A Distributed Authentication Scheme Based on Smart Contract for Roaming Service in Mobile Vehicular Networks

Kaiping Xue, Senior Member, IEEE, Xinyi Luo, Graduate Student Member, IEEE, Yongjin Ma, Jian Li, Member, IEEE, Jianqing Liu, Member, IEEE, David S.L. Wei, Senior Member, IEEE

Abstract—Secure and real-time communication is an essential condition in mobile vehicular networks, and this requires secure authentication and seamless access enabled by roaming services. As a security inspector, roaming authentication ensures that legitimate users can access the network securely. However, today’s roaming authentication protocols authenticate users with the help of centralized authentication servers, leading to the risk of the single point of failure and roaming fraud. The massive device access in 5G networks further exacerbates the losses when problems occur. In light of it, we propose a decentralized fraud-proof roaming authentication framework based on blockchain. We leverage smart contracts to implement a roaming authentication protocol, including user/AP registration, authentication, and revocation. For higher efficiency, we utilize the Bloom filter for the revocation process. In addition, we design an unforgeable and undeniable billing scheme based on hash chain technology. Security and performance analysis show that the proposed roaming authentication scheme can provide the required security features while incurring an acceptable authentication delay.

Index Terms—distributed authentication, roaming authentication, mobile vehicular network, blockchain, smart contract

I. INTRODUCTION

In recent years, with the rapid development of electric vehicles and wireless mobile networks, mobile vehicle networks have gained wide attention and significant development [1, 2]. The emergency of vehicular networks has enabled various new applications, e.g., real-time traffic and road-block analysis, traffic signal control, road examination, city event updates, etc [3, 4]. Seeing that no application can be independent of secure and real-time communication, efficient and secure roaming authentication is indispensable [5]. Besides, several communication types are usually considered in mobile vehicular networks, e.g., vehicle to vehicle (V2V), vehicle to infrastructure (V2I), and vehicle to pedestrian (V2P), and so on. Thus, the roaming and authentication service should be adaptive to heterogeneous networks, including, for example, vehicular networks, wireless networks, and cellular networks. All these factors lead to a dramatic increase in the importance and complexity of roaming services, and the demand for scalability is also increasing. According to Kaleido Intelligence, 5G data roaming traffic generated by mobile devices and IoT applications will exceed 500 Petabytes in 2024 [6]. Thereupon, roaming authentication, as a security barrier for roaming services that guarantees users’ secure access and prevents fraudulent use of services, has gained extensive investigation.

So far, according to the number of participating entities, roaming authentication protocols fall into two categories: three-party roaming authentication schemes, e.g., the work in [7], and two-party roaming authentication schemes, e.g., the work in [8]. A typical three-party roaming authentication scheme involves three participants: a roaming user, a visiting foreign server and a home server. First, a user sends the access credentials to the foreign server, and then the foreign server forwards the credentials to the home server. Finally, the home server verifies the credentials and notifies the foreign server of the result. The three-party scheme relies on the home server’s real-time participation, causing high authentication delays and the risk of a single point of failure. For example, the home server is vulnerable to denial of service (DoS) attacks, which may cause the collapse of the entire roaming system.

In two-party authentication schemes, a foreign server can directly authenticate a roaming user without the participation of the home server, thereby reducing the authentication delay compared with three-party schemes. To achieve this, two-party schemes adopt a public key system for authentication, hence bringing additional overhead for key management such as establishing a public key infrastructure (PKI). Moreover, roaming users have to store the public key of each visited network server for authentication, which is a heavy burden for mobile devices when the number of networks increases. Besides, in practice, roaming partnerships between different service operators are not changeless but dynamically established or revoked. However, on the basis of traditional two-party schemes, a service operator needs to notify all its’ responding users when partnership changes, which brings a significant system overhead. What’s more, two-party schemes generally distinguish foreign users from home users and use different protocols to authenticate different types of users, making the protocols not universal. In a word, although...
two-party schemes are superior to three-party schemes in terms of authentication efficiency, there are still significant shortcomings such as the complexity of the system architecture and inflexibility of partnership changes.

![Diagram](image.png)

**Fig. 1.** Architecture of the mobile vehicular network and the communication types.

With the development of cryptocurrencies such as Bitcoin [9], blockchain technology has been proven both theoretically and practically to be able to guarantee the security of decentralized systems through cryptography and consensus mechanisms. Recently, there have been some schemes that introduce blockchain to build security facilities and implement key technologies [10–12]. Considering the risk of single point of failure of centralized authentication and the semi-trust relationships between different network operators, we introduce blockchain for users and access points (APs) registration, revocation, and authentication. Our main idea is to record the credentials and revocation information on the blockchain, and utilize smart contract to automatically provide users/APs with authentication services, thus achieving distributed and direct mutual authentication between users and APs. It is to be noted that the blockchain has limitations in both storage capabilities and data querying, and meanwhile blockchain confirmation could cause high latency. To this end, we leverage the Bloom filter for storage optimization and mapping tables for efficient querying. In addition to the authentication phase, secure and efficient billing is also worthy of attention. In order to prevent operators from cheating for higher billing revenues or users deceiving to avoid payment, we design a billing scheme based on hash chain technology.

4) We analyze the security of our proposed scheme in theory. In addition, we make a prototype implementation and evaluate the performance. Security and performance analysis show that our proposed scheme provides the required security while incurring an acceptable performance overhead.

**II. RELATED WORK**

Roaming, as a key service of wireless mobile networks, has been widely studied in academia and industry. According to the number of participating entities, roaming authentication protocols can be classified into two categories: three-party roaming authentication schemes and two-party roaming authentication schemes. Three-party roaming authentication schemes require the cooperation of the roaming user, the foreign server, and the home server. In 3G/4G networks, the foreign server forwards a user’s authentication request to the home server, and then the home server authenticates the user based on the pre-shared key mechanism. Aiming at reducing the computation and communication overhead and enhancing the security and robustness of the authentication protocols, researchers have proposed a number of roaming schemes.

In 2004, Zhu and Ma [7] first proposed a three-party roaming authentication scheme. This scheme guarantees one-way authentication of the server to the user, and the session key is unilaterally generated by the user, thus causing a certain security risk. In addition, it cannot guarantee user anonymity. To take action against the problems, Lee et al. [13] proposed a new scheme to enhance security. However, Lee’s scheme does not achieve actual backward security and anonymity. Thus, Wu et al. improved this scheme in [14]. Although Wu’s scheme encrypts user identities, as in [15], the encrypted identities remain unchanged. Hence, the user identities are still traceable. In 2013, Jiang et al. [16] proposed an anonymous roaming protocol based on quadratic residual. A user’s identity is encrypted and transmitted by the pre-stored large integer and quadratic residual algorithm. The server can restore the user’s true identity according to the Chinese remainder theorem. However, this scheme cannot resist replay attacks [17].

