ADDRESSING THE SECURITY AND EFFICIENCY CHALLENGES IN INTERNET OF THINGS

By

Xinyu Lei

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2021

ABSTRACT

ADDRESSING THE SECURITY AND EFFICIENCY CHALLENGES IN INTERNET OF THINGS

By

Xinyu Lei

Nowadays, Internet of things (IoT) devices (e.g., smart cameras, Amazon Alexa, GPS navigation devices) are increasingly popular in our daily life. In practice, IoT devices are usually supported by their infrastructures (such as cloud servers, blockchain systems) to provide a variety of services. Some examples are given as follows. First, smart home Wi-Fi IoT devices can connect to their IoT vendor servers over the Internet, and they can be remotely monitored and controlled. Second, IoT devices along with blockchain systems have been implemented in various industries including financial, supply chain management, smart agriculture, cryptocurrency-supported vending machine, etc. Third, IoT devices can produce/collect datasets (e.g., locations) and upload them to powerful public cloud servers for storage. Then, the cloud server (serves as the IoT infrastructure) can deliver different data queries (e.g., kNN queries) services to data users. For both IoT devices and IoT infrastructures, there are many security and efficiency challenges that are needed to be addressed. For example, IoT devices usually have limited hardware capabilities, so they may not support secure communications (i.e., SSL/TLS connections). Moreover, blockchain systems may suffer from double-spending attacks and public clouds may steal the datasets in their storage. In this work, we propose various solutions to address these security and efficiency challenges. They are introduced as follows.

To address security and efficiency challenges in IoT devices, we have two studies. First, in our project targeting smart home Wi-Fi-connected IoT devices, we conduct an empirical study on how the cryptographic/security protocols (e.g., SSL/TLS) are supported on 40 popular Wi-Fi smart home IoT devices. Surprisingly, we discover two security vulnerabilities and show that adversaries can exploit them to hijack the victims' IoT devices or peek at victims' activities. To secure these smart home IoT devices, we present SecWIR (**Sec**ure **Wi**-Fi **I**oT communication **R**outer) framework, which is deployed on the commercial off-the-shelf (COTS) home Wi-Fi routers. Our experimental results show that SecWIR can secure IoT devices at the expense of only a small reduction in the routing performance. Second, in our project on home digital voice assistants (HDVAs), we study the insecurity of HDVA services by using Amazon Alexa and Google Home as case studies.

We disclose three security vulnerabilities that root in their insecure access control. The insecure access control means that HDVA devices not only solely rely on single-factor authentication but also take voice commands even if no people are around them. To address the venerability, we devise a **V**irtual **S**ecurity **Button** (VSButton), which leverages a real-time outlier detection algorithm on Wi-Fi signal to detect indoor human motions. Only when indoor human motions are detected, VSButton activates the HDVA devices and allows them to accept voice commands. At last, we conduct experiments to demonstrate the efficiency and effectiveness of VSButton.

To address security and efficiency challenges in IoT infrastructures, we have two studies. First, in our project on reducing the transaction validation time on Bitcoin blockchain, we focus on designing fast Bitcoin transaction validation protocols which can help to promote the IoT-blockchain services (e.g., Bitcoin-supported vending machine). Currently, a secure Bitcoin transaction requires the payee to wait for at least 6 block confirmations (one hour) to be validated. In our project, we propose BFastPay scheme to accelerate the Bitcoin transaction validation. BFastPay employs a smart contract called BFPayArbitrator to host the payer's security deposit and fulfills the role of a trusted payment arbitrator which guarantees that a payee always receives the payment even if attacks occur. BFastPay is a routing-free solution that eliminates the requirement for payment routing in the traditional transaction routing network (e.g., Lightning Network). The theoretical and experimental results show that BFastPay is able to significantly reduce the Bitcoin transaction waiting time. Second, in our project on providing secure IoT-cloud service, we focus on k nearest neighbor (kNN) queries service. Nowadays, location service providers (LSPs) often resort to IoT devices (e.g., GPS navigation devices) to collect geospatial data. In practice, LSPs may rely on commercial cloud services, e.g., Dropbox, to store the tremendous geospatial data and deal with a number of user queries. However, it is challenging to achieve a secure and efficient location-based query processing over encrypted geospatial data stored on the untrusted cloud. In this project, we propose SecEQP (**Sec**ure and **E**fficient **Q**ueries **P**rocessing) scheme to address the secure kNN query problem. Our theoretical analysis and experimental evaluation demonstrate that SecEQP is secure and efficient.

In summary, we successfully address the security and efficiency challenges in different IoT devices (including smart home IoT devices and HDVAs) and IoT infrastructures (including blockchain systems and cloud servers) in this work. We believe that our work can promote the fast growth of the IoT industry.

Dedicated to my parents

ACKNOWLEDGMENTS

First, I would like to give my deep gratitude to my Ph.D. advisors Guan-Hua Tu and Prof. Alex X. Liu. During the past five years, I had the precious opportunity to work with them. In their labs, I have learned how to conduct research from choosing good topics, conducting experiments to writing high-qualified papers. Their strong skills for academic research and paper writing have been, and will always be, beneficial for my future career.

Besides, I thank another two professors who serve as my Ph.D. committee members (i.e., Prof. Xiao Li and Prof. Yuying Xie). They provide valuable feedback on my research during the comprehensive exam and the final defense. Based on the feedback, the quantity of my research work has been significantly improved.

Then, I would like to thank all members in "Security, Networking and Mobile Systems Research Lab" (including Tian Xie, Yiwen Hu, Sihan Wang, and Jingwen Shi) as well as all members in "System and Security Lab" (including Ali Munir, Faraz Ahmed, Kamran Ali, and Liyan Wang). I have learned a lot from them during the collaboration and discussion we had in the lab.

Next, I would like to thank other friends in MSU. Most of them are Ph.D. students in the department of CSE at MSU. These friends (such as Yichun Shi, kaixiang Lin, Zhiwei Wang, Yao Ma) introduce to me some new problems in machine learning, data mining, and computer vision. Thus, I have gained extra knowledge which would be helpful to my future interdisciplinary research.

Last but not least, I thank my parents for their love and support. Without them, I would not have the chance to pursue a Ph.D. degree at MSU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction and Motivation

1.1 Motivation

Internet of things (IoT) is the extension of Internet connectivity into physical devices. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with their infrastructures (such as cloud servers). The number of IoT devices is forecasted to grow from 115 million in 2018 to 320 million in 2020 with a compound annual growth rate of 40.65% [107]. Of the many different types of wireless IoT devices available (including cellular, ZigBee, and SigFox), the Wi-Fi-connected IoT devices are particularly popular among home users. For example, according to the Amazon selling records, seven of the top 10 best sellers of electrical outlet switches are Wi-Fi-connected devices. A recent report [114] also forecasts that the number of global Wi-Fi devices in homes will reach 17 billion in 2030 from 4 billion in 2019, and 60% of them are smart home devices.

In practice, IoT devices are usually supported by their infrastructures (such as blockchain systems, cloud servers) to provide a variety of services. For example, smart home Wi-Fi IoT devices can connect to their IoT cloud servers over the Internet, and they can be remotely monitored and controlled. The home digital voice assistant (HDVA) devices (such as Amazon Alexa) can connect to remote voice processing cloud servers via the Internet. The voice processing cloud servers can help to analyze the users' voice commands, and the HDVA devices can respond accordingly. In addition, IoT devices can connect to blockchain systems to provide different services. The IoT devices create tamper-resistant records of shared transactions on blockchain systems. Each transaction can be verified to prevent disputes and build trust among all blockchain network nodes. The IoT devices along with blockchain systems has been implemented with across industries including financial services [64], supply chain management [33], smart agriculture [83], cryptocurrency-supported vending machine [39], etc. Furthermore, IoT devices can produce/collect datasets (e.g., locations) and upload them to cloud servers for storage. Then, the cloud server can deliver differ-

ent data queries (e.g., kNN queries) services to data users. In this application, the cloud servers provide the infrastructure support to the IoT devices.

1.2 Problems in Current IoT Communication, Authentication, Blockchain, and Cloud Services

In this section, we introduce the security and efficiency challenges in both IoT devices (including smart home IoT devices, HDVAs) and IoT infrastructures (including blockchain systems and public cloud servers).

1.2.1 Insecure IoT Communication

Due to limited hardware capabilities and heterogeneous architectures, some common supported secure solutions (e.g., anti-virus software) cannot be deployed on some IoT devices. Therefore, IoT devices are more likely to have security vulnerabilities than a PC. For example, one popular IoT platform, Delta DFCM-NNN40, has only 256 KB flash memory [84]. Supporting SSL/TLS protocols for IoT devices requires as a minimum a lightweight SSL/TLS library and a list of trusted certificate authorities. The former library (e.g., WolfSS- L [117]) needs around 20 KB-100 KB flash memory space, while the latter requires 250 KB [45]. That is, a minimum memory space of 270 KB is required, which exceeds the 256 KB available on the DFCM-NNN40 platform. In addition, not all IoT devices support remote software/firmware updates. Consequently, it is not easy for IoT vendors to patch these security vulnerabilities of their devices without expensive recalls. Therefore, IoT devices are likely to have security vulnerabilities in their communications and the vulnerabilities are hard to be addressed by remote software/firmware updates.

1.2.2 Imprudent IoT User Authentication

For HDVAs, to provide users with usage convenience, most HDVA devices (e.g., Amazon Echo, Google Home) adopt an always-listening mechanism which takes voice commands all the time. Specifically, users are not required to press or hold a physical button on HDVA devices before speaking commands. This weak single-factor authentication mechanism may pose security threats to HDVA users. For example, Amazon Alexa employs a single-factor authentication method based

on a password-like voice word, to authenticate users who intend to access the voice service. The voice command recognition mechanism of Alexa services does not consider if the speakers are authorized users (e.g., the Alexa device owner or the owner's family members) but the semantics of the received voice commands. Therefore, any users who know the voice commands can access the Alexa services on behalf of the victims. Thus, we believe a second-factor user authentication should be designed for HDVAs. The second-factor user authentication mechanism should not significantly reduce user experience.

1.2.3 Low Efficiency of Blockchain-based IoT Services

IoT devices often resort to blockchain systems to provide different services. For blockchain systems, IoT devices may broadcast transactions on blockchain systems. The IoT-blockchain service models may need blockchain systems to record the transactions that are tamper-resistant. However, most blockchain systems are not efficient. For example, Bitcoin blockchain require 6 block confirmations for a transaction to be validated. It takes about 1 hour. Such a long waiting time thwarts the wide deployment of some IoT-blockchain services (e.g., Bitcoin vending machine). The 6 block confirmations are required to resist the possible double-spending attacks. Note that 6 block confirmations are based on an assumption that adversaries do not control more than 10% of the global hash power of the Bitcoin network and a double-spending probability of less than 0.1% is acceptable [91]. To provide more efficient IoT-blockchain services (e.g., Bitcoin vending machine), we need to develop solutions to reduce the Blockchain transaction validation time while still defending against double-spending attacks.

1.2.4 Relying on Untrusted Public Clouds for IoT Services

IoT devices often rely on public clouds for IoT services. In practice, IoT devices may offload datasets to public commercial clouds. The IoT-cloud service models may lead to security and privacy concerns because the datasets may contain the private information (such as biometric datasets, financial datasets, location datasets) of the IoT users and because the public clouds are typically not fully trusted. For instance, the clouds have financial incentives to collect and sell the IoT devices' uploaded datasets. In addition, the corrupted cloud employee may peak and spy on the datasets uploaded by IoT devices. Moreover, the public clouds may be compromised and all of the stored

information may be leaked to hackers. For instance, it is reported that Dropbox is hacked and more than 68 million Dropbox account information is now for sale on the DarkNet marketplace [5]. A straightforward solution is to require IoT devices to encrypt (such as using AES) their datasets before outsourcing to the public cloud. However, the encrypted datasets are pseudo-numbers, making it impossible for the cloud to use the datasets. Therefore, we need to develop solutions to support secure IoT dataset storage on the public cloud while preserving the cloud's ability to use the IoT data to deliver services (e.g., kNN search).

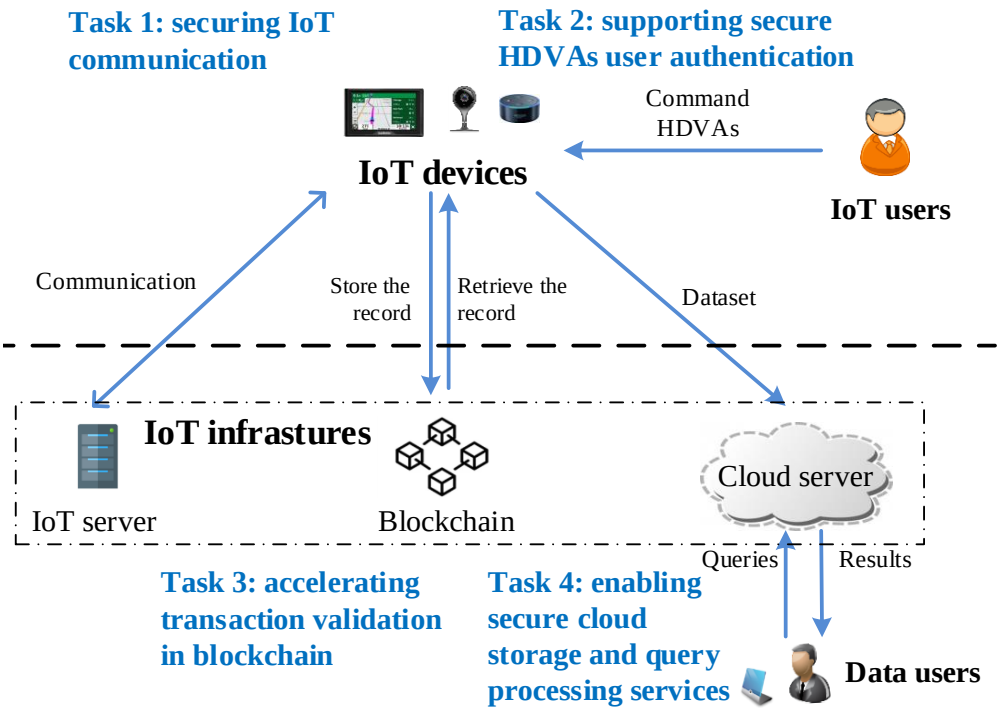


Figure 1.1: Research tasks overview.

