Secure Communication for Smart Home Environments Using Blockchain

by

Anna Vuong

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2021 by the
Graduate Supervisory Committee:

Stephen S. Yau, Chair
Adam Doupé
Samira Ghayekhloo

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Many residences from student apartment units to family homes use a range of smart devices to make the day-to-day lives of the residents safer and more convenient. The ability to remotely access these devices has further increased their convenience, but it comes with the increased risk of vulnerable devices being exploited to achieve unauthorized access or to conduct surveillance on the users. This highlights the need for an access control system to securely restrict home device access to authorized users only. Existing approaches for securing smart homes use less secure authentication methods, do not allow for data ownership or fine-grained access control, and do not reliably store credential modification records, access records, or access policy modification records. These records can be a valuable resource to have available in the case of a security incident.

In this thesis, a secure and efficient remote mutual authentication system with fine-grained access control integrating blockchain and digital signatures to authenticate users, authenticate the home gateway, and provide reliable auditing of the credential modifications, access history, and access policy modifications of the devices is presented. The immutability and verifiability properties of blockchain make it useful for securely storing these records. In this approach, a smart contract is created in the blockchain to keep track of authorized users, manage the access policy, and record requests for access or control of the home devices. A private blockchain is used to provide trust and privacy, which is necessary for a smart home system. Elliptic curve digital signatures are used to

verify identities because the shorter key sizes and signature times are more adapted to Internet of Things contexts. The approach presented in this thesis is better than existing approaches because it provides fine-grained access control, and reliably stores credential modification records, access records, and access policy modification records. The approach was implemented and evaluated using Hyperledger, a private open-source blockchain, and the results show that this approach has significant additional security benefits with negligible additional overhead cost.

DEDICATION

Hôm nay con khôn lớn, bay đi khắp mọi miền! Như tất cả những gì mà con làm trong cuộc sống này, con đã hoàn thành khóa luận tốt nghiệp này cho Mẹ và Pà Pá. Có Mẹ, có Pà Pá, thì có quê hương.

ACKNOWLEDGEMENTS

I would like to thank Dr. Stephen S. Yau, Dr. Adam Doupé, and Dr. Samira Ghayekhloo for agreeing to serve on my committee and patiently providing me with feedback.

I appreciate the support from National Science Foundation under the NSF SFS Arizona Cyber Defense Scholarship Program at Arizona State University.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Applying Internet of Things to homes has enhanced day-to-day convenience for an increasing number of users. Certain smart devices that are connected to the Internet can be accessed remotely. Using a smart air conditioner as an example, a user could turn their air conditioner off while they go out or turn it on as they are on their way home. Unauthorized users should not be able to access these smart devices, view the contents of an authorized user's communications with these devices, or feed false feedback from the devices to an authorized user. The smart devices in the home as well as the channel used to communicate with them can be vulnerable to compromise. The consequences of such compromise could result in unauthorized access to or control of a home's smart devices and could lead to the harvesting of sensitive data [1]-[3]. When enough of this data is collected, especially data from cameras, microphones, or smart locks, it can be used for stalking or profiling the residents of a home. This highlights the need for a secure communication between users and smart devices.

There exists a number of approaches for remotely accessing smart home devices by a user through a wireless device. Many of these approaches use less secure authentication methods that are especially vulnerable when considering smart home attacker profiles. Most of these approaches do not provide data ownership or fine-grained access control. Even more of these approaches do not keep credential modification records, access records for the devices, or access

policy modification records [4]. The approaches that do keep these records usually store them in a database, which makes them potentially modifiable without the user's knowledge. Storing the credential modification records, access records, and access policy modification records of these devices not only expands the functionality and convenience of a smart home, it can also be a valuable tool for users in the case of a security incident. An approach to securing smart homes should securely mutually authenticates users and the home gateway, allow for flexible participation, provide fine-grained access control and data ownership, and keep records of credential changes, device access, and access policy changes. Blockchain presents an interesting solution to these issues due to its inherent properties of immutability and auditability. Existing blockchain approaches are not compatible with regular usage of smart homes and do not address fine-grained access control. The major innovations of this thesis are summarized as follows:

1. This thesis presents an approach combining a private blockchain and smart home systems to provide fine-grained access control to smart home devices.
2. This thesis uses a smart contract stored on the private blockchain to keep track of authorized users, manage the access policy, and record requests for access or control of the home devices. Any changes to user credentials or the access policy, and any successful user requests are reliably recorded to the blockchain to provide additional functionalities and additional security for smart home users.

This approach is better than existing approaches because it provides fine-grained access control and data ownership and reliably stores credential modification records, access records, and access policy modification records.

Organization: Chapter 2 of this article surveys existing popular smart home systems as well as other smart home security solutions that have been proposed. Chapter 3 explains the functional components of the approach. Chapter 4 explains how the major innovations are achieved. Chapter 5 describes the overall approach that is being proposed. Chapter 6 demonstrates an illustrative example of the overall approach. Chapter 7 evaluates the performance metrics of the approach and compares the features of the approach against other approaches. Chapter 8 concludes the article and outlines future work.