In order to enhance anonymity and untraceability of a user’s identity, smart card-based two-factor roaming authentication schemes, e.g., [18–23] were proposed. These schemes not only implement authentication through pre-shared keys, but also further enhance security with short passwords entered by users. Xie et al. [18] first proposed this type of scheme, but Xie’s scheme cannot resist counterfeiting attacks. They therefore proposed a security-enhanced scheme et al. [19]. However, it fails to provide strong user anonymity, and it has inefficient typo-detection. It requires an online verification of the password and cannot be verified on smart cards. Gope et al. [20] used a pseudonym mechanism to enhance user anonymity. However, this scheme does not resist desynchronization attacks and requires smart cards with large storage capacity. Odelu et al. [21] designed a roaming scheme that puts password verification on a smart card, but it also creates password
**TABLE I**

A SUMMARY OF ROAMING AUTHENTICATION SCHEMES

<table>
<thead>
<tr>
<th>Scheme Type</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-party schemes</td>
<td>[7, 13–17]</td>
</tr>
<tr>
<td>smart-card-based schemes</td>
<td>[18–23]</td>
</tr>
<tr>
<td>2-party schemes</td>
<td>[8, 26–30]</td>
</tr>
<tr>
<td>IBE-based schemes</td>
<td>[32–34]</td>
</tr>
<tr>
<td>blockchain-based schemes</td>
<td>[10–12]</td>
</tr>
</tbody>
</table>

**III. Preliminaries**

In this section, We first give a review of background information on blockchain and smart contract. Then, we introduce the principle of Bloom filter. At last, we briefly...
describe the definition of elliptic curve digital signature algorithm.

A. Blockchain and Smart Contract

Blockchain originates from Bitcoin [9], a distributed cryptocurrency proposed by Nakamoto. Nowadays, blockchain has gone beyond the scope of cryptocurrency and becomes a distributed tamper-proof ledger. The ledger is a chronologically ordered chain of blocks and each node of blockchain network holds a copy of the chain to prevent a single point of failure. Each block consists of a block body and a block header. The block body records the transactions or facts, which can be of any type such as token transfer transactions, smart contract transactions, health data, system logs, etc. In addition, the transactions are hashed into a Merkle hash tree. The block head records the root hash of the Merkle tree and the hash of the previous block, and hence those who attempt to tamper with the backdated transactions have to modify the target block and all the following blocks, which is considered difficult due to the consensus mechanism.

As a distributed system, blockchain guarantees consistency and synchronization among nodes through a consensus mechanism, ensuring that all nodes maintain the same copy of the blockchain. The consensus mechanism mainly consists of Proof of Work (PoW) [9, 35], Proof of Stake (PoS) [36, 37], and Practical Byzantine Fault Tolerance (PBFT) [38].

Ethereum [36] expands the functionality of the blockchain to support not only distributed cryptocurrencies but also more complex and flexible smart contracts. An Ethereum smart contract is simply a program that runs on the Ethereum blockchain. Once deployed, the program in the smart contract will be executed honestly. Users can design various complex functions utilizing smart contracts.

The deployment and invocation of a smart contract is shown in Fig. 2. A user sends a transaction to the blockchain via an externally owned account to generate a smart contract and save the address of the contract. All the miners will receive the transaction, and then deploy it in the blockchain through consensus. Anyone who knows the contract address can call the contract through a transaction. The contract will be executed by every miner. The execution results are stored in databases maintained by miners and returned to the caller after consensus.

B. Bloom Filter

A Bloom filter [39] is a space-efficient probabilistic data structure used to check whether an element is a member of a set. In formulation, a Bloom filter is an m-bit array and each bit is initially set to zero. There are k hash functions, \( \text{Hash}_i \), \( 1 \leq i \leq k \), used to map ID to a random number ranging from 1 to m. In the construction process of a Bloom filter, for all \( ID \) in the set, we calculate \( k \) hash values \( \text{Hash}_i(ID) \) for \( i \leq 1 \leq k \) as the index values, and set the mapping values of the corresponding positions in the array to be one. If multiple \( ID \) are mapped to the same position, the corresponding value remains one. In the query process of a Bloom filter, we need to calculate \( \text{Hash}_i(ID) \) for \( 1 \leq i \leq k \) to obtain the index values. If any of the mapping value in the array is zero, the ID is definitely not in the set. Otherwise, it is deemed to be in the set with a certain false positive rate which is shown as follows:

\[
f = (1 - e^{-\frac{m}{n}})^k,
\]

where \( n \) is the number of elements in the data set. For more details, interested readers can refer to [39, 40].

C. Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is an instantiation of digital signature (DSA) by elliptic curve, which is generally specified by the following three algorithms in ANSI standard [41]:

- **EC.Keygen()**: The key generation algorithm generates a key pair for an entity. The key pair is computed under a particular set of elliptic curve domain parameters, which consists of a suitable chosen elliptic curve \( E \) defined over a finite field \( F_q \) of characteristic \( p \), and a base point \( G \in E(F_q) \). To generate the secret/public key pair, the entity first selects a random integer \( d \) mod \( n \), where \( n \) is a sufficiently large prime, then computes \( Q = d \cdot G \). Thus, the key pair is \( (d,Q) \).

- **EC.Sign(d, m)**: The algorithm generates a signature for message \( m \), which is implemented by the following steps:
  1. Select a random integer \( k \) (1 \( \leq k \leq n - 1 \));
  2. Compute \( k \cdot G = (x_1, y_1) \) and \( r = x_1 \) mod \( n \). If \( r = 0 \), go to step (1);
  3. Compute \( k^{-1} \) mod \( n \), \( e = Hash(m) \) and \( s = k^{-1}(e + d \cdot r) \) mod \( n \). If \( s = 0 \), go to step (1);
  4. The signature for message \( m \) is \( \sigma = (r,s) \).

