : status$ abort if False
leak(updateState, "deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: transfer ($b_0$, $addr_{pk_2}$):
leak ("transfer", $b_0$, $addr_{pk_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$
await $tx_{off}$ from $\mathcal{A}$
leak(transfer, $tx_{off}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: transfer ($tx_{off}$):
leak ($tx_{off}$, Payee ($pk_2$, $sid_2$, payee)) to $\mathcal{A}$
await ($ack_{win_2} \mid ack_{fail_2}$) from $\mathcal{A}$
if $ack_{win_2}$:
  leak ("save", $ack_{win_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$
  leak(updateState, "pay", $ack_{win_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$
if $ack_{fail_2}$:
  leak(updateState, "abort", $ack_{fail_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: updateState ("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw):
leak("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: updateState ("pay", $ack_{win_2}$):
leak("pay", $ack_{win_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: updateState ("abort", $ack_{fail_2}$):
leak("abort", $ack_{fail_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: leak():
leak("leakState",($pk_i$, $sid_i$, (payer | payee))) to $\mathcal{A}$
if ($pk_i$, $sid_i$, (payer | payee)) is honest:
  await ($\sigma^i_{att}$) from $\mathcal{A}$
if ($pk_i$, $sid_i$, (payer | payee)) is corrupted:
  await ($\sigma^i_{att}$, $\{state^i_{offchain}\}$ without $\{msk_i\}$) from $\mathcal{A}$

function: receive($tx_{off}$):
parse $tx_{off}$ as ($pk_1$, $addr_{pk_1}$, $addr_{pk_2}$, $b_0$, $T_{gen}$, $\sigma^1_{att}$, $mpk_1$, $\sigma 1$)
if accept $b_0$:  leak("receive", $tx_{off}$, ($pk_2$, $sid_2$, payee)) to $\mathcal{A}$
  await $ack_{win_2} \mid ack_{fail_2}$ from $\mathcal{A}$
  leak ($ack_{win_2} \mid ack_{fail_2}$, Payer ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$
if refuse $b_0$:  leak("reject", $tx_{off}$, ($pk_2$, $sid_2$, payee)) to $\mathcal{A}$
  await $ack_{fail_2}$ from $\mathcal{A}$
  leak ($ack_{fail_2}$, Payer ($pk_1$, $sid_1$, payer)) to $\mathcal{A}$

function: reconciliation():
leak("onchainUpdate", ($pk_i$, $sid_i$, (payer | payee)) to $\mathcal{A}$
await ($tx^1_{off_{j|\forall j \in \{1..n\}}}$, $ack^j_{win_2}$, $ack^j_{fail_2}$, $\sigma^i_{att}$) from $\mathcal{A}$
leak (($tx^1_{off_{j|\forall j \in \{1..n\}}}$, $ack^j_{win_2}$, $ack^j_{fail_2}$, $\sigma^i_{att}$), ($pk_{ledger}$, $sid_{ledger}$,
  ledger)) to $\mathcal{A}$
await $(state_{onchain} = tx^{receipt}_{reconcile} \mid \perp) from \mathcal{A}$
verify $state_{onchain}$ abort if $\perp$
if $tx^{receipt}_{reconcile} : status$ is False:
  $\mathcal{F}_{ledger}$ freezes ($addr_{pk_i}$, $pk_i$)
  leak("freeze", $tx^{receipt}_{reconcile}$, ($pk_i$, $sid_i$, (payer | payee)) to $\mathcal{A}$
if $tx^{receipt}_{reconcile} : status$ is True:
  $\mathcal{F}_{ledger}$ updates $state_{onchain_i}$
  leak("finalize", $tx^{receipt}_{reconcile}$, ($pk_i$, $sid_i$, (payer | payee))) to $\mathcal{A}$

function: logout():
leak(reconciliation, ($pk_i$, $sid_i$, (payer | payee))) to $\mathcal{A}$
await $tx^{receipt}_{reconcile}$ from $\mathcal{A}$
parse $tx^{receipt}_{reconcile}$ as $tx^{receipt}_{reconcile}:status$
if status is True: leak("logout", ($pk_i$, $sid_i$, (payer | payee))) to $\mathcal{A}$ abort if False
await $tx_{logout}$ from $\mathcal{A}$
parse $tx_{logout}$ as ($pk_i$, $addr_{pk_i}$, $addr_{pk_x}$, $state_{offchain_i}$:$balance$, $T_{gen}$,
  $\sigma i$, ($\sigma^i_{att}$, $mpk_i$))
leak ($tx_{logout}$, ($pk_{ledger}$, $sid_{ledger}$, ledger)) to $\mathcal{A}$
await $(state_{onchain} = tx^{receipt}_{logout} \mid \perp) from \mathcal{A}$
verify $state_{onchain}$ abort if $\perp$
verify $tx^{receipt}_{logout} : status$ abort if False
leak("clean", $tx_{logout}$, $tx^{receipt}_{logout}$,($pk_i$, $sid_i$, (payer | payee))) to $\mathcal{A}$
```

Algorithm 1: Ideal Functionality $\mathcal{F}$

```
(1) On receive ("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw):
parse $tx_{deposit}$ as ($pk_{pid}$, $pk_{caller}$, $addr_{pk_{caller}}$, $\sigma$, $b_0$)
parse $tx^{receipt}_{deposit}$ as ($status$)
verify $\sigma$ and $status$, abort if False
if withdraw False:
  $state_{offchain_i} : balance+ = b_0$
if withdraw True:
  $state_{offchain_i} : balance- = b_0$
append ($tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw) to Storage[$\mathcal{P}_i$]

(2) On receive ("save", $ack_{win_2}$):
store $ack_{win_2}$

(3) On receive ("pay", $ack_{win_2}$):
parse $ack_{win_2}$ as ($pk_2$,
  $tx_{off} = (pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1)$, $\sigma_2$,
  $status_{tx_{off}} = ok$)
verify ($\sigma_2$, $status_{tx_{off}} = ok$, $\sigma^1_{att}$, $\sigma 1$) and abort if False
$state_{offchain_1} : balance- = b_0$

(4) On receive ("receive", $tx_{off}$):
parse $tx_{off}$ as ($pk_1$, $addr_{pk_1}$, $addr_{pk_2}$, $b_0$, $T_{gen}$, $\sigma^1_{att}$, $mpk_1$, $\sigma 1$)
if verify ($\sigma_1$, $\sigma^1_{att}$) is True:
  build and store $ack_{win2}$; $state_{offchain_2} : balance+ = b_0$
  leak $ack_{win2}$ to $\mathcal{A}$
if verify ($\sigma_1$, $\sigma^1_{att}$) is False:
  build $ack_{fail2}$
  leak $ack_{fail2}$ to $\mathcal{A}$

(5) On receive ("transfer", $b_0$, $addr_{pk_2}$) :
build $tx_{off}$ as ($pk_1$, $addr_{pk_1}$, $addr_{pk_2}$, $b_0$, $T_{gen}$, $\sigma^1_{att}$, $mpk_1$, $\sigma 1$)
leak $tx_{off}$ to $\mathcal{A}$

(6) On receive ("logout") :
build $tx_{logout}$ as ($pk_i$, $addr_{pk_i}$, $addr_{pk_x}$, $state_{offchain_i}$ : $balance$, $T_{gen}$,
  $\sigma^i_{att}$, $mpk_i$, $\sigma_i$)
leak $tx_{logout}$ to $\mathcal{A}$

(7) On receive ("reject", $tx_{off}$):
  build $ack_{fail2}$
  store $ack_{fail2}$
  leak $ack_{fail2}$ to $\mathcal{A}$

(8) On receive ("abort", $ack_{fail_2}$):
parse $ack_{fail_2}$ as ($pk_2$, $tx_{off}$ =($pk_1$, $addr_{pk_1}$, $addr_{pk_2}$, $b_0$, $T_{gen}$, $\sigma^1_{att}$,
  $mpk_1$, $\sigma 1$), $\sigma_2$, $status_{tx_{off}} = fail$)
verify ($\sigma_2$, $status_{tx_{off}} = fail$, $\sigma^1_{att}$, $\sigma 1$) and abort if False
store $ack_{fail_2}$

(9) On receive ("clean", $tx_{logout}$, $tx^{receipt}_{logout}$):
verify $tx_{logout}$:$\sigma$ and $tx^{receipt}_{logout}$:$status$ abort if False
delete($pk_i$, $sk_i$)