CHAPTER 2

CURRENT STATE OF ART

Smart home systems integrate with other smart devices, and even some third-party devices, to allow authenticated users to remotely access or control smart devices. The most popular existing smart home systems [5]-[8] all opt for a username and password-based login and use cookies to authenticate user actions after that. This method is generally the most simple and easy to use authentication method for the average smart home user. However, it is less secure than signature-based authentication. If the authentication server were to be hacked or spoofed, an attacker could potentially learn the login credentials of the user because the user is forced to send their credentials to the server to log in. The profile of an attacker of a smart home would also tend to be someone that is already at least somewhat familiar with the residents of the home. This is because the only logical objective for a cyberattacker of a smart home would be more information about the residents and their behaviors. Since people tend to use simple passwords, or reuse passwords, this makes the probability of this kind of attacker being able to correctly guess the authorized user's username and password especially higher [9]. Multi-factor authentication could mitigate these vulnerabilities, but it is not usually enabled by default, and the average smart home user would find it cumbersome to use. As of 2018, less than 10% of Google accounts had two-factor authentication enabled [10]. If users lose access to the device that they use to multi-factor authenticate, then they could be left temporarily unable to access their devices at all, which is a critical drawback in

the case of smart homes. Using cookies for authentication also leaves the user vulnerable to cookie injection attacks. Google has been found to have cookie-related vulnerabilities before [11]. Using these systems requires storing the user's credential information on the smart home system provider's servers. In the case of a data leak, the user's information could be compromised through no fault or action of the user. Furthermore, these popular username and password-based systems [5]-[8] all only allow granting users full access to the home devices, or no access to the home devices at all, which means that the users are unable to achieve fine-grained access control. Smart home users often want to have different access levels for different groups of users so this is a major deficiency. Existing approaches that do allow for fine-grained access control do not generate any records of user activity [12].

Although signature-based schemes are generally more difficult to compromise than username-password based authentication schemes, not all signature-based schemes are appropriate for authenticating users to smart home devices. Since the average household size is only 2.53 people as of 2020 [13], for smart homes, anonymous signatures are not worth their computational costs [14]-[15]. Furthermore, anonymous signatures would make it impossible to achieve fine-grained access control. Signature schemes that do not allow for flexible participation for signers are also not compatible with smart homes since it is not uncommon for new users to be added to a smart home system after initialization [15].

While using blockchain would increase energy consumption, it could allow smart home users to achieve immutability and complete data ownership. However, not all forms of blockchain would be appropriate for smart homes. Blockchain-based smart home security solutions that use public blockchain networks or computing power-based consensus mechanisms [14]-[16] have incompatibly high time overhead and incur financial costs with each transaction. This is a major flaw for smart homes because users would not be motivated to switch from a smart home system that does not cost money to a smart home system that does cost money and is slower. Most blockchain-based smart home solutions proposed fail to provide fine-grained access control for different users using different devices within the same home [15][16][17][18]-[20]. Other blockchain-based approaches try to provide access control [21] but opt to store the access policy as a header of each block. With this design, a record would need to be stored in this header for each device, each requester, and each action. So, this header would need to hold a very large amount records, even with just a few users and a few devices, and since it is a header, it would need to be chained as a part of every block. That makes this design is extremely inefficient. Since there is no way to directly change the access policy without doing anything else, if the access policy needed to be changed, it would not happen until another block is chained after some number of requests transactions are collected. This delay is not tolerable, especially in the case of needing to revoke access of a malicious user. Additionally, storing the policy as a header means that any node chaining the blocks can be used to change the

access policy. This effectively renders this policy unenforceable in a multi-user home where other users will want to have access to the records, because the policy can be changed by anyone that has access to these nodes.

For the reasons outlined in this chapter, a local, private blockchain-based approach integrated on top of an existing home gateway using a smart contract to manage users and devices is a promising solution for smart home systems. Such an approach would enable users to have complete ownership and control of their data, reliably storing credential modification records, access records, and access policy modification records, and achieve fine-grained access control.

CHAPTER 3

FUNCTIONAL COMPONENTS OF THE SYSTEM

In this chapter, I will explain the communication environment that I am considering, as well as explain the entities communicating in my approach.

3.1 Communication Architecture

The architecture for communication in a typical smart home environment is shown in Figure 1. The user devices and the home smart devices communicate with each other through the home gateway.

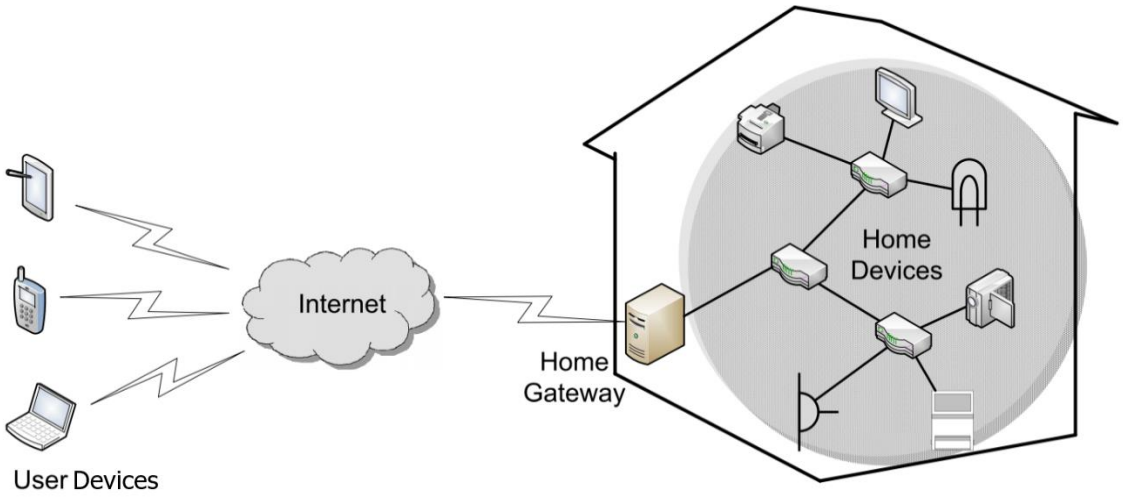

Figure 1: Typical Smart Home Communication Architecture [15]