- **EC.Verify(Q, σ)**: The algorithm verifies the signature \( σ \) of \( m \), which is implemented by the following steps:
  1. Verify whether \( r \) and \( s \) are two integers in the interval \([1, n - 1]\). If yes, continue;
  2. Compute \( e = Hash(m) \), \( w = s^{-1} \) mod \( n \), \( u_1 = e \cdot w \) mod \( n \) and \( u_2 = r \cdot w \) mod \( n \);
  3. Compute \( X = u_1 \cdot G + u_2 \cdot Q \). If \( X = O \), reject the signature. Otherwise, compute \( v = x_1 \) mod \( n \) where \( X = (x_1, y_1) \). Accept the signature if and only if \( v = r \). For more details, we refer the interested readers to [41].
IV. System Model, Security Model and Security Requirements

A. System Model

The requirement of global network access for mobile users makes it necessary for wireless mobile networks to provide roaming services. The system model of a roaming service is illustrated in Fig. 3. The system consists of multiple domains, each of which contains a network control center (NCC), a number of access points (APs) and several blockchain nodes. The following illustrates the functions and roles of each entity.

- **NCC** is the management center of a network domain and is responsible for providing roaming services, including user/AP registration, authentication and revocation. It issues credentials for users/APs at the registration stage, provides related information for authentication, and updates revocation records. In addition, NCC deploys and maintains smart contracts to support roaming services.

- **Blockchain nodes** are maintained by their respective network operators, and collectively they form a global consortium blockchain. The blockchain maintains block information, runs smart contracts and verifies transactions.

- **APs**, as the entrance to the network, authenticate users through smart contracts and provides network access services for them.

- **Users** including e-vehicles and cellular users leave the home network to access a foreign network, and obtain the subscribed service through the roaming agreement.

B. Security Model

We assume that the network control center (NCC) is trustworthy for its domain users but is semi-trusted to other NCCs and their domain users. That is to say, an NCC is considered to follow the agreement to provide roaming services to users from other NCCs, but it is possible to misrepresent users’ consumption. It is also assumed impossible for any adversary to compromise NCC. Besides, we assume that there exist a secure channel between APs and blockchain nodes, between NCC and its domain APs, and between NCC and its domain users. The secure channels can be constructed by the TLS or SSL protocol. At last, we assume that a polynomial time adversary, who can modify, interrupt or forge the messages exchanged between users and APs, tries to break the proposed roaming authentication protocol when users access to a foreign network.

C. Security Requirements

Our scheme should satisfy the following security requirements.

- **Mutual Authentication**: The system should have the ability to detect unauthorized users’ access and abort their requests. Meanwhile, users should have the ability to check the legitimacy of any access point.

- **Key Establishment**: A random session key should be negotiated between a user and an AP to ensure the security of subsequent communications between them.

- **Forward/Backward Secrecy**: It requires that the disclosure of the current session key would not affect the security of its future and previous session keys.

- **Revocation Checking**: A roaming user may be revoked by the system (e.g., the user’s subscription period has expired), and thus an AP should be able to find out whether a user is revoked.

- **Unforgeability and undeniability of billing**: The foreign operator cannot forge billing information to charge more fees to the home operator. Users cannot deceive the amount of services received in order to reduce payments to the home operator.

- **Robustness**: Even if the home NCC (HNCC) or the foreign NCC (FNCC) fails, the system can still run stably for a certain period of time.

V. Our Proposed Scheme

In this section, we give a detailed description of our scheme, which mainly consists of five phases: System Initialization, User Authentication, Dynamic User Enrollment and Revocation, Roaming Partnership Establishment and Billing.

A. Overview

Each NCC issues a set of smart contracts for roaming authentication. As illustrated in Fig. 4, the proposed smart contract framework is composed of a main contract (MC), an authentication contract (AC) and a revocation contract (RC). MC records the address of its related AC and a mapping table of other NCCs to their MC addresses. AC consists of two parts. The first part performs the authentication function and the second part stores the address of RC, which is used to verify whether a certificate has been revoked. By recording the addresses of AC and RC into MC, users and APs only need to store MC’s address to complete the authentication process.

During system initialization phase, users and APs register with their HNCC, and the HNCC issues them credentials $CR_U$ or $CR_{AP}$ and the address of HNCC’s MC. When a user accesses a foreign network, he/she generates an access request $M_U$ based on his/her credential $CR_U$ and sends
Then, HNCC computes the credential by:

$$CR_U = EC.Sign(SK_{NCC}, ID_U||PK_U).$$

Finally, HNCC sends \(\{ID_U, ID_{NCC}, SK_U, PK_U, CR_U, MADDR\}\) to the user via a secure channel where MADDR is HNCC’s MC address.

2) AP Registration: APs need to register with HNCC as well. The same as user registration, HNCC generates an ECDSA’s private/public key pairs \((SK_{AP},PK_{AP})\) for the user. Then, HNCC computes the credential by:

$$CR_{AP} = EC.Sign(SK_{NCC}, ID_A||PK_A).$$

Finally, HNCC sends \(\{ID_{AP}, ID_{NCC}, SK_{AP}, PK_{AP}, CR_{AP}, MADDR\}\) to the AP.

C. User Authentication

The user authentication phase is implemented when a mobile user roams to a foreign network and accesses the network for obtaining services. In this phase, the AP and the user need to verify each other’s legitimacy. If the verification is passed, a secure channel can be further established between them. The detailed steps are given as follows. (It’s noted that we mainly focus on roaming authentication scheme in this paper, while authentication for accessing home network can also be achieved by implementing the following authentication processes.)

1) The user firstly generates an access request \(M_U\) and sends it to the corresponding AP. The details for generating \(M_U\) are illustrated in Algorithm 1.

Algorithm 1: Access Request Generation

Input: The user’s identity \(ID_U\), public key \(PK_U\), private key \(SK_U\), credential \(CR_U\), and HNCC’s identity \(ID_{NCC}\);

Output: Access request \(M_{AP}\);

1. Set \(R_U = r_U \cdot G\);
2. Generate timestamp \(ts_U\);
3. Set \(T_U = h(R_U||ts_U)\);
4. Set \(V_U = EC.sign(SK_U, T_U)\);
5. Set the access request message as \(M_{AP} = ID_U||PK_U||CR_U||R_U||ID_{NCC}||V_U||ts_U\);
6. return Access request \(M_{AP}\);

2) Upon receiving \(M_U\), the AP verifies its validity and generates the access response message \(M_{AP}\) by implementing Algorithm 2. Firstly, the AP checks whether the timestamp \(ts_U\) is within an allowed range compared with its current time and then verifies whether \(T_U\) is valid. If both of them are positive, the AP then generates MC’s input as \(TX_{in} = ID_{NCC}||T_U||ID_U||PK_U||CR_U||V_U\), and invokes its HNCC’s MC by the stored MADDR and sends \(TX_{in}\) for user authentication. MC’s process is shown in Algorithm 3. As illustrated in Fig. 4, MC stores \(ID_{NCC}\), AC’s address and the mapping table of \(ID_{NCC}\) to the corresponding MC’s address. If \(ID_{NCC}\) is equal to \(ID_{NCC}\), MC calls AC (Algorithm 4) to verify whether the credential \(CR_{U}\) and signature \(V_U\) is valid. Otherwise, MC finds the mapping of \(ID_{NCC}\) to MC’s address, and call the corresponding MC to verify \(TX_{in}\). If the verification fails, the AP will receive a fail signal, and the access request will be rejected. Otherwise, the AP generates the access response \(M_{AP}\) as shown in Algorithm 2, and computes the session key \(SK = r_{AP} \cdot R_U\). Finally, the AP sends \(M_{AP}\) to the user.