1.3 Research Tasks Overview

To address the above security and efficiency challenges in IoT devices and IoT infrastructures, we have four main research tasks in this work. Figure 1.1 overviews the four research tasks. The tasks can be divided into two categories: IoT devices-side tasks and IoT infrastructure-side tasks. For IoT device security, we have two tasks. The first task is to secure the IoT communications. The second is to support secure HDVAs user authentication. For IoT infrastructure security, we have two tasks. The first one is to accelerate transaction validation in the blockchain. The second one is

to enable secure cloud storage of IoT data while preserving the cloud's ability to provide queries processing services.

1.4 Our Contributions

We address some security and efficiency challenges both in IoT devices and IoT Infrastructures. Our research contribute to more secure and efficient IoT devices and IoT Infrastructures.

1.4.1 Toward Secure IoT Communication and User Authentication

We seek to make IoT devices more secure and efficient via the following two studies.

We first study 40 of the best selling Wi-Fi smart home IoT devices on the Amazon platform. It is shown that 29 of these devices have either no security protocols deployed, or have problematic security protocol implementations. Seemingly, these vulnerabilities can be easily fixed by installing security patches. However, many IoT devices lack the requisite software/hardware resources to do so. To address this problem, the present study proposes a SecWIR framework designed for implementation on top of the users' existing home Wi-Fi routers to provide IoT devices with a secure IoT communication capability. However, it is way challenging for SecWIR to function effectively on all home Wi-Fi routers since some routers are resource-constrained. Thus, several novel techniques for resolving this implementation issue are additionally proposed. The experimental results show that SecWIR performs well on a variety of COTS Wi-Fi routers at the expense of only a small reduction in the non-IoT data service throughput (less than 8%), and small increases in the CPU usage (4.5%$\sim$7%), RAM usage (1.9 MB$\sim$2.2 MB), and the IoT device access delay (24 ms$\sim$154 ms) while securing 250 IoT devices.

Second, we study the insecurity of HDVA services by using Amazon Alexa and Google Home as case studies. We disclose three security vulnerabilities which root in their insecure access control. We then exploit them to devise two proof-of-concept attacks, home burglary and fake order, where the adversary can remotely command the victim's HDVA device to open a door or place an order from Amazon.com or Google Express. The insecure access control is that HDVA devices not only rely on a single-factor authentication but also take voice commands even if no people are around them. We thus argue that HDVAs should have another authentication factor, a physical

presence based access control; that is, they can accept voice commands only when any person is detected nearby. To this end, we devise a VSButton, which leverages the WiFi technology to detect indoor human motions. Once any indoor human motion is detected, the HDVA device is enabled to accept voice commands. Our evaluation results show that it can effectively differentiate indoor motions from the cases of no motion and outdoor motions in both laboratory and real world settings.

1.4.2 Toward Efficient and Secure IoT Service Infrastructure

We seek to make IoT infrastructures more secure and efficient via the following two studies.

First, we study how to accelerate the Blockchain transaction validation. IoT devices may broadcast transactions on blockchain systems to provide different services. The IoT-blockchain service models may need blockchain systems to validate and record the transactions as fast as possible. In our study, we use the most popular Bitcoin blockchain as a case study. In practice, a secure Bitcoin transaction requires the payee to wait for at least 6 block confirmations (one hour) to be validated. Such a long waiting time thwarts the wide usage of the Bitcoin blockchain system because many usage scenarios require a much shorter waiting time. In our project, we propose BFastPay to accelerate the Bitcoin transaction/payment validation. BFastPay employs a smart contract called BFPayArbitrator to host the payer's security deposit and fulfills the role of a trusted payment arbitrator which guarantees that a payee always receives the payment even if attacks occur. BFastPay is a routing-free solution that eliminates the requirement for payment routing in the traditional payment routing network (e.g., Lightning Network). The theoretical and experimental results show that BFastPay is able to significantly reduce the Bitcoin payment waiting time (e.g., from 60 mins to less than 1 second) with nearly no extra operation cost.

Second, we study how to protect IoT data security against untrusted cloud servers, while still preserving the cloud's ability to answer kNN queries. IoT devices may offload datasets to public commercial clouds, which can provide queries processing services to the IoT data users. In our project, we focus on kNN queries over IoT-uploaded location datasets. Nowadays, location-based services are proliferating and being widely deployed. For example, a Yelp user can obtain a list of the recommended restaurants near his/her current location. Location service providers (LSPs) often resort to IoT devices (e.g., GPS navigation devices) to collect geospatial data. In practice,

LSPs may rely on commercial cloud services, e.g., Dropbox, to store the tremendous geospatial data and deal with a number of user queries. However, it is challenging to achieve a secure and efficient location-based query processing over encrypted geospatial data stored on the cloud. In this project, we propose the SecEQP scheme to address the secure k nearest neighbor (SkNN) query problem. SecEQP employs the projection function-based approach to code neighbor regions of a given location. Given the codes of two locations, the cloud server only needs to compare whether codes equal or not to check the proximity of the two locations. The codes are further embedded into an indistinguishable Bloom filter tree to build a secure and efficient index. The security of SecEQP is formally proved in the random oracle model. We further prototype SecEQP scheme and evaluate its performance on both real-world and synthetic datasets. Our evaluation results show that SecEQP is a highly efficient approach, e.g., top-10 NN query over 1 million datasets only needs less than 40 msec to get queried results.

1.5 Dissertation Structure

Now we lay out the structure of this dissertation. Chapter 3 and 4 focus on addressing the security and efficiency challenges in IoT devices, while Chapter 5 and 6 focus on addressing these challenges in IoT infrastructures.

Chapter 3 presents our SecWIR project [78]. We first present the introduction of our project. We then present the related work. Next, we introduce the threat model, assumptions, and security guarantees underlying the development of the proposed SecWIR framework. We next introduce our empirical security study. We discover two security vulnerabilities on the tested IoT devices: (V1) a lack of security protocol support and (V2) flawed certificate validation. To address them, we design SecWIR. SecWIR consists of three modules: 1) IoT secure tunneling module, 2) stream security validation module, and 3) resource monitoring module. Subsequently, we have the security analysis of SecWIR, and then, implements and evaluates SecWIR, Last, we discuss some potential issues.

Chapter 2 presents the background of the studied IoT devices and IoT infrastructures. Then, we further present the related state-of-the-art IoT devices and infrastructure security studies.

Chapter 4 introduces our VSButton project [79]. We first present the introduction of our project.

Then, some related works are reviewed. Next, the VSButton design is introduced in detail. The VSButton adopts the channel state information (CSI) to detect human motions. It consists of the CSI processing phase and the outlier detection phase. Finally, we evaluate its performance in real-world settings and discuss several remaining issues.

Chapter 5 presents our BFastPay project [80]. First, the project introduction and related works are presented. Then, we introduce some background knowledge. Next, we introduce the adopted threat model and assumptions. We present an overview of BFastPay. BFastPay is designed based on two key insights. First, it employs a decentralized smart contract to host the payer's security deposit and fulfill the role of a trusted payment arbitrator which guarantees that a payee always receives the payment even if attacks occur. Second, BFastPay takes advantage of the fast consensus property of the emerging programmable smart contract (PSC)-supported blockchains to reduce the waiting time of the Bitcoin transaction. Subsequently, the arbitration mechanism used by BFastPay is illustrated. We then perform security analysis and evaluates the operation cost of BFastPay. Last, we discuss some remaining issues.

Chapter 6 introduces our SecEQP project [80]. First, the project introduction and related works are presented. We then introduce how to realize space encoding via projection functions. Next, we describe how to process kNN queries in the plaintext domain. In SecEQP, a prefix-free encoding method is used to embed the codes into an indistinguishable Bloom filter tree to build a secure index. By using the binary search over the indistinguishable Bloom filter tree, SecEQP can ensure a sublinear search time. Subsequently, we introduce how to transform the designed kNN protocol to be a secure and sublinear protocol. The related analysis is also well presented. Last, we conduct the performance evaluation.

In Chapter 7, we conclude the dissertation.

Chapter 2

Background and State-of-the-Art

In this chapter, we introduce the background of the considered IoT devices and IoT infrastructures. We further present the related state-of-the-art IoT devices and infrastructure security studies.

2.1 Background

2.1.1 IoT Devices Background

The background of smart home IoT devices and HDVAs is introduced below.

Smart Home IoT Devices. Figure 2.1 illustrates the service model which is commonly used by Wi-Fi-connected IoT devices nowadays. As shown, the devices, e.g., a smart camera and Amazon Alexa voice assistant, communicate with the IoT server in the vendor realm (e.g., *.tplink.com) via the Wi-Fi router and the owner communicates with the devices using a vendor-specific IoT application installed on a smartphone.

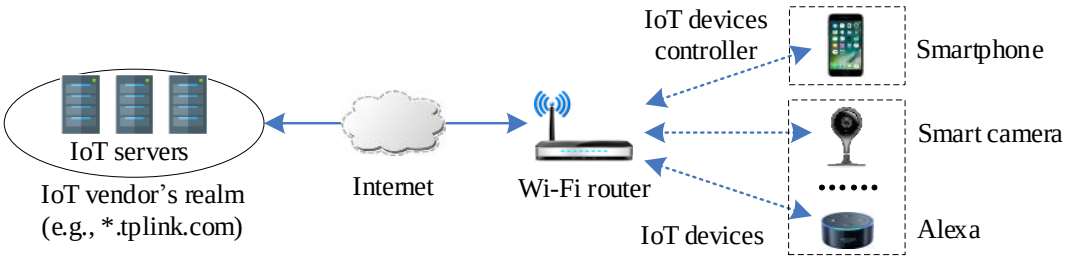


Figure 2.1: Wi-Fi smart home IoT service model.

HDVAs. Among all smart home Wi-Fi-connected smart home IoT devices, we are particularly interested in HDVAs. HDVAs rely on cloud servers to processing voice commands and respond accordingly. In the following, we use Amazon Alexa devices as examples to introduce the background of HDVAs. There are five kinds of Alexa devices: Amazon Echo, Amazon Tap, Echo Dot, Echo Spot, and Echo Show, as shown in Figure 2.2. To support voice commands, they connect to a cloud-based Amazon voice service, *Alexa*. Amazon Echo is the first generation Alexa device.

It always stays in a listening mode, so it does not take any voice commands until a voice word “Alexa" wakes it up. Every time it wakes up, it serves only one voice command and then returns to the listening mode. It appears as a 9.25-inch-tall cylinder speaker with a 7-piece microphone array. Amazon Tap is a smaller (6.2-inch-tall), portable device version with battery supply, but has similar functions. Echo Dot is the mini version of Echo, which is a 1.6-inch-tall cylinder with one tiny speaker. Amazon Echo Spot and Echo Show are the latest versions which provide a small-size display screen and a large-size display screen, respectively. It can be expected that more and more Alexa devices are going to be released in the future. Currently, all of the Alexa devices (except the Amazon Tap), which require plug-in power supplies, are usually deployed at a fixed location (e.g., inside a room). In this work, we focus on these non-portable Alexa devices and mainly use Echo Dot to examine the Alexa voice service.

Figure 2.2: Alexa devices.

The Alexa voice service supports the recognition of voice commands to Alexa devices. Figure 2.3 illustrates how the voice service works with Alexa devices to control smart home devices (e.g., smart bulb, thermostat, etc.). To control a smart device, a user can speak a voice command to an Alexa device after waking it up with voice “Alexa". The Alexa then sends the sounds of that voice command to a remote voice processing cloud via its connected Wi-Fi network. Once the cloud recognizes the sounds as a valid command, it is forwarded to a server, called *smart home skill adapter*, which is maintained by Amazon to enable the cooperation with third-party service providers. Afterward, that command is sent to another cloud which can control the corresponding smart device remotely. Note that in addition to the control of smart devices, some functions (e.g., checking the weather, placing orders on Amazon.com, etc.) provided by Alexa devices can also be accessed by voice commands.

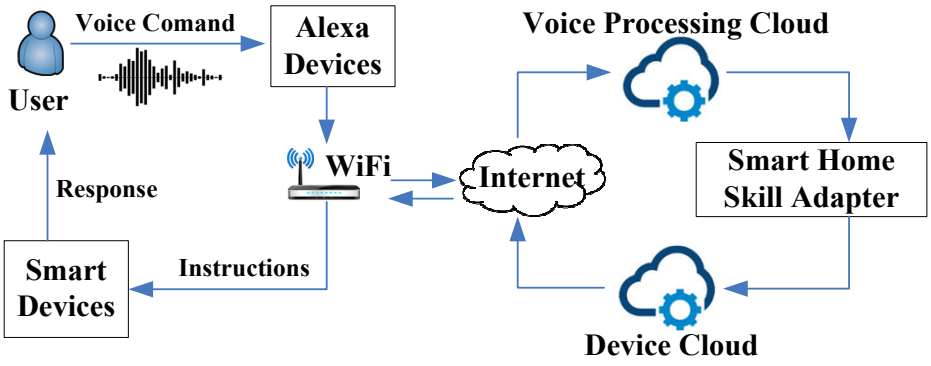


Figure 2.3: Alexa voice service model.