For this approach, consensus nodes and a designated home manager is introduced. The following are the participants in this approach:

- Home Gateway: This is a machine that handles authenticating users and maintaining sessions. It also integrates with the home smart devices to handle executing user requests on the appropriate smart devices and

handle communicating feedback from the smart devices to the users. This machine is connected to the home smart devices through the home network. The users can connect to the home gateway through the Internet.

- Consensus Nodes: These nodes chain transactions that will be described in Section 3.2 to the blockchain, which stores the user credentials, device access policy, records of credential changes and access policy changes, and records of user requests for access or control. The responsibilities of these nodes could be integrated into the home gateway itself or could be done by any servers in the home that the home manager chooses.
- Home Manager: This is a trusted user of the smart home that is designated by the household to decide the authorized users and the access policy of the home. The home manager handles the setup of the system and is the only user that can add keys or devices, update keys or devices, or revoke keys. There could be multiple home managers, but these additional capabilities are not made available to all general users to ensure that users that may have less permissions according to the access policy cannot bypass it by registering as another user or directly changing their own permissions.
- Users: Users use wireless devices to authenticate themselves to the home gateway through the Internet to send requests for access or control of smart devices.

- Smart Devices: Smart devices receive user requests for access or control from the home gateway through the home network and can send feedback from the device to the user through the same way.

While the consensus nodes could potentially handle the authentication directly, the rest of the responsibilities of the home gateway are out of the scope of the blockchain. So, I present the home gateway and the consensus nodes as two separate entities with separate responsibilities that could potentially be integrated.

3.2 Smart Contract Algorithms Used

A smart contract is essentially some executable logic stored on the blockchain. In this approach, transactions are used to trigger the smart contract for user registration, key update, key revocation, updating the access policy, and uploading a request for access or control of a device. Since transactions trigger a record of the action to be chained to the blockchain, the records will be highly reliable. Updating keys or the access policy and revoking keys can only be done by a designated home manager. To send requests for access or control of a device, the users interact with the home gateway, who sends then sends verified transactions to generate a record. The smart contract is shown in Algorithms 1-7.

Algorithm 1: Constructor

```
function SmartHome()
% Constructor, automatically invokes while being deployed
     % Defines user table
     struct user {
          int UID;
          int key;
          int GID;
     } user[] public UT;
     % Defines device table
     struct device {
          int DID;
          vector<int> GIDs;
     } user[] public DT;
     % Defines request from user
     struct request{
          int key;
          string msg;
     } public RQ;
     % Designates sender as home manager
     int HM = msg.sender;
     return 1;
```

Algorithm 2: Update User Table

```
function updateUT(UID, key, GID)
% Invoked by home manager to update UT
     require(msg.sender == HM);
     if exist(UT[i].UID == UID)
          UT[i].key = key;
          UT[i].GID = GID;
     else
          UT[UT.size].UID = UID;
          UT[UT.size].key = key;
          UT[UT.size].GID = GID;
     return 1;
```

Algorithm 3: Query User Table

```
function queryUT(key)
% Invoked to retrieve user information
     if exist(UT[i].key == key)
          return UT[i];
     else
          return 0;
```

```
Algorithm 4: Revoke User
function revokeUT(key)
% Invoked by home manager to revoke a user
     require(msg.sender == HM);
     if exist(UT[i].key == key)
          release(UT[i]);
          for i < UT.size; i++
               UT[i] = UT[i+1];
          return 1;
     else
          return 0;
```

```
Algorithm 5: Update Device Table
function updateDT(DID, GIDs)
% Invoked by home manager to update DT
     require(msg.sender == HM);
     if exist(DT[i].DID == DID)
          DT[i].GIDs = GIDs;
     else
          DT[DT.size].DID = DID;
          DT[DT.size].GIDs = GIDs;
     return 1;
```

```
Algorithm 6: Query Device Table
function queryDT(DID, GID)
% Invoked to retrieve device information
     if exist(DT[i].DID == DID)
          if exist(DT[i].GIDs[j] == GID)
               return 1;
     else
          return 0;
```

```
Algorithm 7: Change Current Request
function changeRequest(key, msg)
% Invoked by home gateway to set most recent request
     require(msg.sender == HG);
     new request tempRQ;
     tempRQ.key = key;
     tempRQ.msg = msg;
     RQ = tempRQ;
     return 1;
```

The smart contract manages three variables that are initialized when the smart contract is deployed:

1. *User Table*: The smart contract manages a table of user identifiers, hashed user public keys, and group identifiers. An example of what the user table might look like is shown in Table 1. This gives users control of their own data without needing to have their credentials managed by a third party. The public keys are hashed so that nobody can view them in plaintext. As discussed in Chapter 2, smart homes are especially vulnerable to dictionary attacks. Since this approach is hashing and storing keys instead of passwords, this approach is much more resistant to dictionary attacks than password-based approaches.