(10) On receive ("leakState"):
if honest: leak $\sigma_{att}$ to $\mathcal{A}$
if corrupted: leak ($\sigma_{att}$, ($state_{offchain}$ without $msk$)) to $\mathcal{A}$

(11) On receive ("finalize", $tx^{receipt}_{reconcile}$):
leak $tx^{receipt}_{reconcile}$ to $\mathcal{A}$
parse $tx^{receipt}_{reconcile}$ as $tx^{receipt}_{reconcile}$:$status$
if $tx^{receipt}_{reconcile}$:$status$ is True:
  free storage ($tx^1_{off_{j|\forall j \in \{1..n\}}}$, $ack^j_{win_2}$, $\sigma^i_{att}$)
if $tx^{receipt}_{reconcile}$:$status$ is False: abort

(12) On receive ("onchainUpdate"):
collect params as ($tx^1_{off_{j|\forall j \in \{1..n\}}}$, $ack^j_{win_2}$, $ack^j_{fail_2}$ $\sigma^i_{att}$)
leak params to $\mathcal{A}$
```

This work extends $\mathcal{G}_{att}$ with the ability to sign messages, verify signatures, and hash data under the random oracle model. Algorithm 1 portrays the security objective of our offline protocol in the ideal functionality $\mathcal{F}$.

The Adversary $\mathcal{A}$ and The Corruption Model. Our work adopts the static corruption model, where the dummy adversary $\mathcal{A}$ corrupts a subset of entities after their trusted initialization [28].

We consider an entity as corrupted if at least one of its subroutines is corrupted, rather than requiring all subroutines to be corrupted. Our work abstracts the corruption protocol to be used and insights to design an interesting corruption protocol are available in [28].

When an entity becomes corrupted, the internal state of such entity is leaked to the adversary that gains full control over the corrupted entity. In this work, $\mathcal{G}_{att}$ responds to a specific restricting message (**leakState**) that requests its complete internal state. We adopt the transparent enclave model $\hat{\mathcal{G}}_{att}$ [33] that leaks the full state of a corrupted player to the dummy adversary $\mathcal{A}$, while keeping the manufacturer signing key msk secret.

We feature a malicious adversary that instructs corrupted parties to send several messages on its behalf. We match those messages to the set of offline attacks we investigate in this manuscript: coin forgery, data tampering, double-spend, double-deposit, and man-in-the-middle attacks.

Algorithm 2: Setup sub-Protocol $\mathcal{P}_{setup}$

```
On receive ("install"):
For each instance $G^j_{att}$: $(mpk_j, msk_j) := \Sigma.genKey(1^n)$

On receive ("init"):
$p_i : (pk_i, ski) := \Sigma.genKey(1^n)$
$mpk_i := \mathcal{G}_{att}.getpk()$
$addr_{pk_i} := G_{att}.H(pk_i)[20]$
return $(pk_i, mpk_i, addr_{pk_i})$
```

THE OFFLINE PAYMENT PROTOCOL $\mathcal{P}$. We model our offline payment protocol as a real protocol $\mathcal{P}$ that exploits four ideal sub-functionalities as subroutines: $\mathcal{G}_{att}$, $\mathcal{F}^{\delta=1}_{BD-SMT}$, $\mathcal{F}_{CLOCK}$, and $\mathcal{F}_{ledger}$. $\mathcal{P}$ embodies three roles: setup, payer, and payee. We leverage the digital signature scheme Σ and the random oracle hash function H both embedded in $\mathcal{G}_{att}$. We illustrate our overall framework in Fig. 1.

To enable the secure generation and global distribution of system initial parameters, we implement the setup role as a single machine that spawns a single instance over multiple entities. We implement the roles payer and payee as two separate machines. We require every instance of those machines to handle exactly one entity. The instances above characterize an actual implementation of our protocol and each new run leads to a new program instance.

Setup role. The setup sub-protocol $\mathcal{P}_{setup}$ manages the initialization of players. It leverages system parameters analog to the global common reference string (GCRS) model.

During the execution of $\mathcal{P}_{setup}$, parties send the install command to their instance of $\mathcal{M}_{install,resume}$ to load the enclave program $prog_{enclave}$, and obtain a manufacturer key pair (mpk, msk). Parties send the init command to $\mathcal{M}_{install,resume}$ to generate an off-chain account with a unique public/private key pair (pk_{pid}, sk_{pid}), generate an account address based on pk_{pid} and initialize the off-chain state $state_{off-chain}$ with the value $state^0_{off-chain}$.

Algorithm 2 reveals the details of the sub-protocol $\mathcal{P}_{setup}$ in our construction.

Payer role. We define several sub-routines of the sub-protocol $\mathcal{P}_{payer}$ that aim at the following four operations: onchainDeposit, transfer, updateState, and leak. We combine existing operations to provide two additional functionalities for the role payer: withdraw and recharge. Algorithm 3 reveals the details of the sub-protocol $\mathcal{P}_{payer}$ in our scheme.

$\mathcal{P}_{payer}$ leverages $\mathcal{F}^{\delta=1}_{BD-SMT}$ and $\mathcal{F}_{CLOCK}$ to query a party through the authenticated channel, and wait for a guaranteed response within the threshold δ. Our scheme can be extended to support any arbitrary value of δ.

The onchainDeposit subroutine first checks the on-chain connectivity. It exploits the restricting message **checkConnection** sent to $\mathcal{M}_{ledger}$, which returns a Boolean value.

When **checkConnection** returns True, the onchainDeposit subroutine leverages $\mathcal{M}_{ledger}$ to send the input transaction $tx_{deposit}$ to the on-chain smart contract address. $tx_{deposit}$ includes the account address generated from pk_{pid} as the receiver address, a caller address, the caller's public key, the caller's signature σ, and the required cryptocurrency amount b_0. We consider σ was generated using $\mathcal{F}_{ledger}$ based on the private key of the on-chain contract caller. Such a caller, the sender in $tx_{deposit}$, must possess an existing on-chain account filled with cryptocurrencies.

The onchainDeposit subroutine queries the status of $tx_{deposit}$ using the algorithm **checkStatus(**$tx_{deposit}$**)** over $\mathcal{M}_{ledger}$. **checkStatus(**$tx_{deposit}$**)** aims to return the current on-chain state $state_{onchain}$ that is either none or the transaction receipt $tx^{receipt}_{deposit}$, after validation from the on-chain contract $contract_{onchain}$ (i.e., $balance(calleraddress) >= value + fees$). Based on the status in $tx^{receipt}_{deposit}$, the onchainDeposit subroutine can call the subroutine updateState that sends the message deposit to $\mathcal{M}_{install,resume}$ with $tx_{deposit}$ and $state_{onchain}$ as parameters. To support withdrawal and recharge operations, we leverage a Boolean variable $withdraw$ as input parameter to the onchainDeposit subroutine.

The transfer subroutine leverages the amount of cryptocurrency b_0 to be sent to the payee, and the payee's address $addr_{pk_2}$, which was generated through the payee's public key pk_2. The transfer subroutine sends the command transfer to $\mathcal{M}_{install,resume}$ with b_0 and $addr_{pk_2}$ as parameters. $\mathcal{M}_{install,resume}$ checks that the balance of the local off-chain state $state_{offchain_1}$:$balance$ is greater than b_0. It builds an offline transaction tx_{off} that includes pk_1, the address of the payer $addr_{pk_1}$, $addr_{pk_2}$, b_0, the timestamp T_{gen}, the enclave measurement σ^1_{att}, mpk_1, and the offline transaction signature σ_1.To generate σ^1_{att}, $\mathcal{M}_{install,resume}$ leverages pk_1, $state_{offchain_1}$:$balance$, and msk_1 for enclave signature. To generate σ_1, $\mathcal{M}_{install,resume}$ signs tx_{off} using sk_1.The transfer subroutine forwards tx_{off} to the payee's machine instance and waits for a guaranteed response: tx_{off} can succeed or fail. We define a failed offline transaction as one that was rejected by the payee. The payee generates an acknowledgment of failure ack_{fail_2} whenever it rejects an offline transaction. The payee generates an acknowledgment of success ack_{win_2} whenever it accepts an offline transaction. ack_{fail_2} contains pk_2, tx_{off}, $status_{tx_{off}}$=fail, and the signature σ_2 over $\{pk_2, tx_{off}, status_{tx_{off}} = fail\}$; ack_{win_2} contains pk_2, tx_{off}, $status_{tx_{off}} =$ ok, and the signature σ_2 over $\{pk_2, tx_{off}$, $status_{tx_{off}} =$ ok$\}$.

If the transfer subroutine receives an acknowledgment of failure ack_{fail_2} from the payee, it invokes the updateState subroutine that sends the command abort to $\mathcal{M}_{install,resume}$ together with ack_{fail_2} as the parameter. If the transfer subroutine receives an acknowledgement of success ack_{win_2} from the payee: it sends the command save to $\mathcal{M}_{install,resume}$ with ack_{win_2} as parameter; it invokes the updateState subroutine that sends the command pay to $\mathcal{M}_{install,resume}$ with ack_{win_2} as parameter. Following the command save, $\mathcal{M}_{install,resume}$ stores ack_{win_2} locally.

The updateState subroutine aims to update the balance of $state_{offchain_1}$. It sends three types of commands to $\mathcal{M}_{install,resume}$: deposit, pay, and abort. The command deposit serves to update $state_{offchain_1}$:$balance$ after an on-chain deposit. It takes $tx_{deposit}$ and $state_{onchain}$ as input parameters. The command deposit instructs $prog_{enclave}$ to verify σ in $tx_{deposit}$ and to check the on-chain status of $tx_{deposit}$ using $state_{onchain}$. If both verification processes succeed, $\mathcal{M}_{install,resume}$ credits $state_{offchain_1}$:$balance$ by b_0.

Algorithm 3: Payer sub-Protocol $\mathcal{P}_{payer}$

