HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes

Chao Lin, Debiao He, Neeraj Kumar, *Member, IEEE*, Xinyi Huang, Pandi Vijayakumar, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—Increasingly, governments around the world, particularly in technologically advanced countries, are exploring or implementing smart homes, or the related smart facilities for the benefits of the society. The capability to remotely access and control Internet of Things (IoT) devices (e.g., capturing of images, audios, and other information) is convenient but risky, as vulnerable devices can be exploited to conduct surveillance or perform other nefarious activities on the users and organizations. This highlights the necessity of designing a secure and efficient remote user authentication solution. Most of the existing solutions for this problem are generally based on a single-server architecture, which has limitations in terms of privacy and anonymity (leading to users' daily activities being predicted), and integrity and confidentiality (resulting in an unreliable behavior auditing). While blockchain-based solutions may mitigate these issues, they still face some critical challenges (e.g., providing regulation of behaviors and privacy protection of access policy). Motivated by these facts, in this article, we construct a novel secure mutual authentication system, which can be applied in smart homes and other applications. Specifically, the proposed approach integrates blockchain, group signature, and message authentication code to provide reliable auditing of users' access history, anonymously authenticate group members, and efficiently authenticate home gateway, respectively. We also prove the security and privacy requirements, including anonymity, traceability, and confidentiality, that the proposed system satisfies, with an implementation and evaluation to demonstrate its practicality.**

***Index Terms*—Blockchain, Internet of Things (IoT), mutual authentication, smart contract, smart homes.**

Manuscript received July 1, 2019; revised August 30, 2019 and September 21, 2019; accepted September 25, 2019. Date of publication September 30, 2019; date of current version February 11, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61932016, Grant 61822202, Grant 61972294, Grant 61772377, and Grant 61841701, and in part by the Opening Project of Guangdong Provincial Key Laboratory of Data Security and Privacy Protection under Grant B030301004-11. The work of K.-K. R. Choo was supported by the Cloud Technology Endowed Professorship. *(Corresponding author: Debiao He.)*

C. Lin is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: linchao91@qq.com).

D. He is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Guangdong Provincial Key Laboratory of Data Security and Privacy Protection, Guangzhou 510000, China (e-mail: hedebiao@163.com).

N. Kumar is with King Abdul Aziz University, Jeddah, Saudi Arabia, and also with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology (Deemed University), Patiala 147004, India (e-mail: neeraj.kumar@thapar.edu).

X. Huang is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350000, China, and also with the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fuzhou 350000, China (e-mail: xyhuang81@gmail.com).

P. Vijayakumar is with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Villupuram 604001, India (e-mail: vijibond2000@gmail.com).

K.-K. R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA, and also with the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JIOT.2019.2944400

I. Introduction

INTERNET of Things (IoT) is increasingly pervasive and becoming popular in our society for both civilian and military applications [1]. One particular IoT application is in smart homes, or smart cities and smart nations [2], where Internet-connected devices can be remotely accessed or controlled. For example, in a smart home environment, an authorized user can remotely access or control home devices to perform mundane tasks like switching on/off lights, heaters, and washing machines, regardless of the user's geographical location. Such activities can be for the purpose of convenience, leisure, or cost savings, for example, the heater is switched on 30 min remotely before the user returns home, so that the user can return to a heated home in winter. In addition, domestic accidents (e.g., an elderly falling at home) or incidents (e.g., housebreaking) may be avoided via remote monitoring using these devices.

Here, we briefly introduce the communication architecture in a typical smart home environment (see Fig. 1). In the system, residential users use wireless devices (e.g., Android and iOS devices) to remotely communicate with the home gateway via the Internet. Upon receiving relevant messages, such as access or control order from the users, the home gateway executes the corresponding tasks with the relevant home devices. Therefore, the home gateway not only provides network connectivity with the nodes at home but also interconnects the wireless sensor networks formed by various home devices (e.g., sensors installed on the home devices).

As previously discussed, the devices and the (wireless) communication channel can be targeted and subject to exploitation. The challenges of securing the devices and the communications are compounded in resource constrained devices (e.g., smart switches). Consequences include harvesting and exfiltration of sensitive information from these devices [3]–[5]. The information from any single device may not be deemed sensitive, but collectively these data can be extremely revealing. For example, audios and videos covertly collected via the smart

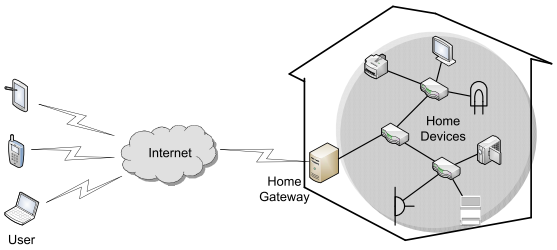


Fig. 1. Example communication architecture in smart home environment.

TV's speakers and camera, and data from the smart locks and smart switches, can be used for stalking, profiling, etc.

This highlights the importance of having in place some secure remote user authentication system, which is also sufficiently lightweight for deployment on resource constrained devices. While there are solutions designed to authenticate nodes in WSNs [6]–[8], in this article, we mainly focus on secure (remote) mutual authentication between users and home gateway.

There are a number of remote authentication approaches. The simplest and the most direct approach is password-based authentication. Such an approach is also inexpensive to implement [9], but it does not protect against direct wiretapping attacks if we do not have a secret, secure channel. One-time password (OTP) authentication [10], one of the most popular methods for two-factor authentication, appears to resist a number of common attacks. Remote user authentication schemes based on smart cards [11] have also been widely used in applications involving remote user login, Web access, and other online services, due to their efficiency, convenience, and low computational cost. However, once an attacker gets a legitimate user's smart card, he/she can impersonate this user to access the remote system [12]. To mitigate such weaknesses, Vaidya *et al.* [12] utilized HOTP [13] and smart cards to implement a strong password-based approach for secure access in smart home devices. However, this comes at a higher computation cost. Recently, several mutual authentication protocols [14]–[16] have been designed for other applications, but almost all of them are based on a single-server architecture. Existing remote authentication approaches based on single-server architectures generally have the following limitations.

1) *Lack of Privacy and Anonymity:* A simple password authentication approach does not protect the user's real identity when they access the smart home services and devices, for a malicious or compromised home gateway can easily predict the user's daily activities.
2) *Lack of Completeness and Confidentiality:* Users' historical access records stored by the home gateway, say in a local database, may be modified without the user's knowledge which may also lead to user privacy leakage.

To solve these limitations, the role of blockchain technology[1] has been highlighted in IoT scenarios due to its properties such as decentralization, verifiability, and immutability. These properties are beneficial in constructing a more secure, reliable, and convenient IoT systems [17]. Hence, a rising number of blockchain-based authentications with anonymity and even confidentiality (e.g., [18]–[20]) have been proposed. However, they still face with the following critical challenges. The inherent anonymity afforded by blockchain may be suitable for IoT and smart homes to keep users private, but in these systems, the user behaviors cannot be effectively regulated. In other words, there is no an efficient mechanism to trace the users who execute malicious requests. Worse still, all the request transactions or access policy are in plaintext, which can be collected to statistically analyze user daily life, and thus, compromising user privacy.

Contributions: To further provide traceability and privacy protection of access policy, we integrate both blockchain and group signature [21] to anonymously authenticate group members, as well as message authentication code [22] to efficiently authenticate home gateway, in our HomeChain. All request records from group members (or revocation requests from the group manager) will be chained into the blockchain. Due to the immutability of blockchain and traceability of group signature, these records are not easy to be tampered or deleted and hence providing reliable behavior auditing. Moreover, we avoid using the access control policy table, but only adopt a revocation list to revoke authorities of malicious users, which efficiently achieves privacy protection of access policy. Specifically, we prove the security properties (including comparison with other related solutions) that our proposal can satisfy and implement a prototype of the proposal on *JUICE*[2] to show its utility.

Organization: We organize the reminder of this article as follows. Section II reviews the current blockchain-based smart home systems. Then in Section III, we present the security requirements of a blockchain-based secure mutual authentication system. Before constructing our HomeChain in Section V, we first describe the adopted cryptographic primitives in Section IV. To show the feasibility of HomeChain, we further give the security analysis and performance evaluation in Sections VI and VII, respectively. Section VIII concludes this article.

