A Blockchain-based Shamir's Threshold Cryptography Scheme for Data Protection in Industrial Internet of Things Settings

Keping Yu, *Member, IEEE,* Liang Tan, Caixia Yang, Kim-Kwang Raymond Choo, *Senior Member, IEEE,* Ali Kashif Bashir, *Senior Member, IEEE*, Joel J. P. C. Rodrigues, *Fellow, IEEE,* and Takuro Sato, *Life Fellow, IEEE*

Abstract—The Industrial Internet of Things (IIoT), a typical Internet of Things (IoT) application, integrates the global industrial system with other advanced computing, analysis, and sensing technologies through Internet connectivity. Due to the limited storage and computing capacity of edge and IIoT devices, data sensed and collected by these devices are usually stored in the cloud. Encryption is commonly used to ensure privacy and confidentiality of IIoT data. However, the key used for data encryption and decryption is usually directly stored and managed by users or third-party organizations, which has security and privacy implications. To address this potential security and privacy risk, we propose a Shamir threshold cryptography scheme for IIoT data protection using blockchain: STCChain. Specifically, in our solution the edge gateway uses a symmetric key to encrypt the data uploaded by the IoT device and stores it in the cloud. The symmetric key is protected by a private key generated by the edge gateway. To prevent the loss of the private key and privacy leakage, we use a Shamir secret sharing algorithm to divide the private key, encrypt it, and publish it on the blockchain. We implement a prototype of STCChain using Xuperchain, and the results show that STCChain can effectively prevent attackers from stealing data as well as ensuring the security of the encryption key.

Index Terms—Industrial Internet of Things, data protection, blockchain, Shamir secret sharing.

1 Introduction

The Industrial Internet of Things is a system composed of networked smart objects, network physical assets, related general information technology, and, in some settings, cloud or edge computing platforms that support various processes and services in an industrial environment. Examples of such processes and services include real-time, intelligent and autonomous access to data and their collection, analysis, communication and exchange to optimize the overall output value. There are, however, security and privacy considerations in the deployment of such systems that necessitate the design of robust security and privacy-preserving solutions in IIoT settings.

A typical, simplified IIoT system architecture generally comprises three key functional systems, namely, network, platform and security systems. The latter (i.e., security system) is designed to support the security requirements in applications, data, networks, controls and devices, for example, by identifying, detecting and mitigating various security threats [1], [2].

One challenge of designing security and privacy-preserving solutions for IIoT systems is that the devices underpinning such systems are generally limited in terms of storage and computing capabilities and are power-constrained [3], [4]. Hence, data sensed, collected and disseminated by these IIoT devices are generally stored and processed in the cloud. To minimize the risk of data leakage (e.g., due to a malicious cloud employee), protective measures such as data encryption are used [5], [6]. Such approaches generally require the data encryption/decryption key to be stored and/or managed directly by users or by a centralized third-party institution. There are limitations to both approaches; for example, the limitations associated with the local storage of keys include single point of failure/attack and loss of data/service access if the local storage

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grants JP18K18044 and JP21K17736, in part by the National Natural Science Foundation of China under Grant No. 61373162, in part by the Sichuan Provincial Science and Technology Department Project under Grant No. 2019YFG0183, in part by the Sichuan Provincial Key Laboratory Project under Grant No. KJ201402, in part by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/50008/2020, and in part by Brazilian National Council for Scientific and Technological Development - CNPq, via Grant No. 313036/2020-9. The work of K.-K. R. Choo was supported only by the Cloud Technology Endowed Professorship. (Corresponding author: Liang Tan)

- *Keping Yu is with the College of Computer Science, Sichuan Normal University, Chengdu 610101, China and the Global Information and Telecommunication Institute, Waseda University, Tokyo, Japan. (e-mail: keping.yu@aoni.waseda.jp).*
- *Liang Tan is with the College of Computer Science, Sichuan Normal University, Chengdu, China and the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China (e-mail: jkxy_tl@sicnu.edu.cn).*
- *Caixia Yang is with the College of Computer Science, Sichuan Normal University, Chengdu, Sichuan, China (e-mail: doreamon95@163.com).*
- *Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, and the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249, USA (e-mail: raymond.choo@fulbrightmail.org).*
- *Ali Kashif Bashir is with the Department of Computing and Mathematics, E-154, John Dolton, Chester Street, M15 6H, Manchester Metropolitan University, Manchesterm UK (email: dr.alikashif.b@ieee.org)*
- *Joel J. P. C. Rodrigues is with the Senac Faculty of Ceará, 60160-194 Fortaleza-CE, Brazil; Instituto de Telecomunicações, 6201-001 Covilhã, Portugal (e-mail: joeljr@ieee.org).*
- *Takuro Sato is with Research Institute for Science and Engineering, Waseda University, Tokyo 169-8555, Japan (e-mail: t-sato@waseda.jp).*

medium is corrupted (unless there is a backup copy of the key on another storage medium). If the keys are stored and managed by a trusted third-party center (e.g., certificate management organization (CA)), then we must trust the trusted third-party center to do the right thing and not leak the key.

Seeking to mitigate these limitations and based on [7], we propose a **S**hamir's **T**hreshold **C**ryptography approach that utilizes block**Chain** for IIoT key data (hereafter referred to as **STCChain**). [7] is only a synopsis, while this paper proposes a detailed solution for the two limitations of encryption/decryption key protection mentioned above. Also, this paper validates the proposed method from multiple dimensions. The key features are as follows:

- **Decentralized IIoT key data threshold encryption scheme**. First, the edge gateway encrypts the data uploaded by the IoT device with a temporary symmetric key generated by the edge gateway and stored in the cloud; meanwhile, the symmetric key is encrypted by the public key generated by the edge gateway. Second, we use the Shamir threshold scheme [8] to split the private key, and the private key fragments are encrypted to store on blockchain [9], [10]. The user can decrypt the data only after successfully reconstructing the decryption key. Finally, the STCChain design includes a detailed data security storage process and data security reading process.
- **Key decentralized storage**. To prevent the loss and disclosure of privacy for the private key used for decryption, we use the Shamir threshold scheme to split the private key, and its fragments are encrypted with the participant's public key and published on blockchain, effectively guaranteeing key privacy.
- **Security**. STCChain can effectively guarantee the confidentiality, integrity, availability, and accountability of data and can effectively protect the security of the encryption key.

Sections 2 and 3, respectively, discuss the related work and introduce the relevant background material. Then, Section 4 presents our proposed STCChain scheme. The security and performance evaluations of STCChain are presented in Sections 5 and 6, respectively. Finally, Section 7 provides the conclusion.

2 Related Work

In this section, we briefly review several related studies on IIoT data encryption and existing blockchain-based approaches.

Existing approaches include those based on public-key authentication and encryption. For example, [11] proposed a secure channel-free and certificateless searchable multi-keyword public-key encryption scheme (SCF-MCLPEKS) for IIoT deployment and proved the security of the scheme in a random prediction model. [12] proposed a multi-acceptor certificateless encryption with keyword search (CLKS) scheme without pairing and proved that its efficiency is higher than that of the current CLKS scheme and is suitable for cloud-assisted IIoT scenarios. Other examples include those of [13], [14]. Other approaches utilize attribute-based encryption, for example, to facilitate access control that publishes and revokes attributes [15]. Another example is a support online/offline data-sharing framework [16].

Finally, [17] proposed an efficient key management scheme named HKFS-KM to address the issue of high key storage overhead and difficult key management. [18], aiming to address the issue of key escrow in data authentication and the difficulty of implementing the mapping to point hash function and random prediction model, proposed a certificateless signature scheme, but the key storage was centralized.

There have been attempts to integrate both blockchain and IIoT in various contexts. First, for transactions, [19] proposed a decentralized IIoT platform based on blockchain technology named BPIIoT that uses blockchain as a trusted intermediary for transactions between users. [20] proposed a blockchain-based secure energy trading system to address the untrustworthy and opaque nature of the energy market and designed a credit-based payment scheme to reduce the transaction restrictions caused by the delay of transaction confirmation in the energy blockchain. For service provision, [21] proposed a nonrepudiation service provision scheme based on blockchain to solve issues related to an untrusted service provider, a user who refuses to provide the service, or one who uses the service for their own benefit. [22] proposed a blockchain-based IIoT supply chain management system that used the characteristics of blockchain to solve the problem of fair commodity exchange between merchants and suppliers. [23] proposed a lightweight IIoT traffic classification service to classify real-time traffic in the IIoT.

In addition, the use of blockchain to solve IIoT security problems has become a current research hotspot, but most of these studies are aimed at specific security aspects [24]. For system security, [25] proposed a credit-based IIoT device proof-of-work mechanism that used a directed acyclic graph–structured blockchain and proposed a data rights management method to ensure sensitive data confidentiality. For security authentication, [26] proposed a blockchain-based system named BSeIn that combined fine-grained access control to achieve secure mutual authentication, and combined blockchain and attribute-based signatures for anonymous authentication. For secure data sharing, [27] proposed a high-efficiency data based on Ethereum to address issues related to how the intelligent mobile terminal (MT) in the IIoT can achieve high-quality data collection and how to ensure the security of data sharing between MTs. The scheme created a reliable and secure environment using blockchain and deep reinforcement learning. For data storage, [28] proposed a blockchain-based image encryption scheme that encrypted and stored image pixel values in blockchain to ensure the privacy and security of image data, but the scheme relies on a CA to publish certificates.

In conclusion, although academic circles have reported many research results related to the above aspects, many deficiencies remain in using blockchain to protect IIoT key data, especially the lack of research results with threshold encryption protection.