3) Upon receiving \(M_{AP}\), the user’s processing is the same as the AP’s. Specifically, the user first checks the validity of \(ts_{AP}\) and \(T_{AP}\), and then generates MC’s input as \(TX_{in} = ID_{NCC}||T_{AP}||ID_A||PK_A||CR_{AP}||V_{AP}\), and invokes his/her HNCC’s MC and sends \(TX_{in}\) for AP authentication. If the verification fails, the user will receive a false signal, and the access response will be rejected. Otherwise, the user computes the session key by \(SK = r_U \cdot R_{AP}\) and establishes a secure channel with the AP.
Algorithm 2: Access Response Generation

Input: Access request $M_U$, MC’s address $MADDR$;
Output: Access response $M_{AP}$;
1 Check whether timestamp $t_{SU}$ is within an allowed range;
2 Set $T_U' = h(R_U||t_{SU})$;
3 Check whether $T_U$ is equal to $T_U'$;
4 if $t_{SU}$ or $T_U$ is invalid then
5 Reject the access request;
6 return false;
7 else
8 Set $TX_{in} = ID_{NCC}||T_U'||ID_U||PK_U||CR_U||V_U$;
9 Use $TX_{in}$ as input to generate transaction and call HNCC’s main contract;
10 if $MC(TX_{in}) == false$ then
11 Reject the access request;
12 return false;
13 else
14 Select a random number $r_{AP}$;
15 Set $R_{AP} = r_{AP} \cdot G$;
16 Generate timestamp $t_{SP}$;
17 Set $T_{AP} = h(R_{AP}||t_{SP})$;
18 Set $V_{AP} = EC.sign(SK_{AP}, T_{AP})$;
19 Set the access request message as $M_{AP} = ID_{AP}||PK_{AP}||CR_{AP}||R_{AP}||ID_{NCC}||V_{AP}||t_{SP}$;
20 Set $SK = r_{AP} \cdot R_U$;
21 return Access response $M_{AP}$;
22 end

Algorithm 3: Main contract

Input: $TX_{in}$;
Output: true or false;
1 Check whether $ID_{NCC}$ is equal to $ID_{HNCC}$;
2 if $equal$ then
3 $return AC(TX_{in})$
4 end
5 Check whether $ID_{NCC}$ exists in address table;
6 if $inexistent$ then
7 $return$ false;
8 else
9 Find the mapping of $ID_{NCC}$ to MC’s address;
10 $return MC(TX_{in})$
11 end

D. Dynamic User Enrollment and Revocation

Dynamic user enrollment means that the system allows a new user to join at any time after system initialization. This is an indispensable feature for any practical roaming authentication system. In our proposed scheme, when a new user wants to join the system, he/she only needs to perform the registration process to register with the HNCC.

Besides, some users may leave the system halfway due to key loss, illegal usage, etc. The authentication system should support revocation of these users. To this end, HNCC maintains a Bloom filter (RBF) to store all revoked users’ identities $ID_U$. The RBF is preserved as a RBF array in the revocation contract (RC), and can be updated through modifying RC’s variable. HNCC periodically (e.g., daily) updates the RBF based on the latest undo user by invoking RC to change the RBF array. As shown in Algorithm 4, when AC verifies the user’s legitimacy, it invokes RC to check whether the user has been revoked. RC’s procedure is illustrated in Algorithm 5. RC checks every bit in $Hash_i(ID_U)$. If any of them is zero, $ID_U$ is definitively not in the revocation set. Otherwise, it judges that $ID_U$ has been revoked with a certain probability of misjudgment. Considering that some $ID_U$ may be misidentified as revocation, our scheme allows the AP to query the HNCC through the FNCC whether these $ID_U$ have been revoked. It is worth noting that the AP’s enrollment and revocation mechanism is the same as users’.

Algorithm 4: Authentication contract

Input: $TX_{in}$;
Output: True or false;
1 Check the credential $CR_{U/AP}$ by $EC.Verify(PK_{NCC}, CR_{U/AP})$;
2 if $not passed$ then
3 $return$ false;
4 end
5 Check signature $V_{U/AP}$ by $EC.Verify(PK_{U/AP}, V_{U/AP})$;
6 if $not passed$ then
7 $return$ false;
8 else
9 Call $RC$ to check whether the entity has been revoked;
10 if $RC(ID_{U/AP}) == true$ then
11 $return$ false;
12 else
13 $return$ true;
14 end
15 end

Algorithm 5: Revocation contract

Input: $ID_{U/AP}$;
Output: True or false;
1 Check whether $ID_{U/AP}$ is in the RBF;
2 for $i = 1:k$ do
3 if $RBF[Hash_i(ID_U)] == 0$ then
4 $return$ false
5 end
6 end
7 $return$ true

E. Roaming Partnership Establishment

In practice, a roaming user can access a foreign network only if the home and the foreign network operators have
Algorithm 3 thus, when AP belonging to FNCC invokes HNCC’s MC for and FNCC respectively erase the corresponding mapping. As of the roaming partnership is just in a reversed way. HNCC network operators may also cancel the partnership due to trust and interests. Therefore, our scheme is designed to support dynamic establishment and revocation of roaming partnerships.

In our proposed scheme, when two network operators decides to establish a roaming partnership, all they need to do is to update the corresponding items of the mapping table in their MC. Specifically, the HNCC adds the mapping of \( ID_{FNCC} \) to FNCC’s MC address by sending a transaction to the blockchain network, and the FNCC also adds the mapping of \( ID_{HNC} \) to HNCC’s MC address. The revocation of the roaming partnership is just in a reversed way. HNCC and FNCC respectively erase the corresponding mapping. As thus, when AP belonging to FNCC invokes HNCC’s MC for user/AP authentication, as shown in Algorithm 3, if \( ID_{FNCC} \) does not exist in the mapping table, user/AP will know that the HNCC and FNCC have not yet established a roaming partnership and will terminate the access process.