II. Related Work

To provide security and privacy for IoT and smart home, there are numerous literatures have proposed. In 2014, Skarmeta *et al.* [23] proposed a distributed access control for protecting privacy of sensitive information, but which causes excessive communication overheads and may reveal user identity information. To provide authentication and privacy, Groß *et al.* [24] introduced a novel method based on IPsec and TLS. However, the expensive computation cost required in [24] is intolerable for resource-limited IoT devices. While Ukil *et al.* [25] proposed a privacy management method for evaluating the risk of sharing data with others, the risk of privacy leakage is often lower than the perceived benefit of IoT services. There, it should be highlighted to provide privacy-aware data sharing without compromising user privacy.

[1]Blockchain is a distributed ledger, where blocks are chained chronologically according to the hash (i.e., the current block contains the hash of the previous block and a timestamp). One cannot change any backdated transaction existing in the blockchain unless he/she can change all blocks subsequent to the modified block. An anonymous node (i.e., a miner) is elected/voted to maintain the blockchain based on a consensus mechanism (e.g., proof of work (PoW), proof of stake (PoS), or PBFT).

[2]*JUICE* is an open permissioned blockchain service platform, https://open.juzix.net/.

Despite many privacy-preserving authentication protocols (e.g., [26]–[28]) for IoT scenarios, these highly centralized solutions have limitations such as difficulty of scalability and single point of failure. Roman *et al.* [29] introduced a modularized method (including data collector, data receiver, and result module) to protect users' privacy in the smart home. Although their proposal can achieve access control of user's data and data unlinkability, it cannot protect user privacy during sharing data with a service provider.

Upon the benefits of distributed IoT, Dorri *et al.* [30] first argued the role of blockchain in providing privacy and security for IoT. They utilized different types of blockchain according to the different network hierarchy a transaction occurs, and adopted distributed trust methods to achieve a decentralized topology. In [31], Dorri *et al.* also summarized the various core components of the smart home tier, together with discussing the corresponding various transactions and procedures, and thus optimizing blockchain in the context of smart homes. Moreover in [32], they proposed an optimized blockchain by eliminating the overhead associated with the classic blockchain but preserving its security and privacy benefits. Their proposal requires no mining and hence avoids additional delays in chaining transactions. In addition, some blockchain-based authentications for IoT have also been proposed, such as [18]–[20]. Although these blockchain-based smart home systems may mitigate the aforementioned security and privacy issues, they still face the critical challenges (i.e., providing regulation of behaviors and privacy protection of access policy).

III. Security Requirements

Based on a review of the literature (e.g., [33] and [34]), the following security requirements are fundamental for a secure blockchain-based mutual authentication system.

Single Registration: To ease the process of remote access/control, the system should provide single registration in the sense that only a one-off registration is needed to issue requesting transactions.

Mutual Authentication: To ease the process of remote access/control, the property of mutual authentication between group user and home gateway is essential.

User Anonymity: For privacy protection, the system should guarantee the anonymity of users. That is, the adversary cannot obtain the real identity of the user by simply analyzing transactions.

Traceability: The system should be capable of tracing the disputed transaction to the original signer.

Session Key Agreement: For security of the message dissemination, the system should support session key agreement. That is, a session key can be established with another device or user upon successful connection.

Perfect Forward Secrecy: For security of previously transmitted message(s), the system should provide perfect forward secrecy. That is, although an attacker has already acquired the private/public keys of both the user and home gateway, it cannot obtain previous session key(s).

No Verifier List: In order to minimize communication overhead and protect against both theft and denial of service attacks, the system should be designed without any verifier list. That is, mutual authentication can be achieved without a verifier list.

No Online Central Registry: To reduce communication overhead and ease mutual authentication, the system should avoid an online central registry.

Relay of Current Timestamp: To establish a reliable blockchain with chronological order, some trustworthy nodes are required to relay current timestamp which will be recorded into blocks.

Resistance of Birthday Collision: To avoid situation where two same blocks are generated simultaneously, the system should resist block birthday collision.

Resistance of Interception and Alteration: To ensure the integrity of transmitted information, the system should protect data-in-transit (e.g., message or transaction) from being intercepted and modified.

Resistance of Hijack: To facilitate a smooth transaction, the system should prevent an attacker from hijacking the transaction.

Resistance of Various Attacks: To establish a secure communication environment, the system should resist various attacks (e.g., user impersonation attack, distributed denial-of-service attack, modification attack, replay attack, and man-in-the-middle attack).

IV. Cryptographic Primitives

A. Public Key Encryption

In our scheme, we use the public key encryption to provide confidentiality of transmitted message including the request transaction data and the response data. Here, we suggest using elliptic curve integrated encryption scheme (ECIES) [35] for our implementation, which not only provides the confidentiality of plaintext using a symmetric cryptographic primitive (e.g., AES) but also authenticates the resulting ciphertext via a MAC (i.e., message authentication codes) tool. It comprises four algorithms, that is, PPG, KGen, Enc, and Dec.

1) *PPG:* This public parameter generation algorithm is given a security parameter λ, it generates public parameters $\text{PP} = \{E, \mathbb{G}, p, q, a, b, P, h\}$, where p and q are large prime numbers of length λ bits, E is a nonsingular elliptic curve defined by $y = x^3 + ax + b \bmod p (a, b \in \mathbb{F}_p)$, $\mathbb{G}$ is an additive cyclic group which consists of all points on E as well as the point at infinity O, P of order q is one of $\mathbb{G}$'s generators, and h is the cofactor (i.e., $h = \#E(\mathbb{F}_p)/q$).
2) *KGen:* This key generation algorithm is given public parameters PP; it randomly chooses $d \xleftarrow{R} \mathbb{Z}_q$ and obtains $Q = dP$. It returns d as the private key; Q as the corresponding public key.
3) *Enc:* This encryption algorithm is given public parameters PP, public key Q, and a plaintext msg; it randomly chooses $k \xleftarrow{R} \mathbb{Z}_q$ and obtains $R = kP$, $Y = hkQ$. If $Y = \infty$, then it rechooses k and repeats the previous operation; otherwise, it continues to compute $(k_1, k_2) \leftarrow \text{KDF}(x_Y, R)$, where KDF is a key derivation algorithm, and x_Y is the x-coordinate of point Y. It also computes

$C = \mathsf{AEnc}_{k_1}(\text{msg})$ and $t = \mathsf{MAC}_{k_2}(C)$, where AEnc is the encryption algorithm of AES and MAC is an MAC algorithm. It finally returns (R, C, t) as the ciphertext of message m.

4) *Dec:* This decryption algorithm is given public parameters PP, private key d, and a ciphertext (R, C, t). It first computes $Y = hdR$, if $Y = \infty$, then rejects this (R, C, t); otherwise, it computes $(k_1, k_2) \leftarrow \mathsf{KDF}(x_Y, R)$. Then, it computes $t' = \mathsf{MAC}_{k_2}(C)$. If $t' \neq t$ then it rejects this ciphertext; otherwise, it computes $\text{msg} = \mathsf{ADec}_{k_1}(C)$, where ADec is the decryption function of the above mentioned symmetric-key encryption scheme (e.g., AES). It finally returns msg as the message of the input ciphertext.

This ECIES scheme has been proven secure, assuming that the AES scheme and the MAC algorithm are secure. Interested readers can refer to [35] for more information.

B. Group Signatures

In a blockchain-based system, using group signatures [21], one can sign transactions to allow a group member to anonymously request a remote access or control. For others, the request is coming from the group (e.g., The University of Texas at San Antonio Group) rather than any individual member. The group manager can perform traceability whenever needed, for instance, to identify a member after he/she has been found to be misbehaving after an investigation has been undertaken. In addition, a group signature scheme with a shorter signature length can reduce the communication overhead. Thus, such a scheme is suitable for deployment in lightweight remote user authentication system. Specifically, we propose using a short group signature scheme such as the scheme presented in [36], for its size of both the group public key and group signature are constant. We now briefly introduce this scheme [36].

1) *Setup:* This initialization algorithm is given a security parameter λ. It first produces system public parameters $\text{PP} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, \mathcal{H}(\cdot))$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of order q (of length λ bits), $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing, P_1 is a generator of $\mathbb{G}_1$ and P_2 is $\mathbb{G}_2$'s one generator, and $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{256}$ is a secure hash function. It then chooses randomness $d, s, u \overset{R}{\leftarrow} \mathbb{Z}_q^*$ to compute $D = d \cdot P_1$, $S = s \cdot P_2$ and $U = u \cdot P_1$. It sets $sk = (d, s)$ as the group manager's private key, $u \overset{R}{\leftarrow} \mathbb{Z}_q^*$ as the tracing key (i.e., $tk = u$), and $gpk = (D, S, U)$ as the group public key. The final return is (PP, sk, tk, gpk).