3 Problems and Related Knowledge

3.1 Edge computing

Cloud computing is part of the core infrastructure of IIoT. The IIoT integrates industrial and commercial networks to promote industrial intelligence by uploading data to the cloud for analysis and processing. On the one hand, the security and privacy requirements for edge device data in the IIoT are relatively high; on the other hand, in the era of the Internet of Everything, the number of edge devices in the IIoT has skyrocketed, and the volume of data has exploded. Since the cloud adopts a centralized data processing model, users must upload all data to the cloud before they can use the services it provides. However, this method has several problems, such as unsatisfactory real-time performance, insufficient bandwidth, high energy consumption, and difficulty in ensuring data security and privacy. Edge computing has emerged to address these issues [29], [30].

Unlike cloud computing, which processes data in the cloud, edge computing is a new computing model that can perform calculations at the edge of a network; it has emerged as a supplement to cloud computing. The combination of edge computing and cloud computing provides a better service experience for the IIoT and mobile computing. Edge computing operation includes two aspects, as shown in Figure 1: it can process downlink data from cloud services and it can process uplink data from data sources and consumers.

In summary, edge computing provides three obvious advantages:

- Edge computing processes temporary data at the edge of the network instead of uploading all data to the cloud, so it can reduce network bandwidth pressure.
- Compared to cloud computing, which processes data in the cloud, edge computing processes data very close to the data source, so it offers low latency and can, to a certain extent, improve service response capabilities.
- /With edge computing, storage and computing services can be provided for private data to protect data privacy.

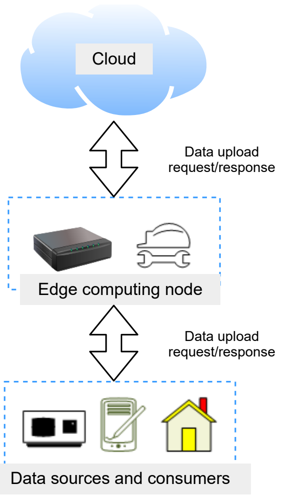

Cloud
Data upload request/response
Edge computing node
Data upload request/response
Data sources and consumers

Fig. 1. Edge computing bidirectional computing flow model.

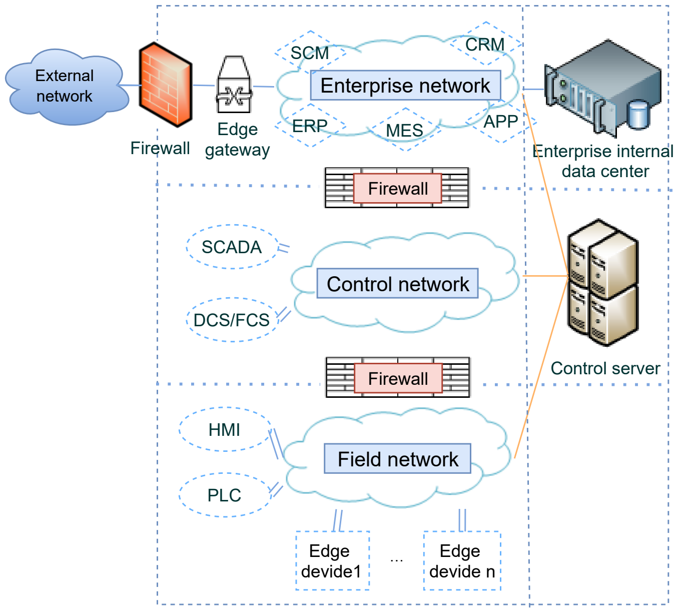


Fig. 2. IIoT network model.

3.2 IIoT Network Model

The IIoT is essentially constituted by the interconnection of commercial and industrial networks and results from the integration of global industrial systems with advanced computing, analysis, and sensing technologies and Internet connections.

As shown in Figure 2, the IIoT model is composed of three main parts—a field network, a control network, and an enterprise network:

- The enterprise network is composed of various systems, such as the manufacturing enterprise management execution system, supply chain management system, customer relationship management (CRM) system, enterprise resource planning and industrial applications.
- The control network is composed of supervisory control and data acquisition, the program logic control system, the fieldbus control system, etc.
- The field network includes the human-machine interface, programmable logic controller, and edge devices.

In the IIoT, data are usually collected by edge devices in the field network, monitored and controlled by the control network, and then stored in the enterprise data center for unified processing. Finally, the data are passed through the edge device or edge gateway to access the external network for operations [31].

3.3 Current IIoT data storage model

Considering the limited storage and computing power of IIoT devices, most enterprises or individuals choose to store IIoT data in the cloud. Once IIoT data are stored in the cloud,

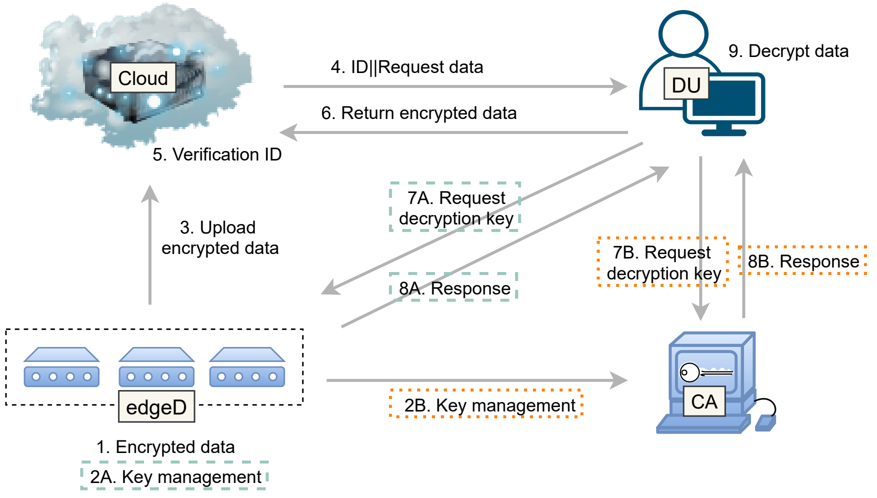


Fig. 3. Current IIoT data storage model.

enterprises or individuals lose absolute management rights over their data. Because the cloud adopts a centralized management mechanism, untrusted clouds or other attackers may use the data illegally, causing privacy disclosure. Data encryption is one of the main ways to ensure the confidentiality of data stored by an IIoT device in the cloud. Data owners, such as edge devices ($edgeD$), usually encrypt data before storage in the cloud, and the decryption key is stored and managed by a user or a trusted third-party organization (such as a CA). When a data user (DU) wants to access the data, the DU must request the decryption key from the user or CA.

We can thus abstract the current IIoT data storage model as shown in Figure 3, which contains three main entities: $edgeD$, $Cloud$, and DU. Regarding the storage of the key, some $edgeD$ will choose to hold the key themselves, as shown in step 2A, while the resource-constrained $edgeD$ will choose to have a trusted third-party organization (such as a CA) escrow the key, as shown in step 2B, and will then obtain the key from the CA when needed.

3.4 Existing Problems

The current IIoT data storage model still has the following two problems:

- **Problem 1**. $edgeD$ storage has limited computing power, and it is difficult to encrypt data autonomously. Generally, the encryption method is composed of a symmetric and an asymmetric key. However, even if symmetric key encryption can provide a lightweight solution for $edgeD$, due to the low capacity and low performance of $edgeD$, autonomous encryption is very difficult.
- **Problem 2**. Centralized key storage is vulnerable to attacks. On the one hand, malicious CA administrators could illegally use $edgeD$ encryption and decryption data keys, thereby stealing $edgeD$ data and causing privacy leakage. On the other hand, an attacker could attack the CA key database and steal $edgeD$ data, thereby destroying data confidentiality and privacy. In addition, $edgeD$ stores the key locally, which is prone to single point of failure and privacy leakage.

3.5 Shamir's Secret Sharing

In some cryptography problems, it is necessary to share secrets s between n users. The Shamir secret sharing (SSS) scheme proposed by Shamir [8] is based on the Lagrange interpolation formula, which requires at least k ($n \geq k$) secret holders to participate to reconstruct s, while $k-1$ participants cannot reconstruct s. This is called the (k,n) threshold secret sharing scheme, where k is the threshold. The process is described as follows:

Step 1. **Initialization**: Let $GF(q)$ be a finite field generated by a large prime q, where $q \geq n+1$. Secret distributor D randomly selects n different $x = \{x_r \neq 0 | r = 1,2,...,n\}$, where x_r is used to represent each secret segment holder $U = \{U_r | r = 1,2,...,n\}$, and then discloses x_r and U_r.

Step 2. **Secret distribution**: D wants to distribute s, where $s \in Z_q | q$ is a large prime number. First, D arbitrarily selects $k-1$ elements within $GF(p)$ to construct $a_i(i=1,2,...,k-1)$ and then constructs the $k-1$ order polynomial:

$$f\left(x\right) = \sum_{i=1}^{k-1} a_0 + a_i x^i (mod \quad q) \tag{1}$$

that is,

$$f\left(x\right) = a_0 + a_1 x + \cdots + a_{k-2} x^{k-2} + a_{k-1} x^{k-1} \tag{2}$$

where $q > s$, q is a large prime number, and $s = f(0) = a_0$. Finally, D generates n secret fragments

$$s_r = f\left(x_r\right) = \sum_{i=1}^{k-1} a_0 + a_i x_r^i (mod \quad q) \tag{3}$$

where $r = 1,2...,n$ and $U_r \in U$. It then sends s_r to U_r.