```
$state_{offchain_1} := state^0_{offchain_1}$
Storage[$p_1$] $:= \phi$

function: onchainDeposit ($tx_{deposit}$, withdraw:bool):
$state_{onchain}$ := upload $tx_{deposit}$ to $\mathcal{M}_{ledger}$
assert $state_{onchain} \neq \perp$
assert $(state_{onchain} := tx^{receipt}_{deposit}) \rightarrow status \neq$ False
call updateState ("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw:bool, $(pk_1, sid_1, payer)$)
call leak()

function: transfer ($b_0$, $addr_{pk_2}$):
$tx_{off}$ = upload ("transfer", $b_0$, $addr_{pk_2}$, $(pk_1, sid_1, payer)$) to $\mathcal{M}^1_{install,resume}$
call transfer($tx_{off}$, $(pk_1, sid_1, payer)$)
call leak()

function: transfer ($tx_{off}$):
output := upload ($tx_{off}$, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}_{payee}$
if output == $ack_{win_2}$:
  upload ("save", $ack_{win_2}$, $(pk_1, sid_1, payer)$) to $\mathcal{M}^1_{install,resume}$
  call updateState ("pay", $ack_{win_2}$, ($pk_1$, $sid_1$, payer))
if output == $ack_{fail_2}$:
  call updateState ("abort", $ack_{fail_2}$, ($pk_1$, $sid_1$, payer))
call leak()

function: updateState ("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, $withdraw : bool$):
upload ("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw:bool, ($pk_1$, $sid_1$, payer) ) to
  $\mathcal{M}^1_{install,resume}$
call leak()

function: updateState ("pay", $ack_{win_2}$):
upload ("pay", $ack_{win_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}^1_{install,resume}$
call leak()

function: updateState ("abort", $ack_{fail_2}$):
upload ("abort", $ack_{fail_2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}^1_{install,resume}$
call leak()

function: leak():
output := upload ("leakState", ($pk_1$, $sid_1$, payer)) to $\mathcal{M}^1_{install,resume}$
if ($pk_1$, $sid_1$, payer) is honest:
  return $\sigma^1_{att}$
if ($pk_1$, $sid_1$, payer) is corrupted:
  return $\sigma^1_{att}$, $state^1_{offchain} =$
   $(mpk_1, balance, pk_1, sk_1, addr_{pk_1}, tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2},$
$ack^{j|j\in\{1..n\}}_{fail_2})$
function: reconciliation():
output := upload ("onchainUpdate", ($pk_1$, $sid_1$, payer)) to $\mathcal{M}^1_{install,resume}$
parse output as $(tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2}, ack^{j|j\in\{1..n\}}_{fail_2}, \sigma^1_{att})$
receipt := upload output to $\mathcal{M}_{ledger}$
receipt := receipt || output
if $receipt \rightarrow status$ is False:
  upload ("freeze", receipt, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}^1_{install,resume}$
if $receipt \rightarrow status$ is True:
  upload ("finalize", receipt, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}^1_{install,resume}$
call leak()

function: logout():
status:= call reconciliation(($pk_1$, $sid_1$, payer))
if status is True:
  $tx_{logout}$ = upload ("logout", ($pk_1$, $sid_1$, payer) to $\mathcal{M}^1_{install,resume}$
  $state_{onchain}$ := upload $tx_{logout}$ to $\mathcal{M}_{ledger}$
  assert $state_{onchain} \neq \perp$ and assert $state_{onchain} \rightarrow status \neq$ False
  upload ("clean", $tx_{logout}$, $state_{onchain}$, ($pk_1$, $sid_1$, payer) to $\mathcal{M}^1_{install,resume}$
if status is False abort

function: recharge($b_0$, $addr_{pk_j}$):
status:= call reconciliation(($pk_1$, $sid_1$, payer))
if status is True:
  $tx_{recharge}$ := upload ("transfer", $b_0$, $addr_{pk_j}$, $(pk_1, sid_1, payer)$) to
   $\mathcal{M}^1_{install,resume}$
  call onChainDeposit ($tx_{recharge}$, withdraw:=False, $(pk_1, sid_1, payer)$)
  call leak()
if status is False:
  call leak()
  abort
```

Algorithm 3: Payer sub-Protocol $\mathcal{P}_{payer}$

```
function: withdrawal($b_0$, $addr_{pk_j}$):
status:= call reconciliation(($pk_1$, $sid_1$, payer))
if status is True:
  $tx_{withdraw}$ := upload ("transfer", $b_0$, $addr_{pk_j}$, $(pk_1, sid_1, payer)$) to
   $\mathcal{M}^1_{install,resume}$
  call onChainDeposit ($tx_{withdraw}$, withdraw:=True, $(pk_1, sid_1, payer)$)
  call leak()
if status is False:  call leak() and abort

(1) On receive ("deposit", $tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw:bool):
if withdraw is True:
  parse $tx_{deposit}$ as $(pk_1, addr_{pk_j}, addr_{pk_1}, b_0, Tgen, \sigma^1_{att}, mpk_1, \sigma_1)$
  verify ($\sigma^1_{att}$, $\sigma_1$ ) and assert $(status == True)$ abort if false
  $state_{offchain_i} : balance- = b_0$
if withdraw is False:
  parse $tx_{deposit}$ as $(pk_{pid}, pk_{caller}, addr_{pk_{caller}}, \sigma, b_0)$
  verify $\sigma$ and assert $(status == True)$ abort if false
  $state_{offchain_i} : balance+ = b_0$
append ($tx_{deposit}$, $tx^{receipt}_{deposit}$, withdraw) to Storage[$\mathcal{P}_1$]

(2) On receive ("save", $ack_{win_2}$): store $ack_{win_2}$ to Storage [$p_1$]

(3) On receive ("transfer", $b_0$, $addr_{pk_2}$) :
build $tx_{off} as(pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1)$
upload ($tx_{off}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$

(4) On receive ("logout"):
build $tx_{logout}$ as $(pk_1, addr_{pk_1}, addr_{pk_x}, state_{offchain_1} : balance, T_{gen},$
  $\sigma^1_{att}, mpk_1, \sigma 1)$
upload ($tx_{logout}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$

(5) On receive ("pay", $ack_{win_2}$):
parse $ack_{win_2}$ as ($pk_2$,
  $tx_{off} = (pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1), \sigma_2,$
  $status_{tx_{off}} = ok)$
verify ($\sigma_2$, $\sigma^1_{att}$, $\sigma 1$) and assert $(status_{tx_{off}} == ok)$ abort if False
$state_{offchain_1} : balance- = b_0$