UID	**key**	**GID**
dad	405905bc13959309f48a6d06e45a827fabd59b026dc974d7	PARENT
thao	03042cf8100db386818cee4ff0f2972431a62ed78edbd09a	THAOROOM
thanh	8920f3ce9287b4af6d9a5730bdda240fc74b3a5deb25e0b0	CHILD

Table 1: Example User Table

2. *Device Table*: The smart contract manages a table of device identifiers and lists of group identifiers. An example of what the device table might look like is shown in Table 2.

DID	**GIDs**
KITCHENLIGHT	PARENT, CHILD, THAOROOM
GARAGEDOOR	PARENT
THAOSPEAKER	THAOROOM

Table 2: Example Device Table

3. *Most Recent Request*: The smart contract stores the most recent successful user request for access or control of a device in the home. Examples of these user requests would be a request for a recording from a security camera or a request to open a smart lock.

CHAPTER 4

MAJOR INNOVATIONS

In this chapter, I will explain how I achieve my two major innovations: fine-grained access control and reliable records of credential modifications, access history, and access policy modifications.

4.1 Fine-Grained Access Control

This approach achieves fine-grained access control of smart devices by using groups of users to determine who can and cannot access certain devices. The Device Table variable stored in the smart contract allows home managers to customize their access policy to achieve fine-grained access control of their home devices. The home manager can initialize the access policy by choosing which groups of users can access which devices and invoking updateUT accordingly. Following initialization, the home manager could change the existing access policy at any point after it is initialized by invoking updateUT accordingly. The home gateway can enforce the access policy by invoking queryUT upon receiving a user request to check if a user's group has access to a certain device according to the access policy.

4.2 Generation and Storage of Reliable Records

This approach generates and reliably stores three different kinds of records of user actions:

- *Credential Modification Records*: Credential modification records are an extremely valuable security resource to have available for any authentication system. In this approach, whenever users want to register, update a key, or

revoke a key, they must send a transaction to the smart contract to execute the appropriate function to perform their desired action. These functions would be updateUT and revokeUT. Since these functions write to the User Table variable stored in the smart contract, invoking any of these functions will create an immutable, timestamped record of the action.

- *Access Records*: Access records of the home devices are useful as a security resource, but also it is generally useful for users to have timestamped access records for reference in their regular usage of their smart devices. After successful authentication, the home gateway would receive the request and then call changeRequest to change the variable storing the most recent request. Since this function writes to the Request variable stored in the smart contract, invoking this function will create an immutable, timestamped record of the action. The hashed public key of the user is chained rather than the user directly sending the transaction and authenticating themselves for the reasons described in Chapter 3.
- *Access Policy Modification Records*: Access policy modification records are an extremely valuable security resource to have available for any access control system. In this approach, whenever users want to add a new device or update the access policy of an existing device, they must send a transaction to the smart contract to execute updateDT according to their desired action. Since this function writes to the Device Table variable stored in the smart contract, invoking this function will create an immutable, timestamped record of the action.

CHAPTER 5

OVERALL APPROACH

For this approach, I elect to use elliptic curve cryptography (ECC). Elliptic curves are applicable for digital signatures and key agreement and can be indirectly used for encryption. Interested readers can refer to [22]-[23] for more in-depth information about ECC. ECC is used here because the relatively short key sizes [24] and signature times [25] are more adapted to IoT.