Step 3. **Secret reconstruction**: any k shadow secret holders can reconstruct s through the Lagrange interpolation formula according to the $(x_i, f(x_i))_{x_i \neq 0 | i=1,2,...,k}$, that is,

$$\begin{aligned} s &= f(0) = a_0 \\ &= (-1)^{k-1} \sum_{i=1}^{k} f(x_i) \cdot \prod_{j=1, j\neq i}^{k} \frac{-x_j}{x_i - x_j} (mod \quad q) \end{aligned} \tag{4}$$

4 STCChain

To solve the problems in Section 3.4, we propose STCChain, a new threshold encryption protection scheme for critical IIoT data based on blockchain. This section details the initialization, secure data storage and secure data reading. The STCChain model is shown in Figure 4, including the following five entities:

- $EdgeG$: Edge gateway, which is responsible for processing the data uploaded by $edgeD$.
- BC: Blockchain, which is open, transparent, tamper-proof, and irreversible. It is the same as the distributed database, and we use it as a key storage database for STCChain.
- $Cloud$: Cloud storage, which provides identity authentication for DU and non-real-time data storage for $EdgeG$. $Cloud$ verifies the identity of the DU and returns $eDataID$ to the user.
- DU: Data users, if DU obtains encrypted data from $Cloud$ and collects a sufficient number of key fragments from BC, the data can be decrypted.
- FH: If the holder of the key fragment of $eDataID$ verifies the identity of the DU, he or she will send his or her owner key fragment to DU.

The overall process is as follows:

Step 1. $edgeD$ uploads the data to $EdgeG$. After $EdgeG$ receives the data, $EdgeG$ generates $eDataID$, uses ks_m (the symmetric key generated by $EdgeG$) to encrypt the data, encrypts ks_m by SK_{pub} (public key in the asymmetric key SK generated by $EdgeG$) and uploads the encrypted data to $Cloud$. Finally, $EdgeG$ uses the SSS algorithm to split the decryption key SK_{pri} (private key of SK) to obtain n key fragments.

Step 2. A smart contract is used to publish key distribution transactions; that is, each key fragment is encrypted with each FH's public key and then published to BC for storage.

Step 3. $Cloud$ uploads $eDataID$ and the encrypted data to $Cloud$.

If DU wants to access resources, first, DU sends an access request to the $Cloud$, $Cloud$ verifies DU, and then $eDataID$ is returned to DU. Next, DU initiates a private key transaction to BC, triggering BC to send a transaction reminder to $FH_i (i = 1, 2, ..., n)$. Each FH_i verifies and then uses a smart contract to publish the key fragment transaction to BC, where each key fragment uses DU's public key for encryption to ensure privacy. Finally, DU collects a sufficient number of key fragments to satisfy $k \leq k' \leq n$ and uses the SSS scheme to synthesize the key and decrypt the data.

Next, we introduce the symbols that will be used later in this paper, as shown in Table 1, and then introduce some important fields and collections, as shown in Table 2.

TABLE 1
Symbol table.

Symbol	Description
E(key,M) D(key,N)	E() encrypts M with the *key*, and D() decrypts N with the *key*, where the key can be a symmetric key or an asymmetric key; that is, the key can be SK or ks.
SK_{User}	$User$'s asymmetric key, $SK_{User} = \{SK_{pubUser}, SK_{priUser}\}$.
SK_C	$Cloud$'s asymmetric key, $SK_C = \{SK_{pubC}, SK_{priC}\}$.
SK_{EdgeD}	$EdgeD$'s asymmetric key, $SK_{EdgeD} = \{SK_{pubEdgeD}, SK_{priEdgeD}\}$.
SK_{DU}	DU's asymmetric key, $SK_{DU} = \{SK_{pubDU}, SK_{priDU}\}$.
SKW	An asymmetric key pair for generating the wallet address, $SKW = \{SKW_{pub}, SKW_{pri}\}$.
ks	Symmetric key.
Addr	Blockchain wallet address, that is, user identity.

TABLE 2
Field and data structure table.

Field/Set	Description
metaDInfo	Metadata information collection, $metaDInfo$={ $edgeDDomain$, $edgeDID$, $metaData$}, where $edgeDDomain$ is the domain of the edge device, $edgeDID$ is the unique identifier of the edge device, $metaData$ are the data.
metaDInfo_S	Encrypted $metaDInfo$ collection.
eDataID	Data unique identifier.
bcaddInfo	Blockchain data information set, $bcaddInfo$={ $actionInfo$, $dataInfo$}, where $actionInfo$ means behavior information, and $actionInfo$={ $storageKeyAct$,$requestKeyAct$,$responseKeyAct$}, $dataInfo$ is specific data information.
tran	Blockchain transaction collection, $tran$={$tranID$,$Addr_{from}$,$Addr_{to}$,$value$,$addData$}. Where $tranID$: transaction number, $Addr_{from}$: transaction sender, $Addr_{to}$: transaction receiver, $value$: transaction value, $addData$: transaction additional information.

4.1 Initialization

First, $EdgeG$, DU and FH must register with BC to become blockchain users. Since the registration process for $EdgeG$, DU and FH is the same, we collectively refer to it as blockchain user registration. Before designing the registration process, we introduce KGen(), AGen(), and SynData(). These three functions are all realized based on BC. For ease of understanding, the formal definition is as follows.

Definition 1. KGen() is a function to generate SKW. The function input is $null$, and its outputs is SKW. For example, DU calls KGen() to generate $SKW_{DU} =< SKW_{pubDU}, SKW_{priDU} >$, and FH calls KGen to generate $SKW_{FH} =< SKW_{pubFH}, SKW_{priFH} >$. $EdgeG$ and $Cloud$ are similar.

Definition 2. AGen(SKW_{pub}) is a function to create

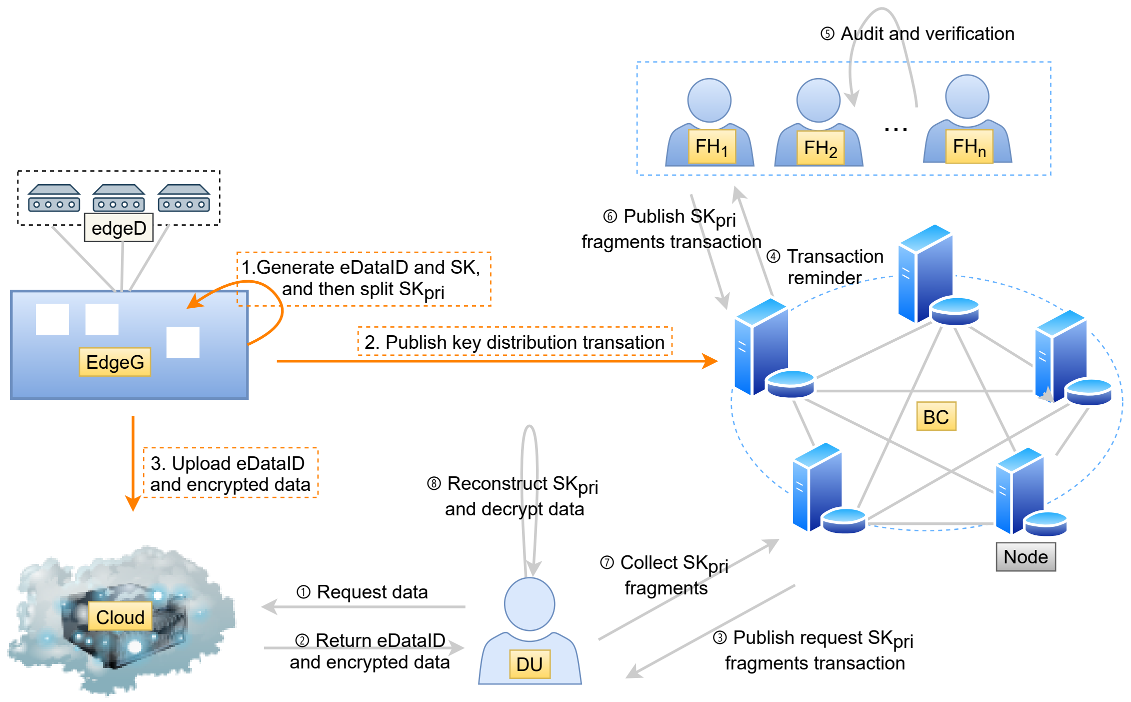

⑤ Audit and verification
FH1
FH2
FHn
edgeD
1.Generate eDataID and SK, and then split SKpri
EdgeG
⑥ Publish SKpri fragments transaction
④ Transaction reminder
2. Publish key distribution transation
BC
3. Upload eDataID and encrypted data
⑧ Reconstruct SKpri and decrypt data
Node
⑦ Collect SKpri fragments
① Request data
Cloud
② Return eDataID and encrypted data
DU
③ Publish request SKpri fragments transaction

Fig. 4. STCChain architecture.

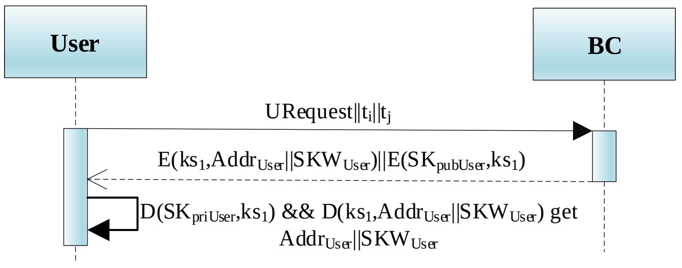

User
BC
URequest||ti||tj
E(ks1,AddrUser||SKWUser)||E(SKpubUser,ks1)
D(SKpriUser,ks1) && D(ks1,AddrUser||SKWUser) get AddrUser||SKWUser

Fig. 5. EdgeG, DU and FH registration process.

a wallet address. The function input is SKW_{pub}, and the output is $Addr$. For example, DU calls AGen(SKW_{pubDU}) to generate $Addr_{DU}$, and FH calls AGen(SKW_{pubFH}) to generate $Addr_{FH}$. $EdgeG$ and $Cloud$ are similar.

Definition 3. SynData(t_i, t_j) is a function to synchronize data from time t_i to t_j. The function inputs are t_i and t_j, and its output is $blockdata_{ij}$.

Next, we design the blockchain user registration process shown in Figure 5.