F. Billing

To prevent operators from cheating for higher billing revenues or from users’ evasion of payment, we employ the hash chain technology for billing [8]. Specifically, after the user and the AP have established a secure channel, the AP stores the user’s public key. The user first selects a random integer \( M \) and calculates a hash chain \( h^\tau(M) = h(h(...h(M))) \), where \( \tau \) denotes the maximum service (e.g., data traffic) of one communication session. Then, the user calculates the signature \( \sigma_\tau = EC.Sign(SK_U, h^\tau(M)[|h(ts)|] \), where \( ts \) is a timestamp. The user sends \( (h^\tau(M), ts, \sigma_\tau) \) to the AP. If \( ts \) and \( \sigma_\tau \) are valid, the AP starts to provide network services to the user. After consuming a (pre-agreed) certain amount of service, the user provides the AP with the previous round’s hash value \( v = h^{\tau-1}(M) \). The AP then verifies whether \( h(v) = h^\tau(M) \). If the verification fails, the AP stops the service. Otherwise, The user continues to receive the service and then provides the previous rounds’ hash value periodically. When the session ends, the AP collects a set of hashes \( h^\tau(M), h^{\tau-1}(M), ..., h^{\tau-n+1}(M) \). The AP then sends \( (\sigma_\tau, ts, n, h^{\tau-n+1}(M), h^\tau(M)) \) to NCC as a service provision proof. NCC saves these proofs in its database. At the end of the day, NCC sends all the proofs to HNCC and bills HNCC. HNCC can also charge users based on these proofs.

VI. Security Analysis

In this section, we theoretically analyze the security of our proposed scheme to verify whether the security requirements introduced in Section IV-C have been satisfied. Our analysis includes mutual authentication, key establishment and forward/backward secrecy, revocation checking, unforgeability and undeniability of billing, and resistance to some common attacks.

A. Mutual Authentication

Mutual Authentication means that a user and its accessing AP can verify each other’s authenticity and legitimacy of the identity. Authenticity requires that an adversary cannot disguise as a valid user/AP, and legitimacy requires that the user/AP has registered to the corresponding HNCC and has not been revoked. Taking an AP authenticating a user as an example, the AP authenticates the user by verifying the challenge-response pair \((T_U, V_U)\), where \( T_U = h(R_U)[ts_U] \) and \( V_U = EC.Sign(SK_U, T_U) \). Since the ECDSA has been proven secure under the assumption that the discrete logarithm problem is hard without knowing the private key, it is infeasible to forge a valid signature on the fresh \( T_U \) without \( SK_U \), thereby ensuring the authenticity. Moreover, a valid credential is signed by the HNCC with the private key \( SK_{NCC} \), and \( SK_{NCC} \) is secretly held by NCC, adversaries therefore cannot forge a credential for themselves. And revoked users will be timely recorded to RC. The credential together with RC ensure the legitimacy. Similarly, the user authenticates the AP by the challenge-response pair \((T_A, V_A)\).

B. Key establishment and forward/backward secrecy

In each session, the session key \( SK \) is computed from key negotiation parameters \( R_U = r_u \cdot G \) and \( R_{AP} = r_{AP} \cdot G \). Computing \( SK \) from these two parameters without knowing \( r_u \) and \( r_{AP} \) is equivalent to solve the discrete logarithmic problem (DLP), which is considered computationally infeasible. Therefore, the session key cannot be derived by any adversary. Besides, the key forward/backward secrecy is mainly achieved by the independency of the session key \( SK \) in different sessions. In our scheme, each session uses different fresh \( r_U \) and \( r_{AP} \) for key establishment, leading to the independency of the session keys. As thus, even if an adversary has obtained the current session key, he/she cannot derive the next or the previous session key.

C. Revocation Checking

Secure revocation checking requires that the AP should be able to find out whether a user requesting access has been revoked. Suppose that a revoked user \( RU \) tries to conceal its revocation when accessing an AP, \( RU \) should first generates \( M_{RU} \) and send it to the AP. The AP then calls the authentication contract (AC) to verify the information contained in \( M_{RU} \). Note that, as Algorithm 4 shows, AC will invoke the revocation contract (RC) to check whether \( RU \) has been revoked. Since the revocation is recorded in the smart contract, \( RU \) can only conceal its revocation by modifying RC, which is considered impossible due to the blockchain consensus mechanism.

D. Unforgeability and undeniability of billing

The unforgeability of billing means that the FNCC charges as much for the service it provides. In our proposed scheme, the FNCC bills HNCC by providing the billing proofs \((\sigma_\tau, ts, n, h^{\tau-n+1}(M), h^\tau(M))\). On account of the
hash chain, the FNCC cannot derive $h^{n+1}(M)$ from $h^{n}(M)$ where $n' > n$. The roaming users will not conspire with FNCC to generate more billing proofs, because providing more billing proofs means that users have to pay more to FNCC. Thus, the FNCC cannot charge more than it provides. Besides, since the roaming user should provide $h^{n}(M)$ once finishing the $n$th service, the FNCC can then charge the user accordingly. Therefore, users cannot deny the received services.

E. Resistance to modification attacks

Suppose that an adversary intercepts and modifies a user’s access request $M_U$. If the adversary aims at modifying $R_U$ or $ts_U$, since he/she does not possess the user’s private key $SK_U$, the adversary cannot compute a valid $V_U'$ on a modified message $T'_U$. Therefore, the modified $V_U' = EC.Verify(PK_U, V_U')$ does not pass the signature verification $EC.Verify(PK_U, V_U)$. If the adversary modifies other parameters of $M_U$, since he/she does not possess the private key $SK_{NCC}$, he/she cannot compute a valid $CR_U'$ on a modified user identity. Therefore, the modified $M_U$ cannot pass the credential verification $EC.Verify(PK_{NCC}, CR_U')$. Similarly, for $M_{AP}$, if the adversary modifies $R_{AP}$ or $ts_{AP}$, the modified $M_{AP}$ cannot pass $EC.Verify(PK_{AP}, V_{AP})$. If the adversary modifies other parameters of $M_{AP}$, the modified $M_{AP}$ cannot pass $EC.Verify(PK_{NCC}, CR_{AP})$. As a result, our scheme successfully prevents unauthorized modifications.

F. Resistance to replay attacks

It is noted that the access request message $M_U = ID_U||PK_U||CR_U||R_U||ID_{NCC}||V_U||ts_U$ contains a timestamp $ts_U$, which is included when computing $T_U = h(R_U||ts_U)$. Then, $T_U$ is signed as $V_U = EC.Sign(SK_U, T_U)$. Because of the above steps, the timestamp cannot be modified and replaced. The access response message $M_{AP}$ is also appended by a timestamp $ts_{AP}$ and signed by $V_{AP} = EC.Sign(SK_{AP}, T_{AP})$. The AP will first check if $ts_U$ is within the valid range, and if it expires, the AP rejects the access request. Besides, upon receiving the access response, the user will also check the timestamp $ts_{AP}$. Thus, any replaying message could be recognized by checking the timestamps and signatures. Therefore, the proposed scheme is able to resist unauthorized modifications.