2.1.2 IoT Infrastructures Background

IoT devices often resort to IoT Infrastructures to provide different services. The commonly used IoT infrastructures include blockchain systems and public cloud servers. We introduce the IoT-blockchain service model and the IoT-cloud service model as follows.

IoT-Blockchain Service Model. The typical IoT-blockchain service model is shown in Figure 2.4. The IoT devices can send transactions to record information on blockchain systems. The records are tamper-resistant and maintained by blockchain systems. Each transaction can be verified to prevent disputes and build trust among all blockchain network nodes. Later, the IoT devices can retrieve information from the blockchain systems to support different applications. As mentioned before, IoT-blockchain service has been implemented across industries including financial services, supply chain management, smart agriculture, cryptocurrency-supported vending machine, etc. In this work, we focus on accelerating blockchain transaction validation. As a result, our study can improve the efficiency of many IoT-blockchain services.

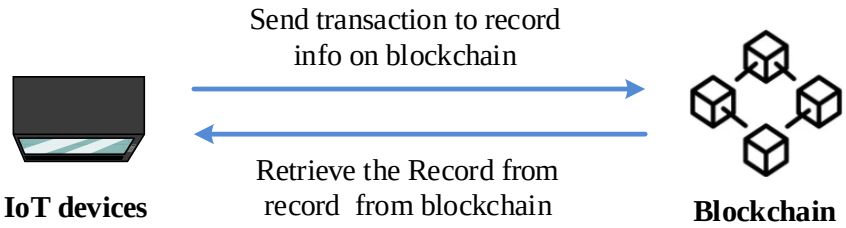


Figure 2.4: Targeted IoT-blockchain service model.

IoT-Cloud Service Model. The typical IoT-cloud service model is depicted in Figure 2.5. The IoT devices produce or collect some datasets and outsource them to a powerful cloud for data storage.

Later, some data users can selectively download some data items from the server according to the submitted queries. The supported queries include keyword queries, range queries, and kNN queries. In this work, we focus on kNN queries in the IoT-cloud service model.

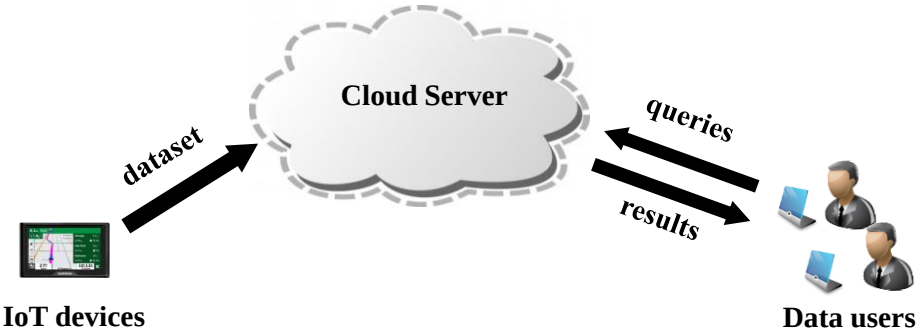


Figure 2.5: Targeted IoT-cloud service model.

2.2 State-of-the-Art

The problem of securing IoT devices has been extensively examined in recent years. Broadly speaking, existing proposals for IoT device security can be categorized as either device-based approaches [110, 94, 72, 57, 76] or infrastructure-based approaches (e.g., IoT security gateways [44, 120], customized secure systems [88], in-hue secure manager [105], and customized securebox [62]). The device-based approaches require the after-market IoT devices to possess the capabilities required to deploy the proposed security mechanisms. Thus, they may not be suitable for all IoT device platforms; particularly those with resource constraints. Meanwhile, the infrastructure-based approaches typically require the users to purchase additional security hardware components, the cost of which may deter the users, to secure the IoT devices; hence, they also have only a limited practical applicability. Moreover, some IoT security gateways (e.g., Samsung SmartHome and Philips Hue) only support certain IoT devices.

To provide more secure user authentication for HDVAs, several second-factor user authentication solutions have been proposed. They can be roughly classified into two categories: (1) device-free biometric-based solution and (2) wearable-aid biometric-based solution. (1) The device-free solution does not require users to carry any device to finish the user authentication. The authors in [77] proposed the approach to authenticate users by recognizing the users' voice. It has three

limitations. First, it requires the pre-training process. Second, users' voices may vary with their ages, illness, or tiredness. Third, it cannot resist users' voice replay attacks. Meng et al. [86] present WiVo, which has two limitations. First, it requires users to purchase Wi-Fi signal antennas. Second, it cannot ensure high accuracy. (2) The wearable-aid biometric-based solutions for user authentication for HDVAs are illustrated as follows. The wearable-aid solution requires users to wear a special device to finish the user authentication. The approach in [121] requires the user to use the smartphone as a Doppler radar to detect the unique articulatory gesture of the user to help user authentication. Feng et al. [56] develops a proprietary wearable device for user authentication. The wearable-based solution has one major limitation. That is, it forces users to carry a device to finish the authentication, and therefore, it significantly decreases user convenience.

To provide efficient blockchain infrastructure support for IoT devices, we study how to accelerated blockchain transaction validation. We use the most popular Bitcoin blockchain as a case study. The prior solutions are introduced as analyzed below. First, one straightforward solution is to enforce users to use some other cryptocurrencies with faster transaction validation time. However, since Bitcoin has dominated in practical usage [1], it is desirable to develop solutions to support fast Bitcoin payment while keeping Bitcoin as the major payment currency. Second, the solutions proposed in [70, 39, 100] deploy observers in the Bitcoin network to detect the conflicting Bitcoin transactions (i.e., multiple transactions that spent the same Bitcoin). The prevention-based solutions are not highly reliable since the conflicting Bitcoin transactions detection is probabilistic. Third, secure wallet-based solutions [54, 108] require users to trust the secure wallet, so they cannot ensure decentralization. Last, the Lightning Network [101] and Duplex Micropayment Channels [53] are escrow-based solutions which support fast payment via payment routing network.

To provide secure cloud infrastructure support for IoT devices, we study how to protect IoT data security against untrusted cloud servers, while still preserving the cloud's ability to answer kNN queries from data users. The prior art can be classified into five categories, which are introduced as follows. First, schemes based on location obfuscation [89], and data transformation [73, 117] do not use strong standard encryption algorithm. Therefore, they suffer from weak privacy. Second, the Private Information Retrieval (PIR)-based solutions [119] mainly consider protecting query privacy but not data privacy. Besides, PIR-based solutions suffer from long query latency for large-scale datasets. Thrid, fully homomorphic encryption (FHE) [59] enables cloud to perform

*k*NN computation directly over the encrypted data. However, current FHE solutions still lack efficiency. Forth, distance-recoverable encryption (DRE)-based schemes [117, 66] and Order-preserving encryption (OPE)-based S*k*NN schemes [117, 112] achieve weak security, as analyzed in [92]. Last, voronoi-based scheme [118] requires each data user to download and maintain a copy of the large-size index locally for query processing, which seriously impedes its real-world applications.

Chapter 3

SecWIR: Securing Home IoT Devices

3.1 Introduction

3.1.1 Background and Motivation

With the increasing popularity and sophistication of wireless smart home IoT devices nowadays (e.g., smart sockets, smart bulbs, and smart cameras), smart home systems are gradually moving into the mainstream. The number of IoT devices is forecasted to grow from 115 million in 2018 to 320 million in 2020 with a compound annual growth rate of 40.65% [107]. Of the many different types of wireless IoT devices available (including cellular, ZigBee, and SigFox), the Wi-Fi-connected devices are particularly popular among home users. For example, according to the Amazon selling records, seven of top 10 best sellers of electrical outlet switches are Wi-Fi-connected devices. A recent report [114] also forecasts that the number of global Wi-Fi devices in homes will reach 17 billion in 2030 from 4 billion in 2019, and 60% of them are smart home devices. Compared with other types of IoT devices, Wi-Fi-connected devices have two significant advantages. First, they do not require the users to purchase additional IoT vendor home gateways/hubs, such as Samsung SmartThing Hub, to connect to and access them (i.e., they depend only on the users' existing Wi-Fi routers at homes). Second, Wi-Fi IoT devices are usually much cheaper than the alternatives. For example, a Samsung SmartThing outlet costs $35, whereas an Etekcity Wi-Fi smart outlet has a cost of less than $10. Thus, Wi-Fi-connected IoT devices offer users a high degree of convenience and functionality at only a moderate price.

Despite the many advantages which IoT devices can bring to daily life, they also offer an appealing target to malicious adversaries seeking to launch cyberattacks, such as phishing, identity theft, and distributed denial of service (DDoS). Consequently, the security of IoT communication is an important concern. Unfortunately, our study on 40 best-selling smart home IoT devices on the Amazon platform, yields a negative answer. We have two findings. First, many smart home IoT devices do not support any security protocols, and hence data confidentiality and integrity pro-

tection are not supported for the communications between these devices (referred to henceforth as **NonSecIoT** devices) and IoT servers. Second, while a small number of devices do offer some form of security protection, the related protocols are not compliant with standards and fail to protect the devices (referred to henceforth as **InSecIoT** devices) from malicious attacks. By exploiting these vulnerabilities, adversaries can launch a variety of attacks, including (but not limited to) remotely controlling users' appliances and capturing users' real-time images/videos, respectively. The results are summarized in Table 3.1.

At first glance, IoT vendors are guilty of ignoring security issues in marketing and distributing their products. Moreover, it seems intuitive that vendors could easily overcome the problem by simply patching their servers and shipped devices. In practice, however, the situation is far more complex. For example, of the 40 smartphone-controlled IoT devices mentioned above, 38 of the related smartphone control applications were found to communicate with the IoT servers through SSL/TLS security protocols. In other words, most IoT vendors do in fact support security protocols in their communication infrastructures between the IoT control applications and the IoT servers. The question therefore arises as to why most IoT vendors do not deploy any security protocols on the IoT devices themselves. There appear to be two possible reasons for this. First, *Wi-Fi-connected IoT devices simply lack the resources required to deploy mainstream security protocols*. For example, one popular IoT platform, Delta DFCM-NNN40, has only 256 KB flash memory [85]. Supporting SSL/TLS protocols for IoT devices requires as a minimum a lightweight SSL/TLS library and a list of trusted certificate authorities. The former library (e.g., WolfSSL [115]) needs around 20 KB-100 KB flash memory space, while the latter requires 250 KB [47]. That is, a minimum memory space of 270 KB is required, which exceeds the 256 KB available on the DFCM-NNN40 platform. Second, *not all IoT devices support remote software/firmware updates*. Consequently, it is not easy for IoT vendors to patch these security vulnerabilities of their devices without expensive recalls. We thus believe that there is a pressing need to develop novel approaches to secure these IoT devices; otherwise, they may be abused to launch a variety of cyberattacks.

Table 3.1: Security vulnerabilities of 40 Wi-Fi smart home IoT devices (✓: vulnerability detected, ✗: vulnerability not detected, NA: not applicable).

Category	Type	Manufacturer	Model	Price	Num. of customer reviews@Amazon	V1: Lack security protocol support	V2: Flawed certificate validation
Remote Control	Voice Assistant	Amazon	Echo Dot	$50	117048	✗	✗
		Google	Home Mini	$49	NA	✗	✗
Automation	Smart Socket	Etekcity	ESW01-USA	$10	3180	✓	NA
		Belkin Wemo	F7C063	$30	10902	✓	NA
		Geekbes	YM-WS-5	$9	222	✓	NA
		TanTan	TANTANSMART01	$10	1006	✓	NA
		TECKIN	SP10	$10	265	✓	NA
	Smart Strip	Foseal	1700-Joule	$33	70	✓	NA
		GXA	ConsumerElec	$29	26	✓	NA
		TONBUX	Powerstrip02	$30	331	✓	NA
		KMC	B0781SVT8B	$25	51	✓	NA
		mengyasi	B07216SSZY	$33	80	✓	NA
	Smart Bulb	Tp-Link	LB100	$30	2121	✓	NA
		IVIEW	ISB600	$14	297	✓	NA
		UPSTONE	YCL-1001	$13	152	✓	NA
		Lotton	B075882X14	$17	106	✓	NA
		LOHAS	B01MYQCXOH	$17	741	✓	NA
	Thermometer	Honeywell	RTH6580WF	$86	2044	✓	NA
		La Crosse	S85814	$32	224	✓	NA
		Netatmo	NWS01-US	$127	813	✓	NA
		Ecobee	ecobee4	$228	1098	✗	✗
		Emerson	ST55	$107	426	✗	✓
Appliance	Humidifier	DIKLA	B072TZDF76	$38	11	✓	NA
		Essential	B07BF3MFH8	$37	74	✓	NA
		RENPHO	B076VP1LPL	$30	365	✓	NA
		ASAKUKI	B076F73M82	$38	81	✓	NA
		Viva	B071XK49MN	$40	187	✓	NA
Security	Smart Camera	360	D503 HD	$40	181	✗	✗
		Zmodo	CS-S1U-WS-1	$40	4145	✗	✓
		YI Dome	720p HD	$35	4928	✗	✗
		EZVIZ Mini	CS-CV206	$40	710	✗	✓
		Funlux	CH-S1R-WA-Q3	$23	2706	✗	✓
		Logitech	961-000392	$90	407	✗	✗
		Amazon	1080p Full HD	$120	3532	✗	✗
		Nest	MAIN-99991	$166	6866	✗	✗
		Wyze	WYZEC2	$26	1481	✗	✓
	Smart Video Doorbell	Ding	AF-KSH001W	$60	1855	✗	✓
		AKASO	IPC010-US-NEW	$70	114	✗	✗
		Ring	720p HD	$100	27047	✗	✗
		SkyBell	SH02300SL	$148	1297	✗	✗

3.1.2 Our Approach: SecWIR

We thus develop a framework designated as **SecWIR** (**S**ecure **W**i-Fi **I**oT communication **R**outer) to provide smart home IoT devices with secure IoT communications through commercial off-the-shelf (COTS) home routers. The development is based on two key rationales. First, most smart home IoT devices allow users to access them remotely through Internet via smartphone applications, and all of the IoT commands destined to, or the responses sent from, the smart home IoT devices pass through the user's home router. Consequently, the home router represents an ideal choke point from which to monitor and analyze all of the incoming and outgoing IoT traffic, and protect the associated IoT devices without modifying them. Second, by leveraging the embedded computing resources of the home router, the proposed mechanism does not require the users to purchase any additional security hardware, and hence the deployment cost is reduced; thereby improving the likely take-up of the proposed framework.