① $User \rightarrow BC$: $URequest||t_i||t_j$. $User$ sends registration request $URequest||t_|$$|t_j$ to BC.

② $BC \rightarrow User$: E(ks_1, $Addr_{User}||SKW_{User}$)|| E($SK_{pubUser}$, ks_1)||$blockdata_{ij}$. BC calls KGen() to generate $SKW_{User}(SKW_{pubUser}, SKW_{priUser})$ and then calls AGen($SKW_{pubUser}$) to generate $Addr_{User}$. BC uses ks_1 to encrypt $Addr_{User}||SKW_{User}$, and $SK_{pubUser}$ encrypts ks_1 and then sends E(ks_1, $Addr_{User}||SKW_{User}$)||E($SK_{pubUser}$, ks_1)||$blockdata_{ij}$ to $User$. $User$ calls D($SK_{priUser}$, ks_1) and D(ks_1, $Addr_{User}||SKW_{User}$) to obtain $Addr_{User}||SKW_{User}$. Usually, $Cloud$ will automatically call SynData(t_i, t_j) to synchronize $blockdata_{ij}$ to the local database.

4.2 Secure Data Storage by Threshold Encryption

Secure data storage includes decentralized key distribution and encrypted data storage. Before designing the secure data storage process, we first introduce three functions, namely, dataUpload(), SGen(), and ASGen(), and a blockchain transaction interface ISend(). For ease of understanding, the formal definition is as follows.

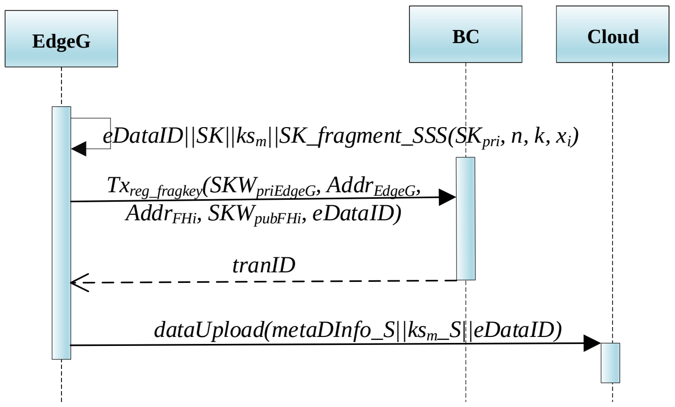
Definition 4. dataUpload($data$) is a function to upload resources from $EdgeG$ to $Cloud$. The function input is $data$, and if the upload is successful, its output is $true$; otherwise, its output is $false$.

Definition 5. SGen() is a function to generate the symmetric key ks_m, which is used to encrypt $metaDInfo$. The function input is $null$, and its output is ks_m.

Definition 6. ASGen() is a function to generate the asymmetric key SK, which is used to encrypt ks_m. The function input is $null$, and its output is ks_m_S.

Definition 7. ISend(SKW_{pri_from}, $Addr_{from}$, $Addr_{to}$, $index$, $content$, $timestamp$) is a blockchain transaction interface used to publish blockchain transactions; that is, $Addr_{from}$ sends a transaction with an asset of $index$ and a transaction value of $content$ to $Addr_{to}$. $Addr_{from}$ is the sender of the transaction, SKW_{pri_from} is the private key of the sender's wallet address (used to sign the transaction), $Addr_{to}$ is the receiver of the transaction, $index$ is the asset unit, $content$ is the asset content, and $timestamp$ is the timestamp. The interface inputs are SKW_{pri_from}, $Addr_{from}$, $Addr_{to}$, $index$, $content$ and $timestamp$, and the output is $tranID$ (transaction number).

We now design the secure data storage process shown in Figure 6, which includes 4 steps.



EdgeG
BC
Cloud
eDataID||SK||ks_m||SK_fragment_SSS(SK_pri, n, k, x_i)
Tx_reg_fragkey(SKW_priEdgeG, Addr_EdgeG, Addr_FHi, SKW_pubFHi, eDataID)
tranID
dataUpload(metaDInfo_S||ks_m_S||eDataID)

Fig. 6. Secure data storage process.

① $EdgeG$: $eDataID||SK||SK\_fragment\_SSS(SK_{pri}$, n, k, $x_i)$. After $EdgeG$ receives the data from the IoT device, $EdgeG$ first generates $eDataID$ for $eDataInfo$, calls system function ASGen() to generate $SK(SK_{pub}$,$SK_{pri})$, and calls SGen to generate ks_m. Then, $EdgeG$ calls $SK\_fragment\_SSS$ to split SK_{pri} into n parts, and each fragment is recorded as $s_i(i = 1, 2, ..., n)$. Next, $EdgeG$ distributes $s_i(i = 1, 2, ..., n)$ to the corresponding FH_i $(i = 1, 2, ..., n)$, who are the fragment holders. Any $k(k \leq n)$ fragments held by the FH can be used to reconstruct SK_{pri}, where $SK\_fragment\_SSS$ is a key division algorithm based on the SSS scheme, as shown in Algorithm 1.

In Algorithm 1:

- Line 1 indicates that for $SK_{pri} \in Z_p$ (p is a large prime number, $p > 2^{128}$), $k-1$ random numbers are randomly selected on the finite group $GF(p)$ and assigned values to $a_1, a_2, ..., a_{k-1}$; among them, n represents the total number of divisions, and k represents the threshold.
- Line 2 lets a_0=SK_{pri} and then substitutes in $a_0, a_1, a_2, ..., a_{k-1}$ to construct a $k-1$ order polynomial (Formula 2), that is, $f(x) = a_0 + a_1x + \cdots + a_{k-2}x^{k-2} + a_{k-1}x^{k-1}$.
- Line 3 initializes an array s of length n+1 to store the divided key fragments.
- Lines 4-8 substitute $x_1, x_2, ..., x_n$ into $f(x)$ to obtain $s_1, s_2, ..., s_n$, where $s_1 = (x_1, f(x_1)), s_2 = (x_2, f(x_2)), ..., s_n = (x_n, f(x_n))$, and then stores s_i into s:
$$s = \{s_1, s_2, ..., s_n\}$$
- Line 9 returns s.

② $EdgeG \rightarrow BC$: $Tx_{reg\_fragkey}(SKW_{priEdgeG}$, $Addr_{EdgeG}$, $Addr_{FHi}$, SKW_{pubFHi}, $eDataID)$. In this step, $EdgeG$ distributes s_i to the corresponding FH_i $(i = 1, 2, ..., n)$. First, $EdgeG$ calls $Tx_{reg\_fragkey}$ to publish the distribution transactions of s_i on BC, where $Tx_{reg\_fragkey}$ is a smart contract. If there are n FH, that is, $FH_i = \{FH_1, FH_2, ..., FH_n\}$, $Tx_{reg\_fragkey}$ is called n times to publish the distribution transactions of the key. $Tx_{reg\_fragkey}$ is shown in Algorithm 2.

In smart contract $Tx_{reg\_fragkey}$:

- Line 1 uses the public key of target FH to encrypt the private key fragment s_i.

Algorithm 1: $SK\_fragment\_SSS$

```
Input: $SK_{pri}$, $n$, $k$
Output: s
1 Randomly select $k-1$ numbers in $GF(p)$ and assign
    them to $(a_i)_{i=1..k-1}$;
2 $f(x) \leftarrow a_0 + a_1x + \cdots + a_{k-2} \cdot x^{k-2} + a_{k-1} \cdot x^{k-1}$
    where $a_0 = SK_{pri}$;
3 Initialize array $s$ with a length of $n+1$ to store the
    split value of $SK_{pri}$;
4 for i=1 to n do
5     Randomly select a number and assign it to $x_i$;
6     $s_i = a_0 + a_1 \cdot x_i + \cdots + a_{k-2} \cdot x_i^{k-2} + a_{k-1} \cdot x_i^{k-1}$;
7     s[i]=$s_i$;
8 end
9 return s;
```

Algorithm 2: Smart Contract $Tx_{reg\_fragkey}$

```
Input: $SKW_{priEdgeG}$, $Addr_{EdgeG}$, $Addr_{FHi}$, $s_i$,
       $SKW_{pubFHi}$, $eDataID$
Output: $tranID$
1 $s_i\_S$ = E($SKW_{pubFH1}$, $s_i$);
2 $bcaddInfo$= storageKeyAct+$s_i\_S$;
3 get current $timestamp$;
4 $tranID$=ISend($SKW_{priEdgeG}$, $Addr_{EdgeG}$,
    $Addr_{FHi}$, $eDataID$, $bcaddInfo$, $timestamp$);
5 return $tranID$;
```

- Line 2 initializes the added value of the blockchain transaction.
- Line 3 obtain the current timestamp.
- Line 4 calls the blockchain publishing interface ISend($SKW_{priEdgeG}$, $Addr_{EdgeG}$, $Addr_{FH_i}$, $eDataID$, $bcaddInfo$, $timestamp$) to send the transaction to BC. $SKW_{priEdgeG}$ represents the private key of the $EdgeG$ wallet address (used for the signature).
- Line 5 returns $tranID_i$ to the corresponding FH_i $(i = 1, 2, ..., n)$.

③ $BC \rightarrow EdgeG$: $tranID$. If the publish transaction is successful, BC will send $tranID$ (transaction number) to $EdgeG$.

④ $EdgeG \rightarrow Cloud$: dataUpload($metaDInfo$_S$||ks_m$_S $||eDataID$). $EdgeG$ uses ks_m to encrypt $metaDInfo$ to obtain $metaDInfo$_S. Meanwhile, SK_{pub} is used to encrypt ks_m to obtain ks_m_S. Finally, $EdgeG$ calls function dataUpload() to upload $metaDInfo$_S$||ks_m$_S$||eDataID$ to $Cloud$.