(6) On receive ("abort", $ack_{fail_2}$):
parse $ack_{fail_2}$ as ($pk_2$,
  $tx_{off} = (pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1), \sigma_2,$
  $status_{tx_{off}} = fail)$
verify ($\sigma_2$, $\sigma^1_{att}$, $\sigma 1$) and assert $(status_{tx_{off}} == fail)$ abort if False
store $ack_{fail_2}$ to Storage[$p_1$]

(7) On receive ("finalize", receipt):
parse receipt as (receipt $\rightarrow$ status, output= $(tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2},$
  $ack^{j|j\in\{1..n\}}_{fail_2}\ \sigma^1_{att}))$
if receipt $\rightarrow$ status is True:
  free Storage[$p_1$](output) and  upload (receipt, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$
if receipt $\rightarrow$ status is False: abort

(8) On receive ("clean", $tx_{logout}$, $state_{onchain}$):
parse $tx_{logout}$ as $(pk_1, addr_{pk_1}, addr_{pk_x}, state_{offchain_1} : balance, T_{gen},$
  $\sigma^1_{att}, mpk_1, \sigma_1)$
verify ($\sigma_1$, $\sigma^{att}_1$) and assert $(state_{onchain} \rightarrow status)$ abort if False
delete($pk_1$, $sk_1$)

(9) On receive ("leakState"):
if ($pk_1$, $sid_1$, payer) is honest:
  upload ( $\sigma^1_{att}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$
if ($pk_1$, $sid_1$, payer) is corrupted:
  upload ($\sigma^1_{att}$, $state^1_{offchain}$=$(mpk_1, balance, pk_1, sk_1, addr_{pk_1},$
   $tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2}, ack^{j|j\in\{1..n\}}_{fail_2}$), ($pk_1$, $sid_1$, payer)) to
   $\mathcal{M}_{payer}$

(10) On receive ("onchainUpdate"):
get $(tx^1_{off_{j|\forall j\in\{1..n\}}}, ack^j_{win_2}, ack^{j|j\in\{1..n\}}_{fail_2}, \sigma^1_{att})$ from Storage[$p_1$]
upload $(tx^1_{off_{j|\forall j\in\{1..n\}}}, ack^j_{win_2}, ack^{j|j\in\{1..n\}}_{fail_2}, \sigma^1_{att}$, ($pk_1$, $sid_1$, payer)) to
  $\mathcal{M}_{payer}$
```

$\mathcal{M}_{install,resume}$ stores $tx_{deposit}$ and $state_{onchain}$ to reconcile off-chain and on-chain states during on-chain synchrony (or weak synchrony). The command pay serves to update the local off-chain state of the payer machine instance after a successful offline transaction. The command pay takes ack_{win_2} as parameter. It instructs $prog_{enclave}$ to perform several operations: $prog_{enclave}$ verifies the signature σ_2 using pk_2; $prog_{enclave}$ ensures tx_{off} was accepted by the payee; $prog_{enclave}$ verifies the enclave signature over σ^1_{att} using mpk_1; $prog_{enclave}$ verifies the payer signature over tx_{off} using sk_1. If all verification processes succeed, $\mathcal{M}_{install,resume}$ debits $state_{offchain_1}$:$balance$ by b_0. The command abort helps to process failed offline transactions. It takes as input ack_{fail_2} and instructs $prog_{enclave}$ to perform the following operations: $\mathcal{M}_{install,resume}$ verifies σ_2 using pk_2; $\mathcal{M}_{install,resume}$ checks that ack_{fail_2} contains rejection proof of tx_{off}; $\mathcal{M}_{install,resume}$ verifies σ_1 in tx_{off} using pk_1; $\mathcal{M}_{install,resume}$ verifies σ^1_{att} in tx_{off} using mpk_1. If no verification fails, $\mathcal{M}_{install,resume}$ simply stores ack_{fail_2} to resolve eventual disputes.

The leak subroutine takes advantage of the restricting message $leakState$ sent to $\mathcal{M}_{install,resume}$, which returns only the enclave measurement σ_{att} for uncorrupted parties (honest players). $leakState$ sent to $\mathcal{M}_{install,resume}$ returns both the enclave measurement σ_{att} and the entirety of the off-chain state $state_{offchain}$, except msk_{pid}, for corrupted parties.

Payee role. We define subroutines of the sub-protocol $\mathcal{P}_{payee}$ that aim at the following two main operations: receive, and leak. Algorithm 4 exposes the details of the sub-protocol $\mathcal{P}_{payee}$ in this work.

The receive subroutine fetches the offline transaction tx_{off} from the network. It checks the cryptocurrency amount b_0 in tx_{off} and decides whether to accept or reject tx_{off}.If the receive subroutine accepts tx_{off}, it sends the command "receive" to $\mathcal{M}_{install,resume}$ with tx_{off} as parameter. $\mathcal{M}_{install,resume}$ checks the signature of the enclave measurement σ^1_{att} in tx_{off} using mpk_1. $\mathcal{M}_{install,resume}$ checks the payer signature σ_1 using pk_1. After the successful validation of both signatures, $\mathcal{M}_{install,resume}$ builds an acknowledgment of success ack_{win_2}, stores a copy locally, and returns ack_{win_2} to the receive subroutine; $\mathcal{M}_{install,resume}$ credits the payee's off-chain state balance $state_{offchain_2} : balance$ by b_0. The receive subroutine exploits $\mathcal{F}^{\delta=1}_{BD-SMT}$ and $\mathcal{F}_{CLOCK}$ to forward ack_{win_2} to the payer's machine instance. If the receive subroutine refuses tx_{off}, the receive subroutine sends the command "reject" to $\mathcal{M}_{install,resume}$ with tx_{off} as parameter. The following process also occurs if the validation of any signature fails. $\mathcal{M}_{install,resume}$ builds an acknowledgment of failure ack_{fail_2}, saves a local copy, and returns ack_{fail_2} to the receive subroutine that forwards it to the payer's machine instance (with guaranteed delivery).

The leak subroutine exploits the leakState restricting message sent to $\mathcal{M}_{install,resume}$. It obeys the same workflow as the leak subroutine in the payer's machine instance.

Reconciliation of off-chain and on-chain states. The reconciliation subroutine applies to both the payer and the payee.

The reconciliation subroutine first assays the on-chain network connectivity by sending the restricting message checkConnection to $\mathcal{M}_{ledger}$. If checkConnection returns True, the reconciliation subroutine sends the command "onchainUpdate" to $\mathcal{M}_{install,resume}$. $\mathcal{M}_{install,resume}$ gathers all the stored offline transactions $tx_{off_{j|\forall j \in \{1..n\}}}$, all the acknowledgments of success $ack^j_{win_2}$, all the acknowledgments of failure $ack^j_{fail_2}$, and the latest enclave measurement for the party initiating the on-chain reconciliation (i.e., σ^1_{att} or σ^2_{att}). $\mathcal{M}_{install,resume}$ returns those data to the reconciliation subroutine that leverages $\mathcal{M}_{ledger}$ to submit them to the on-chain smart contract. $\mathcal{M}_{ledger}$ responds to the reconciliation subroutine with a transaction receipt $tx^{receipt}_{reconcile}$ that indicates the success or failure of the operation through the status field. If $tx^{receipt}_{reconcile}$:$status$ outputs False (the on-chain verification of data failed), $\mathcal{M}_{ledger}$ freezes the on-chain balance of such party, and the reconciliation subroutine sends the command freeze to $\mathcal{M}_{install,resume}$ with $tx^{receipt}_{reconcile}$ as parameter. $\mathcal{M}_{install,resume}$ checks the status in $tx^{receipt}_{reconcile}$. If $tx^{receipt}_{reconcile}$:$status$ outputs False, $\mathcal{M}_{install,resume}$ freezes the off-chain state $state_{offchain}$, which prevents the generation of future enclave measurements during the freezing time slot. If $tx^{receipt}_{reconcile}$:$status$ outputs True (the on-chain verification of data succeeded), $\mathcal{M}_{ledger}$ updates the on-chain state $state_{onchain_{pid}}$ of the corresponding party pid, and the reconciliation subroutine sends the command finalize to $\mathcal{M}_{install,resume}$ with $tx^{receipt}_{reconcile}$ as parameter. $\mathcal{M}_{install,resume}$ checks the status in $tx^{receipt}_{reconcile}$. If $tx^{receipt}_{reconcile}$:$status$ outputs True, $\mathcal{M}_{install,resume}$ frees storage space related to offline transactions, acknowledgments of success and the enclave measurement used to generate $tx^{receipt}_{reconcile}$.

Offline account cancellation. The offline account cancellation relies on the logout subroutine to delete an existing offline account within the abstract TEE. It applies to both the payer and the payee. The logout subroutine begins by the reconciliation of off-chain and on-chain states of a specific party. The rationale behind such subroutine is to carry out an on-chain transfer of the complete balance of the offline account $state_{offchain_i} : balance$ to any address, and to require $\mathcal{M}_{install,resume}$ to delete the account (pk_i, sk_i) after successful on-chain transfer.

Regarding additional operations in our protocol, the withdraw functionality performs an on-chain transfer of cryptocurrency units from the offline account to a recipient address; it requires a prior reconciliation of off-chain and on-chain states. The recharge functionality replenishes the offline account in terms of cryptocurrency units: it requires an initial reconciliation of on-chain and off-chain states, and leverages the onchainDeposit subroutine.

B. Communication Model

In our protocol, the parties, the adversary, and the subfunctionalities are linked in a star topology through an additional router machine that takes instructions from the adversary $\mathcal{A}$. In our work, such a router machine is implemented using an ideal sub-functionality that ensures authenticated channels (authenticated channels prevent an adversary from modifying data in transit while enforcing secrecy through encryption) and weak synchrony among parties. The work of Canetti et al. [34] provides more insights into the role of such a router machine. According to [28], the entities can send and receive messages using the input/output and the network interfaces that belong to their respective roles. This work enforces partial synchrony on the off-chain communication model: we leverage a responsive environment [35] for off-chain settings where messages are delivered reliably within δ. Although we alleviate assumptions regarding the on-chain communication model and rely on underlying properties provided by $\mathcal{F}_{ledger}$, we assume the default asynchrony of the on-chain communication model: our work considers the environment as unresponsive for on-chain interactions.

V. Security Analysis and Proof

In this section, we dive into the security analysis of our proposed protocol. With respect to the threat model in Section III, we separate our analysis into two subsections: informal security analysis, which analyzes the various attacks in our protocol; formal security, which evaluates the security of our construction based on UC.

Algorithm 4: Payee sub-Protocol $\mathcal{P}_{payee}$