3.1.3 Challenges and Solutions

To expand the IoT market and roll out various IoT applications, inexpensive IoT devices have become the mainstream. However, it is challenging to secure these low-cost IoT devices with limited resources through the proposed SecWIR framework. The design of SecWIR needs to address two key challenges. (1) COTS Wi-Fi routers are heterogeneous, and not all of them have sufficient resources to support IoT security functions. For high-end routers, deploying mainstream security protocols on top of the router seems technically straightforward. However, not all the resources of the routers can be used for IoT security functions since they usually need to support rich data services (e.g., DLNA media server, iTunes server, and ReadyCloud server [93]) which may consume many resources. For resource-constrained low-end routers, the problem is far more challenging. For example, the Linksys WRT400N router has 32 MB RAM for both its operating system and other local applications. After it is booted up, only 7 MB RAM is left for secure IoT communications. Furthermore, a user may have multiple IoT devices and it is technically difficult to secure a great number of devices and prevent user-perceived IoT access delays, due to resource constraints. (2) COTS Wi-Fi routers are designed to perform the efficient routing of data packets, not to implement and support security functions for IoT devices. Thus, largely preserving

the original performance of the non-IoT data services while also securing IoT devices is far from trivial.

To address these challenges, we propose four novel mechanisms for making more efficient use of the routers' resources to accomplish secure IoT communications. (1) *IoT-specific SSL/TLS tunneling* saves resources by only allowing NonSecIoT devices to use the secure IoT communications while preventing the access from other devices. The reason why SecWIR adopts the SSL/TLS tunneling mechanism instead of the IPSec tunneling is that the latter consumes more resources [48, 35]. (2) *Priority-based SSL/TLS tunneling management with user-perceived IoT access augmentation* saves resources by using a suite of novel mechanisms including priority-based tunneling management, traffic-outlier detection, and cross-application detection. It provides secure IoT communications for a great number of IoT devices with only a limited number of active SSL/TLS tunnels, while largely preserving user experience of accessing IoT devices. (3) *Stream security validation* secures InSecIoT devices with flawed security protocol implementations using a lightweight stream processing approach. (4) *Resource monitoring* monitors the real-time resources (e.g., CPU and RAM) of the Wi-Fi routers and the user packet routing performance (e.g., the packet drop rate and throughput), and then dynamically assigns/frees resources to/from SecWIR as required.

3.2 Related Work

There are two existing solutions to address the detected two vulnerabilities.

• **Device-based Approaches.** The device-based approaches can be found in [110, 94, 72, 57, 76]. The device-based approaches require the after-market IoT devices to possess the capabilities required to deploy the proposed security mechanisms. Thus, they may not be suitable for all IoT device platforms; particularly those with resource constraints.

• **Infrastructure-based Approaches.** The infrastructure-based approaches include IoT security gateways [44, 120], customized secure systems [88], in-hue secure manager [105], and customized securebox [62]. These approaches typically require the users to purchase additional security hardware components, the cost of which may deter the users, to secure the IoT devices; hence, they also have only a limited practical applicability. Moreover, some IoT security gateways (e.g., Samsung

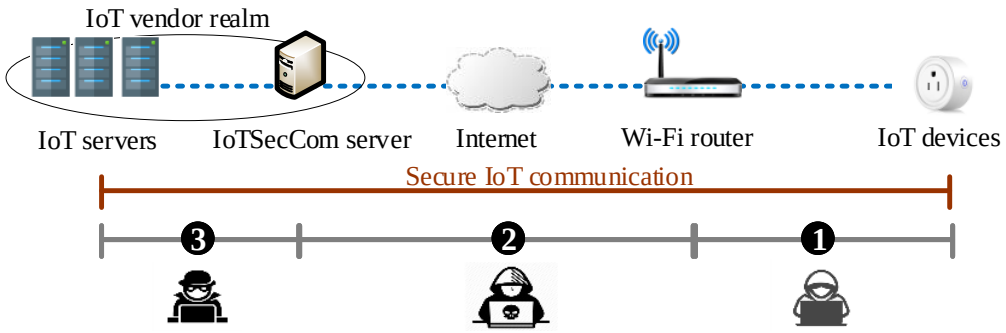


Figure 3.1: Schematic illustration of secure IoT communications provided by SecWIR and three possible attack sources.

SmartHome and Philips Hue) only support certain IoT devices.

3.3 Threat Model, Assumptions, and Security Guarantees

Threat Model: In this study, adversaries are people or organizations which launch remote attacks against victims (e.g., by taking control of their IoT devices). They are assumed to have the following capabilities. (1) They can *intercept*, *modify* or *inject* any messages in the public communication channels. Specifically, the secure IoT communication provided by SecWIR can be divided into three paths, namely the wireless path between the IoT device and the Wi-Fi router, the wired path between the Wi-Fi router and the entrance point of the IoT vendor realm, and that between the entrance point and the serving IoT server (see Figure 3.1). Adversaries may launch attacks from either of these three paths. (2) The adversaries adhere to all cryptographic assumptions, e.g., an encrypted message cannot be decrypted without its decryption key.

Assumptions: SecWIR makes three basic assumptions, namely (1) IoT device vendors are willing to secure their shipped IoT devices for the sake of goodwill if the proposed remedies do not affect the existing IoT services or seriously degrade their profits; (2) the IoT vendor realms in which the IoT servers are deployed are secure; and (3) the owners of the IoT devices have at least one operational Wi-Fi home router which supports Wi-Fi Protected Access II/III (WPA2/3 [29]).

Security Guarantees: SecWIR aims to provide two security guarantees: (1) **secrecy** of all the IoT traffic exchanged between the IoT devices and the IoT vendors (i.e., the IoT packets always have ciphering protection); and (2) **integrity** of the IoT traffic such that even if the packets are intercepted, adversaries cannot use them to generate fabricated packets.

3.4 Empirical Security Study

In this section, we conducted a security study on the security protocol support provided by 40 popular Wi-Fi smart home IoT devices which we purchased on the Amazon. The investigation focused on the SSL/TLS and IPSec protocols due to their widespread deployment. We observed two security vulnerabilities on the tested IoT devices, namely (V1) a lack of security protocol support and (V2) flawed certificate validation. The experimental results are summarized in Table 3.1.

(V1) A lack of security protocol support: Not all the Wi-Fi IoT devices support those studied security protocols. ***Validation:*** we used the `tcpdump` program on Wi-Fi routers to intercept all the IoT packets transmitted between IoT devices and IoT servers. Our study shows that 23 of the 40 devices do not support the studied cryptographic/security protocols (e.g., Etekcity, Belkin, Geekbes, and Tp-Link); that is, the packets are sent in plain-text.

(V2) Flawed Certificate Validation: Some Wi-Fi IoT devices have flaws in validating the IoT server's X.509 certificate [111] which is received during the establishment of an SSL/TLS connection with the server. Such flawed certificate validation can make the IoT devices suffer from various SSL/TLS MITM attacks. ***Validation:*** We first generated self-signed server certificates using the OpenSSL library and then provided the tested IoT devices with the fake server certificates via the SSLSplit [102] tool, which divides an SSL/TLS connection into two sub-connections and allows adversaries to conduct MITM attacks. Our study shows that 6 IoT devices mistakenly accept the forged server certificates.

Security Threats: By exploiting these two vulnerabilities, the adversary can launch a variety of IoT attacks. For example, the adversary can remotely control a victim's IoT devices. Figure 3.2 illustrates that we could generate IoT control messages to freely turn on/off the Etekcity smart socket, which does not support any security protocols. The adversary can also capture real-time images/videos from a victim's camera. Figure 3.3 demonstrates that by providing a security camera with a forged server certificate, we could discover the encryption key of the streaming video and then obtain the real-time images/videos of the victim's home from the camera.

```
00 04 00 01 00 06 84 16  f9 19 9c ba 00 00 08 00   ........ ........
45 00 00 7c 04 f9 00 00  7f 11 8d 43 c0 a8 0a 6b   E..|.... ...C...k
c0 a8 1d 79 10 01 27 11  00 68 49 13 50 4f 53 54   ...y..'. .hI.POST
3a 4e 33 37 36 30 26 4d  41 43 3d 35 43 43 46 37   :N3760&M AC=5CCF7
46 41 44 42 43 46 44 26  49 44 3d 41 44 42 43 46   FADBCFD& ID=ADBCF
44 26 50 57 44 3d 31 32  33 34 26 4c 41 4e 3d 30   D&PWD=12 34&LAN=0
30 26 4d 4f 44 45 4c 49  44 3d 53 4a 41 2d 30 35   0&MODELI D=SJA-05
26 56 45 52 3d 30 30 30  31 26 61 6c 61 72 6d 3d   &VER=000 1&alarm=
30 3c 39 30 30 30 30 30  36 30 0d 0a               0<900000 60..
00 04 00 01 00 06 84 16  f9 19 9c ba 00 00 08 00   ........ ........
45 00 00 79 6a f9 00 00  34 06 ca ee 2d 4f 4b 9d   E..yj... 4...-OK.
c0 a8 17 03 43 79 a7 50  c3 d4 f7 1e 00 00 1d 1c   ....Cy.P ........
50 18 79 70 81 55 00 00  81 4f 7b 22 75 72 69 22   P.yp.U.. .O{"uri"
3a 22 5c 2f 72 65 6c 61  79 22 2c 22 61 63 74 69   :"\/rela y","acti
6f 6e 22 3a 22 62 72 65  61 6b 22 2c 22 63 69 64   on":"bre ak","cid
22 3a 22 30 63 64 36 63  63 31 61 2d 61 61 31 34   ":"0cd6c c1a-aa14
2d 34 33 61 30 2d 38 37  62 36 2d 30 33 62 36 64   -43a0-87 b6-03b6d
34 39 62 64 64 37 31 22  7d                        49bdd71" }
```

Figure 3.2: Hijacking attacks. Top: turn on a smart plug; bottom: turn off a smart plug.

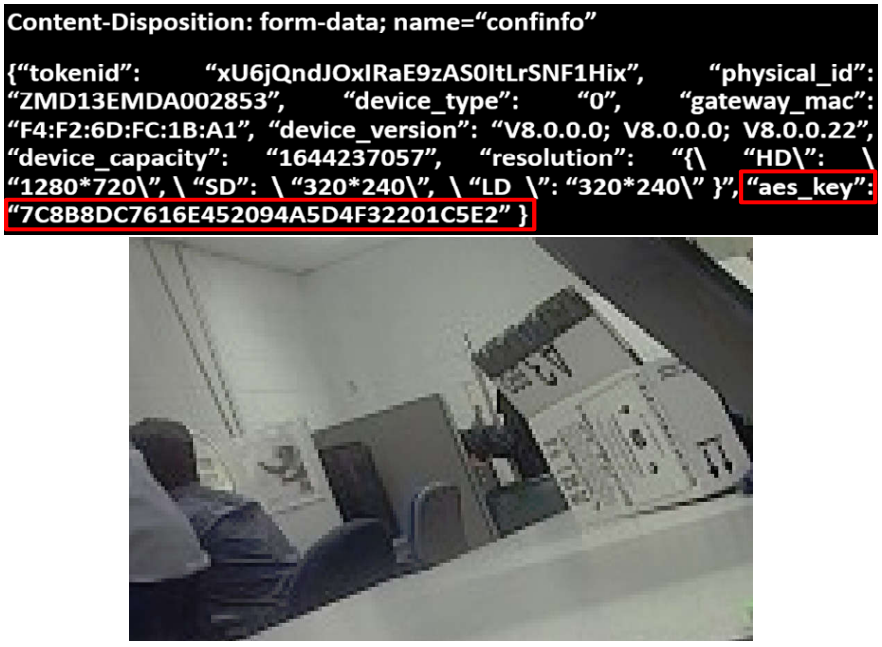


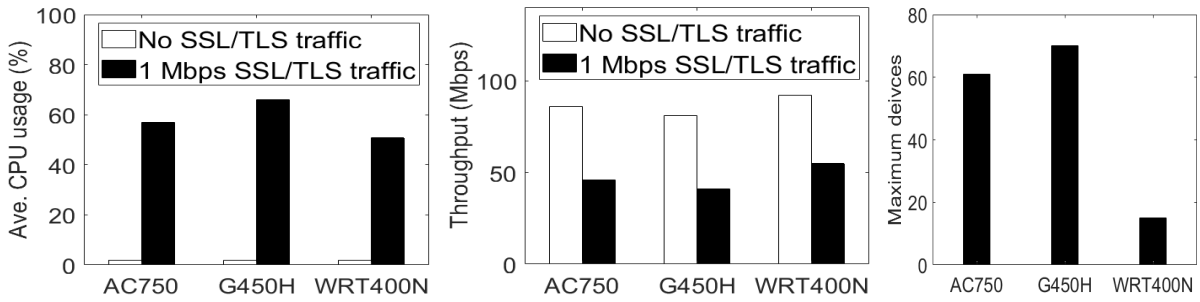
Figure 3.3: Spying attacks. Top: steal the AES encryption key of a camera; bottom: spy on victims.