4.3 Secure Data Access

The secure data reading process includes three main components: reading encrypted data, collecting decentralized stored decryption key fragments and reconstructing the decryption key to decrypt data. Next, we design a secure data reading process, as shown in Figure 7, which includes 7 steps.

① $DU \rightarrow Cloud$: E(ks_2, $eDataID$)||E(SK_{pubC},ks_2). DU sends a read request to $Cloud$.

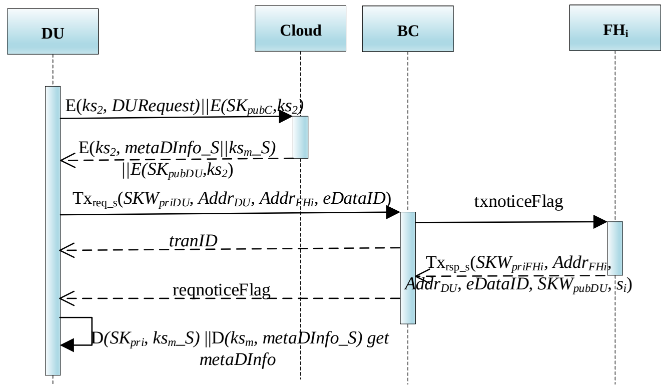

DU
Cloud
BC
FHi
E(ks2, DURequest)||E(SKpubC,ks2)
E(ks2, metaDInfo_S||ksm_S)||E(SKpubDU,ks2)
Txreq_s(SKWpriDU, AddrDU, AddrFHi, eDataID)
txnoticeFlag
tranID
Txrsp_s(SKWpriFHi, AddrFHi, AddrDU, eDataID, SKWpubDU, si)
reqnoticeFlag
D(SKpri, ksm_S) ||D(ksm, metaDInfo_S) get metaDInfo

Fig. 7. Secure data access process.

```
Algorithm 3: Smart Contract $Tx_{req\_s}$
    Input: $SKW_{priDU}$, $Addr_{DU}$, $Addr_{FHi}$, $eDataID$
    Output: $tranID$
1 get current $timestamp$;
2 $bcaddInfo$= requestKeyAct+null;
3 $tranID$=ISend($SKW_{priDU}$, $Addr_{DU}$, $Addr_{FHi}$,
    $eDataID$, $bcaddInfo$, $timestamp$);
4 return $tranID$;
```

② $Cloud \rightarrow DU$: E($ks_2, metaDInfo\_S||ks_m\_S$)||E($SK_{pubDU}, ks_2$). $Cloud$ decrypts to obtain $eDataID$ and then queries the $eDataID$ data, that is, $metaDInfo\_S||ks_m\_S$, and returns the data to DU.

③ $DU \rightarrow BC$: Tx_{req_s}(SKW_{priDU}, $Addr_{DU}$, $Addr_{FHi}$, $eDataID$). DU calls Tx_{req_s} to send a key fragment request transaction to n key holders $FH_i(i = 1, 2, ..., n)$, where Tx_{req_s} is a smart contract, as shown in Algorithm 3.

In smart contract Tx_{req_s}:

- Line 1 obtains the current timestamp.
- Line 2 initializes the additional information stored on blockchain, where $requestKeyAct$ represents the behavior of the requested key fragment.
- Line 3 calls the blockchain publishing interface ISend(SKW_{priDU}, $Addr_{DU}$, $Addr_{FH_i}$, $eDataID$, $bcaddInfo$, $timestamp$) to send the transaction to BC. SKW_{priDU} represents the private key of the DU wallet address (used for the signature).
- Line 4 returns $tranID$ to DU.

④ $BC \rightarrow FH$: $txnoticeFlag$. BC sends a transaction reminder $txnoticeFlag$ to FH_i.

⑤ $FH \rightarrow BC$: Tx_{rsp_s}(SKW_{priFHi}, $Addr_{FHi}$, $Addr_{DU}$, $eDataID$, SKW_{pubDU}, s_i). If FH_i verifies that $Addr_{DU}$ is a user legally requesting private key segments, it calls Tx_{rsp_s} to publish a reply request transaction, as shown in Algorithm 4.

In smart contract Tx_{rsp_s}:

- Line 1 uses SKW_{pubDU} to encrypt s_i.
- Line 2 initializes the additional information stored on blockchain, where $responseKeyAct$ represents the behavior of the response key fragment.
- Line 3 obtains the current timestamp.
- Line 4 calls the blockchain publishing interface ISend(SKW_{priFHi}, $Addr_{FHi}$, $Addr_{DU}$, $eDataID$, $bcaddInfo$, $timestamp$) to send the transaction to BC.
- Line 5 returns $tranID$ to DU.

```
Algorithm 4: Smart Contract $Tx_{rsp\_s}$
    Input: $SKW_{priFHi}$, $Addr_{FHi}$, $Addr_{DU}$, $eDataID$,
           $SKW_{pubDU}$, $s_i$
    Output: $tranID$
1 $s_i$_S′=E($SKW_{pubDU}$, $s_i$);
2 $bcaddInfo$=responseKeyAct + $s_i$_S′;
3 get current $timestamp$;
4 tranID=ISend($SKW_{priFHi}$, $Addr_{FHi}$, $Addr_{DU}$,
    $eDataID$, $bcaddInfo$, $timestamp$);
5 return $tranID$;
```

⑥ $BC \rightarrow DU$: reqnoticeFlag. BC sends the transaction reminder reqnoticeFlag to DU.

⑦ DU: D(SK_{pri}, ks_m_S)||D(ks_m, $metaDInfo\_S$). DU collects at least t transactions and decrypts them to obtain at least t key fragments. $n \geq t \geq k$, where (k, n) is the SSS threshold scheme and k is the threshold value. Any k participants who want to obtain SK_{pri} can use

$$\begin{cases} a_0 + a_1(s_1) + ... + a_{k-1}(s_1)^{k-1} = f(s_1) \\ a_0 + a_1(s_2) + ... + a_{k-1}(s_2)^{k-1} = f(s_2) \\ ... \\ a_0 + a_1(s_k) + ... + a_{k-1}(s_k)^{k-1} = f(s_k) \end{cases}$$

and then substitute the Lagrange interpolation formula (4) to calculate:

$$\begin{aligned} SK_{pri} &= f(0) \\ &= (-1)^{k-1} \cdot \sum_{i=1}^{k} \cdot \prod_{j=1, j \neq i}^{k} \frac{-x_j}{x_i - x_j} (mod \quad q) \end{aligned}$$

Finally, DU calls D(SK_{pri}, ks_m_S)||D(ks_m, $metaDInfo\_S$) to decrypt and obtain $metaDInfo$.

5 Security and Characteristic Analysis

5.1 Characteristic Analysis

1) **High efficiency**. IIoT edge devices usually have low storage capacity and low computing power. Moreover, considering data confidentiality, before outsourcing data to $Cloud$, $EdgeG$ must encrypt data locally before upload it to $Cloud$, so its performance is very inefficient. In STCChain, $edgeD$ only needs to upload data to $EdgeG$, and then high-storage capacity and high-computing power $EdgeG$ will process the data.

2) **Key decentralized storage**. The traditional IIoT data protection solution includes data storage and data reading processes. Usually, $EdgeG$ encrypts and uploads $metaDInfo$ to $Cloud$; then, the decryption key is kept by the user personally or by a trusted third-party institution. Consequently, key storage and management are centralized. In STC-Chain, there is no trusted center. $EdgeG$ encrypts $metaDInfo$ by ks_m to obtain $metaDInfo\_S$ and uses SK_{pub} to encrypt ks_m to obtain ks_m_S; then,

$EdgeG$ uses the SSS algorithm to split the decrypted key SK_{pri} into n fragments $s_i (i = 1, 2, ..., n)$. Finally, $EdgeG$ uses smart contracts to encrypt s_i with the corresponding FH_i's public key and stores it in n FH accounts on blockchain.

5.2 Security Analysis

STCChain has confidentiality, integrity, availability and auditability:

1) **For confidentiality and integrity**.

$$\begin{aligned}
is\_data_{enc} &= E(ks_m, metaDInfo)||E(SK_{pub}, ks_m)\\
is\_data_{dec} &= D(SK_{pri}, ks_m)||D(ks_m, metaDInfo)\\
SK_{pri} &= (-1)^{k-1} \cdot \sum_{i=1}^{k} \cdot \prod_{j=1, j\neq i}^{k} \frac{-x_j}{x_i - x_j} (mod \quad q)\\
is\_comm_{enc} &= E(ks, message)||E(SK_{pub}, ks)\\
is\_comm_{dec} &= D(SK_{pri}, ks)||D(ks, message)\\
is\_s_{enc} &= E(s_i, SKW_{pubFHi})\\
is\_s_{dec} &= D(s_i, SKW_{priFHi})
\end{aligned}$$

First, assume SK_{pri} is divided into n parts in total, and the threshold is t, assuming that the attacker holds k' number of correct s_i.

$$\begin{aligned}
&\because \quad k' < t\\
&\therefore \quad (-1)^{k-1} \cdot \sum_{i=1}^{k'} \cdot \prod_{j=1, j\neq i}^{k'} \frac{-x_j}{x_i - x_j} (mod \quad q) \nrightarrow SK_{pri}\\
&\quad\quad is\_data_{dec} = false
\end{aligned}$$

Thus, only the user who reconstructs SK_{pri} can access data to ensure data confidentiality and integrity.
Second, assume an attacker cannot obtain SK_{priFH}:

$$\begin{aligned}
&\because \quad is\_s_{enc} = true\\
&\quad\quad is\_SK_{priFH} = false\\
&\therefore \quad is\_s_{dec} = false
\end{aligned}$$

Therefore, if SK_{priFH} is not stolen, STCChain can ensure key confidentiality and integrity.

$$\begin{aligned}
&\because \quad is\_comm_{enc} = true\\
&\quad\quad is\_comm_{pri} = false\\
&\therefore \quad is\_comm_{dec} = false
\end{aligned}$$

Thus, STCChain can ensure the confidentiality and integrity of communication.