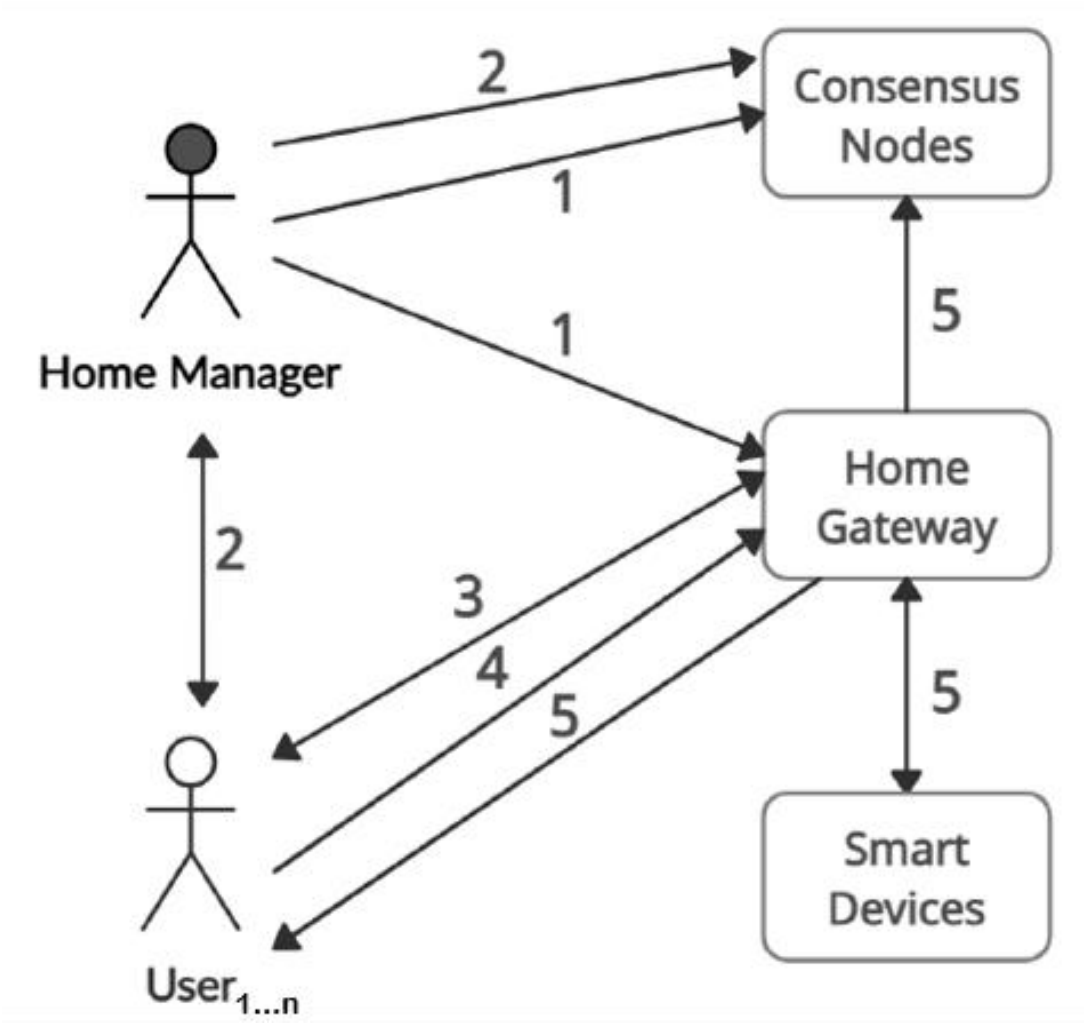

Figure 2: Communication Between Participants for Each Step

The overall approach can be put into 5 steps. A depiction of the communication flow between the participants for each step is shown in Figure 2. Each step can be summarized as follows:

1. *System Setup*: The home manager initializes the system, generates the public parameters, and adds the devices that are integrated with the home gateway. The home manager can add or update devices as needed.
2. *User Registration*: Users are added to the system as valid users that can authenticate themselves for access to the smart devices. After a user is registered, the home manager can update or revoke the user's keys as needed.
3. *Mutual Authentication and Creation of Session Key*: The user verifies their identity to the home gateway, and the home gateway verifies their identity to the user. Upon successful mutual authentication, the user and the home gateway create common session keys using information that they send to each other.
4. *Send Request*: The user uses the session keys to securely send a request for access to or control of a smart device in the home.
5. *Respond to Request*: The home gateway verifies and processes the request and uses the session keys to securely send the device feedback to the user.

I assume that the communication channel used during the Step 1 and Step 2 is secure and private. Since the users will very likely be residents of the

same home, this is a reasonable assumption. Step 2 can occur at any time if Step 1 has already been completed, so user credentials can be added, changed, or removed as needed. Step 4 and Step 5 can occur multiple times in each session, so in each session, as many requests for access or control can be made as needed. The details of each step will be described in the following sections.

5.1 System Setup

The household designates the home manager. Let p and q be large prime numbers of a length to be determined by the home manager as a security parameter. Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$. The home manager picks P, a point in $E(\mathbb{F}_p)$, of order q. The home manager randomly chooses $d_{hm} \leftarrow \mathbb{Z}_q$ and computes $Q_{hm} = d_{hm} * P$. The home manager randomly chooses $d_{hg} \leftarrow \mathbb{Z}_q$ and computes $Q_{hg} = d_{hg} * P$ for the home gateway, which is distributed to the home gateway. The home manager publishes E, P, p, q, and Q_{hg}. The home manager chooses consensus nodes and initializes the blockchain following a specific consensus mechanism (e.g., practical Byzantine fault tolerance in Hyperledger Fabric). The home manager deploys the smart contract shown in Algorithms 1-7 on the blockchain. The home manager determines groups for the home, and decides the access policy by invoking updateDT(DID, GIDs) for each device, where DID is the device identifier and GIDs is a list of the group identifiers of groups that are allowed to access each device. The home manager can invoke updateDT any time they want to add a new device or update the access policy of an existing device.

5.2 User Registration

To register, a user U_i sends their identifier UID_i to the home manager. The home manager randomly chooses $d_i \leftarrow \mathbb{Z}_q$ and computes $Q_i = d_i * P$. The home manager hashes Q_i as key_i = SHA-256(Q_i). The home manager decides on a group with identifier GID_i for U_i and invokes updateUT(UID_i, key_i, GID_i) to add U_i's credentials to the user table. This invocation will create a transaction, which will chain a timestamped record of this action into the blockchain. The home manager distributes (Q_i, d_i) to the U_i. The home manager can register their own keypair in the same way if they will also be using the devices. After a user is registered, if the home manager wants to update a key or revoke a key, then they can do the following:

1. *Update Key*: In the case of U_i wanting to update their keypair, they go through the User Registration step again with the home manager to generate $newkey_i$. The home manager invokes updateUT($newkey_i$, UID_i , GID_i,). This invocation will create a transaction, which will chain a timestamped record of this action into the blockchain. In the case of a compromised keypair, the UID can be updated as well.
2. *Revoke Key*: In the case of the home manager wanting to revoke U_i's key, the manager can invoke revokeUT(key_i). This invocation will create a transaction, which will chain a timestamped record of this action into the blockchain.

5.3 Mutual Authentication and Creation of Session Key

Before a registered user U_i with keypair (Q_i, d_i) can send a request for access or control of a smart device, they must first mutually authenticate and create a session key with the home gateway, which has the keypair (Q_{hg}, d_{hg}). They can do this using a generalized elliptic curve Diffie-Hellman algorithm with timestamped messages. Following this, U_i and the home gateway would share MAC_{hg}, a session key to be used by the home gateway to generate message authentication codes, MAC_i`, a session key to be used by $U_i$ to generate message authentication codes, and EK`, a session key to be used to symmetrically encrypt messages.