G. Resistance to man-in-the-middle attacks

A man-in-the-middle attacker tries to trick two parties into a three-party communication. In the roaming authentication phase, the attacker intercepts data packets communicated between the user and the AP, and attempts to modify the key negotiation parameters to crack the session key. During the access request, the key negotiation parameter $R_U = r_u \cdot G$ is hashed to $T_U = h(R_U||ts_U)$ and then signed as $V_U = EC.sign(SK_U, T_U)$. Therefore, the key negotiation parameter cannot be modified and replaced by the attacker. The key negotiation parameter $R_{AP} = r_{AP} \cdot G$ in the access response phase is also protected by signature $V_{AP} = EC.sign(SK_{AP}, T_{AP})$. Since the attacker cannot obtain the private key of the user and the AP, he/she cannot modify the key negotiation parameters and thus cannot perform a man-in-the-middle attack. Therefore, our scheme is secure against the man-in-the-middle attacks.

VII. PERFORMANCE ANALYSIS

In this section, we analyze the performance of our scheme in terms of authentication delay, revocation overhead and system fault tolerance. In order to test the performance of our scheme and compare it with other schemes, we measure the time to run basic cryptographic algorithms. In addition, we develop a prototype implementation of our scheme. The blockchain is based on Ganache [42] which is a personal blockchain for Ethereum development. The mobile user’s and the AP’s applications are developed using Javascript(node.js). User and APs interact with blockchain via web3.js 1.0. All experiments are completed with Inter(R) Core(TM) CPU i7-4790 @3.6GHz, 20GB RAM and Windows 7 Professional.
The authentication delay is defined as the total time cost during the whole authentication process, including the time cost of computations and communication delay. TABLE II lists the computation time of related cryptographic operations. The communication delay includes the delay from user to FNCC $T_{U-FNCC}$, the delay from AP to FNCC $T_{AP-FNCC}$, and the delay from FNCC to HNCC $T_{H-FNCC}$. In this scheme, the delay from user or AP to the blockchain nodes is also involved, identified by $T_{U-BLN}$ and $T_{AP-BLN}$. We assume that NCC is deployed in a cloud server. We measure the delay from the user to the cloud server to simulate $T_{U-FNCC}$. We select 23 nodes from major cloud server vendors in China for measurement, and measure the ping delay from user to these nodes. Each node is measured 10 times and we take the average delay as the result. As shown in Fig. 5, The ping delays are between 10ms and 60ms, and the average delay is 35.313ms. On this basis, it is reasonable to set $T_{U-FNCC} = T_{AP-FNCC} = 17.656ms$ and $T_{H-FNCC} = 17.656ms$. As for the delay from user to blockchain nodes, we refer to the distribution of Bitcoin nodes. We crawled 4,967 bitcoin node IPs worldwide and measured the ping delay from the user to these nodes. The result is shown in Fig. 6. Experimental results show that the average ping delay is 226.405ms. Therefore, we set $T_{U-BLN} = T_{AP-BLN} = 113.202ms$.

![Fig. 6. Ping delay from user to Bitcoin node](image)

We first analyze the computation delay of our scheme. In the process of user authentication, the protocol also checks whether the user is revoked at the same time. Therefore, the computation delay includes two parts, the first part is the time-consuming of ordinary cryptographic operations in the roaming authentication process, and the second part is the time-consuming of checking whether the user is revoked. TABLE III shows the comparison of typical schemes [8, 26, 27, 32, 43] in computation delay. As can be seen from TABLE III, in terms of both the ordinary computation delay and the revocation computation delay, our scheme is the lowest. Our scheme only requires 4 elliptic curve point multiplication operations, 2 ECDSA signature operations and 4 ECDSA signature verification operations during the authentication process. The delay for revocation computation includes two parts. The first part is to check whether the user is in the Bloom Filter that stores revocation information. After subsequent analysis, this part only needs to perform 10 hash operations, and the delay of the hash operation is negligible. Therefore, this delay is almost 0. The second part is the delay of HNCC querying the revocation list when a false detection occurs. After analysis, we concludes that the false positive rate is $\alpha = 1.58 \times 10^{-6}$, and more rigorous analysis and proof process will be explained in the following subsections. The false positive rate $\alpha$ is very low, and thus the second part delay is almost zero. It can also be seen from TABLE III that the common computation delay of the related schemes is low, within 100ms, and the revocation computation delay is greatly different. We refer to [27] by assuming that the annual user revocation scale is 1,000,000, and compare the revocation delay on this basis. Considering that the scale of user revocation is gradually increasing, the actual computation delay will be lower than when the user scale is at a peak of 1,000,000. When the number of revoked users is 1,000,000, the revocation computation delay of [8, 26, 27] is measured in minutes, which is beyond the user’s tolerance. The reason is that in order to improve anonymity, these schemes perform complex cryptographic operations such as bilinear mapping for each entry of the revocation list during the revocation check. However, the revocation computation relay of the remaining schemes is in the order of milliseconds.

Then we analyze the communication delay of our scheme. The communication delay includes two parts, which are the communication time in the ordinary authentication process and the communication time in obtaining the revocation list. In the compared related schemes, the user’s revocation list is pushed by the HNCC to the FNCC, and thus the FNCC can verify offline whether a user is revoked. Therefore, this part of other schemes takes zero time. The Bloom Filter revocation mechanism used in this solution has a natural false positive rate. In the case of a misjudgment, the user’s revocation information needs to be obtained from the HNCC, and this part of the communication takes time. TABLE IV shows the communication delay comparison of related schemes. Due to the interaction with the blockchain and using the Bitcoin system as a reference example, our communication delay is greater than the rest. The communication delay of our scheme is around 450ms, but the remaining schemes can guarantee the communication delay within 100ms. In practice, users and APs may choose the nearest blockchain node for authentication, and hence the actual communication delay of our scheme may be lower.

Finally, we analyze the authentication delay of our scheme. It can be seen from TABLE IV that the total authentication delay of our scheme is about 460ms, which is higher than that of the schemes of [32, 43], but far lower than the delay of the schemes of [8, 26, 27]. Overall, It is within the user’s acceptable range.