```
$state_{offchain_2} = state^0_{offchain_2}$
Storage[$p_2$] := $\phi$

function: leak():
output := upload ("leakState", ($pk_2$, $sid_2$, payee)) to $\mathcal{M}^2_{install,resume}$
if ($pk_2$, $sid_2$, payee) is honest:
  return $\sigma^2_{att}$
if ($pk_2$, $sid_2$, payee) is corrupted:
  return $\sigma^2_{att}$, $state^2_{offchain} =$
    $(mpk_2, balance, pk_2, sk_2, addr_{pk_2}, tx^2_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2},$
  $ack^{j|j\in\{1..n\}}_{fail_2})$

function: receive(tx_off):
parse $tx_{off}$ as $(pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1)$
if accept $b_0$:
  output:=upload ("receive", $tx_{off}$,($pk_2$, $sid_2$, payee) to $\mathcal{M}^2_{install,resume}$
  parse output as $ack_{win_2}$ or $ack_{fail_2}$
  upload ("save", $ack_{win_2} \mid ack_{fail_2}$, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}^2_{install,resume}$
  upload(output, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$
if refuse $b_0$:
  $ack_{fail_2}$:=upload ("reject", $tx_{off}$, ($pk_2$, $sid_2$, payee) to $\mathcal{M}^2_{install,resume}$
  parse output := $ack_{fail_2}$ as $(pk_2, tx_{off} = (pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen},$
    $\sigma^1_{att}, mpk_1, \sigma 1), \sigma_2, status_{tx_{off}} = fail)$
  upload ("save", output, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}^2_{install,resume}$
  upload(output, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$
call leak()

function: reconciliation():
output := upload ("onchainUpdate", ($pk_2$, $sid_2$, payee)) to $\mathcal{M}^2_{install,resume}$
parse output as $(tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2}, ack^{j|j\in\{1..n\}}_{fail_2}, \sigma^2_{att})$
receipt := upload output to $\mathcal{M}_{ledger}$
receipt := receipt || output
if $receipt \rightarrow status$ False:
  upload ("freeze", receipt, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}^2_{install,resume}$
if $receipt \rightarrow status$ True:
  upload ("finalize", receipt, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}^2_{install,resume}$
call leak()

function: logout():
status:= call reconciliation(($pk_2$, $sid_2$, payee))
if status is True:
  $tx_{logout}$ = upload ("logout", ($pk_2$, $sid_2$, payee) to $\mathcal{M}^1_{install,resume}$
  $state_{onchain}$ := upload $tx_{logout}$ to $\mathcal{M}_{ledger}$
  assert $state_{onchain} \neq \perp$ and assert $state_{onchain} \rightarrow status \neq$ False
  upload ("clean", $tx_{logout}$, $state_{onchain}$, ($pk_2$, $sid_2$, payee) to $\mathcal{M}^1_{install,resume}$
if status is False abort
call leak()
```

Algorithm 4: Payee sub-Protocol $\mathcal{P}_{payee}$

```
(1) On receive ("receive", $tx_{off}$):
parse $tx_{off}$ as $(pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1)$
if verify ($\sigma_1$, $\sigma^1_{att}$) True:
  build $ack_{win2}$ as ($pk_2$,
    $tx_{off} = (pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen}, \sigma^1_{att}, mpk_1, \sigma 1), \sigma_2,$
    $status_{tx_{off}} = ok)$
  store $ack_{win2}$ to Storage[$p_2$]
  $state_{offchain_2} : balance += b_0$
  upload($ack_{win2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$
if verify ($\sigma_1$, $\sigma^1_{att}$) False:
  build $ack_{fail2}$ as $(pk_2, tx_{off} = (pk_1, addr_{pk_1}, addr_{pk_2}, b_0, T_{gen},$
    $\sigma^1_{att}, mpk_1, \sigma 1), \sigma_2, status_{tx_{off}} = fail)$
  store $ack_{fail2}$ to Storage[$p_2$]
  upload($ack_{fail2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$

(2) On receive ("reject", $tx_{off}$):
  build $ack_{fail2}$ and store $ack_{fail2}$ to Storage[$p_2$]
  upload($ack_{fail2}$, ($pk_1$, $sid_1$, payer)) to $\mathcal{M}_{payer}$

(3) On receive ("finalize", receipt):
parse receipt as (receipt $\rightarrow$ status, output= $(tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2},$
  $ack^{j|j\in\{1..n\}}_{fail_2}, \sigma^2_{att}))$
if receipt $\rightarrow$ status True: free Storage[$p_2$](output) abort if False

(4) On receive ("logout"):
build $tx_{logout}$ as $(pk_2, addr_{pk_2}, addr_{pk_x}, state_{offchain_2} : balance, T_{gen},$
  $\sigma^2_{att}, mpk_2, \sigma 2)$
upload ($tx_{logout}$, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}_{payee}$

(5) On receive ("clean", $tx_{logout}$, $state_{onchain}$):
parse $tx_{logout}$ as $(pk_2, addr_{pk_2}, addr_{pk_x}, state_{offchain_2} : balance, T_{gen},$
  $\sigma^2_{att}, mpk_2, \sigma_2)$
verify ($\sigma_2$, $\sigma^{att}_2$) and assert ($state_{onchain} \rightarrow status$) abort if False
delete($pk_2$, $sk_2$)

(6) On receive ("leakState"):
if ($pk_2$, $sid_2$, payee) is honest:
  upload ( $\sigma^2_{att}$, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}_{payee}$
if ($pk_2$, $sid_2$, payee) is corrupted:
  upload ($\sigma^2_{att}$, $state^2_{offchain}$=$(mpk_2, balance, pk_2, sk_2, addr_{pk_2},$
    $tx^1_{off_{j|j\in\{1..n\}}}, ack^{j|j\in\{1..n\}}_{win_2}, ack^{j|j\in\{1..n\}}_{fail_2})$, ($pk_2$, $sid_2$, payee)) to
    $\mathcal{M}_{payee}$

(7) On receive ("onchainUpdate"):
get params := $(tx^1_{off_{j|\forall j\in\{1..n\}}}, ack^j_{win_2}, ack^j_{fail_2}, \sigma^2_{att})$ from Storage[$\mathcal{P}_2$]
upload (params, ($pk_2$, $sid_2$, payee)) to $\mathcal{M}_{payee}$