3.5 SecWIR Design

3.5.1 IoT Secure Tunneling Module

To provide NonSecIoT devices with secure IoT communications, it is necessary to protect all of the packets transmitted between the IoT devices and the IoT servers. As described in §5.4, the secure IoT communication provided by SecWIR can be divided into three paths (see Figure 3.1). This section describes the tunneling module that protects the IoT traffic transferred from/to NonSecIoT devices in the second path (i.e., between the Wi-Fi router and the entrance point of the IoT vendor realm). Note that the first wireless transmission path and last wired path are both skipped here since it is assumed that the former is secured by the Wi-Fi security protocol (e.g., WPA2), and the latter is secure (see Section 3.3).

The aim of the secure tunneling module is to construct a secure SSL/TLS tunnel between the Wi-Fi router and the IoT vendor realm. SecWIR deliberately adopts an SSL/TLS tunnel rather than an IPSec tunnel since the latter protocol consumes more resources [48, 35]. The tunnel is built between two components, *IoTSecComClient* and *IoTSecComServer*, located at the two ends of the tunnel. IoTSecComClient is a SecWIR-specific security module built on the Wi-Fi router, while the IoTSecComServer is a standalone server deployed on the IoT vendor side.



(a) CPU usage (b) Throughput (c) Max IoT devices

Figure 3.4: Performance evaluation of conventional SSL/TLS tunnels on three routers: Buffalo G450H (400 MHz CPU/64 MB RAM), Tp-Link AC750 (580 MHz CPU/64 MB RAM), and Linksys WRT400N (680 MHz CPU/32 MB RAM).

Beyond Conventional SSL/TLS Tunneling: Seemingly, the proposed tunneling solution is similar to the conventional SSL/TLS tunneling technique. However, it presents three unique technical challenges. First, the delivery of SSL/TLS packets consumes more resources (e.g., CPU usage)

than normal IP packets at the router. Figures 3.4a and 3.4b show that a 1 Mbps SSL/TLS data flow can consume more than 50% of the CPU usage and downgrade the throughput performance of non-IoT data services (e.g., FTP and DLNA) by up to 49% (from 81 Mbps to 41 Mbps); this may lead to a new security threat. For example, insider adversaries may launch DoS attacks by sending large amounts of junk data to the secure tunnel, thereby exhausting the resources of the Wi-Fi router. To defend such types of attacks, SecWIR should therefore prevent non-IoT devices from sending data to the tunnel. Second, in the IoT era, users may have multiple IoT devices, and using a separate SSL/TLS tunnel to protect each one of them may overload the Wi-Fi router and affect the performance of non-IoT devices. Our experimental result shows that an SSL/TLS connection consumes about 440 KB RAM using the OpenSSL library v1.1.1e, which is not affordable for some resource-constrained routers to support a great number of IoT devices. For example, Linksys WRT400N can only support 15 IoT devices while no other services/applications are running on the router, as shown in Figure 3.4c. Third, the routers have only limited resources, and hence as the volume of non-IoT traffic increases, the resources available for SSL/TLS tunneling decrease and it becomes difficult to maintain the same level of user experience (e.g., the access time) for the IoT devices.

Therefore, to ensure the successful implementation of SecWIR, an efficient SSL/TLS tunneling approach is required. We thus propose two mechanisms, namely IoT-specific SSL/TLS tunneling (which addresses the first technical challenge) and priority-based SSL/TLS tunneling management (which focuses on the second and third challenges). These two mechanisms shall be applied together to addressing all the challenges.

IoT-specific SSL/TLS Tunneling. Access to the SSL/TLS tunnel is controlled based on the IoT device MAC address, as provided through a device registration procedure implemented using the management application of the Wi-Fi router. Many vendors (e.g., Netgear, Linksys, and TP-Link) provide such management applications [75] for users to control and monitor their Wi-Fi routers via their smartphones. In the SecWIR framework, whenever a new device associates with the Wi-Fi router, the SecWIR management application is enabled to confirm whether or not the new device is an IoT device by sending a notification message to the user. Only registered IoT devices are then permitted to access the SSL/TLS tunnel. Furthermore, to thwart MAC address spoofing attacks in which adversaries attempt to deceive the router by mimicking authorized devices, the remedy,

such as [41], is additionally employed.

Priority-based SSL/TLS Tunneling Management. SecWIR needs to prevent the SSL/TLS tunneling module from downgrading the performance of the non-IoT devices associated with the router. Specifically, when the demand increases and the performance of the non-IoT devices deteriorates, tunneling resource is released to restore their performance. As the number of IoT devices increases, the spare resources at the router may become insufficient to establish all the required SSL/TLS tunnels. The IoT devices may thus experience long user-perceived access delays, which cannot be tolerated for certain IoT operations such as turning on a smart bulb.

A priority-based SSL/TLS tunneling management solution is thus proposed to establish, suspend, and tear down the SSL/TLS tunnels dynamically based on their priorities. To save resources, it is assumed that a tunnel can be used by multiple IoT devices belonging to the same vendor (e.g., all TP-Link IoT devices can communicate with the IoTSecComServer deployed in the TP-Link through the same SSL/TLS tunnel). This design is motivated by an observation that IoT vendors provide customers with different IoT control applications, e.g., TP-Link uses Kasa Smart, whereas Etekcity uses VeSync. It is not rare that users purchase IoT devices from only a few IoT vendors; otherwise, many IoT control applications need to be used. To preserve a similar IoT user experience when the router resources are sparse, the priorities of the IoT devices and tunnels are determined based on two factors: the IoT usage patterns and the waiting time. In particular, devices with a more frequent traffic pattern are assigned a higher priority and hence the corresponding tunnel is canceled or suspended with a lower probability. As the time for which the IoT traffic waits to access the tunnel increases, the priority of the corresponding tunnel increases.

The priority-based tunneling management solution comprises three components: (1) a priority-based SSL/TLS tunneling management algorithm; (2) an IoT device priority function; and (3) an SSL/TLS tunnel priority function. The first algorithm manages the SSL/TLS tunnels based on the properties of the IoT devices and SSL/TLS tunnels, while the latter functions assign priorities to the IoT devices and SSL/TLS tunnels, respectively. We next elaborate on each of them.

(1) Priority-based Tunneling Management Algorithm: In SecWIR, the SSL/TLS tunnel state can be `Active`, `Inactive` or `Waiting` (see Figure 3.5). In the `Active` state, the SSL/TLS tunnel has been established and can be used immediately for the transfer of IoT packets. By contrast, in the `Inactive` state, the tunnel was established previously, but has now been suspended

Figure 3.5: State transitions of an SSL/TLS tunnel.

Algorithm 3.1: SSL/TLS tunneling management

Initialization:

Randomly select some IoT devices to establish SSL/TLS tunnels until the active list reaches its capacity. The remaining unserved IoT devices are moved into the waiting list. Set slot counter k=1. For each SSL/TLS tunnel, set an active state timer $\tau = 0$ to record its `active` state duration. Set a minimum active time threshold τ_{min}.

while (1) **do**

Step 1: For each SSL/TLS tunnel, compute and update its priority value $p_c(k)$ according to Equation (3.3), as well as update timer τ.

Step 2: Find the set S of SSL/TLS tunnels whose active time $\tau > \tau_{min}$;

Step 3: **if** $S = \varnothing$ **then**

k++; continue;

Step 4: Find the SSL/TLS tunnel (represented as c_1) in set S with minimum priority; among the SSL/TLS tunnels with `inactive` or `waiting` state, find the SSL/TLS tunnel (represented as c_2) with maximum priority.

Step 5: **if** $p_{c_1}(k) \geq p_{c_2}(k)$ **then**

k++; continue;

Step 6: **if** $p_{c_1}(k) < p_{c_2}(k)$ && c_2 *is* `waiting` **then**

Step 7: Set c_1 to be `inactive` and move c_1 into the inactive list; set c_2 to be `active` and move c_2 into the active list, set c_2's active state timer $\tau = 0$;

Step 8: **if** $p_{c_1}(k) < p_{c_2}(k)$ && c_2 *is* `inactive` **then**

Step 9: Swap the states and list types of c_1 and c_2, set c_2's active state timer $\tau = 0$;

Step 10: If the inactive list reaches its maximum capacity, the first-in IoT SSL/TLS tunnel is moved from the inactive list to the waiting list.

Step 11: Sleep until the next time slot; k++;

and cannot be used for packet transfer until an SSL/TLS connection resumption process has been performed to restore the tunnel. Note that SecWIR employs the ticket-based SSL/TLS session resumption mechanism specified in RFC5077 [104]. Finally, in the `Waiting` state, the tunnel has not yet been established, but some IoT devices have requested it to be set up for the transfer of their packets. The operational details of the tunneling management process based on these state transitions are shown in Algorithm 3.1.

In Step 1, the Wi-Fi router updates the priority and active time information for each SSL/TLS tunnel at the beginning. We set a minimum active time τ_{min} (e.g., $\tau_{min} = 1s$) to ensure that once an SSL/TLS tunnel turns to be `active`, it at least stays in the `active` state for τ_{min} time. In Steps 2-4, the router collects current priority and active time information to determine whether the state of each SSL/TLS tunnel needs to be changed or not. In Steps 5-9, the router handles two cases for the state update processes. In Step 10, the router handles the situation that the inactive list reaches its maximum capacity. In Step 11, the algorithm sleeps until the next time slot and runs again from the beginning.

(2) IoT device priority function: The secure tunneling module maintains a priority value for each IoT device, where the value is updated every T seconds (T is configurable and set to 0.1 s in this study). It implies that the states of SSL/TLS channels are updated every T s; a larger T leads to lower CPU usage but longer IoT device access time). Let $p(k)$ be the priority function at the kth time slot. The function takes two factors into account, namely $p_u(k)$ and $p_w(k)$, where $p_u(k)$ is the usage frequency function, and $p_w(k)$ is the waiting time function. The priority function $p(k)$ for each IoT device is defined as

$$p(k) = x \times p_w(k) + y \times p_u(k), \tag{3.1}$$

where x and y are adjustable weight coefficients and $x + y = 1$ (they are set to 0.02 and 0.98, respectively, in this study). To reduce the additional IoT access delay, we give a higher weight to $p_u(k)$. The waiting time function, $p_w(k)$, is set to 0 whenever the IoT device has an active SSL/TLS tunnel assigned for its use; otherwise, $p_w(k) = p_w(k-1) + 1$. Meanwhile, the usage frequency function $p_u(t)$ is given by

$$p_u(k) = \begin{cases} 0, & k = 1 \\ \alpha p_u(k-1) + (1-\alpha)c_u(k), & k > 1 \end{cases} \tag{3.2}$$

where $c_u(k)$ is increased by one if any usage of the IoT device is detected during the kth time slot; otherwise, $c_u(k) = 0$. Note that $p_u(t)$ is updated using the exponential moving average (EMA) method [65]. SecWIR detects the IoT device usage to determine $c_u(k)$ using the following two approaches.

- *Traffic-outlier Detection.* The detection process is based on the frequency with which traffic peaks (outliers) are observed in the requested IoT traffic due to the execution of IoT control operations. In our study on the NonSecIoT devices, it is observed that while IoT devices are not being used, they periodically exchange small keep-alive messages (e.g., less than 170 bytes) with the IoT servers. However, while IoT devices are triggered to be used, the IoT servers transmit relatively large IoT command messages (e.g., more than 245 bytes) to the IoT devices. In SecWIR, the outlier detection process is performed using the algorithm proposed in [90].

- *Detection for Manual IoT Device Access.* If a user tries to access an IoT device once from the IoT vendor's control application on his/her smartphone during the kth time slot, the usage, $c_u(k)$, of all IoT devices belonging to the IoT vendor is increased by one. In practice, several techniques are available for performing this detection. For example, on Android phones, the `ActivityManager` module [31] can be used to retrieve the name of the foreground application. While an IoT device access is detected by the traffic-outlier detection, and the foreground application is an IoT control application, a potential manual IoT device access is detected.

(3) *SSL/TLS tunnel priority function:* The SSL/TLS tunnel priority is determined in accordance with the highest priority among all the IoT devices using the tunnel. In other words, the priority of a tunnel c in the kth time slot, is given by

$$p_c(k) = \max\{p_1(k), \ldots, p_n(k)\}, \tag{3.3}$$

where $p_1(k), \ldots, p_n(k)$ respectively represent priority values of n distinct IoT devices which share the same SSL/TLS tunnel c.

3.5.2 Stream Security Validation Module

This module aims to secure the communications of InSecIoT devices by examining whether the procedures used by the IoT device to establish secure channels are compliant with security protocol standards. For each non-compliant establishment procedure, the module may terminate the connection and notify the IoT device owner. The module addresses three common security issues: (1) insecure cryptographic cipher suites, (2) server certificate expiration, and (3) fake server certificates.

Implementing the stream security validation module is challenging for two reasons. First, current validation tools for checking security compliance do not support stream processing (i.e., packet-by-packet examination), but only batch processing. For example, SSLdump [106] outputs the server certificate for validation purposes only after an SSL/TLS connection has been established. Thus, there is no guarantee that insecure channels have not been established and used to send IoT packets to rogue servers. Second, the tools are not optimized for resource saving under resource constraints. For example, they may attempt to verify the same IoT server certificate multiple times within a short time period, thereby wasting the resources of Wi-Fi routers.

To address these problems, SecWIR incorporates two components within the validation module: (1) security standard validation and (2) hash-aided validation.