2) *Enroll:* This enrollment algorithm is given system public parameters PP and the group manager's private key $sk = (d, s)$. It first randomly chooses $x_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and computes $Z_i = (d - x_i)(sx_i)^{-1} \cdot P_1$. Then it computes $\text{tag}_i = \mathcal{H}(x_i \cdot Z_i)$. Finally, it returns $gsk_i = (x_i, Z_i)$ as a group member's private key, and tag_i as its tag. Note that the group manager maintains tag_i in a member list $\text{List} = (\text{ID}_i, GU_i, \text{tag}_i)$, where ID_i is the identity information of member GU_i.

3) *GSign:* This group signing algorithm is given system public parameters PP, a group member's private key gsk_i, the group public key $gpk = (D, S, U)$, and a message msg. It first randomly chooses $k \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and computes $C_1 = k \cdot P_1, C_2 = x_i \cdot Z_i + k \cdot U$, and $Q = e(U, S)^k$. Then it computes $\text{digest} = \mathcal{H}(\text{msg})$, $c = \mathcal{H}(C_1, C_2, Q, \text{digest})$, and $w = kc + x_i$. Finally, it returns the signature (C_1, C_2, c, w) as the signature of msg.

4) *GVerify:* This group signature verification algorithm is given a candidate message/signature pair (msg, σ) and the group key $gpk = (D, S, U)$. It first computes

$$\widehat{Q} = \frac{e(C_2, S) \cdot e(w \cdot P_1, P_2)}{e(c \cdot C_1 + D, P_2)}$$

and $\text{digest} = \mathcal{H}(\text{msg})$. Then it checks that $c \overset{?}{=} \mathcal{H}(C_1, C_2, \widehat{Q}, \text{digest})$ to decide the validity of the (msg, σ).

5) *GTrace:* This tracing algorithm is given a signature $\sigma = (C_1, C_2, c, w)$ and the tracing key $tk = u$. It first computes

$$\text{tag}_i = \mathcal{H}(x_i \cdot Z_i) = \mathcal{H}(C_2 - u \cdot C_1)$$

and searches the $\text{List} = (\text{ID}_i, GU_i, \text{tag}_i)$ to obtain the identity of the signer.

6) *Rev:* This revocation algorithm reveals the group member's private key components $x_i \cdot Z_i$ into a Revocation List revoList to revoke the authority of a group member. Generally, only the group manager who knows the group member's private key components can effectively execute Rev.

In this group signature scheme, the author presented the scheme's security analysis and introduced two group membership revocation methods by revealing group members' private information. Interested readers can refer to [36] for more information.

V. HomeChain: The Proposed System

This section presents the working of our system which is realized upon existing technologies, namely, blockchain and cryptographic primitives such as the group signature and the message authentication code. Here, we suggest using a permissioned type[3] on the basic of the consensus of practical Byzantine fault tolerance (PBFT) [37] for our system. This allows us to achieve hundreds of thousands transactions per second; thus, meeting the design requirements of a blockchain-based mutual authentication system. These consensus nodes could be some servers with sufficient computational and storage costs provided by the relevant families. Before presenting the system design, here, we first introduce the transaction types and the smart contract design in our system.

[3]There are three types (i.e., public, permissioned, and private) of blockchain. In the public type such as Bitcoin and Ethereum, anyone can freely join or quit the generation of blocks upon typical consensus mechanisms, such as PoW and PoS. In the other two, the blockchain is maintained by some reliable nodes (named consensus or permissioned nodes) upon consensus mechanisms, such as PBFT and Raft. The permissioned type allows the users to join after being authorized by consensus nodes, while the private one does not permit anyone to join.

A. Transactions

In a Bitcoin system, a transaction consists of a signature on trade information, including addresses of sender/receiver, and transferred value. In our system, a request transaction consists of version, a fresh public key, device information, and control order, that is, msg = transaction_version‖public_key‖device_information‖control_order. This msg needs to be encrypted by using the targeted home gateway's public key pk_{hg}, and then signed by group signature using user's group private key $gsk[i]$. Assuming that a user, Alice, wishes to switch on the air-conditioner (with heater) in her room before she gets home, she would need to obtain a fresh public/private key pair (pk_a, sk_a), and then compute the transaction $tx = \mathsf{GSign}_{gsk_i}(\mathsf{Enc}_{pk_{hg}}(01\|pk_a\|ac01\|o))$, where 01 is current version, $ac01$ is the air-conditioner identity, o is the control order of "open," Enc denotes the encryption activities using ECIES scheme [35], gpk_{hg} is the targeted home gateway's public key, and gsk_i is her group private key.

B. Smart Contract

Smart contract, together with the ledger, is the key component of some blockchain systems (e.g., Hyperledger Fabric and Ethereum). A smart contract defines some executable logic, which could be complied by a *go* language supported by Hyperledger Fabric or *Solidity* language supported by Ethereum. After that, it will be recorded into the chain as a piece of bytecode in the transaction. Its defined functions and provided application binary interfaces could be triggered through issuing a transaction and a message call from another contract. Notably, this message call will not generate any transaction.

In our system, in order to provide a reliable behavior auditing, we adopt the smart contract to record the requests from users and the responses from home gateways. As shown in Algorithm 1, a user needs to upload its request (including targeted home gateway's public key pk_{hg} and its prepared group signature groupSig) into the smart contract via uploadRequest algorithm. Then, the home gateway is responsible for monitoring the smart contract for a new request and responds to it if the request is valid via invoking uploadResponse algorithm. To release the storage cost of smart contract, the group manager will periodically delete the request list via the deleteRequest algorithm. Note that the consensus nodes will also record the valid requests (based on PBFT consensus mechanism) into the blockchain for a complete auditing.

In addition, to prevent the Revocation List RL in a GS scheme from being maliciously tampered if being stored in a single untrusted server, we use the smart contract to manage RL. Moreover with this RL in the contract, the consensus nodes can easily identify the validity of a transaction. That is, in addition to the correctness of the signature in a transaction, the consensus nodes also check the authority of the issuer by retrieving the RL. A transaction will be regarded as valid if its signature is valid and issuer's private key component (i.e., a GS) has not appeared in the RL. The operations of RL including upload, delete, and read (i.e., addRL, deleteRL, and getRL, respectively) are also shown in Algorithm 1. Note

Algorithm 1 Smart Contract on SmartHome

Require: Function name, invoked parameters
Ensure: *Setting up functions:*
Structure *Req*
% Define the structure of a request from the user.
*uint*256[] *gs*; *% The group signature of a task.*
*uint*256[] *result*; *% The response of a task.*
function SmartHome()
% Constructor, automatically invokes when being deployed.
Manager = *msg.sender*;
mapping(*address* => *Req*) public *reqList*;
*uint*256 [] *revoList*;
function uploadRequest(*address*, *groupSig*)
% Invoked by the user to upload a request.
reqList[*address*].*gs* = *groupSig*;
function uploadResponse(*address*, *state*)
% Invoked by the home gateway to response a task.
require(*msg.sender* == *address*);
reqList[*address*].*result* = *state*;
function view getRequest(*address*)
% Invoked by the home gateway to obtain the request.
require(*msg.sender* == *address*);
return *reqList*[*address*].*gs*;
function view getResult(*address*)
% Invoked by the user to obtain the request.
return *reqList*[*address*].*result*;
function deleteRequest(*address*)
% Invoked by the group manager to clear the request list.
require(*msg.sender* == *Manager*);
reqList[*address*].*gs* = *Null*;
reqList[*address*].*result* = *Null*;
function addRL(*privateInfo*)
% Invoked by the manager to revoke a malicious user.
require(*msg.sender* == *Manager*);
if *revoList*.isNoexist(*privateInfo*) **then**
revoList.push(*privateInfo*);
return 1;
else
return 0;
function deleteRL(*privateInfo*)
% Invoked by the manager to delete the revocation information.
require(*msg.sender* == *Manager*);
r = *revoList*.Find(*privateInfo*);
Delete(*revoList*[*r*]);
function view getRL()
% Invoked by the miner to obtain the list of revocation information.
return *revoList*;

that getRequest, getResult, and getRL are the *view* type of functions, meaning that their invocation does not require any *gas* consumption and transaction confirmation.[4] This reduces