(8) On receive ("save", $input$): store $input$ to Storage [$p_2$]
```

A. Informal Security Analysis

Coin Forgery Attacks. A malicious user p_i may try to update $state_{offchain_i}$ via a fake transaction $tx'_{deposit}$ that lacks receipt from the on-chain smart contract. Since $tx'^{receipt}_{deposit}$ must be produced by the on-chain smart contract, the execution of $tx'_{deposit}$ fails as long as it is an invalid transaction. To modify the balance of a deposit transaction requires the ability to forge the on-chain digital signature scheme. Since on-chain operations are trusted, such endeavor exhibits unfeasibility in our scheme. We consider that our protocol avoids coin forgery attacks.

Forgery of Offline Transaction Data. A malicious player may attempt to tamper the amount b_0 or the transaction timestamp T_{gen} in an offline transaction received from the network. As our work relies on the transparent enclave model, no other entity except the enclave has knowledge of the manufacturer's secret key msk: verifying the signature of such transaction will fail in the secure enclave and the attack will be unsuccessful.

Double Spend and Double Deposit Attacks. A malicious payer could be motivated to send the same transaction multiple times to the payee in hopes to double spend. Since operations in the enclave are trusted and each enclave keeps track of received transactions, $prog_{enclave}$ will fail to process the same offline transaction twice on the recipient side. For multiple use of the same coin, the payer does not have access to msk and lacks the ability to masquerade a fake attestation as valid. A malicious payee may re-transmit the same transaction received from the network to itself (to its enclave), in hopes to double deposit and increase the balance of its offline account. Since the enclave keeps track of received offline transactions as a trusted offline ledger, attempts to double deposit will fail and the malicious payee may under go temporal account freeze. Our protocol thwarts double spend and double deposit attacks.

Man-in-the-middle Attacks. An eavesdropper can intercept an offline payment transaction tx_{off}. It may attempt to impersonate the payee to enjoy the amount of cryptocurrency units in such offline transaction. Since each party has a unique public/private key pair and enclave computations are trusted, it is infeasible for such adversary to confuse the enclave about the ownership of the rightful recipient's public key. Since the environment is responsive, the eavesdropper has no ability to continuously delay the execution of the protocol. Such transaction will be denied by the secure enclave and more actions could be taken such as temporal account freeze to deter man-in-the-middle attacks.

B. Formal Security Proof Based on UC

This subsection analyzes our offline payment solution under the UC framework with off-chain synchronous securities.

Theorem 1: Let $\mathcal{P}$ and $\mathcal{F}$ be two complete but resource-bounded protocols with the same set of public roles [28]. The protocol $\mathcal{P}$ realizes the ideal functionality $\mathcal{F}$ ($\mathcal{P} \leq \mathcal{F}$) if and only if for all adversaries $\mathcal{A}$ that control the network of $\mathcal{P}$, there exists a simulator $\mathcal{S}$ that controls the network of $\mathcal{F}$ such that $\mathcal{A}$, $\mathcal{P}$ exhibits indistinguishability from $\{\mathcal{S}, \mathcal{F}\}$ for all environments $\mathcal{Z}$: $\{\mathcal{Z}, \mathcal{A}, \mathcal{P}\} \equiv \{\mathcal{Z}, \mathcal{S}, \mathcal{F}\}$.

Proof: Proof of indistinguishability. Let E be an event where the simulator $\mathcal{S}$ has access to all leaked messages as well as the internal state of corrupted parties, such that $\mathcal{S}$ can forge signatures, forge data in transit and data at rest. Since our UC model exhibits weakly synchronous and authenticated channels; since computations performed within the TEE are trusted (the hash function $\mathcal{G}_{att}.H$ in the random oracle model, and the digital signature scheme $\mathcal{G}_{att}.\Sigma$); since the environment $\mathcal{Z}$ is responsive and maintains the secrecy of msk even for corrupted parties, it is challenging to construct a PPT algorithm that finds collisions in $\mathcal{G}_{att}.H$ or collisions in random functions: the probability that event E occurs is negligible. Our protocol $\mathcal{P}$ can safely realize the ideal functionality $\mathcal{F}$ in the proposed hybrid UC model.

VI. Implementation and Evaluation

A. Offline Payment Protocol Implementation

We implement two components of our offline payment protocol: 1) on-chain smart contract and 2) offline account. We test the blockchain offline payment protocol using an eight-core 2.4 GHz Intel(R) Core(TM) i7-10510U CPU and 16 GB of RAM. The operating system we used is Ubuntu 20.04.4 TLS with Linux kernel version 5.15.0-43-generic.

On-chain smart contract: We developed the on-chain smart contract of our protocol based on the Ethereum platform [2], using Solidity [36] version 0.8.1. To guarantee the functional requirements of the on-chain smart contract in our work, we deployed such smart contract in a local go-Ethereum environment on our internal server. Users can interact with the deployed smart contract: they can invoke its functions to ensure the functional requirements of our protocol regarding the enclave and the consistency of operations.

Offline account: We use the open source framework Occlum 0.29.2 [37] for TEE OS based on the common TEE platform Intel SGX [38], and develop it using the python language. Occlum is an open-source TEE OS system, designed using the container concept and divided into enclave and host. The user loads and executes the appropriate application from the trusted image using the Occlum run command in the host. In Python, we utilized web3.py [39], which is compatible with Ethernet accounts, to hash the transactions using Keccak256 [40] and sign the transaction hash using ECDSA-secp256k1 [41]. The offline payment protocol we developed via Python is built into Intel SGX via Occlum to enable secure interaction between offline accounts and on-chain smart contracts.

B. Offline Payment Protocol Evaluation

1) Code Size: To develop the blockchain offline payment protocol, we migrate web3.py to SGX via Occlum. We use Python for the development of the offline payment protocol on the off-chain, relying on the web3 package and making relevant modifications to the functions within the package to meet the functional requirements of the protocol. As shown in Table I, we produced 560 LOC of overall Python code for the protocol and another 2849 LOC from dependencies. The protocol was made to run into Intel SGX by packing it with 63 LOC of one configuration file and two script files, using the Occlum build command and making a trusted image. The online contract deployed on the blockchain is implemented with Solidity and has 215 LOC.

TABLE I
Code Size

Component	Code	LOC
Offline payment protocol	Python	560
Dependent codes	Python	2849
Occlum setting	scripts	63
Online-contract	Solidity	215

TABLE II
Gas and Time Consumption of On-Chain Smart Contract

Function	Gas (GWEI)	Time (ms)	Remark
Deploy the Contract	987073	19.88	-
Verification and update	463948	8.80	one transaction
	551844	11.12	threee transactions
	639740	12.89	five transactions
Dynamic balance exchange	420000	8.46	withdrawal/recharge
Offline account cancellation	469884	9.47	-
Freeze account	21916	0.44	-

2) On-Chain Smart Contract Performance: We use Solidity to write on-chain smart contracts and deploy them to the Ethereum test network for testing, implementing the functions required for an offline payment protocol. The specific on-chain smart contract performance is shown in Table II. Deploying the on-chain smart contract to the blockchain consumes 987073gas and takes 19.88ms. The deployment of the contract takes place only once and the corresponding consumption occurs only once to invoke the functionality it is set up for. When the user has a network, offline transactions are uploaded to the chain to validate and update the status. We tested the performance of the contract to validate one, three, and five transactions respectively, and the consumption of gas is 463948, 551844, and 639740, and the time consumption is 8.80ms, 11.12ms, and 12.89ms. With the increasing number of transactions, the on-chain smart contract increases for the consumption of offline transaction information validation. Once the offline transaction information is verified, a transaction is executed on the blockchain to update the account state. In addition, for the dynamic balance exchange function of offline and on-chain accounts, the gas consumption for recharge/withdrawal is 420,000 and the time cost is 8.46ms; for the offline account cancellation function, the gas consumption is 469,884 and the time cost is 9.47ms; for the frozen offline account, the gas consumption is 21916 and the time cost is 0.44ms.