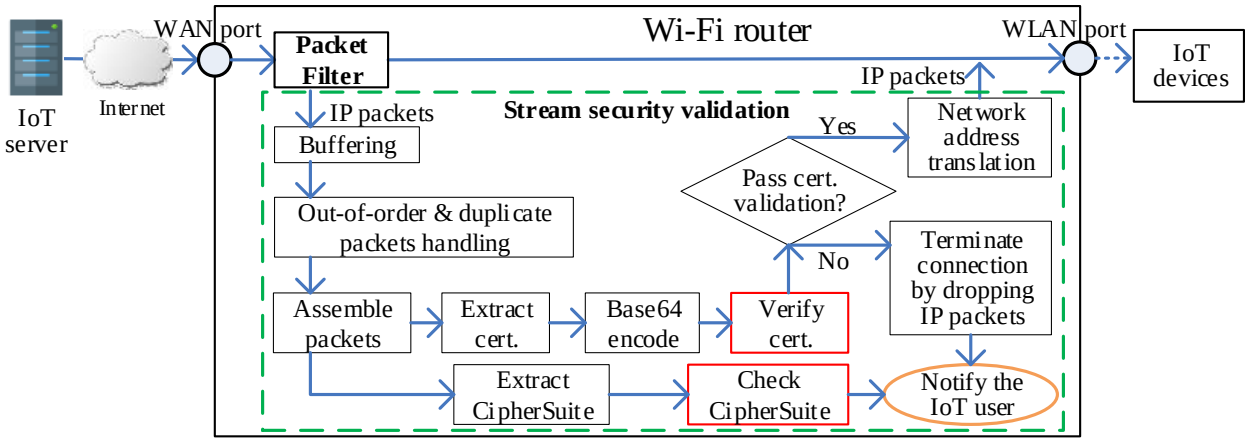


Figure 3.6: Security standard validation flow.

Security Standard Validation: As shown in Figure 3.6, a packet filter is used to dispatch all the packets containing SSL/TLS handshake messages to the validation module. The module extracts cryptographic information from the messages (e.g., ServerHello) and then examines whether an

insecure cipher suite has been used by the IoT servers. The cipher suite is a set of algorithms that secure network communications; it usually contains key exchange, encryption/decryption, and message authentication code algorithms. Several algorithms have been reported as being insecure, including RC4, MD5, DES-CBC, and IDEA-CBC. When the cipher suite selected by the IoT servers contains insecure algorithms, the validation module can send a `warning` message to the device owner through the Wi-Fi router management application (e.g., [75]). Then, the module validates the device's server certificate in terms of its expiration date and the validity of the issuer.

Specifically, the validation process at this module consists of seven tasks: (1) buffering only the related packets; (2) dealing with packet retransmission and out-of-order delivery issues; (3) assembling messages related to a cipher suite; (4) extracting the server certificate from the cipher-related messages; (5) checking if any insecure algorithms are used in the messages (and sending a `warning` message to the device owner if necessary); (6) encoding the server certificate into a Base64 format, which is X.509-compatible, if a server certificate exists; (7) verifying the server certificate. If the certificate is determined to be valid, the module permits the tunnel establishment and then routes the IoT packets through this tunnel. In practice, the server certificate validation process may fail for two reasons, namely the certificate is expired; or the certificate is generated by a non-trusted CA. In both cases, the secure tunnel establishment process is terminated. Although the validation module currently considers only three common security issues, it can easily be extended to consider additional security issues if required.

Hash-aided Validation: In the empirical IoT study described in §5.3, it was found that some of the server certificates were verified many times. In some cases, this was due to the fact that the devices manufactured by the same IoT vendor connected to the same IoT server. In other cases, it was caused by some of the devices having only short-lived SSL/TLS connections, which were terminated as soon as the device request was served and reconstructed whenever a new request was received. The need to verify the server certificates repeatedly not only consumes the resources of the Wi-Fi router, but also delays the IoT access response time.

A hash-aided validation method was further designed to reduce the occurrence of repeated validation operations. In particular, whenever a server certificate is successfully verified, the validation module caches its hash value and this value is then referenced in any future validation process to check whether or not the certificate has been verified previously. Notably, the expiration time of

the cached validation result is configurable in the proposed module.

3.5.3 Resource Monitoring

Wi-Fi routers are designed to efficiently route user packets rather than perform security functions for IoT devices. Thus, SecWIR should not affect the performance of non-IoT devices while securing IoT devices. Accordingly, the resource monitoring module aims to strike a balance between the non-IoT device performance and IoT security. In particular, it monitors the uplink and downlink packet drop status of non-IoT devices, and then dynamically allocates resources to SecWIR by adapting the number of maximum active SSL/TLS tunnels and the maximum data rate of each SSL/TLS tunnel. For example, when the Wi-Fi router starts to drop packets of non-IoT devices, the module gradually reduces the number of maximum active SSL/TLS tunnels.

3.6 Security Analysis

In this section, we examine how SecWIR fulfills the required security guarantees and explain how SecWIR defends against possible attack scenarios over the connections between the home Wi-Fi router and multiple IoT devices (C1), and between the IoTSecCom server deployed in the IoT vendor realm and a home Wi-Fi router of the IoT device owner (C2). Note, as described previously in §5.4, it is assumed that attacks do not occur within the IoT vendor realm.

3.6.1 Security Guarantees

SecWIR supports two security guarantees: **secrecy** and **integrity** for an IoT communication over the connections: C1 and C2.

C1: SecWIR supports the secrecy and integrity of the wireless communications between the IoT devices and the Wi-Fi router by enabling existing Wi-Fi security protocols (e.g., WPA2/WPA3 [25]). In particular, AES_128 and CCMP (CTR mode with CBC-MAC Protocol) are adopted by SecWIR for the secrecy and integrity protection, respectively.

C2: SecWIR supports the secrecy and integrity of the wired communications between the Wi-Fi router and IoT vendor realm using the SSL/TLS security protocols. In particular, SecWIR adopts a secure cipher suite consisting of ECDHE, ECDSA, AES_128, and CBC_SHA256, which con-

tains a key exchange algorithm (ECDHE, Elliptic Curve Diffie-Hellman Ephemeral), a signature algorithm (ECDSA, Elliptic Curve Digital Signature Algorithm), a ciphering algorithm (AES128), and a message authentication code algorithm (SHA256). The first three algorithms guarantee the secrecy of communications, while the last algorithm guarantees their integrity.

3.6.2 Possible Attacks

In examining the robustness of SecWIR against malicious attacks, it is assumed that the attacks can be launched from either inside the home network (C1) or outside the home network (C2).

Inside Home Wi-Fi Attacks:

- *IoT Compromise/DoS/Side-Channel Attacks:* The IoT devices may suffer various forms of internal attacks, including compromising attacks (e.g, Mirai attacks [58]), DoS attacks (e.g., PING flooding), and side-channel attacks (e.g., inferring the IoT device usage by analyzing the intercepted IoT packets). All of these attacks rely on the adversary being able to access the IoT user's home Wi-Fi network. The empirical study of 40 common IoT devices described in §5.3 revealed two important observations: (1) all commands for IoT devices are sent by external IoT servers; and (2) all notifications/alerts/messages produced by IoT devices are sent to the external IoT servers. In other words, the IoT devices do not need to communicate with other internal hosts. Thus, to guard against internal attacks, SecWIR implements a security policy by which the IoT devices are prevented from communicating with any internal hosts other than SecWIR.

- *IoT Masquerading Attacks:* Adversaries may compromise non-IoT devices inside a victim's home Wi-Fi network and masquerade these devices as authentic IoT devices using a MAC address spoofing technique. They can send large quantities of spam data to the IoT server, thereby exhausting its resources. However, SecWIR can easily detect such attacks and report them to the IoT users since each device associated with the Wi-Fi router is assigned a unique security key (i.e., PTK, pairwise transient key) by the WPA2/WPA3 protocol. Furthermore, SecWIR maintains the registration of the MAC addresses of all the legitimate IoT devices. Thus, if multiple devices use the same MAC address as a registered IoT device but employ a different security key, a notification message is immediately sent to the IoT user.

- *Malicious/Compromised IoT Attacks:* Since SecWIR allows multiple IoT devices to share the same SSL/TLS tunnel, a compromised device can thus gain the access to the shared channel and may overwhelm the tunnel to hurt the performance of the other IoT devices. However, such attack damages are limited due to the following two reasons. First, as described previously, SecWIR can protect IoT devices from being remotely compromised. Second, even when an IoT device is compromised by non-cyber approaches (e.g., physical compromise), the attack can be mitigated by employing a per-device rate limit mechanism.

- *Denial-of-IoT-Service (DoIS) Attacks:* Adversaries may compromise the victim's non-IoT devices and then use these devices to generate huge volumes of non-IoT data traffic to consume the resources of the home Wi-Fi router and launch a DoIS attack against the user's IoT devices. However, SecWIR readily defends such attacks due to two reasons. First, the IoT user can assign a small amount of *guaranteed* resources to SecWIR. Since all of the security modules within SecWIR are designed to work efficiently with only limited resources, such an approach can substantially mitigate the impact caused by the DoIS attacks. Second, most COTS Wi-Fi routers support fair bandwidth sharing among the associated devices and hence adversaries are unable to occupy all the resources of the Wi-Fi router using compromised devices.

Outside Home Wi-Fi Attacks:

- *SSL/TLS Protocol Attacks:* Recently, researchers have exploited the vulnerabilities of SSL/TLS to develop various MITM attacks, including BEAST [51], CRIME [84], TIME [40], RC4 BIASES [99], SSL Renegotiation [122], and downgrade attacks [116]. However, since these attacks rely mainly on insecure security algorithms and problematic implementations, they can be easily thwarted by SecWIR. For example, BEAST, CRIME, TIME, RC4 BIASES, and SSL Renegotiation attacks can be addressed by enabling AES256, disabling TLS compression, enabling Encrypt-then-MAC authenticated encryption, disabling RC4, and using TLSv1.2, respectively.

- *Side-channel Attacks:* Researchers have demonstrated that adversaries can infer users' IoT usage by analyzing the encrypted IoT data [36]. However, SecWIR largely protects IoT

users from such attacks by the means of the tunneling mechanism. Since multiple IoT devices share a TLS tunnel with the IoTSecCom server, it is difficult for adversaries to infer a particular IoT device usage due to the natural noise (IoT data) produced by the other IoT devices. Moreover, additional noises can be introduced by both IoTSecCom server and the SecWIR router to defend side-channel attacks.

3.7 SecWIR Evaluation

This section describes the implementation and the evaluation of the prototype SecWIR framework.

3.7.1 Implementation

The SecWIR framework was written in Linux C and was implemented on top of OpenWrt/LEDE-powered Wi-Fi routers. OpenWrt/LEDE [28], a very popular operating system for Wi-Fi routers, has supported 235 Wi-Fi router vendors and 1362 models in its current release [97]. We upgraded all tested Wi-Fi routers to the latest stable OpenWrt/LEDE releases at the time of the paper submission (e.g., Linksys WRT400N uses v17.01.5, whereas Tp-Link AC1750 uses v19.07.2). The ***IoT Secure Tunneling*** module used the OpenSSL library [96] to establish, close, and restore SSL/TLS tunnels. In addition, the SSL/TLS session ticket followed the ASN.1 [38] representation standards. In the ***Stream Security Validation*** module, seven routines were developed to check the validity of the IoT server certificates and ensure they were compliant with the relevant security protocol standards (see Figure 3.6). In the event of validation failures, the related IP packets were dropped by configuring the Wi-Fi router's IP table [26]. The ***Resource Monitoring*** module used the Linux utilities to obtain real-time resource usage of the Wi-Fi router and packet delivery status. Specifically, the `Top` command [27] was used to retrieve CPU/RAM usage, while the `netstat` command was used to acquire TX/RX-DRP (the number of packets dropped) at specific WAN/WLAN interfaces.

3.7.2 Evaluation

The performance of the SecWIR framework was evaluated using the experimental setup shown in Figure 3.7 based on a Linksys WRT400N Wi-Fi router with a 680 MHz CPU, 32 MB RAM, and 8 MB flash memory. The experiments were commenced by evaluating the effectiveness and per-

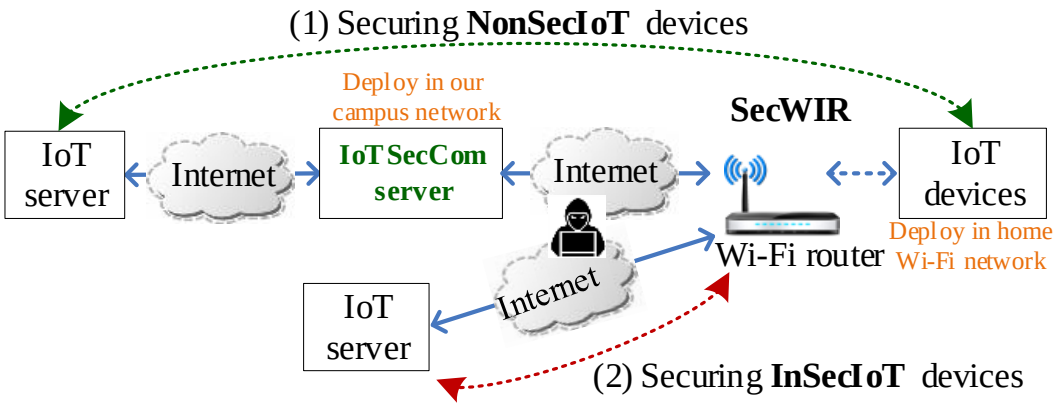


Figure 3.7: Experimental testbed.

formance of the two SecWIR security modules: the IoT secure tunneling module and the stream security validation module. The performance overhead incurred by SecWIR was then evaluated for four additional routers in terms of the access delay, the throughput, the RAM usage and the CPU usage. In performing the experiments, the IoT devices were deployed in a tested home Wi-Fi network. To provide the NonSecIoT devices with secure IoT communications with their IoT servers, an IoTSecCom client was installed on the home Wi-Fi router, and an IoTSecCom server was installed on a campus network. An SSL/TLS tunnel was then established between the IoTSecCom client and the IoTSecCom server. For the InSecIoT devices, SecWIR checked whether the associated secure channel establishment process was compliant with the security protocol standards and notified the device owner if necessary.