[4]https://solidity.readthedocs.io/en/v0.5.7/contracts.html

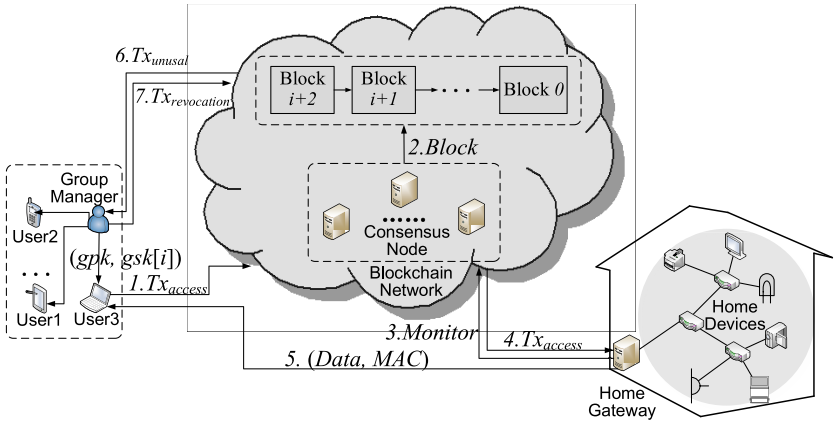


Fig. 2. Blockchain-based secure authentication system architecture. This figure briefly illustrates the main idea of HomeChain. After registering from group manager, a group user can publish its request via TX_{access} into the blockchain. Home gateway monitors the requests recorded in blockchain, and replies the result data to the user. Note that the group manager can trace the real identity in unusual transaction TX_{unusual} and then revoke the malicious user's authority.

the communication delay when the consensus nodes chain a new block.

C. System Design

The proposed system not only provides anonymous authentication for members using a group signature scheme but also authenticates the home gateway using a message authentication code. With this authentication mechanism, members (e.g., those staying in the same household) can remotely access or control home devices without revealing their true identities. Moreover, the freshly chosen key pair and the MAC assure that only the requester is able to receive the response from the legitimate home gateway. Fig. 2 describes the architecture of our design. We will now describe the System Setup, Request Control, State Delivery, Chain Transaction, and Handle Dispute in our system.

1) *System Setup:* For illustration purpose, we assume that family members constitute a group, from which a group manager (maybe one of the parents) is chosen. A group manager invokes Setup and Enroll algorithms to obtain group public key gpk and group private keys $gsk_i (i = 1, 2, \ldots,)$, and PPG in ECIES to generate public parameters. Group members allocate their individual group private keys for signing transactions. Correspondingly, each home gateway holds the group public key for the transaction verification. In addition, home gateway generates his/her public/private key pair (pk_{hg}, sk_{hg}) by invoking the key generation algorithm KGen in ECIES. We denote $\text{MAC} = \text{MAC}_{\text{key}}(\text{message})$ as the MAC generation function, where message is the message, key is the negotiated key, and MAC is the MAC for authenticating message.
2) *Request Control:* When a group member wishes to publish an access or control request with the home gateway, a fresh public/private key pair (pk, sk) is generated. This is the suggested approach to avoid replay attacks and profiling. After obtaining (pk, sk) by invoking the KGen, the group member constructs the transaction from his/her requirement. For example, the group member requests switching off the lights in the room, the transaction computed by GSign algorithm in this case is $TX_{\text{access}} = \text{GSign}_{gsk_i}(\text{Enc}_{pk_{hg}}(01 \| pk \| l01 \| to))$, where 01 is current version, $l01$ is the light identity, to is the control order of "switch off," Enc is the encryption algorithm of ECIES, gpk_{hg} is the target home gateway's public key, and gsk_i is the group member's private key. Then, it uploads this request into the smart contract via uploadRequest.
3) *State Delivery:* The home gateway monitors the smart contract for new requests. Once a group member requests a new access or control service, the request will be retrieved by the home gateway via the getRequest algorithm. If the transaction passes the verification via GVerify algorithm and has not appeared in existing revocation transactions (i.e., is not in the list returned by invoking getRL algorithm), then the home gateway decrypts it via Dec using its private key sk_{hg} to the request information transaction_version ‖public_key‖ device_information‖control_order to obtain the group member's public key, the target device information, and control order. Then, the home gateway connects to the target home device for executing the request. Generally, different sensors are allocated in home devices to perform different tasks or return device status information. Upon receiving the feedback, such as execution result or status information from the target home device, the home gateway encrypts the feedback via the Enc algorithm using the group member's public key pk and computes the corresponding MAC using its private key sk_{hg}. The response is denoted as $\text{Data} = \text{Enc}_{pk}(\text{info})$, where info is the execution result or the device's status information. The corresponding MAC is $\text{MAC} = \text{MAC}_{\text{key}}(\text{Data})$, where $\text{key} = pk^{sk_{hg}}$. As a result, (Data, MAC) is uploaded into the smart contract via the uploadResponse algorithm. f After the response has been received by the group member via getResult, he/she uses his/her private key sk to recompute $\text{MAC}' = \text{MAC}_{\text{key}'}(\text{Data})$, where $\text{key}' = pk_{hg}^{sk}$. If $\text{MAC}' = \text{MAC}$ holds, then it implies that the response is not from an impersonator. Hence, the next step is

decrypting Data via the Dec algorithm using sk to obtain the response information about the request.

4) *Chain Transaction:* Consensus nodes are responsible for retrieving transactions in the smart contract via the getRequest algorithm, and compete with each other for chaining the block to the blockchain. The process is described as follows.
   a) Collect all valid transactions $(Tx_1, Tx_2, \ldots, Tx_n)$ (i.e., signatures are correct and not signed by revoked group members) within the transaction collection period (a certain system period of time). We define invalid transactions to be illegitimate signatures, and such transactions will be discarded.
   b) The consensus nodes use the PBFT consensus mechanism for chaining the valid transactions. Namely, the present recorder first pends some valid transactions into a block. Then, all the consensus nodes reach a consensus on this pending block when there are no less two-third of total consensus nodes approving this block. Hence, this block is finally chained into the blockchain.
5) *Handle Dispute:* Group manager can trace the transaction Tx_{unusual} back to the actual group member when an unusual/abnormal behavior is detected (e.g., frequently changes in device's state or is suspected of maliciously publishing revocation transaction). The group manager can then retrieve the transaction associated with such behavior and invoke the GTrace algorithm for revealing the real identity of the requester, if necessary. Here, the group manager can periodically and selectively revoke illegal group members through invoking the Rev algorithm followed by updating the *revoList* via addRL (as a revocation transaction).

VI. Security Analysis

According to the system model, we define the security model (i.e., anonymity) for HomeChain via a game executed between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$. In the game, $\mathcal{A}$ can make the following queries.

1) *System Setup-Oracle:* $\mathcal{C}$ runs Setup and Enroll to get a group key gpk, group private keys $gsk_i (i = 1, 2, \ldots,)$, and a tracing key tk, and PPG and KGen to obtain public parameters PP and public/private key pair (pk_{hg}, sk_{hg}) (used as home gateway's key pair). Then $\mathcal{C}$ sends $(gpk, \text{PP}, pk_{hg})$ to $\mathcal{A}$.
2) *Request Control-Oracle:* $\mathcal{A}$ can query any request adaptively chosen by it. Specifically, $\mathcal{A}$ prepares a fresh public/private key pair (pk_j, sk_j) and encrypts its request upon the home gateway's public key (i.e., $\text{encData}_j = \text{Enc}(tv||pk_j||di||co)$). After receiving this query, $\mathcal{C}$ randomly chooses a group private key gsk_i to sign it and then replies $TX_{\text{access},j}$ to $\mathcal{A}$.
3) *State Delivery-Oracle:* $\mathcal{C}$ servers as the home gateway upon its generated sk_{hg}, and replies $\text{Data}_j = \text{Enc}_{pk_j}(\text{info}_j)$ and $\text{MAC} = \text{MAC}_{\text{key}_j}(\text{Data}_j)$ (where $\text{key}_j = pk_j^{sk_{hg}}$) to $\mathcal{A}$. Correspondingly, $\mathcal{A}$ verifies the Data_j (upon the MAC) and decrypts it to obtain info_j upon its chosen public/private key pair (pk_j, sk_j).
4) *Chain Transaction-Oracle:* $\mathcal{A}$ can adaptively submit received $TX_{\text{access},j}$ into blockchain. The permissioned nodes will chain its transactions into the blockchain.
5) *Handle Dispute-Oracle:* $\mathcal{A}$ can query this oracle to adaptively trace real identities of transactions. Suppose that it queries a transaction $TX_{\text{access},k}$, then $\mathcal{C}$ will use its tracing key tk to trace $TX_{\text{access},k}$ (i.e., obtaining tag_k), and finally returns tag_k to $\mathcal{A}$.
6) *Test-Oracle:* After $\mathcal{A}$ has queried the above oracles enough times, it invokes this query. That is, it randomly chooses two untraced transactions $(TX^0_{\text{access}}, TX^1_{\text{access}})$ (i.e., $\mathcal{A}$ has never queried them in *Handle Dispute-Oracle*) to $\mathcal{C}$. Then, $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$, and uses its tracing key tk to trace TX^b_{access} (i.e., getting tag_b). After receiving tag_b, $\mathcal{A}$ replies a guess b' to $\mathcal{C}$. If $b' = b$, then $\mathcal{A}$ wins this game; otherwise, it fails.