5.4 Send Request

To send a request for access or control of a device, U_i would construct a message msg = (DID || CO || t_i), where DID is the device identifier, CO is the control order to be enacted on the device, and t_i is the current timestamp. U_i would then send rq = $Encrypt_{EK}$(msg || MAC_{MACi}(msg)) to the home gateway.

5.5 Respond to Request

The home gateway computes msg` || $MAC_{MACi}$(msg`)` = $Decrypt_{EK}$(rq), verifies the MAC, and verifies that $t_{hg}$ - $t_i$` ≤ Δt where t_{hg} is the current timestamp and t_i` is the timestamp retrieved from msg`. The home gateway invokes queryDT(DID`, GID`) and does not proceed if 1 is not returned. The home gateway invokes changeRequest(key_i`, DID`, CO`). This invocation will create a transaction, which will chain a timestamped record of this action into the blockchain. The home gateway connects to the target home device to execute

the request. The home gateway receives feedback from the target device (e.g., execution result, device status, requested data). The home gateway would construct the message msg = (feedback || t_{hg}). The home gateway would then send fb = $Encrypt_{EK}$(msg || MAC_{MAChg}(msg)) to U_i who would verify the MAC and verify t_{hg} according to Δt in a similar way that has been previously described before accepting the feedback.

CHAPTER 6

ILLUSTRATIVE EXAMPLE

In this chapter, an example is presented to illustrate how the overall approach secures communication in smart homes.

Consider a smart home with three users U_1, U_2, and U_3 owning a security camera and a smart lock at the front door. The camera is capable of remotely providing users its live feed and the smart lock is capable of remotely locking and unlocking. This approach is suitable for all kinds of smart devices, such as monitors, locks, lights, speakers, printers, televisions, power plugs etc. All kinds of smart devices will need to receive a request initiated by the user to be accessed or controlled. All kinds of smart devices will have some kind of feedback to return to the user whether it be requested data or the status of the device. The home gateway is integrated with both devices to allow them to be accessed and controlled through the home gateway. Assume that the home manager is designated as U_1.

According to Step 1 of the overall approach described in Chapter 5, U_1 generates the public parameters, and generates keypairs for itself (Q_1, d_1) and for the home gateway (Q_{hg}, d_{hg}). U_1 chooses to initialize the blockchain onto the same machine as the home gateway and deploys the smart contract. U_1 decides on two groups for the household: residents and guests. U_1 invokes updateDT to add the security camera and the smart lock. U_1 sets the security camera's access policy to residents and guests and sets the smart lock's access policy to residents only. U_2 registers to the system as a resident and U_3 registers to the

system as a guest according to Step 2, so they now have (Q_2, d_2) and (Q_3, d_3) respectively. To register U_2 and U_3, U_1 will need send 2 transactions, which will chain records of U_2 and U_3 being added as users to the blockchain. Even after this, U_1 could add more users, update user keys, or revoke users as needed. The user table and device table that would be created in this example situation are shown in Table 3 and Table 4. U_1 and U_2 are registered as residents, so they have access to both the security camera and the smart lock as described in the device table. U_3 is registered as a guest, so they only have access to the security camera, but not the smart lock as described in the device table.

UID	**key**	**GID**
U_1	405905bc13959309f48a6d06e45a827fabd59b026dc974d7	RESIDENT
U_2	03042cf8100db386818cee4ff0f2972431a62ed78edbd09a	RESIDENT
U_3	8920f3ce9287b4af6d9a5730bdda240fc74b3a5deb25e0b0	GUEST

Table 3: Illustrative Example User Table

DID	**GIDs**
CAM	RESIDENT, GUEST
LOCK	RESIDENT

Table 4: Illustrative Example Device Table

Users can select which groups they allow to access which devices, and can construct the groups to be as fine-grained as they choose. For instance, to restrict access to certain smart devices in different users' bedrooms that should not be accessible to users that do not sleep in those bedrooms, the users could potentially set a different group for each user of the smart home and create the access policy accordingly.

In this example, suppose U_2 is not at home, but their hired dog walker has arrived at the front door and needs access to the house so that they can enter and walk the dog. U_2 uses (Q_2, d_2) to authenticate with the home gateway according to the method described in Step 3. The home gateway uses (Q_{hg}, d_{hg}) to authenticate itself to the U_2. Three shared session keys are generated to create the session, providing perfect forward secrecy. This is because even if an attacker was somehow able to steal both (Q_2, d_2) and (Q_{hg}, d_{hg}), they still would not be able to view previously made requests from the user and previously provided feedback from the home gateway since a new encryption key based on randomly chosen secrets is generated each time users authenticate for a new session. Next, according to Step 4, U_2 sends a message to the home gateway msg = (LOCK || 0 || t_2) where 'LOCK' and '0' are indicating that the sender wants to unlock the front door. Note that if U_3 had authenticated and sent this message, they would not receive a response from the home gateway. This is because according to the device table shown in Table 4, guests are not included in the list of groups that have access to the smart lock. Upon receiving a message for control of the smart lock from U_3, the home gateway would check the device

table using queryDT and not proceed once 0 would be returned. Continuing with U_2's communications, in response to U_2's message, according to Step 5, the gateway sends a transaction chaining a record of U_2 requesting to unlock the front door to the blockchain. The gateway connects with the smart lock and executes the unlock, and the smart lock sends a '1' back to the home gateway to indicate that the request for unlocking was successful. The home gateway sends the feedback msg = $(1 \,||\, t_{hg})$ to U_2.