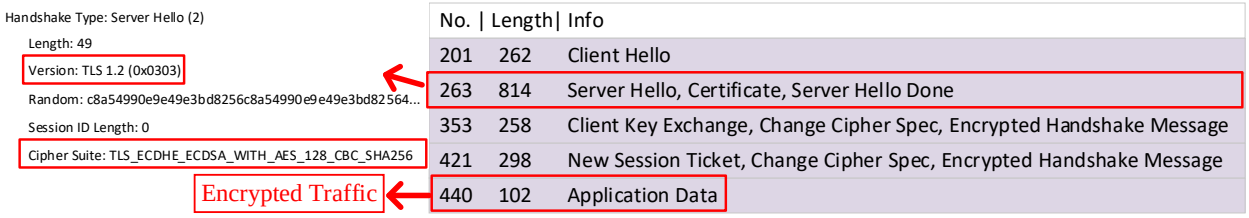
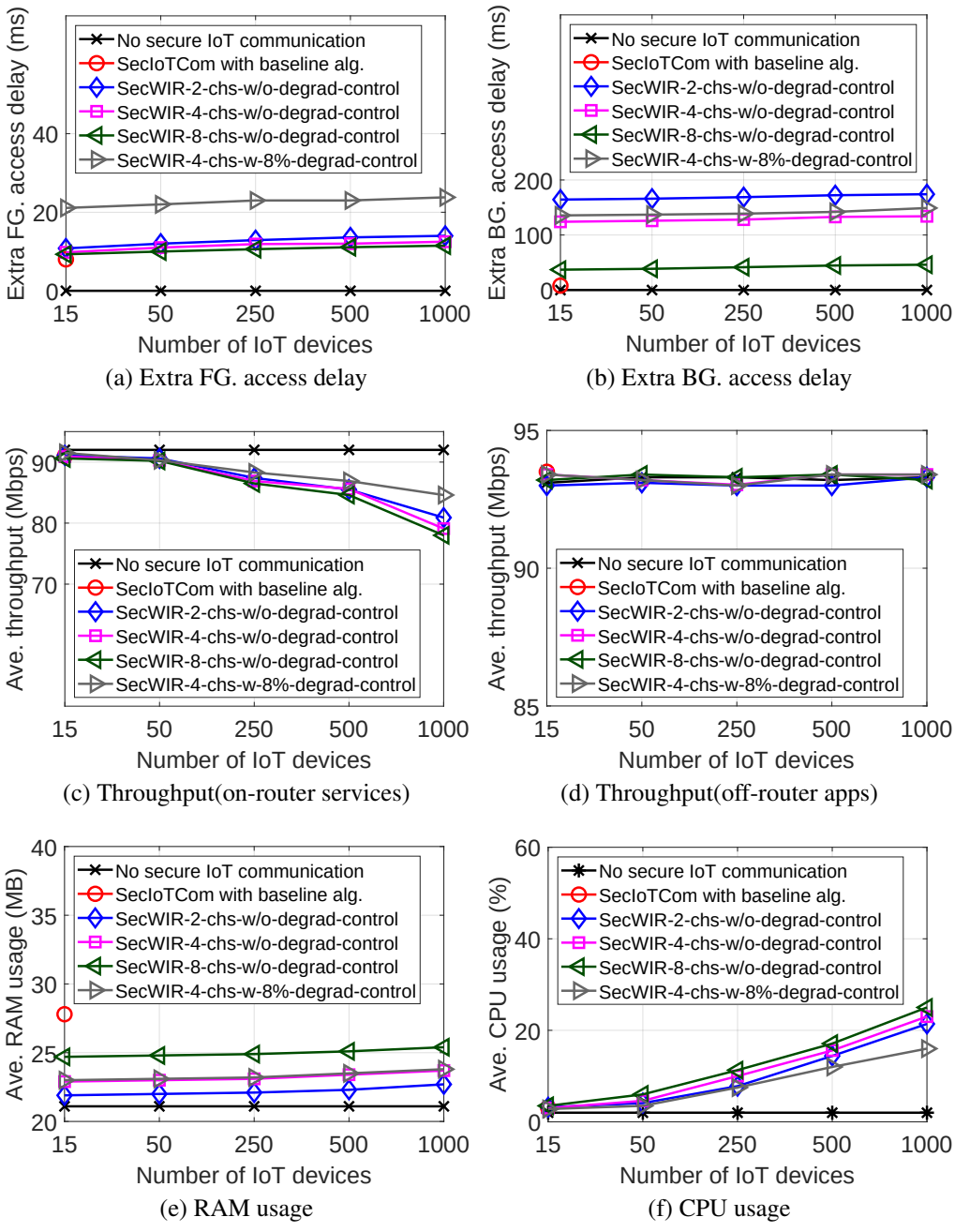


Figure 3.8: A TLS secure tunnel is successfully established.

SecWIR Security Function Evaluation.

1) ***IoT Secure Tunneling Module:*** We conducted an experiment to verify if the NonSecIoT devices can communicate with the IoT servers without any issues through the proposed secure IoT tunnels. There are 8 NonSecIoT test devices spanning four categories: socket (Geekbes and Etekcity), bulb (IView and TP-Link), humidifier (Essential and ASAKUKI), and strip (KMC and Teckin). Fig-



(a) Extra FG. access delay

(b) Extra BG. access delay

(c) Throughput(on-router services)

(d) Throughput(off-router apps)

(e) RAM usage

(f) CPU usage

Figure 3.9: Evaluation of secure IoT communication on Linksys WRT400N Wi-Fi router.

ure 3.8 shows that a TLS secure tunnel was successfully established between the IoTSecCom server and IoTSecCom client, based on a CipherSuite consisting of ECDHE, ECDSA, WITH_AES_128, and CBC_SHA256. Note that the elliptic curve-based algorithms are deliberately chosen since they provide strong security even with limited key lengths, and hence are suitable for resource-constrained routers. We further launched the IoT hijacking attacks (as shown in Figure 3.2) against these IoT devices by sending them fake IoT commands. It was observed that all the fake IoT commands were discarded by SecWIR, and all the IoT devices could be accessed normally through the secure IoT tunnels.

2) ***Stream Security Validation Module:*** The effectiveness of the stream security validation module in verifying the security of the channel establishment procedure was evaluated by deploying a server as an adversary between the tested InSecIoT devices and their IoT servers in order to intercept and modify the SSL/TLS messages and launch MITM attacks. In this experiment, we launched the spying attacks (as shown in Figure 3.3) and the CipherSuite downgrade attacks against two InSecIoT devices (i.e., Zmodo and Wyze security cameras) by sending them fake server certificates or fake ServerHello messages with a downgraded CipherSuite via our SSLSplit server. The experimental results are described in the following.

i) Expired & Forged Certificate Detection: The server intercepted the certificate sent by the IoT server and replaced it with a fake one (e.g., an expired certificate or a certificate issued by a non-trusted CA). The fake certificate was then forwarded to the tested InSecIoT device. The experimental result showed that the validation module could detect insecure server certificates and prevented the corresponding SSL/TLS connections from being established.

ii) Insecure CipherSuite Detection: The server intercepted the ServerHello message sent by the IoT server and changed the CiperSuite selected by the IoT server to an insecure CiperSuite (e.g., using RC4 and MD5). A fake ServerHello message was then sent to the tested InSecIoT device. The experimental result showed that the validation module successfully detected the insecure CipherSuite usage and sent a warning message to the IoT user. Note that SecWIR does not explicitly forbid the use of an insecure CiperSuite but simply warn the user, since current IoT devices may have some restrictions which make them use those weak CiperSuites.

SecWIR Overhead Evaluation.

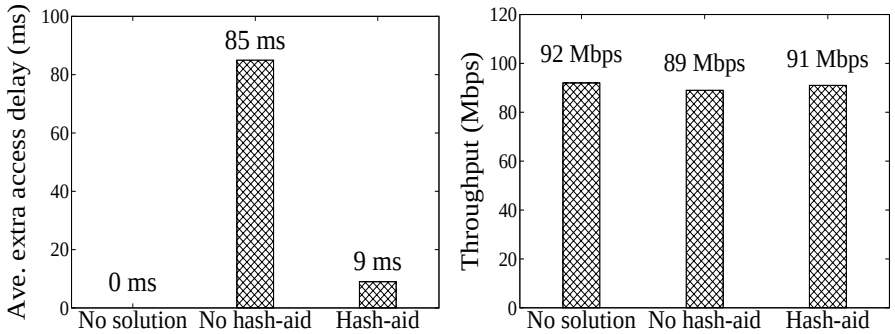
In evaluating the performance overhead of the SecWIR framework, four metrics were consid-

ered, namely the extra IoT device access delay incurred by SecWIR, the non-IoT traffic throughput, the RAM usage, and the CPU usage. Note that the extra IoT device access delay was defined as $t_1 - t_0$, where t_0 and t_1 represent the access delays (i.e., response time) of the IoT device before and after running SecWIR, respectively. The throughput was measured using Netperf [69] to emulate the non-IoT traffic generated by on-router services (e.g., router-side FTP, DLNA, and iTune services) and off-router applications (e.g., accessing Internet from the user's laptop).

1) *IoT Secure Tunneling Module:* We evaluated this module for three different scenarios: (1) no secure IoT communication, (2) support of SSL/TLS using a baseline tunneling management algorithm, in which each IoT device established a dedicated SSL/TLS connection with its IoT server; and (3) support of SSL/TLS using the proposed priority-based tunneling management algorithm. The experiments were performed using different numbers of IoT devices different numbers of active SSL/TLS tunnels, and different allowed degradation values of the maximum non-IoT traffic throughput.

i) Experiment Settings: We deployed four types of the NonSecIoT devices as described in Section 3.7.2. Commands were issued to the IoT devices with three different levels of usage frequency: (1) once per 1 min (25% of tested devices), (2) once per 10 mins (25% of tested devices), and (3) once per hour (the remaining tested devices). The baseline algorithm established and maintained as many SSL/TLS connections as the number of tested IoT devices when spare resources are sufficient. Note that, to evaluate the scalability of SecWIR, when the number of IoT devices is more than 16, an IoT device emulation server is deployed to emulate tested IoT devices based on their IoT traffic patterns, including both the foreground (FG) (e.g., IoT access commands) and background (BG) traffic (e.g., keep-alive messages).

ii) Experimental Results: The results are shown in Figure 3.9. We have four observations. First, the CPU and RAM usage volumes of the priority-based algorithm are increased with an increasing number of active secure channels. For example, 2- and 8-active-secure-channel methods increase the CPU usage by 2.1% and 4%, respectively, to secure 250 IoT devices. However, more active secure channels lead to shorter IoT device access delays. Second, the RAM usage of the priority-based management algorithm is much less than that of the baseline algorithm. For example, compared with the non-secure communication scenario, the priority-based algorithm with 4 active secure channels only increases the RAM usage by 2.7 MB to support 1,000 IoT devices,



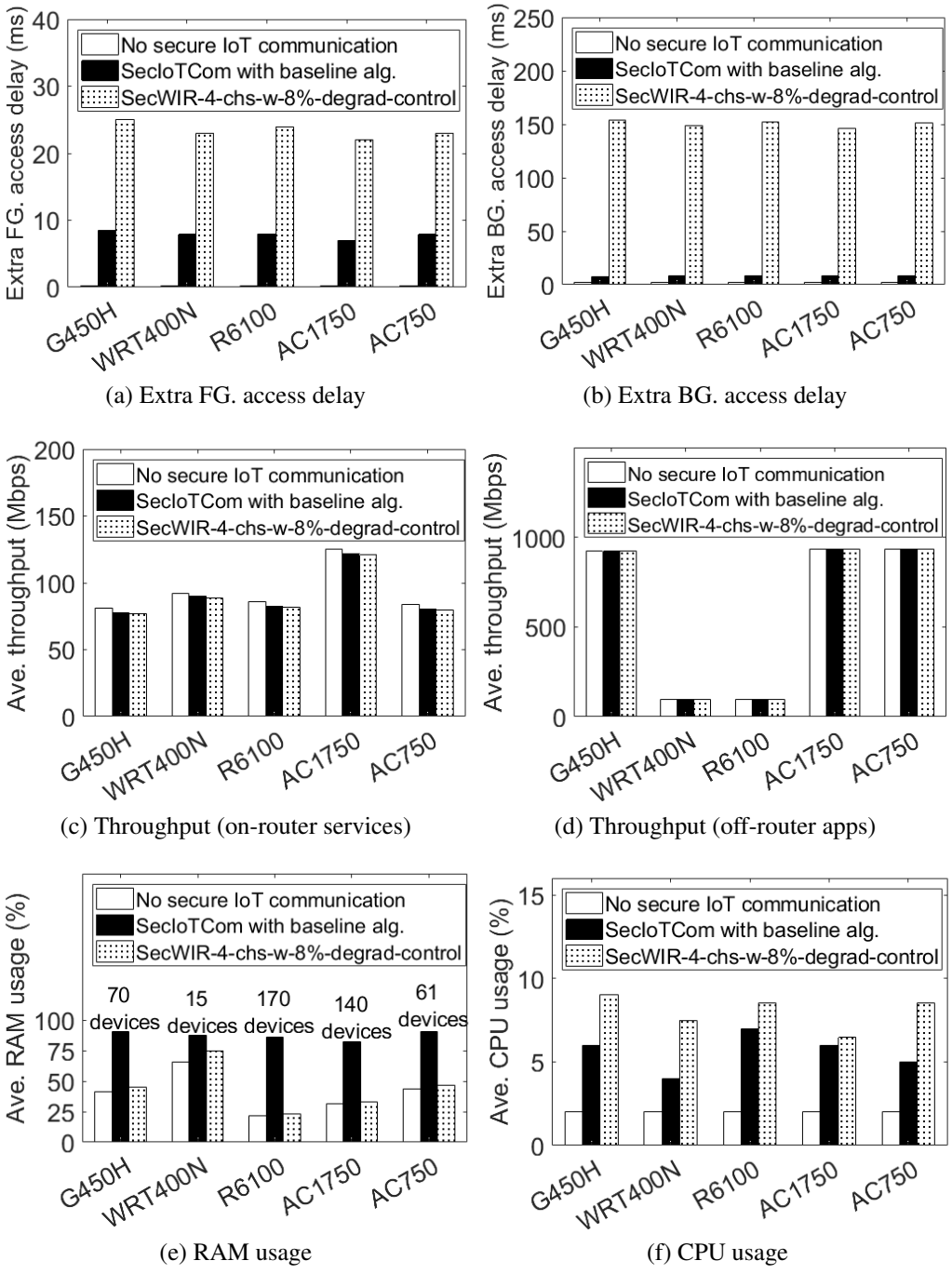
(a) Ave. extra access delay (b) Throughput