Here, we denote $\text{Adv}^{\text{An}}_{HC}(\mathcal{A})$ as the advantage that $\mathcal{A}$ wins the above game, that is

$$\text{Adv}^{\text{An}}_{HC}(\mathcal{A}) = \left|\Pr\left(b' = b\right) - 1/2\right|.$$

Definition 1: A HomeChain for smart home is anonymous if and only if $\text{Adv}^{\text{An}}_{HC}(\mathcal{A})$ is negligible for any polynomial attacker $\mathcal{A}$.

On the basic of the security model defined above, we first prove the anonymity of our HomeChain as follows.

Theorem 1: The proposed HomeChain for smart home is with anonymity if the adopted group signature is anonymous.

Proof: Assuming that there exists an attacker $\mathcal{A}$ that could distinguish two transactions $(TX^0_{\text{access}}, TX^1_{\text{access}})$ with a non-negligible advantage $\text{Adv}^{\text{An}}_{HC}(\mathcal{A})$, then, we can construct another attacker $\mathcal{B}$ to break anonymity of the group signature. Given a group signature oracle $gs \leftarrow \mathcal{O}_{GS}(m, \cdot)$, a tracing oracle $\text{tag} \leftarrow \mathcal{O}_{\text{Trace}}(gs, \cdot)$, a test oracle $\mathcal{O}_{\text{Test}}(gs_0, gs_1, \cdot)$, and a group public key gpk, $\mathcal{B}$ simulates the following oracles queried by $\mathcal{A}$.

1) *System Setup-Oracle:* $\mathcal{B}$ runs PPG and KGen to obtain public parameters PP and public/private key pairs (pk_{hg}, sk_{hg}) (used as home gateway's key pair). Then $\mathcal{B}$ sends (PP, gpk, pk_{hg}) to $\mathcal{A}$.
2) *Request Control-Oracle:* To reply the $\mathcal{A}$'s query encData_j [where the request of $\mathcal{A}$ is encrypted under a chosen key pair (pk_j, sk_j)], $\mathcal{B}$ first queries $\mathcal{O}_{GS}$ to obtain gs_j. Then, $\mathcal{B}$ replies the transaction $TX_{\text{access},j} = gs_j$ to $\mathcal{A}$.
3) *State Delivery-Oracle:* $\mathcal{B}$ serves as the home gateway upon sk_{hg}, and replies $\text{Data}_j = \text{Enc}_{pk_j}(\text{info}_j)$ and $\text{MAC} = \text{MAC}_{\text{key}_j}(\text{Data}_j)$ (where $\text{key}_j = pk_j^{sk_{hg}}$) to $\mathcal{A}$. Correspondingly, $\mathcal{A}$ verifies Data_j (upon the MAC) and decrypts it to obtain info_j upon its key pair (pk_j, sk_j).
4) *Chain Transaction-Oracle:* After receiving $TX_{\text{access},j}$ from $\mathcal{A}$, the permissioned nodes will chain them into the blockchain.
5) *Handle Dispute-Oracle:* To reply the $\mathcal{A}$'s query of tracing transactions $TX_{\text{access},k}$, $\mathcal{B}$ queries $\mathcal{O}_{\text{Trace}}$ to obtain tag_k and then returns tag_k to $\mathcal{A}$.

6) *Test-Oracle:* When $\mathcal{A}$ challenges two untraced transactions $(TX^0_{\text{access}}, TX^1_{\text{access}})$, $\mathcal{B}$ trivially queries $(TX^0_{\text{access}}, TX^1_{\text{access}})$ as (gs^0, gs^1) to $\mathcal{O}_{\text{Test}}$. Then $\mathcal{B}$ will get a response tag tag_b, and it sends tag_b to obtain the guess b' from $\mathcal{A}$. Finally, $\mathcal{B}$ uses b' as its answer.

Denote $\text{Adv}^{\text{An}}_{GS}(\mathcal{B})$ as the advantage of $\mathcal{B}$ breaking the anonymity of group signature, and we have

$$\text{Adv}^{\text{An}}_{GS}(\mathcal{B}) = \text{Adv}^{\text{An}}_{HC}(\mathcal{A}).$$

This means that $\mathcal{B}$ can break the anonymity of group signature with a nonnegligible advantage, which contradicts the anonymity of adopted group signature scheme in HomeChain. Thus, our HomeChain is anonymous against any polynomial attacker. ■

Then, we explain how our proposed system meets all the security requirements as mentioned is Section III.

Single Registration: There exists a group manager in our proposal for initializing group members's private keys so that group members can send remote access or control requests to the home gateway. Even though the group manager revokes the request permission of one or more group members, remaining group members do not need to update their private keys. Therefore, the proposed system provides single registration.

Mutual Authentication: The home gateway authenticates the group member by verifying the validity of the transaction (i.e., group signature). From a group member's perspective, he/she can determine the authenticity of the home gateway by recomputing the MAC. Without solving the underlying CDH problem, the attacker will not be capable of computing a valid MAC for a target message.

User Anonymity: As proved in Theorem 1, the anonymity of the adopted group signature scheme can prevent the group member's true identity from being revealed. In addition, the suggested strategy of generating fresh key pair for each request helps prevent replay attacks or the device being identified by someone diligently observing, collecting, and analyzing the network traffic. The encryption algorithm used in each request transaction ensures that only the targeted home gateway can obtain the request information, which further provides the privacy protection.

Traceability: Due to the traceability property offered by the group signature scheme used in the system, only a group manager can trace a transaction (deemed to be suspicious) to the original signer by invoking the GTrace algorithm. If any other user wishes to trace the group signature, he/she needs to break the security of the underpinning Elgamal encryption.

Session Key Agreement: A fresh one-time key pair is required during each session to encrypt data and compute the corresponding MAC, which can provide authentication and confidentiality in future communications.

Perfect Forward Secrecy: An attacker may be able to successfully steal both private/public key pairs of the group member and home gateway, however, this attacker can only decrypt the cipher text belonging to the current session. The use of fresh key pair of the group member ensures perfect forward secrecy.

No Verifier List: In our system, the central registry does not need to maintain a verifier list, since group members and home gateway only need to generate new public/private keys and store these keys, respectively, for the authentication.

No Online Central Registry: The group member and home gateway directly authenticate with each other based on the group signature and generated MAC without involving an online central registry.

Relay of Current Timestamp: Our design suggests using hardcoded DNS servers for relaying correct timestamp.

Resistance of Birthday Collision: For the permissioned blockchain adopted in our system, which is using the PBFT consensus mechanism to record a block. Some trusted nodes (i.e., servers provided by some families) are in charge of chaining. Thus, no "fork" situation will appear, which intuitively resists blocks' birthday collisions.

Resistance of Interception and Alteration: Each broadcasted request transaction is signed with group private keys. Any modifications to an interception transaction will invalidate the transaction. Moreover, our system binds the response data with an MAC to resist modification attacks.

Resistance of Hijack: Signing transactions under the group signature scheme can also prevent transaction hijacking, since an attacker cannot modify the request context with still guaranteeing the validity of signatures.

Resistance of Various Attacks:

1) *User Impersonation Attack:* In our system, any legitimate group member in the group can have remote access to authorized devices. Any impersonation can be prevented via mutual authentication, and misbehaving user identified and revoked.
2) *DDoS Attack:* Our system inherits Bitcoin's solutions to resist DDoS attack, e.g., restrict the block size and the maximum number of group signature checks for the transaction input.
3) *Modification Attack:* Suppose the attacker modifies the broadcasted transaction or replied data, then the behavior will be discovered and refused because of the group signature and message authentication code.
4) *Replay Attack:* The group member generates a new key pair in each request which is utilized to encrypt response information and compute the message authentication code. Since the group member's generated key pair is fresh, the group member and home gateway can detect any replay attack.
5) *Man-in-the-Middle Attack:* On the basic of above discussion, we can trivially find that the system owns the property of secure mutual authentication. Therefore, it can also mitigate man-in-the-middle-attacks.