2) **For auditability**. STCChain splits SK_{pri} using the Shamir threshold and obtains n fragments, that is, s_i (i=1,2,...n). On the one hand, in the key distribution stage, $EdgeG$ uses a smart contract to distribute s_i to the corresponding FH on blockchain. On the other hand, in the key recovery phase, DU uses a smart contract to publish s_i requests on blockchain, and after the FH verification is passed, FH publishes a reply transaction on blockchain. The entire process of key distribution and key recovery is recorded on blockchain, so it can be audited and verified when security incidents or disputes occur.

3) **For availability**. The core of STCChain is that $EdgeG$ encrypts the data uploaded by the edge device with ks_m temporarily generated by $EdgeG$, encrypts ks_m with the SK_{pub} temporarily generated by $EdgeG$, splits SK_{pri} to obtain s_i using the Shamir threshold, and finally encrypts s_i (i=1,2,...,n) and stores it on blockchain. Since blockchain is jointly maintained by the nodes of the entire network, STCChain can work quickly and cannot reject the key request of legitimate users. For example, first, a DU with legal authority requests the key fragment of SK_{pri}, that is, s_i. Then, the DU uses a smart contract to initiate a request to the FH holding s_i, and FH verifies the DU. If verification is passed, FH will publish the corresponding s_i reply transaction to blockchain; finally, if the number of correct s_i collected by the DU is not less than the threshold, SK_{pri} can be successfully reconstructed.

Additionally, STCChain can effectively resist common attacks such as collusion attacks and impersonation attacks.

1) A **collusion attack** refers to two or more malicious parties deceiving others to illegally steal data. STCChain guarantees the decentralized storage of keys based on threshold encryption and blockchain and divides the decryption key into n parts using the SSS scheme. If the threshold is t, assuming there are m colluding personnel, when $m = t$:

$$\begin{aligned}
&\because \quad f(s) = \begin{cases} a_0 + a_1(s_1) + ... + a_{t-1}(s_1)^{t-1} = f(s_1)\\ a_0 + a_1(s_2) + ... + a_{t-1}(s_2)^{t-1} = f(s_2)\\ ...\\ a_0 + a_1(s_t) + ... + a_{t-1}(s_t)^{t-1} = f(s_t)\end{cases}\\
&\because \quad m = t, SK_{pri} = f(0)\\
&\therefore \quad SK_{pri} = (-1)^{t-1} \cdot \sum_{i=1}^{t} \cdot \prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j} (mod \quad q)\\
&\quad\quad is\_data_{dec} = true
\end{aligned}$$

Therefore, STCChain can resist collusion attacks. Only when the attacker collects at least t correct fragments can SK_{pri} be reconstructed to decrypt the data.

2) **Impersonation attack** refers to an attacker pretending to be a legitimate user to obtain data illegally. In STCChain, data encryption is stored in $Cloud$. Assuming that an attacker pretends to be a legal DU, (is_cloud_{verify} represents $Cloud$ verifying the identity of the visitor):

$$\begin{aligned}
&\because \quad is\_cloud_{verify} = true\\
&\therefore \quad data = metaDInfo\_S||ks_m\_S\\
&\because \quad D(SK_{pri}, ks_m\_S) = false\\
&\therefore \quad D(ks_m, metaDInfo\_S) = false
\end{aligned}$$

Thus, it is difficult for an attacker to obtain resources through impersonation attacks.

6 Experiment and Performance Evaluation

6.1 Experiment Environment

We implement the STCChain prototype based on XuperChain. XuperChain is Baidu's self-developed blockchain underlying technology that implements state-of-the-art technologies such as in-chain parallel technology, a pluggable consensus mechanism, and integrated smart contracts, and is characterized by compatibility, strong scalability, and high performance. XuperChain is suitable for the application scenarios of consortium blockchain between enterprises and institutions and public blockchain.

Our experimental environment is Alibaba Cloud Ubuntu 18.04, configured with 2-core, 16 GB and 100 GB storage. We adopt the XuperChain-crypto library module to realize asymmetric data encryption and secret sharing. Finally, we use the XuperChain-xbench platform to conduct performance tests.

In addition, we use the SSS algorithm to divide SK_{pri} into n shares. Assuming that t is the threshold, DU must obtain at least t correct key fragments to reconstruct SK_{pri}. We test STCChain, as shown in Table 3.

It can be seen from Table 3 that if the DU passes the identity verification and holds at least t correct key fragments, the key can be successfully reconstructed, and the data can be successfully decrypted; if the DU passes the identity verification but the number of correct key fragments held is less than t, the key cannot be reconstructed, and the decryption fails. If the DU or SA fails to pass the identity verification, the key fragment cannot be obtained, and even if she or he obtains the encrypted data, the decryption will fail due to lack of a decryption key.

6.2 Performance Evaluation

In this section, we will analyze the performance of STCChain in terms of processing time and throughput.

First, we consider the time overhead of key splitting and reconstruction. STCChain decentralizes decryption key SK_{pri} and uses the SSS algorithm to split SK_{pri} into n shares (t is the threshold). Assuming that DU has collected at least t correct key fragments, the SK_{pri} can be reconstructed. We test the time overhead for the segmentation and reconstruction of SK_{pri}, as shown in Figure 8 and Figure 9.

Figures 8 and 9 show that when $n = 7$ and threshold $t = 3$, the split $time \approx 100$ ms and the reconstruction $time \approx 1$ ms; when $n = 10andt = 6$, the split $time \approx 150$ ms and the reconstruction $time \approx 2$ ms; when $n = 20andt = 12$, the split $time \approx 330$ ms and the reconstruction $time \approx 22$ ms. Thus, the SK_{pri} segmentation and reconstruction time is related to the number of segments n and the size of the threshold t. When the number of divisions n is large, despite the increase in the division and reconstruction time, the total time overhead is not excessive.

The key fragment encryption and decryption time overhead is considered next. STCChain must distribute the decryption key fragments to n accounts on blockchain and uses the public key of key holder FH for encryption. When FH receives the distribution transaction, he or she uses their private key to decrypt the key fragments. We test the time overhead for the encryption and decryption of a single key fragment, as shown in Figure 10 and Figure 11.

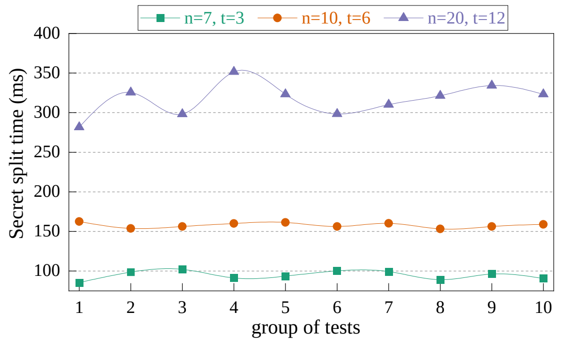


Fig. 8. Time overhead of key split.

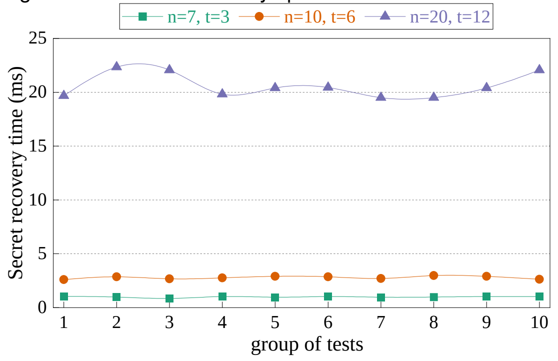


Fig. 9. Key reconstruction time.

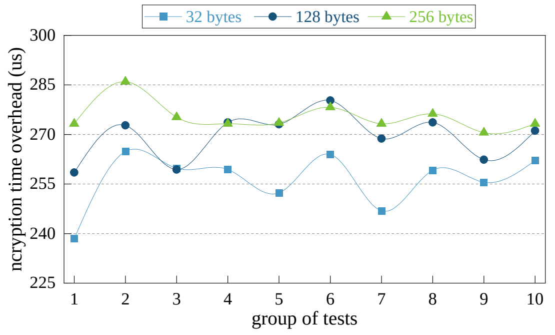


Fig. 10. Encryption time cost of key fragment.

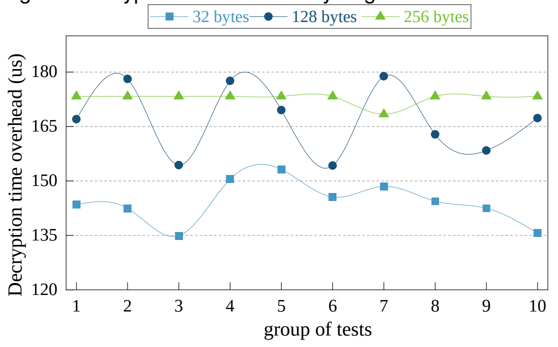


Fig. 11. Decryption time overhead of key fragment.

As shown in Figure 10 and Figure 11, we tested the encryption and decryption time overhead for key segment sizes of 32, 128, and 256 bytes. When the key segment size is 32 bytes, the encryption $time \approx 255$ us and the decryption $time \approx 144$ us; when the key segment size is 128 bytes, the

TABLE 3
Reconstruct the key to decrypt the data result.