TABLE III
IMPLEMENTED FUNCTIONS COMPARISON WITH EXISTING WORKS

Implemented Functions	**Scheme Name**						
	Bitcoin1 [21]	Bitcoin2 [22]	LN [10]	SVLP [12]	xLumi [11]	PW [23]	Ours
Interact with smart contracts	×	×	×	✓	✓	✓	✓
Coin forgery attack	✓	✓	✓	✓	✓	✓	✓
Offline transaction data forgery	✓	✓	✓	✓	✓	×	✓
Double-spend and double-deposit attack	✓	✓	✓	✓	✓	✓	✓
Man-in-the-middle attack	✓	✓	✓	✓	✓	✓	✓
Open transaction	✓	✓	×	×	×	✓	✓
Coin redistribution	✓	✓	✓	✓	✓	×	✓
Instant settlement for offline payments	×	×	✓	✓	×	✓	✓

Through the analysis of the on-chain smart contract's gas and time consumption, the performance of our developed contract is sound and implementable.

3) Offline Account Performance: We implement the offline account functionalities in the enclave to support the proper working of the offline payment protocol. Fig. 3 shows the time consumption of the developed offline payment protocol running 50 times in the enclave. Specifically, the five specific operations of the offline payment protocol are included, namely from P1 to P5. P1 is the initialization of the offline account operation, where the time used in the enclave in generating the offline account public and private key pairs and initializing the account state is focused on 46 to 50 ms. P2 is the offline transaction operation, where the average time consumption is about 32 ms. The time consumption here only includes the time consumption for execution within the TEE, in practice the time consumption should also include the communication time between the TEE and the host, NFC or Bluetooth. P3 is the upload verification and update operations, which corresponds to the enclave time cost of submitting 3 offline transactions for upload verification and update, with an average time cost of 50ms. P4 is the dynamic balance exchange operation, with an average time cost of 37ms for recharge/withdrawal in the enclave. P5 is the offline account logout operation, with an average time cost of 55ms in the enclave.

By analyzing the time consumption of offline accounts operating in the enclave, our protocol performs well in the enclave and is implementable.

C. Comparison With Existing Works

This protocol implements functions in an offline payment scenario in blockchain and compares them with existing relevant blockchain offline payment solutions. The basic security requirements for offline payments are coin forgery attacks, offline transaction data forgery, double-spend and double-deposit attacks, and man-in-the-middle attacks. As well as the real flexibility requirements of users in practical application scenarios, including open transactions, offline coin redistribution, and instant settlement for offline payments, are addressed. The comparison results are shown in Table III.

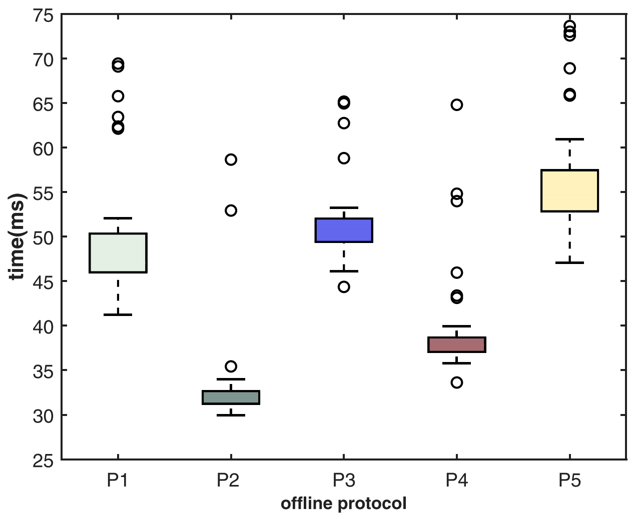


Fig. 3. Time consumption for 50 executions of the offline payment protocol in the enclave.

From Table III, comparing with the latest blockchain offline payment solutions, it is clear that all existing protocols have some missing functionality, while our protocol is more complete than the others. This protocol meets the security requirements of offline payments while giving users the practical flexibility they need in real-world scenarios. Specifically, Bitcoin wallet [21], [22] can meet the basic security requirements, but lacks the function of instant settlement of offline payments; for [10], [11], [12], it is the payment channel that represents the offline payment solution, both have the disadvantage of not being able to achieve open transactions, and are greatly limited in the application scenario; Ethereum offline payment framework PW wallet [23] does not meet the basic security requirements, and the offline balance is non-distributable.

Our offline payment protocol can meet all the above functional requirements and satisfy both basic security requirements and flexibility requirements. We realize the advanced level of current blockchain-based offline payment solutions, with security and flexibility.

VII. Conclusion

The main goal of this paper was to determine whether two offline parties can proceed with blockchain-based offline payments without sacrificing security and flexibility. To achieve our aim, we designed a flexible and secure offline payment protocol under partial network asynchronism. Our work leverages on-chain smart contracts during good on-chain connectivity and secure off-chain transactions that leverage compatible TEEs under bad on-chain networks. We characterized a threat model for blockchain-based offline payment over an asynchronous network. We found that our construction meets the basic security requirements in practical offline payment scenarios: it is secure and robust against coin forgery attacks, offline transaction forgery, double-spend and double-deposit attacks, as well as man-in-the-middle attacks. As another finding, our protocol exhibits flexibility thanks to the use of open transactions that support coin redistribution and instant settlement. We evaluated our work under the Intel SGX and provided an empirical performance analysis of our work on the Ethereum blockchain. We provided a theoretical comparison between our design and state-of-the-art constructions. The results suggest that our protocol achieves a threshold between security and flexibility compared to existing works. The findings will be of interest to businesses, governments, academicians, and practitioners. The most important limitation lies in the fact that our scheme lack support for fully asynchronous mode: this would be a fruitful area for further work. Our work shows that there is a definite need for secure and flexible offline payment solutions that improve the user experience and widen the adoption of blockchain technology.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, 2008, Art. no. 21260.

[2] V. Buterin et al., "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, no. 37, pp. 2–1, 2014.

[3] "Biggest cryptocurrency in the world - Both coins and tokens - Based on market capitalization on November 11, 2022." Statista. Accessed: Nov. 2022. [Onlline]. Available: https://www.statista.com/statistics/1269013/biggest-crypto-per-category-worldwide/

[4] B. Sriman and S. G. Kumar, "Decentralized finance (DEFI): The future of finance and defi application for Ethereum blockchain based finance market," in *Proc. Int. Conf. Adv. Comput., Commun. Appl. Informat. (ACCAI)*, 2022, pp. 1–9.

[5] P. Kar, K. Chen, and J. Shi, "DMACN: A dynamic multi-attribute caching mechanism for NDN-based remote health monitoring system," *IEEE Trans. Comput.*, vol. 72, no. 5, pp. 1301–1313, May 2023.

[6] M. Xu, S. Liu, D. Yu, X. Cheng, S. Guo, and J. Yu, "Cloudchain: A cloud blockchain using shared memory consensus and RDMA," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3242–3253, Dec. 2022.

[7] M. Xu, F. Zhao, Y. Zou, C. Liu, X. Cheng, and F. Dressler, "BLOWN: A blockchain protocol for single-hop wireless networks under adversarial SINR," *IEEE Trans. Mobile Comput.*, vol. 22, no. 8, pp. 4530–4547, Aug. 2023.

[8] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu, and X. Cheng, "wChain: A fast fault-tolerant blockchain protocol for multihop wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6915–6926, Oct. 2021.

[9] V. Buterin. "Ethereum. On sharding blockchains." GitHub. Accessed: Nov. 19, 2019. [Online]. Avalable: https://github.com/ethereum/wiki/

[10] J. S. Bellagarda, "The potential effect off-chain instant payments will have on cryptocurrency scalability issues-the lightning network," in *Proc. Int. Conf. Inf. Resour. Manage.* (*CONFIRM)*, 2019, vol. 2, pp. 1–14.

[11] N. Ying and T. W. Wu, "xlumi: Payment channel protocol and off-chain payment in blockchain contract systems," 2021, *arXiv:2101.10621*.

[12] L. Zhong, Q. Wu, J. Xie, J. Li, and B. Qin, "A secure versatile light payment system based on blockchain," *Future Gener. Comput. Syst.*, vol. 93, pp. 327–337, Apr. 2019.

[13] E. Rohrer, J. Malliaris, and F. Tschorsch, "Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, 2019, pp. 347–356.

[14] A. Mizrahi and A. Zohar, "Congestion attacks in payment channel networks," in *Financial Cryptography and Data Security*, N. Borisov and C. Diaz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 170–188.

[15] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[16] "The Raiden network." Raiden. Accessed: Jul. 2023. [Online]. Available: https://raiden.network/

[17] J. Coleman, L. Horne, and L. Xuanji. "Counterfactual: Generalized state channels." Available: https://www.bgp4.com/wp-content/uploads/2019/05/statechannels.pdf