To show the advancement of our HomeChain, we further compare it with the existing authentication protocols in IoT in terms of the following features. "Mutual authentication" represents that the participants can identify each other during the communication, "Anonymity" means that the client (i.e., group member in HomeChain) can issue requests in an anonymous method, "Confidentiality" refers to the privacy of recorded requests in blockchain, "Traceability" is the ability to trace malicious behaviors (being considered if the

TABLE I
COMPARISON BETWEEN OUR PROPOSAL WITH OTHER BLOCKCHAIN-BASED AUTHENTICATION PROTOCOLS IN IOT

Feature	HomeChain (proposed)	Out-of-band Authen. [18]	BSeIn [19]	Bubbles of Trust [20]	FairAccess [38]
Mutual authentication	√	√	√	√	√
Anonymity	√	×	√	×	×
Confidentiality	√	×	√	×	×
Traceability	√	\	×	\	\
Privacy of access policy	√	×	×	\	×
Revocation	√	√	√	×	√

* Note that the notation √ denotes as a supported feature, × an unsupported one, and \ not involving one.

"Anonymity" is provided), "Privacy of access policy" represents that the privacy protection of access policy stored in blockchain, and "Revocation" is the functionality of revoking the authority of malicious users. From the results in Table I, our proposed HomeChain not only owns the functionalities of mutual authentication and revocation with anonymity and confidentiality, but also provides the traceability and privacy of access policy. The latter has not been achieved by other blockchain-based authentication protocols recently.

VII. IMPLEMENTATION AND PERFORMANCE EVALUATION

We implemented and evaluated a prototype of the proposed system on *JUICE* to evaluate the utility of our proposed architecture. This platform can support the design of smart contracts using *Solidity* and possess a *Java* and *Javascript*-based Web/client tools for managing and monitoring. Moreover, it provides abundant cryptographic API calls, including homomorphic encryption, GS, and zero-knowledge proof for constructing secure and privacy-preserving applications.

Specifically, we used *JUICE* of *client version* to initialize a test chain (consisting of three permissioned nodes) in our personal computer, where the system configuration is Ubuntu 16.04 (64 bits) with an Intel Core i7-6700 CPU 3.40 GHZ and 3-GB RAM, which was also configured with *nginx-1.11.3*, *truffle-4.1.13*, and *JUICE-client*. Then, we realized an easy smart contract shown in Algorithm 1 for maintaining the request transaction and RL.

In the design of our smart contract, we used some random uint256 s to represent the request transactions (i.e., group signatures) and the corresponding responses (i.e., an MAC and ciphertext) for simplicity, because the 256 bits length is the maximum supported by Solidity (namely, uint256 and int256). However, in real-world applications, these items could be stored as an array of unit256. This implies that our realization could be extended for different settings.

After the deployment of our smart contract, we used Web3j[5] to test the functionality of deploying smart contract, uploadRequest, uploadReponse, getRequest, getResult, deleteRequest, addRL, deleteRL, and getRL. Moreover, we obtained the approximate time cost of these algorithms using *shell* script and *javascript* with 1000 running times. The results are shown in Table II, which are consistent with the block-generation time about 12 s level in Ethereum. Note

[5]A lightweight library for *Java* applications, available: https://docs.web3j.io/.

TABLE II
TIME COST OF ALGORITHMS IN THE SMART CONTRACT (S)

Algorithm	Adverage time	Max time	Min time
Deployment	18.583	29.913	14.627
uploadRequest	10.501	15.723	7.323
uploadResponse	8.072	10.809	5.397
getRequest	1.749	2.44	0.958
getResult	1.485	2.195	0.799
deleteRequest	14.101	18.808	11.247
addRL	12.158	14.532	10.616
deleteRL	13.232	16.154	10.626
getRL	1.525	2.255	0.804

TABLE III
NOTATIONS AND EXECUTING TIME (MS)

Notation	Description	Time	
		128-bit	192-bit
T_{g1a}	A point addition in $\mathbb{G}_1$	0.0811	0.3096
T_{g1sm}	A scale multiplication in $\mathbb{G}_1$	8.8517	176.057
T_{g2sm}	A scale multiplication in $\mathbb{G}_2$	19.731	332.763
T_h	A general hash function	0.00060	0.00064
T_{bp}	A bilinear pairing $\mathbb{G}_T$	119.3940	1371.01
T_{ebp}	A exponentiation in $\mathbb{G}_T$	50.3876	579.632
T_{mbp}	A multiplication in $\mathbb{G}_T$	0.5946	2.057
T_{mi}	A modular inversion in $\mathbb{Z}_q$	0.0471	0.1856
T_{mm}	A modular multiplication in $\mathbb{Z}_q$	0.0097	0.0501
T_{enc}	AES encryption algorithm	0.000346	0.000369
T_{dec}	AES decryption algorithm	0.000362	0.000374

* Note that, for the security levels of 128-bit and 192-bit, the adopted hash function is SHA256 and SHA768 respectively, and the used AES is AES128 and AES192 respectively.

that getRequest, getResult, and getRL are *view* type of functions, which do not cause any *gas* consumption and transaction confirmation, but only some communication delay. These results show that the time cost of invoking algorithms in SC will lead to an order of magnitude in seconds. This delay may not be tolerated in the real applications in smart homes. However, the time cost could be in terms of milliseconds if we reduce the blockchain generation time (e.g., in some specialized platform in the future).

In addition, we evaluated the performance, including computation and communication costs of our proposal, which is executed on a personal computer (Dell with an i5-4210U 1.70-GHz processor, 4-GB memory and Window 7 OS) based on the miracl library (a popular cryptographic library, version 7.0). Specifically, inspired by [39], we consider two different system security parameters (i.e., $\lambda = 128, 192$) and utilize the Barreto–Naehrig (BN) [40] over different base fields (i.e., $\mathbb{F}_p$-256 and $\mathbb{F}_p$-768, respectively) for the evaluation and comparison. Hence, the corresponding group elements in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, and $\mathbb{Z}_q$ are presented in 64 bytes, 128 bytes, 384 bytes, and 32 bytes (and in 192 bytes, 384 bytes, 576 bytes, 96 bytes, respectively).

Here, we first list the notations and the corresponding executing time in Table III. Then, to show the utility of HomeChain, we compare it with BSeIn [19] by counting the operations of invoking cryptographic algorithms and smart contract. From the results shown in Table IV, only the computation cost of state delivery in HomeChain is more than that in BSeIn, which is due to the recorded result into smart contract and causes additional smart contract invocations (i.e., getRequest, getRL, uploadResponse, and getResult). However, this can further guarantee the reliable behavior auditing of home gateways. Except this, the proposed HomeChain has a

TABLE IV
COMPUTATION COST (S) AND COMMUNICATION COST (BYTES)