User	Identity Authentication	s_i held by $FH_i(i=1,2,...,n, t <= n)$					Reconstruction Key	Decryption Data
		s_1	s_2	$\cdots$	s_{t-1}	s_t		
DU	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
DU	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$
	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$
	$\surd$	The number of correct $s_i < t$					$\times$	$\times$
DU	$\times$	-	-	-	-	-	$\times$	$\times$
SA	$\times$	-	-	-	-	-	$\times$	$\times$

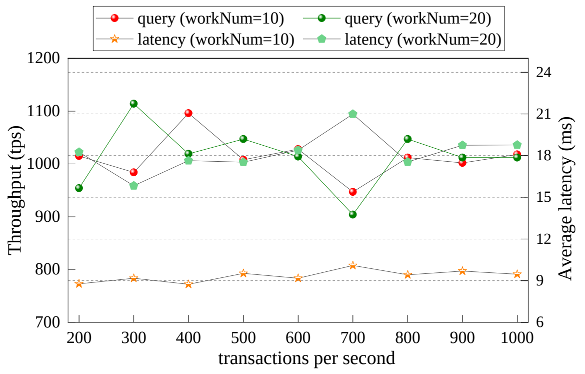


Fig. 12. Throughput overhead.

encryption $time \approx 270$ us and the decryption $time \approx 167$ us; when the key segment size is 256 bytes, the encryption $time \approx 275$ us and the decryption $time \approx 173$ us. Clearly, the encryption and decryption performance is high, but even when the key segment size is 256 bytes, the total time overhead for encryption and decryption is less than 500 us.

Finally, we calculate the read throughput and read latency overhead. Read throughput is a measure of how many read operations are completed within a specified period of time, usually expressed in reads per second, that is, transactions per second (TPS). Read latency is the time between submitting a read request and receiving a reply. We use xbench to test the XuperChain single-node performance. We assess the throughput and latency of 200-1000 transactions, where workNum is the number of threads. As shown in Figure 12, when the number of threads is 10, the read throughput of STCChain is approximately 1015 TPS and the delay is approximately 9 ms; when the number of threads is 20, the read throughput of STCChain is 1013 TPS and the delay is approximately 18 ms. Therefore, the STCChain read throughput performance is also high.

7 Conclusion

The IIoT includes industrial control systems and industrial networks, as well as commercial network infrastructure such as big data storage analysis, cloud computing, business systems, and customer networks. Currently, the IIoT is widely used in manufacturing, logistics, transportation, health care, energy, and utilities. However, the IIoT faces several challenges, among which security and privacy preservation of IIoT data are the most crucial concerns. Moreover, blockchain is a shared database, and the data or information stored in it are characterized as "unforgeable", "remain trace", "traceable", "open and transparent" and "collectively maintained", so it can transform industries by enabling anonymous and trustful transactions in a decentralized and trustless environment. We propose a threshold encryption protection scheme for critical IIoT data based on blockchain: STCChain solves the security and privacy issues caused when the key used for data encryption and decryption is directly stored and managed by users or third-party organizations. The experimental results show that STCChain can not only effectively prevent attackers from stealing data illegally but also protect the privacy of keys.

Potential future directions include the implementation of the proposed scheme in a real-world IIoT setting using the alliance chain fabric, and identify other security and privacy properties that can be addressed.

Acknowledgment

This article was presented in part at the 1st ACM MobiCom Workshop on Artificial Intelligence and Blockchain Technologies for Smart Cities with 6G (6G-ABS'21) [7].

References

[1] K. Yu, M. Arifuzzaman, Z. Wen, D. Zhang, and T. Sato, "A key management scheme for secure communications of information centric advanced metering infrastructure in smart grid," *IEEE transactions on instrumentation and measurement*, vol. 64, no. 8, pp. 2072–2085, 2015.

[2] Z. Guo, K. Yu, A. Jolfaei, A. K. Bashir, A. O. Almagrabi and N. Kumar, "A Fuzzy Detection System for Rumors through Explainable Adaptive Learning," *IEEE Transactions on Fuzzy Systems*, doi: 10.1109/TFUZZ.2021.3052109.

[3] H. Li, K. Yu, B. Liu, C. Feng, Z. Qin and G. Srivastava, "An Efficient Ciphertext-Policy Weighted Attribute-Based Encryption for the Internet of Health Things," *IEEE Journal of Biomedical and Health Informatics*, 2021, doi: 10.1109/JBHI.2021.3075995.

[4] K.-K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial internet-of-things: Research challenges and opportunities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3567–3569, aug 2018.

[5] K. Yu, Z. Guo, Y. Shen, W. Wang, J. C. Lin, T. Sato, "Secure Artificial Intelligence of Things for Implicit Group Recommendations", *IEEE Internet of Things Journal*, 2021, doi: 10.1109/JIOT.2021.3079574.

[6] L. Tan, K. Yu, F. Ming, X. Cheng, G. Srivastava, "Secure and Resilient Artificial Intelligence of Things: a HoneyNet Approach for Threat Detection and Situational Awareness", *IEEE Consumer Electronics Magazine*, 2021, doi: 10.1109/MCE.2021.3081874.

[7] L. Tan, K. Yu, C. Yang, and A. K. Bashir, "A blockchain-based Shamir's threshold cryptography for data protection in industrial internet of things of smart city, " in Proc. *1st ACM MobiCom Workshop on Artificial Intelligence and Blockchain Technologies for Smart Cities with 6G (6G-ABS '21)*, 2021, pp. 13-18, doi: https://doi.org/10.1145/3477084.3484951.

[8] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[9] C. Feng, B. Liu, Z. Guo, K. Yu, Z. Qin, K. -K. R. Choo, “Blockchain-based Cross-domain Authentication for Intelligent 5G-enabled Internet of Drones”, *IEEE Internet of Things Journal*, 2021, doi: 10.1109/JIOT.2021.3113321.

[10] L. Tan, K. Yu, N. Shi, C. Yang, W. Wei and H. Lu, ”Towards Secure and Privacy-Preserving Data Sharing for COVID-19 Medical Records: A Blockchain-Empowered Approach,” *IEEE Transactions on Network Science and Engineering*, doi: 10.1109/TNSE.2021.3101842.

[11] M. Ma, D. He, N. Kumar, K.-K. R. Choo, and J. Chen, “Certificateless searchable public-key encryption scheme for industrial internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 759–767, feb 2018.

[12] Y. Lu, J. Li, and Y. Zhang, “Privacy-Preserving and Pairing-Free Multirecipient Certificateless Encryption With Keyword Search for Cloud-Assisted IIoT,” *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2553–2562, Apr. 2020.

[13] T.-Y. Wu, C.-M. Chen, K.-H. Wang, and J. M.-T. Wu, “Security analysis and enhancement of a certificateless searchable public-key encryption scheme for IIoT environments,” *IEEE Access*, vol. 7, pp. 49 232–49 239, 2019.

[14] H. Xiong et al., “Heterogeneous Signcryption with Equality Test for IIoT environment,” *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[15] D. Ziegler, A. Marsalek, B. Prünster, and J. Sabongui, “Efficient access-control in the iiot through attribute-based encryption with outsourced decryption,” in *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications: SECRYPT*. SciTePress-Science and Technology Publications, 2020.

[16] Y. Miao, Q. Tong, K.-K. R. Choo, X. Liu, R. H. Deng, and H. Li, “Secure Online/Offline Data Sharing Framework for Cloud-Assisted Industrial Internet of Things,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8681–8691, Oct. 2019.

[17] Y. Miao, X. Liu, R. H. Deng, H. Wu, H. Li, J. Li, and D. Wu, “Hybrid keyword-field search with efficient key management for industrial internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3206–3217, jun 2019.

[18] A. Karati, S. H. Islam, and M. Karuppiah, “Provably secure and lightweight certificateless signature scheme for IIoT environments,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3701–3711, aug 2018.

[19] A. Bahga and V. K. Madisetti, “Blockchain platform for industrial internet of things,” *Journal of Software Engineering and Applications*, vol. 9, no. 10, pp. 533–546, 2016.

[20] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, “Consortium blockchain for secure energy trading in industrial internet of things,” *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2017.

[21] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, “A blockchain-based nonrepudiation network computing service scheme for industrial IoT,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3632–3641, jun 2019.

[22] A. Alahmadi and X. Lin, “Towards secure and fair iiot-enabled supply chain management via blockchain-based smart contracts,” in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[23] H. Qi, J. Wang, W. Li, Y. Wang, and T. Qiu, “A Blockchain-Driven IIoT Traffic Classification Service for Edge Computing,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2124–2134, Feb. 2021.

[24] N. Teslya and I. Ryabchikov, “Blockchain platforms overview for industrial iot purposes,” in *2018 22nd Conference of Open Innovations Association (FRUCT)*. IEEE, 2018, pp. 250–256.

[25] J. Huang, L. Kong, G. Chen, M.-Y. Wu, X. Liu, and P. Zeng, “Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, jun 2019.

[26] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, “Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0,” *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, 2018.

[27] C. H. Liu, Q. Lin, and S. Wen, “Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3516–3526, jun 2019.

[28] P. W. Khan and Y. Byun, “A blockchain-based secure image encryption scheme for the industrial internet of things,” *Entropy*, vol. 22, no. 2, p. 175, feb 2020.

[29] F. Ding, G. Zhu, M. Alazab, X. Li, and K. Yu, “Deep-Learning-Empowered Digital Forensics for Edge Consumer Electronics in 5G HetNets”, *IEEE Consumer Electronics Magazine*, doi: 10.1109/MCE.2020.3047606.

[30] L. Liu et al., ”Blockchain-Enabled Secure Data Sharing Scheme in Mobile-Edge Computing: An Asynchronous Advantage Actor–Critic Learning Approach,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2342-2353, 15 Feb.15, 2021, doi: 10.1109/JIOT.2020.3048345.

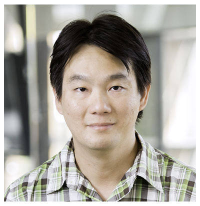
[31] C. Feng, et al., ”Attribute-Based Encryption with Parallel Outsourced Decryption for Edge Intelligent IoV,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13784-13795, Nov. 2020, doi: 10.1109/TVT.2020.3027568.