[18] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A secure payment network with asynchronous blockchain access," in *Proc. 27th ACM Symp. Operating Syst. Princ.*, 2019, pp. 63–79.

[19] M. Elsheikh, J. Clark, and A. M. Youssef, "Short paper: Deploying payword on Ethereum," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, New York, NY, USA: Springer, 2019, pp. 82–90.

[20] N. Ivanov and Q. Yan, "System-wide security for offline payment terminals," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, New York, NY, USA: Springer, 2021, pp. 99–119.

[21] A. Dmitrienko, D. Noack, and M. Yung, "Secure wallet-assisted offline bitcoin payments with double-spender revocation," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 520–531.

[22] T. Takahashi and A. Otsuka, "Short paper: secure offline payments in bitcoin," in *Proc. Int. Conf. Financial Cryptography Data Secur.,* New York, NY, USA: Springer, 2019, pp. 12–20.

[23] I. S. Igboanusi, K. P. Dirgantoro, J.-M. Lee, and D.-S. Kim, "Blockchain side implementation of pure wallet (PW): An offline transaction architecture," *ICT Exp.*, vol. 7, no. 3, pp. 327–334, 2021.

[24] H. Wang, X. Li, J. Gao, and W. Li, "Mobt: A kleptographically-secure hierarchical-deterministic wallet for multiple offline bitcoin transactions," *Future Gener. Comput. Syst.*, vol. 101, pp. 315–326, Dec. 2019.

[25] S. Verbücheln, "How perfect offline wallets can still leak bitcoin private keys," *MCIS 2015 Proceedings. 2*. [Online]. Available: https://aisel.aisnet.org/mcis2015/2

[26] W. Dai, J. Deng, Q. Wang, C. Cui, D. Zou, and H. Jin, "SBLWT: A secure blockchain lightweight wallet based on trustzone," *IEEE Access*, vol. 6, pp. 40638–40648, 2018.

[27] J. Katz, U. Maurer, B. Tackmann, and V. Zikas, "Universally composable synchronous computation," in *Theory of Cryptography*, A. Sahai, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 477–498.

[28] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch, "iUC: Flexible universal composability made simple," in *Proc. Adv. Cryptol. (ASIACRYPT)*, S. D. Galbraith and S. Moriai, Eds., Cham, Switzerland: Springer International Publishing, 2019, pp. 191–221.

[29] R. Kuesters, M. Tuengerthal, and D. Rausch, "The IITM model: A simple and expressive model for universal composability," Cryptol. ePrint Arch., Paper, 2013, 2013/025. [Online]. Available: https://eprint.iacr.org/2013/025

[30] R. Canetti, R. Pass, and A. Shelat, "Cryptography from sunspots: How to use an imperfect reference string," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, 2007, pp. 249–259.

[31] M. Graf, D. Rausch, V. Ronge, C. Egger, R. Küsters, and D. Schröder, "A security framework for distributed ledgers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA: ACM, 2021, pp. 1043–1064.

[32] R. Pass, E. Shi, and F. Tramèr, "Formal abstractions for attested execution secure processors," in *Proc. Adv. Cryptol. (EUROCRYPT)*, J.-S. Coron and J. B. Nielsen, Eds., Cham, Switzerland: Springer International Publishing, 2017, pp. 260–289.

[33] F. Tramèr, F. Zhang, H. Lin, J.-P. Hubaux, A. Juels, and E. Shi, "Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2017, pp. 19–34.

[34] R. Canetti, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in*

Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9216. Berlin, Germany: Springer Verlag, 2015, pp. 3–22.

[35] J. Camenisch, R. R. Enderlein, S. Krenn, R. Küsters, and D. Rausch, “Universal composition with responsive environments,” in *Proc. Adva. Cryptol. (ASIACRYPT)*, J. H. Cheon and T. Takagi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 807–840.

[36] “Solidity is an object-oriented, high-level language for implementing smart contracts.” Solidity. Accessed: Jul. 2023. [Online]. Available: https://docs.soliditylang.org/en/v0.8.17/

[37] Y. Shen et al., “Occlum: Secure and efficient multitasking inside a single enclave of intel SGX,” in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2020, pp. 955–970.

[38] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, “Innovative technology for CPU based attestation and sealing,” in *Proc. 2nd Int. Workshop Hardware Archit. Support Secur. Privacy*. New York, NY, USA: ACM, 2013, 13(7).

[39] “Web3.py is a python library for interacting with Ethereum.” [Online]. Available: https://web3py.readthedocs.io/en/v5/

[40] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “The road from panama to Keccak via RadioGatún,” in *Dagstuhl Seminar Proceedings* Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[41] D. R. Brown, “Sec 2: Recommended elliptic curve domain parameters,” *Standards for Efficient Cryptography*, 2010. [Online]. Available: https://www.secg.org/sec2-v2.pdf

Wanqing Jie received the B.S. degree from the Department of Management Science and Engineering, Tianjin University of Finance and Economics, China, in 2020, and the M.S. degree from the Institute of Artificial Intelligence and Blockchain, Guangzhou University, China, in 2023. Currently, she is working toward the Ph.D. degree with Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, China. Her research interests include blockchain and privacy computing.

Wangjie Qiu (Member, IEEE) received the B.S. degree, in 2008, and the Ph.D. degree, in 2013, both in mathematics from Beihang University. He is currently an Associate Professor with Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University. He is also the Deputy Secretary General of the blockchain committee, China Institute of Communications. His research interests include Information security, blockchain, and privacy computing. As the founding team, he successfully developed ChainMaker, an international advanced blockchain technology system. He has been authorized a number of invention patents in the fields of information security, blockchain, and privacy computing. He won the first prize of China National Technology Invention in 2014.

Arthur Sandor Voundi Koe (Member, IEEE) received the B.E. degree in network and computer maintenance from The African Institute of Computer Science (IAI), Cameroon branch, in 2010, the B.S. degree in fundamental computer science from the University of Yaoundé 1, Cameroon, in 2011, the M.S. degree in computer and application technology from Hunan University, China, in 2015, and the Ph.D. degree in computer science and application technology from Hunan University, China in 2020. He was a Postdoctoral Researcher with Guangzhou University, from 2020 to 2023. He is currently a Lecturer with Xidian University. He serves as a Reviewer for many renowned international journals. His research interests include security and privacy issues in blockchain technology, cloud computing security and privacy issues, and machine learning (adversarial learning and federated learning).

Jianhong Li was born in 1998. He received the bachelor of engineering degree in information security from Guangzhou University. He is now working toward the master’s degree in cyberspace security with Guangzhou University. His research interests are blockchain and cybersecurity.

Yin Wang was born in 1999 and received the B.S. degree from the Computer Science Department of Jiangxi Agricultural University. He is now working toward the M.S. degree with Guangzhou University. His research interests are blockchain and consensus algorithms.

Yaqi Wu was born in 2000. He is currently working toward the M.Eng. degree with the Institute of Artificial Intelligence and Blockchain, Guangzhou University. His research interests include AI security and blockchain.

Jin Li (Senior Member, IEEE) received the B.S. degree, in 2002, M.S. degree, in 2004, both in mathematics, from Southwest University and Sun Yat-sen University respectively, and the Ph.D. degree in information security from Sun Yat-sen University, in 2007. He is currently a Professor and an Executive Dean of the Institute of Artificial Intelligence and Blockchain, Guangzhou University. His research interests include the design of secure protocols in cloud computing and cryptographic protocols. He has published more than 100 papers in international conferences and journals, including IEEE INFOCOM, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, and ESORICS. His work has been cited more than 18000 times at Google Scholar and the H-Index is 56. He also served as program chair and committee for many international conferences. He received NSFC Outstanding Youth Foundation in 2017.

Zhiming Zheng received the Ph.D. degree in mathematics from the School of Mathematical Sciences, Peking University, Beijing, China, in 1987. He is currently a Professor with the Institute of Artificial Intelligence, Beihang University, Beijing, China. His research interests include refined intelligence, blockchain, and privacy computing. He is one of the initiators of Blockchain-ChainMaker. He is a member of Chinese Academy of Sciences.