Phase	Computation Cost		Communication Cost	
	HomeChain (proposed)	BSeIn [19]	HomeChain (proposed)	BSeIn [19]
System Setup	$(2+2t)T_{g1sm}$+tT_h+ T_{g2sm}+$2tT_{mm}$+tT_{mi}	$(2+2l)T_{g1sm}$+ lT_h+ $(2c+1)T_{g2sm}$+ lT_{mm}	$t\|\mathbb{G}_1\|+t\|\mathbb{Z}_q\|$	$(2l+\alpha+2)\|\mathbb{G}_1\|+2l\|\mathbb{Z}_q\|$
Request Control	$6T_{g1sm}+T_{g1a}+4T_h$ $+2T_{mm}+T_{ebp}+T_{enc}$ + uploadRequest	$(4r+2rc+6)T_{g1sm}+5T_h+$ $(2rc+2r-c+2)T_{g1a}+lT_{mm}$ $+T_{enc}$ + getPDHT + uploadTX	$3\|\mathbb{G}_1\|+2\|hash\|$ $+\|\mathbb{Z}_q\|+\|c_{AES}\|$	$3\|\mathbb{G}_1\|+(2+l)\|\mathbb{Z}_q\|$ $+(r+c)\|\mathbb{G}_2\|$
State Delivery	$3T_{bp}+8T_{g1sm}+T_{g1a}+$ $10T_h+3T_{mm}+2T_{dec}+$ T_{enc} + getResult + getRL + uploadResponse + getRequest	$3T_{g1sm}+\frac{1}{2}(l^2+l-2)T_{mm}+$ $5T_h+2T_{dec}+T_{enc}$ + getTX	$4\|\mathbb{G}_1\|+4\|hash\|$ $+\|\mathbb{Z}_q\|+2\|c_{AES}\|$	$3\|\mathbb{G}_1\|+(2+l)\|\mathbb{Z}_q\|+\|c_{AES}\|$ $+(r+c)\|\mathbb{G}_2\|+2\|hash\|$
Chain Transaction	$3T_{bp}+2T_{g1sm}$ $+T_{g1a}+2T_h$ + getRL	$(r+c-3)T_{bp}+2T_{g1sm}+T_{g1a}+$ $4T_h+\frac{1}{2}(l^2+l-2)T_{mm}+T_{dec}$ $+2rT_{g2sm}+rT_mbp$ + 2getTX	$3\|\mathbb{G}_1\|+2\|hash\|$ $+\|\mathbb{Z}_q\|+\|c_{AES}\|$	$3\|\mathbb{G}_1\|+(2+l)\|\mathbb{Z}_q\|$ $+(r+c)\|\mathbb{G}_2\|$
Handle Dispute	$T_{g1sm}+T_{g1a}+T_h$ + addRL	\	$3\|\mathbb{G}_1\|+2\|hash\|$ $+\|\mathbb{Z}_q\|+\|c_{AES}\|$	\

* Note that t is the number of group members, (r,c,l) are parameters in BSeIn [19], α is the number of attributes for some user in BSeIn, $|hash|$ is the length of a hash digest (256-bit or 768-bit in our system), getPDHT, uploadTX and getTX are smart contract algorithms in BSeIn, and $|c_{AES}|$ is the length of a AES ciphertext (128-bit or 192-bit in our system). Here, in the phase of Chain Transaction, we only consider the situation of one transaction.

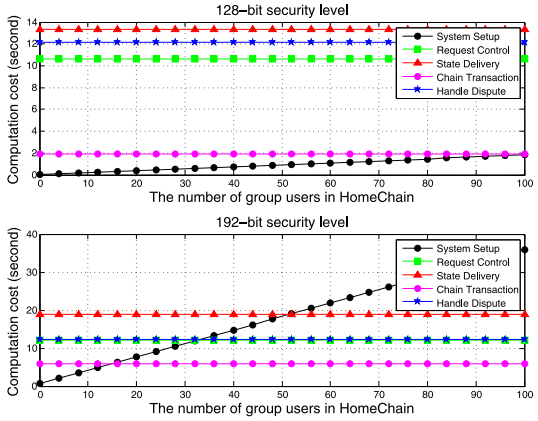


Fig. 3. Scalability of computation costs in different security levels.

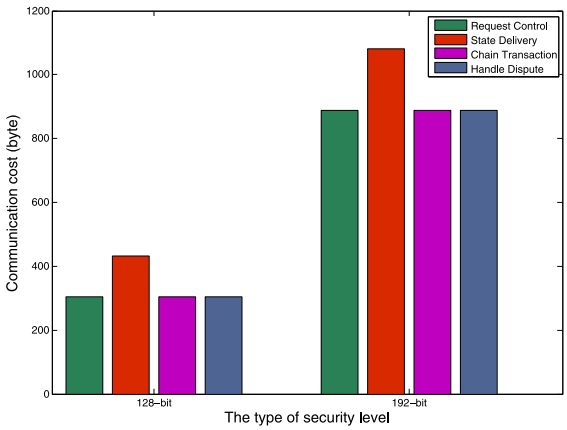


Fig. 5. Impact of security parameter on communication cost.

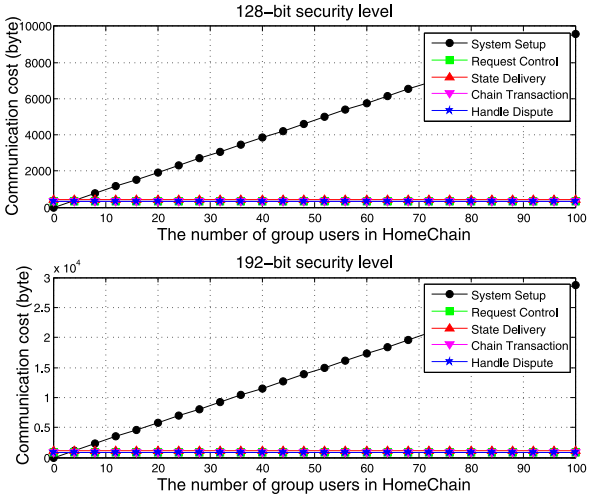


Fig. 4. Scalability of communication costs in different security levels.

better performance than BSeIn in terms of computation and communication costs, especially the performance of the former will not linearly increase as the number of permissioned nodes increases.

To further show the performance scalability from the view of delay, we depict the asymptotic line charts of the communication and time costs (see Figs. 3 and 4) in different security levels. From the line charts, in both 128-bit and 192-bit security levels, only the phase of System Setup is affected by the number of group users (actually, caused by the initialization of group private keys for group users). In the rest of the phases, both the computation and communication costs are independent of the number of group users and permissioned nodes. This also demonstrates the advantage of our HomeChain when compared to BSeIn.

Finally, we discuss the impact of security parameters on communication and computation costs. As shown in Figs. 5 and 6, both communication and computation costs are affected by the security parameter. That is, the higher security level of HomeChain will incur more communication and computation costs. A tradeoff between security and utility in our HomeChain can be achieved via using a proper security parameter. Note that we omit these impacts during the phase of System Setup, which is only executed once and will not negatively impact the user's experience.

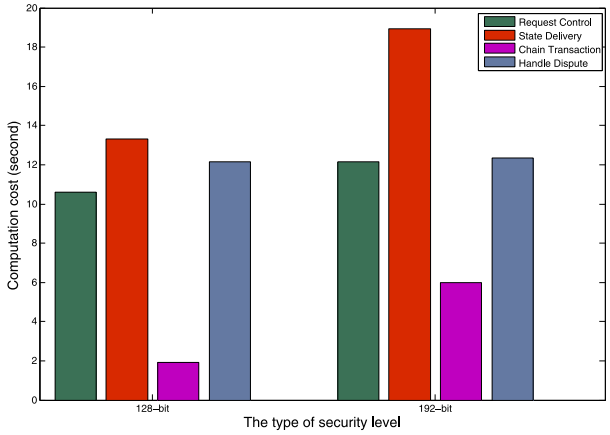


Fig. 6. Impact of security parameter on computation cost.

VIII. Conclusion

Smart homes and blockchain are two trending topics at the moment, and in this article, we demonstrated how blockchain can be utilized with other techniques to ensure mutual authentication between users and the home gateway in a smart home. Specifically, the proposed system utilizes a GS and an MAC to authenticate a requestor without leaking information on the specific member and the home gateway with perfect forward secrecy, respectively. The system also allows one to efficiently trace any users subsequently found to be misbehaving. We finally demonstrated the security and utility of our proposed system.

In the future, we will consider the attribute-based cryptographic approaches into our proposal for achieving the fine-grained access control (but not compromising the privacy of access policy). Moreover, we will consider the secure protocol design in the universal composability setting, for guaranteeing the security while running in parallel with other protocols. We also intend to implement and evaluate a prototype of the extended system for achieving higher utility in the real world.

References

[1] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.

[2] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything you wanted to know about smart cities," *IEEE Consum. Electron. Mag.*, vol. 5, no. 3, pp. 60–70, Jul. 2016.

[3] Q. Do, B. Martini, and K.-K. R. Choo, "Cyber-physical systems information gathering: A smart home case study," *Comput. Netw.*, vol. 138, pp. 1–12, Jun. 2018.

[4] Q. Do, B. Martini, and K. R. Choo, "Is the data on your wearable device secure? An Android wear smartwatch case study," *Softw. Pract. Exp.*, vol. 47, no. 3, pp. 391–403, 2017.

[5] Q. Do, B. Martini, and K.-K. R. Choo, "A data exfiltration and remote exploitation attack on consumer 3D printers," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 10, pp. 2174–2186, Oct. 2016.

[6] M. L. Das, "Two-factor user authentication in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1086–1090, Mar. 2009.

[7] K. H. M. Wong, Y. Zheng, J. Cao, and S. Wang, "A dynamic user authentication scheme for wireless sensor networks," in *Proc. IEEE Int. Conf. Sensor Netw. Ubiquitous Trustworthy Comput. (SUTC)*, Taichung, Taiwan, Jun. 2006, pp. 244–251.