Since the authentication communications blinds the user's public key and is signed by the keys provided during user registration, the confidentiality of secrets, the integrity of the authentication messages, and the identity of the home gateway and the user is guaranteed. Since the communications for requests from the user and feedback from the home gateway are encrypted and signed by the three shared session keys that were established securely during authentication, the confidentiality of the user's actions and integrity of the requests and feedback are guaranteed. Since all of the authentication messages as well as the request and feedback messages are timestamped, this method achieves replay attack resistance.

Suppose U_3 has (Q_3, d_3) stolen by MAL. MAL uses (Q_3, d_3) to successfully send a request to view the live feed of the security camera. The home gateway responds with the live feed because Q_3 is registered in the user table and the GUEST group is allowed camera access according to the device table. Any user in the house can check the blockchain and see a record of U_3 viewing this

camera feed, even though U_3 Is not actually doing so. U_1 can then update or revoke (Q_3, d_3) according to how it is described in Step 2.

CHAPTER 7

PROTOTYPE IMPLEMENTATION AND EVALUATION

In this chapter, I will describe how I evaluated the performance as well as the qualities of this approach.

7.1 Prototype Implementation with Times

To evaluate the smart contract operations, I implemented a prototype of the blockchain using Hyperledger Fabric using the round robin consensus mechanism on my Ubuntu 20.04.1 machine with 2 cores and 16 GB of RAM. I elect to use round robin because a network-based consensus mechanism is more practical for this approach than using a computing power-based consensus mechanism. I generated a mock set of messages to be sent and stored, matching the size of how I described each message in each step. After deploying the smart contract, I used a Python script to obtain the average running time of the operations over 1000 trials. The results are shown in Table 5.

Using the same machine, I also timed each of the cryptographic functions that were involved in operational steps as well as cryptographic functions used in other similar approaches. Those results are shown in Table 6.

The results for the estimated total time taken for each operational step is shown in Table 7. Note that the time for Respond to Request does not include the time that it takes for the home gateway to connect with the target device, and for the target device to respond to the home gateway. The runtime of negligible, lightweight operations is omitted. Also, the evaluation of the System Setup is omitted because their runtime does not impact the regular use of the system.

	updateUT	queryUT	revokeUT	updateDT	queryDT	changeRequest
Avg Time (s)	4.157	1.553	4.159	4.144	1.448	4.145

Table 5: Average Times for Smart Contract Algorithms

	Encrypt/Decrypt	* on $\mathbb{G}$	Hash
Avg Time (s)	0.166	0.053	0.005

Table 6: Average Times for Cryptographic Algorithms

	User Registration	Mutual Authentication and Creation of Session Key	Send Request	Respond to Request
Avg Time (s)	4.215	3.532	0.337	6.267

Table 7: Average Times for Overall Steps

7.2 Comparison with Existing Systems

In this subsection, I will compare the costs of the overall approach with costs of other smart home system approaches.

This authentication design takes more time to process compared to the existing popular method of using cookies, which can be attributed to the additional cryptographic operations. The time it takes for takes for the home gateway to respond to a request from a user and the time it takes to update keys, revoke keys, and update the access policy is also longer than a username/password-based approach due to the chaining of the transaction. However, the time cost for each operational step is still of an order of seconds, which does not render this approach completely incompatible with smart homes. I argue that these time costs as well as the additional energy costs due to the blockchain could be considered worth it given the significant authentication, privacy, and auditability benefits. This approach is less easy to for regular smart home users to use than a traditional username/password-based approach, but is much more secure, and offers the additional feature of reliable records. To make the more cumbersome authentication of the approach need to happen less often, it would be trivial to implement session resumption by associating session IDs with each session.

7.3 Innovative Aspects of This Approach

The important features of this approach to a smart home system comparison to related approaches are shown in Table 8.

	[5]-[8]	[15]	[16]	[17]	[18]-[20]	[21]	This Approach
Data Ownership	No	Yes	No	Yes	No	Yes	Yes
Device Access Policy	No	No	No	No	No	No*	Yes
Credential Change Records	No	Yes	Yes	Yes	Yes	No*	Yes
Access Records	No	Yes	Yes	Yes	Yes	Yes	Yes
Time Overhead	L	H	H	M	H	H	M
Energy Consumption	L	M	H	M	H	M	M

L = Low, M = Moderate, H = High

*For the reasons outlined in Chapter 2, this approach to access control is not enforceable or appropriate for a multi-user home.

Table 8: Comparison of Smart Home System Approaches

Data ownership is important to allow users to have control of their own data. Device access policies allow users to achieve fine-grained access control. Of course, approaches that do not allow users to adjust access policies also do not record access policy changes, while my approach does. Credential modification records are important to keep in the case of a security incident. Access records are important to keep in the case of a security incident, and

convenient to keep for regular use. Increased time overhead and energy consumption are limitations of using blockchain, but they can be minimized. Based on the above comparison, this approach can achieve secure communication in smart homes because it has the first four properties while minimizing the negative impact of the last two properties.