Further, we build a prototype system of our scheme through the private chain Ganach to analyze the performance. We divide the authentication process into user request, AP contract call, AP response and user contract call. We measure the detailed processing time from four steps as shown in
TABLE II
CRYPTOGRAPHY OPERATION COST

<table>
<thead>
<tr>
<th>Operation Domain</th>
<th>Exponentiation</th>
<th>Multiplication</th>
<th>ECDSA</th>
<th>Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>(G), (G_T)</td>
<td>(T_{exp}), (T_{G_mul})</td>
<td>(T_{E_sign}), (T_{E_verify})</td>
<td>(T_{pair})</td>
</tr>
<tr>
<td>Delay (ms)</td>
<td>1.086</td>
<td>0.131</td>
<td>0.328</td>
<td>1.095</td>
</tr>
</tbody>
</table>

TABLE III
COMPUTATION COST COMPARISON

<table>
<thead>
<tr>
<th>Computation delay (ms)</th>
<th>Revocation Computation delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>[26] (29T_{G_exp} + 11T_{G_exp} + 7T_{pair} + T_{E_sign} + T_{E_verify} = 38.438)</td>
<td>(</td>
</tr>
<tr>
<td>[27] (75T_{G_exp} + 20T_{G_exp} + 4T_{pair} + T_{E_sign} + T_{E_verify} = 94.326)</td>
<td>0.4(</td>
</tr>
<tr>
<td>[8] (16T_{G_exp} + 7T_{G_exp} + 7T_{pair} + T_{E_sign} + T_{E_verify} = 23.796)</td>
<td>(</td>
</tr>
<tr>
<td>[43] (6T_{G_exp} + 2T_{pair} + 4T_{G_mul} + T_{E_sign} + T_{E_verify} = 13.004)</td>
<td>(</td>
</tr>
<tr>
<td>[32] (T_{G_exp} + T_{pair} + 9T_{G_mul} + T_{E_sign} + T_{E_verify} = 12.37)</td>
<td>(</td>
</tr>
<tr>
<td>Ours (4T_{G_mul} + 2T_{E_sign} + 4T_{E_verify} = 6.884)</td>
<td>(\alpha</td>
</tr>
</tbody>
</table>

TABLE IV
AUTHENTICATION DELAY COMPARISON

<table>
<thead>
<tr>
<th>Operation</th>
<th>Total computation delay</th>
<th>Communication delay</th>
<th>Authentication delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>[26]</td>
<td>11.318min</td>
<td>3 ·</td>
<td>(U)-FNCC = 52.968ms</td>
</tr>
<tr>
<td>[27]</td>
<td>7.242min</td>
<td>3 ·</td>
<td>(U)-FNCC = 52.968ms</td>
</tr>
<tr>
<td>[8]</td>
<td>22.633min</td>
<td>3 ·</td>
<td>(U)-FNCC = 52.968ms</td>
</tr>
<tr>
<td>[43]</td>
<td>15.161ms</td>
<td>3 ·</td>
<td>(U)-FNCC = 52.968ms</td>
</tr>
<tr>
<td>[32]</td>
<td>14.531ms</td>
<td>3 ·</td>
<td>(U)-FNCC = 52.968ms</td>
</tr>
<tr>
<td>Ours</td>
<td>6.884ms</td>
<td>4 ·</td>
<td>(A)P-BLN + (\alpha)A)P-HNCC ≈ 452.808</td>
</tr>
</tbody>
</table>

TABLE V
PROCESSING TIME OF EVERY STEP IN AUTHENTICATION

<table>
<thead>
<tr>
<th>Step detail</th>
<th>User</th>
<th>AP</th>
<th>User</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>1.242ms</td>
<td>68.568ms</td>
<td>1.013ms</td>
<td>68.942ms</td>
</tr>
</tbody>
</table>

TABLE V. During the user request phase, the user needs to perform a dot multiplication, a hash and a signature operation and the total time is 1.242 ms. After receiving the user’s access request, the AP processes a hash operation and then calls the MC from the blockchain. It costs 68.566 ms. It is worth noting that the MC will run locally in one of the blockchain nodes in a CALL manner, and no transaction will result in a consensus across the blockchain network. Therefore, it avoids the time of block consensus, such as 10 minutes for Bitcoin and 10 seconds for Ethereum. The AP response phase is similar to the user request phase, except that there is an additional dot multiplication operation. It costs 1.618 ms. Similarly, the user contract call phase has only one more point multiplication operation than the AP contract call phase. It costs 68.942 ms. The total time cost of computations is 140.368 ms, which is still within the user’s acceptable range. However, the result is two orders of magnitude higher than the result shown in TABLE III. The main reason is that the contract calling process needs to perform some operations including cryptographic algorithms in the virtual environment of Ganach, which is exactly the bottleneck. Considering that the actual performance of the blockchain virtual machine is high, and with the development of software technology, this part of the delay will be effectively reduced.

B. Revocation Overhead

During the roaming authentication stage, the AC calls the RC to check whether the user is revoked. The RC then queries the Bloom Filter to feed back the results. Due to the limitation of smart contract capacity, our solution increases the capacity of Bloom Filter by sub-contract storage. Ethereum limits the size of the smart contract to 24KB, we therefore set the size of the RC Bloom Filter to \(20\)KB. When \(m\) is fixed, we analyze the characteristics of the Bloom Filter to find a better setting, i.e., the false positive rate under different numbers of hashes (from 1 to 50) and revocation records (from 1 to 15000). We further analyze the revocation space overhead and compare our scheme with schemes based on revocation list (\(n\)). We further analyze the time overhead and compare our scheme with schemes based on revocation list (\(n\)). Note that the revocation runtime overhead has been provided in Section VII-A.

Fig. 7 shows the relationship between the revocation false positive rate and the revocation entity size, \(n\), and Bloom Filter hash times, \(k\). It can be seen from Fig. 7 that when \(n\) is increased to about 5,000, choosing an appropriate \(k\) can keep the false positive rate at a low value. According to Eq. 1, when the size of the Bloom Filter is fixed, the larger the number of revoked entities, the higher the false positive rate. When \(n\)
between the revocation space and the number of revoked entities. The revocation space based on Bloom Filter increases slowly with the number of revoked entities. For every 5,000 revoked users, our revocation space increases by 20KB. If based on the revocation list mechanism, referring to the paper [27], we set the entity ID size to 16. Compared with our scheme, the revocation space of these schemes is more drastic as the number of revoked entities increases. The required space is about 4 times of that of ours. Considering that the storage space of the blockchain is more valuable, our scheme can better improve the storage performance.