Keping Yu (Member, IEEE) received the M.E. and Ph.D. degrees from the Graduate School of Global Information and Telecommunication Studies, Waseda University, Tokyo, Japan, in 2012 and 2016, respectively. He was a Research Associate and a Junior Researcher with the Global Information and Telecommunication Institute, Waseda University, from 2015 to 2019 and 2019 to 2020, respectively, where he is currently a Researcher. He is also a Visiting Professor with the College of Computer Science, Sichuan Normal University, Chengdu, China.

Dr. Yu has hosted and participated in more than ten projects, is involved in many standardization activities organized by ITU-T and ICNRG of IRTF, and has contributed to ITU-T Standards Y.3071 and Supplement 35. He received the Best Paper Award from ITU Kaleidoscope 2020, the Student Presentation Award from JSST 2014. He has authored 100+ publications including papers in prestigious journal/conferences such as the IEEE Wireless Communications, ComMag, NetMag, IoTJ, TFS, TII, T-ITS, TVT, TNSE, TGCN, CEMag, IoTMag, ICC, GLOBECOM etc. He is an Associate Editor of IEEE Open Journal of Vehicular Technology, Journal of Intelligent Manufacturing, Journal of Circuits, Systems and Computers. He has been a Lead Guest Editor for Sensors, Peer-to-Peer Networking and Applications, Energies, Journal of Internet Technology, Journal of Database Management, Cluster Computing, Journal of Electronic Imaging, Control Engineering Practice, Sustainable Energy Technologies and Assessments and Guest Editor for IEICE Transactions on Information and Systems, Computer Communications, IET Intelligent Transport Systems, Wireless Communications and Mobile Computing, Soft Computing, IET Systems Biology. He served as general co-chair and publicity co-chair of the IEEE VTC2020-Spring 1st EBTSRA workshop, general co-chair of IEEE ICCC2020 2nd EBTSRA workshop, general co-chair of IEEE TrustCom2021 3nd EBTSRA workshop, session chair of IEEE ICCC2020, TPC co-chair of SCML2020, local chair of MONAMI 2020, Session Co-chair of CcS2020, and session chair of ITU Kaleidoscope 2016. His research interests include smart grids, information-centric networking, the Internet of Things, artificial intelligence, blockchain, and information security.

Liang Tan received a BA degree in computer science from the University of Electronic Science and Technology of China in 2002 and a PhD degree in computer science from the University of Electronic Science and Technology of China in 2007. He is currently a professor at the College of Computer Science, Sichuan Normal University, and a postdoctoral research associate at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include cloud computing, big data, trusted computing and network security. For the TPM 2.0 Key Migration Protocol, he proposed a migration protocol based on Duplication Authority. It uses the Duplication Authority (DA) as an authentication and control center to divide the key migration process into an initialization phase, an authentication and attribute acquisition phase, and a control and execution phase. The DA determines the migration process by the duplicate attributes and types of the migration key and by the handle type of the new parent key. He considered a variety of reasonable combinations of attributes and designed 12 different migration processes before settling on the protocol analyzed and simulated. The results show that the protocol is not only fully compliant with the "TPM-Rev-2.0-Part-1-Architecture-01.38", but also has integrity, confidentiality and supports authentication.

Caixia Yang received the bachelor's degree in engineering from Sichuan Normal University, in 2018, where she is currently pursuing the master's degree in computer science and technology. Her main research direction is in information security.



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the founding co-Editor-in-Chief of ACM Distributed Ledger Technologies: Research & Practice, and founding Chair of IEEE TEMS's Technical Committee on Blockchain and Distributed Ledger Technologies. He also serves as the Department Editor of IEEE Transactions on Engineering Management, the Associate Editor of IEEE Transactions on Dependable and Secure Computing, and IEEE Transactions on Big Data, and the Technical Editor of IEEE Network Magazine. He is an ACM Distinguished Speaker and IEEE Computer Society Distinguished Visitor (2021 - 2023), and included in Web of Science's Highly Cited Researcher in the field of Cross-Field - 2020. In 2015, he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher), the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the British Computer Society's 2019 Wilkes Award Runner-up, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He has also received best paper awards from the IEEE Systems Journal in 2021, IEEE Consumer Electronics Magazine for 2020, Journal of Network and Computer Applications for 2020, EURASIP Journal on Wireless Communications and Networking in 2019, IEEE TrustCom 2018, and ESORICS 2015; the IEEE Blockchain 2019 Outstanding Paper Award; and Best Student Paper Awards from Inscrypt 2019 and ACISP 2005.

Ali Kashif Bashir (Senior Member, IEEE) is a Reader/Associate Professor and Program Leader of BSc (H) Computer Forensics and Security at the Department of Computing and Mathematics, Manchester Metropolitan University, United Kingdom. He is also with School of Electrical Engineering and Computer Science, National University of Science and Technology, Islamabad (NUST) as an Adjunct Professor and School of Information and Communication Engineering, University of Electronics Science and Technology of China (UESTC) as an Affiliated Professor and Chief Advisor of Visual Intelligence Research Center, UESTC. He is a senior member of IEEE, member of IEEE Industrial Electronic Society, member of ACM, and Distinguished Speaker of ACM. His past assignments include Associate Professor of ICT, University of the Faroe Islands, Denmark; Osaka University, Japan; Nara National College of Technology, Japan; the National Fusion Research Institute, South Korea; Southern Power Company Ltd., South Korea, and the Seoul Metropolitan Government, South Korea. He has worked on several research and industrial projects of South Korean, Japanese and European agencies and Government Ministries. In his career, he has obtained over 2.5 Million USD funding. He received his Ph.D. in computer science and engineering from Korea University South Korea. He has authored over 180 research articles; received funding as PI and Co-PI from research bodies of South Korea, Japan, EU, UK and Middle East; supervising/co-supervising several graduate (MS and PhD) students. His research interests include internet of things, wireless networks, distributed systems, network/cyber security, network function virtualization, machine learning, etc. He is serving as the Editor-in-chief of the IEEE FUTURE DIRECTIONS NEWSLETTER. He is also serving as area editor of KSII Transactions on Internet and Information Systems; associate editor of IEEE Internet of Things Magazine, IEEE Access, Peer J Computer Science, IET Quantum Computing, Journal of Plant Disease and Protection. He is leading many conferences as a chair (program, publicity, and track) and had organized workshops in flagship conferences like IEEE Infocom, IEEE Globecom, IEEE Mobicom, etc.

Joel J. P. C. Rodrigues (Fellow, IEEE) is with Senac Faculty of Ceará, Brazil, head of research, development, and innovation; and senior researcher at the Instituto de Telecomunicações, Portugal. He is a collaborator of the Post-Graduation Program on Teleinformatics Engineering at the Federal University of Ceará (UFC), Brazil. Prof. Rodrigues is the leader of the Next Generation Networks and Applications (NetGNA) research group (CNPq). He has authored or coauthored over 1000 papers in refereed international journals and conferences, three books, two patents, and one ITU-T Recommendation. He is a member of the Internet Society and a Senior Member of ACM. He received several outstanding leadership and outstanding service awards from the IEEE Communications Society and several best papers awards. He is an IEEE Distinguished Lecturer, a Member Representative of the IEEE Communications Society on the IEEE Biometrics Council, and the President of the scientific council at ParkUrbis Covilha Science and Technology Park. He was Director of the Conference ˜ Development - IEEE ComSoc Board of Governors, the Technical Activities Committee Chair of the IEEE ComSoc Latin America Region Board, the past Chair of the IEEE ComSoc Technical Committee on eHealth and the IEEE ComSoc Technical Committee on Communications Software, a Steering Committee Member of the IEEE Life Sciences Technical Community, and the Publications Co-Chair. He is also the Editor-in-Chief of the International Journal of E-Health and Medical Communications and an Editorial Board Member of several high-reputed journals. He has been the General Chair and the TPC Chair of many international conferences, including IEEE ICC, IEEE GLOBECOM, IEEE HEALTHCOM, and IEEE LatinCom.

Takuro Sato (Life Fellow, IEEE) was born in Niigata Prefecture, Japan, on January 16, 1950. He received his B.E. and Ph.D. degrees in Electronics Engineering from Niigata University. He has been a member of the Research and Development Laboratories of Oki Electric Industry Co., Ltd., in Tokyo, Japan, where he worked on PCM transmission equipment, mobile telephone technology and the standardization of mobile data transmission and CDMA systems for the international standardization committee. During 1977-1978, Sato developed ATT AMPS (EIA/TIA-553) cellular phone equipment at Oki Electric Industry Co., Ltd. He developed a high-speed cellular modem for the AMPS cellular system in the USA in 1983. This technology was proposed to be standardized in the CCITT (now ITU) SG17 standard. In 1990, he developed a data transmission system based on digital cellular technology. He developed a W-CDMA system named IS-665 under the auspices of TIA for nextgeneration cellular systems. In 1990, the T1P1/TIA Joint Technical Committee (JTC) was organized to evaluate the proposed 2nd-generation, 1.9 GHz, Personal Communications Systems. He proposed W-CDMA, which passed the evaluation tests and became TIA Standard IS-665 and T1P1 Standard JSTD-015 in 1996. He became a Professor in the Department of Information and Electronics Engineering, Niigata Institute of Technology, in 1995. He contributed to the standardization process for IEEE 802.11a. He established the venture company Key Stream to provide LSI integrated circuits for 802.11 wireless LAN systems. In 2004, he became a Professor with the Faculty of Science Engineering at Waseda University. Recently, he has been interested in smart grid technologies in cooperation with ICT systems, including wireless communication. He is also interested in mobile edge computing technologies based on ICN (information-centric networking) and their applications in 5G mobile communication networks. He is a life fellow of the IEEE and IEICE, a fellow of JSST and a member of the Engineering Academy of Japan