CHAPTER 8

CONCLUSION AND FUTURE WORK

To prevent surveillance and tampering of smart devices, communication in smart homes is crucial to secure. The approach that I propose in this thesis achieves significant additional security properties while keeping computation and communication costs minimal and reasonable. The highlights of this approach are that it allows users to achieve fine-grained access control, and makes highly reliable, timestamped credential modification records, access records, and access policy modification records available to users.

In the future, I could implement a more extended prototype to better simulate real world scenarios to evaluate this approach, such as evaluating the performance of the cryptographic operations on a smart phone instead of just on a computer. Future work in the area of securing smart home communication against surveillance likely lies in protecting smart homes against privacy attacks on encrypted IoT traffic [26].

REFERENCES

[1] Do, Quang, et al. “Cyber-Physical Systems Information Gathering: A Smart Home Case Study.” Computer Networks, vol. 138, 2018, pp. 1–12. Crossref, doi:10.1016/j.comnet.2018.03.024.

[2] Do, Quang, et al. “Is the Data on Your Wearable Device Secure? An Android Wear Smartwatch Case Study.” Software: Practice and Experience, vol. 47, no. 3, 2016, pp. 391–403. Crossref, doi:10.1002/spe.2414.

[3] Do, Quang, et al. “A Data Exfiltration and Remote Exploitation Attack on Consumer 3D Printers.” IEEE Transactions on Information Forensics and Security, vol. 11, no. 10, 2016, pp. 2174–86. Crossref, doi:10.1109/tifs.2016.2578285.

[4] Huang, Danny Yuxing, et al. “IoT Inspector.” Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 4, no. 2, 2020, pp. 1–21. Crossref, doi:10.1145/3397333.

[5] https://www.amazon.com/echo-smart-hub/s?k=echo+smart+hub

[6] https://store.google.com/

[7] https://www.wink.com/products/

[8] https://www.samsung.com/us/smart-home/

[9] Hanamsagar, Ameya et al. “How Users Choose and Reuse Passwords.” (2016).

[10] Paganini, Pierluigi. “Less than 10% of Gmail Users Enabled Two-Factor Authentication.” Security Affairs, Security Affairs, 24 Jan. 2018, securityaffairs.co/wordpress/68125/digital-id/gmail-two-factor-authentication.html.

[11] Zheng, Xiaofeng. “Cookies Lack Integrity: Real-World Implications.” 24th USENIX Security Symposium, 2015, pp. 707–21, www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/zheng.

[12] Kim, Ji Eun, et al. “Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes.” 2012 Eighth International Conference on Intelligent Environments, 2012. Crossref, doi:10.1109/ie.2012.57.

[13] US Census Bureau. “Historical Households Tables.” The United States Census Bureau, Census, 23 Nov. 2020, www.census.gov/data/tables/time-series/demo/families/households.html.

[14] Zhang, Jinxin, and Meng Wu. “Blockchain Use in IoT for Privacy-Preserving Anti-Pandemic Home Quarantine.” Ring, 2020. electronic, www.mdpi.com/2079-9292/9/10/1746/pdf.

[15] Lin, Chao, et al. “HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes.” IEEE Internet of Things Journal, vol. 7, no. 2, 2020, pp. 818–29. Crossref, doi:10.1109/jiot.2019.2944400.

[16] Hammi, Mohamed Tahar, et al. “Bubbles of Trust: A Decentralized Blockchain-Based Authentication System for IoT.” Computers & Security, vol. 78, 2018, pp. 126–42. Crossref, doi:10.1016/j.cose.2018.06.004.

[17] Singh, Saurabh, et al. “SH-BlockCC: A Secure and Efficient Internet of Things Smart Home Architecture Based on Cloud Computing and Blockchain Technology.” International Journal of Distributed Sensor Networks, vol. 15, no. 4, 2019. Crossref, doi:10.1177/1550147719844159.

[18] Dorri, Ali, et al. “Blockchain in internet of things: Challenges and Solutions.” ArXiv abs/1608.05187 (2016): n. pag.

[19] Dorri, Ali, et al. “Blockchain for IoT Security and Privacy: The Case Study of a Smart Home.”

[20] Dorri, Ali, et al. “Blockchain for IoT Security and Privacy: The Case Study of a Smart Home.” 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2017. Crossref, doi:10.1109/percomw.2017.7917634.

[21] Xue, Jingting, et al. “Private Blockchain-Based Secure Access Control for Smart Home Systems.” KSII Transactions on Internet and Information Systems, vol. 12, no. 12, 2018. Crossref, doi:10.3837/tiis.2018.12.024.

[22] Hankerson, Darrel, et al. Guide to Elliptic Curve Cryptography. Vol. 46, vol. no. 1, New York-United States, United States, Springer Publishing, 2006.

[23] Johnson, Don, et al. “The Elliptic Curve Digital Signature Algorithm (ECDSA).” International Journal of Information Security, vol. 1, no. 1, 2001, pp. 36–63. Crossref, doi:10.1007/s102070100002.

[24] Lauter, Kristin. “The Advantages of Elliptic Curve Cryptography for Wireless Security.” IEEE Wireless Communications, vol. 11, no. 1, 2004, pp. 62–67, doi:10.1109/MWC.2004.1269719.

[25] De Win, Erik, et al. “On the Performance of Signature Schemes Based on Elliptic Curves.” Lecture Notes in Computer Science, 1998, pp. 252–66. Crossref, doi:10.1007/bfb0054867.

[26] Apthorpe, Noah et al. “Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic.” ArXiv abs/1708.05044 (2017): n. pag.