Fig. 8. Relationship between false positive rate and number of hashes

On the basis of the above, we further analyze the revocation performance of our scheme. Fig. 9 shows the relationship between the revocation space and the number of revoked entities stored in an RC Bloom Filter to 5,000. In practice, the scale of revocation entity gradually increases. We have to choose the appropriate \( k \) to ensure a low false positive rate. From Fig. 7, we can see that when \( k = 10 \), the gradually increasing of \( n \) can still maintain a low false positive rate. Then, we select different \( n \) and analyze \( k \). As shown in Fig. 8, when \( n \) is less than 5,000, the false positive rate takes the lowest value at \( k = 10 \), and the lowest value is not much different. When \( n \) is higher than 5,000, the false positive rate can still reach the lowest value in different \( k \) values, but it increases significantly compared to when \( n < 5000 \). A smaller value of \( k \) can improve the efficiency of revoking query. Thus, we set \( k = 10 \) and \( n \) to be less than 5,000. In summary, we set \( m = 20KB \), \( k = 10 \), and make the capacity of an RC Bloom Filter 5,000. Finally, through Eq. 1, the maximum false positive rate is \( \alpha = 1.58 \times 10^{-6} \).

Fig. 7. Change of false positive rate

C. System Fault Tolerance

Our solution enhances the system’s fault tolerance by introducing distributed blockchain nodes. In order to compare the fault tolerance rate of our scheme with other schemes, we assume that the main nodes of the system have the same failure probability, set to \( x \), and assume there are 100 blockchain nodes. We compare the probability of roaming system failure of our proposed scheme with the three-party and two-party schemes under different node failure probabilities.

Fig. 10 shows how the probability of system failure changes as the probability of node failure increases. The main node of the two-party roaming scheme is the foreign server. If the foreign server fails, the roaming system will collapse. Therefore, the system failure probability is \( x \), so the system failure probability increases linearly with the node failure probability. In addition to foreign server, the three-party roaming scheme also has the participation of the home server. Any node failure will cause the system to crash, so the probability of system failure is higher than the two-party scheme. The probability of system failure is \( 1 - (1 - x)^2 = 2x - x^2 \). Our scheme involves blockchain nodes participating in authentication, and any blockchain node can authenticate the users or APs. The roaming system will only collapse if all blockchain nodes fail. When the scale of blockchain nodes reaches a certain level, our system will hardly fail. At a scale of 100 blockchain nodes, the system failure probability is \( x^{100} \). As shown in Fig. 10, when the node failure probability is less than 1, our system failure probability is almost 0. Only when

Fig. 9. Revocation space comparison
the node failure probability is close to 1, the system failure probability is close to 1, which is almost impossible to happen in practice. Therefore, our system has very high stability.

![Graph showing system stability](image)

Fig. 10. Comparison of system fault tolerance

In addition, we analyze the fault tolerance of our roaming system under different node failure probabilities (NFP). We assume NFP={x} and the number of nodes is k, then the roaming system crash probability is \( x^k \). As shown in Fig. 11, as the NFP probability increases, the system quickly reaches high stability (the failure probability is close to 0). But no matter how big the NFP is, even as high as 90%, when the number of blockchain nodes increases to about 50, it will always stabilize. Therefore, our scheme has strong stability and fault tolerance.

![Graph showing system stability](image)

Fig. 11. System stability

VIII. CONCLUSION

In this work, by leveraging blockchain and smart contracts, we designed a distributed and secure roaming mechanism for mobile vehicle networks, which can be directly applied to other mobile network scenarios. In our scheme, we utilized smart contracts to implement roaming protocols including user/AP registration, authentication, and revocation, enabling secure and automatic roaming authentication. Considering blockchain’s limitations on storage and computation, we introduced the Bloom filter to achieve more efficient revocation process. Moreover, we designed an unforgeable and undeniable billing scheme based on hash chain, preventing operators from cheating for higher billing revenues or users from payment evasion. Our security and performance analysis shows that the proposed scheme can provide the required security features while incurring an acceptable authentication delay.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

REFERENCES

Kaiping Xue (M'09-SM'15) received his bachelor’s degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Science and Technology, USTC. His research interests include next generation Internet architecture design, transmission optimization and network security. He has authored and co-authored more than 100 technical papers in various archival journals and conference proceedings. He serves on the Editorial Board of several journals, including the IEEE Transactions of Dependable and Secure Computing (TDSC), the IEEE Transactions on Wireless Communications (TWC) and IEEE Transactions on Network and Service Management (TNSM). He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE Journal on Selected Areas in Communications (JSAC), IEEE Communications Magazine, and IEEE Network. He is an IET Fellow and an IEEE Senior Member.

Jianqing Liu (M'18) received his bachelor’s degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013 and received his Ph.D. degree with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, in 2018. Currently, he is an assistant professor in the Department of Electrical and Computer Engineering at the University of Alabama in Huntsville. His research interests include wireless networking and network security in cyber-physical systems.

Xinyi Luo received her bachelor’s degree in Information Security from the School of the Gifted Young, University of Science and Technology of China (USTC) in July, 2020. She is currently a graduated student in the School of Cyber Science and Technology, USTC. Her research interests include Network security and Cryptography.

Jian Li (M’20) received his B.S. degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and received Ph.D degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. From Nov. 2019 to Nov. 2020, he was a visiting scholar with the Department of Electronic and Computer Engineering, University of Florida. He is currently a Post-Doctoral researcher with the School of Cyber Science and Technology, USTC. His research interests include wireless communications, satellite networks and next-generation Internet.

Xinyi Luo received her bachelor’s degree in Information Security from the School of the Gifted Young, University of Science and Technology of China (USTC) in July, 2020. She is currently a graduated student in the School of Cyber Science and Technology, USTC. Her research interests include Network security and Cryptography.

Yongjin Ma received his bachelor’s degree from the Department of Information Security, USTC, in July, 2017, and the master’s degree in communication and information system from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2020. His research interests include next-generation Internet and mobile network security.

David S.L. Wei (SM’07) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and co-authored more than 130 technical papers in various archival journals and conference proceedings. He is currently a Professor of Computer and Information Science Department at Fordham University. His research current interests include Cloud and Mobile Edge Computing, Cyber Security, Quantum Engineering, and Machine Learning and its Applications. He was a lead guest editor or a guest editor for several special issues in the IEEE Journal on Selected Areas in Communications, the IEEE Transactions on Cloud Computing and the IEEE Transactions on Big Data. He also served as an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018, an Associate Editor of IEEE Transactions on Cloud Computing, 2014-2018 and an Associate Editor of IEEE JSAC for the Series on Network Softwarization & Enablers, 2018-2020.