[8] K. Ren, K. Zeng, W. Lou, and P. J. Moran, "On broadcast authentication in wireless sensor networks," in *Proc. 1st Int. Conf. Wireless Algorithms Syst. Appl. (WASA)*, Xi'an, China, Aug. 2006, pp. 502–514.

[9] C.-S. Tsai, C.-C. Lee, and M.-S. Hwang, "Password authentication schemes: Current status and key issues," *Int. J. Netw. Security*, vol. 3, no. 2, pp. 101–115, 2006.

[10] N. M. Haller, "The s/key one-time password system," in *Proc. Internet Soc. Symp. Netw. Distrib. Syst.*, 1995, pp. 151–157.

[11] J.-Y. Liu, A.-M. Zhou, and M.-X. Gao, "A new mutual authentication scheme based on nonce and smart cards," *Comput. Commun.*, vol. 31, no. 10, pp. 2205–2209, 2008.

[12] B. Vaidya, J. H. Park, S. Yeo, and J. J. P. C. Rodrigues, "Robust one-time password authentication scheme using smart card for home network environment," *Comput. Commun.*, vol. 34, no. 3, pp. 326–336, 2011.

[13] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "Hotp: An HMAC-based one-time password algorithm," Internet Eng. Task Force, RFC 4226, 2005.

[14] K. Y. Choi, J. Y. Hwang, D. H. Lee, and I. S. Seo, "ID-based authenticated key agreement for low-power mobile devices," in *Proc. Inf. Security Privacy 10th Aust. Conf. ACISP*, Brisbane, QLD, Australia, Jul. 2005, pp. 494–505.

[15] Y.-H. Chuang and Y.-M. Tseng, "Towards generalized id-based user authentication for mobile multi-server environment," *Int. J. Commun. Syst.*, vol. 25, no. 4, pp. 447–460, 2012.

[16] Y.-P. Liao and C.-M. Hsiao, "A novel multi-server remote user authentication scheme using self-certified public keys for mobile clients," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 886–900, 2013.

[17] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the Internet of Things: Research issues and challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.

[18] L. Wu, X. Du, W. Wang, and B. Lin, "An out-of-band authentication scheme for Internet of Things using blockchain technology," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Mar. 2018, pp. 769–773.

[19] 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," *J. Netw. Comput. Appl.*, vol. 116, pp. 42–52, Aug. 2018.

[20] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of trust: A decentralized blockchain-based authentication system for IoT," *Comput. Security*, vol. 78, pp. 126–142, Sep. 2018.

[21] D. Chaum and E. van Heyst, "Group signatures," in *Proc. Adv. Cryptol. Workshop Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, vol. 547. Brighton, U.K., Apr. 1991, pp. 257–265.

[22] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *J. Comput. Syst. Sci.*, vol. 61, no. 3, pp. 362–399, 2000.

[23] A. F. Skarmeta, J. L. H. Ramos, and M. V. M. Cano, "A decentralized approach for security and privacy challenges in the Internet of Things," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Seoul, South Korea, Mar. 2014, pp. 67–72.

[24] H. Groß, M. Hölbl, D. Slamanig, and R. Spreitzer, "Privacy-aware authentication in the Internet of Things," in *Proc. 14th Int. Conf. Cryptol. Netw. Security (CANS)*, Marrakesh, Morocco, Dec. 2015, pp. 32–39.

[25] A. Ukil, S. Bandyopadhyay, and A. Pal, "IoT-privacy: To be private or not to be private," in *Proc. IEEE INFOCOM Workshops*, Toronto, ON, Canada, Apr./May 2014, pp. 123–124.

[26] A. Rachedi and A. Benslimane, "Security and pseudo-anonymity with a cluster-based approach for MANET," in *Proc. Glob. Commun. Conf. (GLOBECOM)*, New Orleans, LA, USA, Nov./Dec. 2008, pp. 1956–1961.

[27] A. Rachedi and A. Hasnaoui, "Advanced quality of services with security integration in wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 15, no. 6, pp. 1106–1116, 2015.

[28] A. Rachedi and A. Benslimane, "Multi-objective optimization for security and QoS adaptation in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–7.

[29] R. Roman, J. Zhou, and J. López, "On the features and challenges of security and privacy in distributed Internet of Things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, 2013.

[30] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and solutions," *CoRR*, vol. abs/1608.05187, pp. 1–13, 2016.

[31] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 618–623.

[32] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *Proc. 2nd Int. Conf. Internet Things Design Implement. (IoTDI)*, Pittsburgh, PA, USA, Apr. 2017, pp. 173–178.

[33] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.
[34] D. He, S. Zeadally, N. Kumar, and W. Wu, "Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 9, pp. 2052–2064, Sep. 2016.
[35] D. Hankerson, S. Vanstone, and A. Menezes, "Guide to elliptic curve cryptography," *Comput. Rev.*, vol. 46, no. 1, p. 13, 2005.
[36] T. Ho, L. Yen, and C. Tseng, "Simple-yet-efficient construction and revocation of group signatures," *Int. J. Found. Comput. Sci.*, vol. 26, no. 5, pp. 611–624, 2015.
[37] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, New Orleans, LA, USA, Feb. 1999, pp. 173–186.
[38] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Cham, Switzerland: Springer, 2017, pp. 523–533.
[39] D. Abbasinezhad-Mood and M. Nikooghadam, "Efficient design of a novel ECC-based public key scheme for medical data protection by utilization of nanopi fire," *IEEE Trans. Rel.*, vol. 67, no. 3, pp. 1328–1339, Sep. 2018.
[40] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. 12th Int. Workshop Select. Areas Cryptography (SAC)*, Kingston, ON, Canada, Aug. 2005, pp. 319–331.

Chao Lin received the bachelor's and master's degrees from the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, China, in 2013 and 2017, respectively. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China.

His current research interests include authentication of graph data and blockchain security.

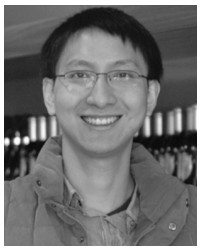
Debiao He received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009.

He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, and also with the Guangdong Provincial Key Laboratory of Data Security and Privacy Protection, Guangzhou, China. His current research interests include cryptography and information security, in particular, cryptographic protocols.

Neeraj Kumar (M'16) received the Ph.D. degree in CSE from Shri Mata Vaishno Devi University, Katra, India.

He is currently a Full Professor with the Department of Computer Science and Engineering, Thapar University, Patiala, India. He is also an Adjunt Professor with King Abdul Aziz University, Jeddah, Saudi Arabia. He has over 400 technical research papers in leading journals, such as the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TWPS, IEEE SYSTEMS JOURNAL, *IEEE Communications Magazine*, *IEEE Wireless Communications Magazine*, *IEEE Network Magazine*, and conferences. His research is supported from DST, TCS, and UGC. He has guided many students leading to M.E. and Ph.D. His current research interests include mobile computing, parallel/distributed computing, multiagent systems, service-oriented computing, routing and security issues in mobile ad hoc, sensor, and mesh networks.

Prof. Kumar was a recipient of many prestigious awards from IEEE Systems Journal and IEEE ICC in 2018.



Xinyi Huang received the Ph.D. degree from the University of Wollongong, Wollongong, NSW, Australia.

He is currently a Professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, and the Co-Director of the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fuzhou. His current research interests include applied cryptography and network security.

Prof. Huang is an Associate Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. He serves on the Editorial Board for the *International Journal of Information Security* (Springer), and has served as the Program/General Chair or a Program Committee Member in over 80 international conferences.

Pandi Vijayakumar received the B.E. degree in computer science and engineering from Madurai Kamaraj University, Madurai, India, in 2002, the M.E. degree in computer science and engineering from the Karunya Institute of Technology, Coimbatore, India, in 2005, and the Ph.D. degree in computer science and engineering from Anna University Chennai, Chennai, India, in 2013.

He is the Former Dean and currently an Assistant Professor with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Villupuram, India, which is a constituent college of Anna University Chennai. His current research interests include key management in network security, VANET security, and multicasting in computer networks.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. degree in information security from the Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio (UTSA), San Antonio, TX, USA. In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015, he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg.

Dr. Choo was a recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) 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 ESORICS 2015 Best Paper Award, 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 is also a Fellow of the Australian Computer Society, and the Co-Chair of IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group.