Figure 3.10: Evaluation of the stream security validation module on the Linksys WRT400N Wi-Fi router.

whereas the baseline algorithm almost runs out of all the available RAM (7 MB) to support 15 IoT devices. Third, compared with the absence of the secure IoT communication, the baseline and priority-based SSL/TLS tunneling management algorithms increase the IoT device access delays by 8 ms and 8$\sim$175 ms (extra foreground delays: 8$\sim$24 ms; extra background delays: 35$\sim$175 ms), respectively. Although the SecWIR's priority-based tunneling management algorithm results in a longer access time, in practice, the increased access time compared to the case in which no security is deployed is unlikely to be perceived by the user since it is so short. Our study shows that SecWIR offers the IoT device access delay that is comparable to commercial IoT security gateways (less than 1 second, see Table 3.2). Fourth, when the resource monitoring module is not activated (e.g., SecWIR-2/4/8-chs-w/o-degrad-control), the non-IoT data throughout degradation caused by the priority-based management algorithm increases with an increasing number of IoT devices (i.e., from 0.5% with 15 devices to 15.2% with 1,000 devices). After activating the monitoring module (e.g., SecWIR-4-chs-with-8%-degrad-control), the throughput downgrade can be reduced to 8% by slowing down the speed of processing IoT traffic at SecWIR based on the introduction of a 10 ms sleep time.

2) *Stream Security Validation Module:* We next evaluate the performance of the stream security validation module.

i) Experimental Settings: Two tested smart cameras (i.e., Zmodo and Wyze) were connected to the Wi-Fi router. It was observed that each of the cameras established an SSL/TLS connection with the IoT server when being powered up. Therefore, the performance of the stream security validation



(a) Extra FG. access delay

(b) Extra BG. access delay

(c) Throughput (on-router services)

(d) Throughput (off-router apps)

(e) RAM usage

(f) CPU usage

Figure 3.11: Evaluation of secure IoT communication on five Wi-Fi routers: WRT400N (CPU: 680MHz, 32MB RAM), G450H (CPU: 400MHz, 64MB RAM), AC750 (CPU: 580MHz, 64MB RAM), R6100 (CPU: 560MHz, 128MB RAM), and AC1750 (CPU: 720MHz, 128MB RAM).

module in examining the CiperSuite selected by the IoT servers and the IoT server certificates was evaluated by deliberately cycling the cameras on and off. In the experiment, a security standard examination request was generated every 6 seconds on average.

ii) Experimental Results: Figure 3.10 shows the performance evaluation results. It is seen that the security standard examination based on the proposed hash-aided certificate validation process results in a lower extra IoT device access delay than that without hash-based optimization (i.e., 9 ms vs. 85 ms). In addition, the hash-aided validation process has only a small effect on the throughput of the Wi-Fi router. Notably, compared to the benchmark scenario in which secure IoT communications were not deployed, the stream security validation module was found to increase the RAM and CPU usage by less than 1 MB and 3% CPU, respectively, irrespective of whether or not the hash-aided optimization was employed. The results confirm that the validation module does not impose any significant performance overhead when securing InSecIoT devices.

Performance Evaluation on Low-cost Wi-Fi Routers. The performance of SecWIR was further evaluated for the case in which the SSL/TLS tunneling module and stream security validation module were both enabled. In addition to the Linksys WRT400N Wi-Fi router, the experiments were also conducted on four other low-cost Wi-Fi routers, namely Buffalo WZR-HP-G450H ($35), Tp-Link AC750 ($30), Netgear R6100($50), and Tp-Link AC1750 ($50).

1) Experimental Settings: We conducted the same experiments as those previously evaluating the SSL/TLS tunneling module and stream security validation module while supporting 250 IoT devices. For the tunneling module, three mechanisms were evaluated: (1) no secure IoT communication, (2) support of SSL/TLS using a baseline tunneling management algorithm, in which each IoT device established a dedicated SSL/TLS connection with its IoT server, and (3) support of SSL/TLS using the proposed priority-based tunneling management algorithm using 4 active SSL/TLS tunnels and imposing the maximum 8% of non-IoT traffic throughput downgrade. For the validation module, a security standard examination request was generated every 6 seconds on average.

2) Experimental Results: Figure 3.11 shows the performance evaluation results. Among all the considered routers, the maximum foreground delay of the IoT device access is 25 ms, which is incurred by the Buffalo router. Furthermore, the Linksys router results in 8% of the maximum throughput degradation (92 Mbps $\rightarrow$ 84.6 Mbps). Finally, SecWIR increases the CPU and RAM

usage by 4.5%∼7% and 1.9 MB∼2.2 MB over all the five routers to secure 250 IoT devices, whereas the conventional SSL/TLS tunneling mechanism runs out of all the available RAM resources of the routers while supporting 80∼235 fewer IoT devices than SecWIR. Overall, the results show that SecWIR provides IoT users with secure IoT communications even on COTS low-cost Wi-Fi routers and causes no significant degradation on non-IoT data service performance.

Comparison with Samsung SmartThing Hub Samsung SmartThing system is a popular smart home IoT system. To use Samsung SmartThing IoT devices, the user needs to first purchase a SmartThing hub and connects the hub with his/her home router, and then connect Samsung SmartThing IoT devices with the hub. Finally, the user can access the IoT devices via the Samsung SmartThing application. Similar to SecWIR, the SmartThing hub plays the role of the IoT security gateway in this system. We thus compare SecWIR with the Samsung SmartThing hub. The comparison results are summarized in Table 3.2. We observe that both SecWIR and Samsung SmartThing provide comparable security functions (e.g., TLS v1.2 and AES encryption algorithms) and performance (e.g., accessing an IoT device only takes 0.8 s). However, SecWIR has two advantages over Samsung SmartThing. First, SecWIR supports IoT devices from various vendors, whereas the SmartThing hub is specific to Samsung IoT devices. Second, the cost of deploying a Wi-Fi-connected smart device is relatively inexpensive. A Samsung smart socket (GP-U9995JVLDAA) costs $35, whereas the Geekbes (YM-WS-5) smart socket takes only $9. Moreover, SecWIR does not require users to purchase additional IoT security gateways before securely using their smart home devices.

3.8 Discussions

Incentives for IoT vendors. Providing NonSecIoT devices with secure IoT communications with their IoT servers inevitably requires the support of IoT vendors. However, as described previously in §5.3, it is reasonable to expect that most IoT vendors will be willing to deploy security mechanisms in their infrastructures in order to secure their IoT devices if the proposed solutions do not affect their existing IoT services or degrade their profit. Thus, the present study has deliberately developed standalone IoTSecCom servers for deployment in the IoT vendor realm (see Figure 3.1). In practice, the IoT servers do not need to be aware of this IoTSecCom server. In addition, SecWIR

Table 3.2: Comparison between using Samsung SmartThing IoT system and SecWIR.

IoT security infrastructure		**SmartThing Hub**	**SecWIR**
Hardware capabilities	CPU	528 Mhz	680 Mhz
	RAM	256 MB	32 MB
	Flash	4 GB	8 MB
Secure IoT communication	Security protocol	TLS v1.2	TLS v1.2
	key exchange algorithm	ECDHE	ECDHE
	Signature algorithm	RSA	ECDSA
	Ciphering algorithm	AES_256	AES_128
	Integrity algorithm	GCM_SHA384	CBC_SHA256
Performance	Time of accessing IoT devices	740-800 ms	700-750 ms
	Time of establishing TLS connections	115-462 ms	150-350 ms
	Time of validating a server cert.	25-120 ms	72-98 ms
	Time of re-validating a server cert.	25-120 ms	2-5 ms
Security	Defend IoT hijacking attack?	Yes	Yes
	Defend spying attacks?	Yes	Yes
	Reject forged/expired server certs.?	Yes	Yes
Price		$70	free*

*: SecWIR relies on the IoT users' existing home router.
NOTE: The hardware capabilities and performance of SecWIR is based on Linksys WRT400N and Geekbes smart plugs.

yields a win-win situation for the IoT vendors and IoT users. In particular, the IoT vendors can preserve their profit when deploying secure IoT communications since they can continue to employ low-cost IoT device platforms for their smart home devices, while IoT users do not need to purchase expensive IoT devices to ensure secure IoT communications, but can continue instead to use cheaper IoT devices supporting SecWIR.

Deploying SecWIR. To support SecWIR, the home Wi-Fi routers must be upgraded. Most Wi-Fi router vendors provide users with a web/app-based interface to manually upgrade their routers. Since SecWIR is deployed on top of the popular OpenWrt/LEDE system, which has supported 235 Wi-Fi router vendors and 1362 models, it is a relatively simple task for most router vendors to adopt SecWIR and release an appropriate update. Regarding the other Wi-Fi routers, we believe that it is not difficult for router vendors to adopt SecWIR since it is a low-overhead, resource-efficient security framework. Note that the deployment of SecWIR requires IoT users to have some knowledge of using web services or smartphone apps to upgrade their routers, if an automatic router upgrade is not supported.

Customizing OpenWrt/LEDE. Customizing OpenWrt/LEDE operating systems of home routers can be one option to give more resources to secure IoT devices. However, the available resources can still be exhausted rapidly by the inefficient, conventional SSL/TLS tunnel mechanism, when the number of IoT devices increases. Moreover, with many built-in services currently deployed on the COTS Wi-Fi routers (e.g., iTunes server), we believe that the proposed light-weight IoT security framework is still desirable.

Applying SecWIR to securing other IoTs. Although this study aims to secure IoT communication of the Wi-Fi-connected IoT devices, the techniques adopted by SecWIR can be also applied to other IoT technologies (e.g., LoRaWAN [34]). For example, the proposed priority-based SSL/TLS tunneling management and stream security validation modules can be deployed on LoRaWAN gateways to secure the IoT communications between LoRAWAN gateways and LoRAWAN customer IoT servers [34].

Chapter 4

VSButton: Securing Home Digital Voice Assistants

4.1 Introduction

4.1.1 Background and Motivation

In recent years, more and more home digital voice assistant (HDVA) devices are deployed at home. Its number is forecasted to grow thirteen-fold from 2015 (1.1 million) to 2020 (15.1 million), a compound annual growth rate of 54.74% [113]. Thanks to the continuous efforts of the leading HDVA device manufacturers (e.g., Amazon and Google) and the third-party voice service developers (e.g., CapitalOne, Dominos, Honeywell), users can do a great number of things using voice commands. These commands include playing music, ordering pizzas, shopping online, scheduling an appointment, checking the weather, making a payment, controlling smart devices (e.g., garage doors, plug, thermostats), to name a few. To provide users with usage convenience, most of HDVA devices (e.g., Amazon Echo, Google Home) adopt an always-listening mechanism which takes voice commands all the time. Specifically, users are not required to press or hold a physical button on HDVA devices before speaking commands. This is the major difference between the HDVAs and phone assistants. The phone assistants are carried by users and only take voice commands after the phones are unlocked. Such great convenience may expose users to security threats due to the openness nature of voice channels. Therefore, we believe that the HDVA security should be specially considered. At this point, a natural question is: *Do these commercial off-the-shelf (COTS) HDVAs employ necessary security mechanisms to authenticate users and protect users from acoustic attacks?*

Unfortunately, our study on Amazon Alexa and Google Home yields a negative answer. We identify three security vulnerabilities from them. (1) The vulnerability V1 (weak single-factor authentication) indicates that for any person/machine who speaks the correct simple authentication

word(s) (e.g., “Alexa”, “Hi, Google”) ahead of a voice command, the command can be accepted by the HDVA devices. (2) The vulnerability V2 (no physical presence based access control) reveals that the device can accept voice commands even if no people are around it. It will accept the command if the command sound that reaches it is at the sound pressure level (SPL) 60 dB or higher. (3) The vulnerability V3 (insecure access control on Alexa-enabled device cloud) discovers that many vendors support the default names and commands. It is thus easy for the adversary to guess the voice command to control a smart device. Based on the discovered vulnerabilities, we devise two proof-of-concept attacks: home security breach attack and fabricated online shopping attack. In the former attack, an adversary can burglar the house of an Alexa device owner by requesting the Alexa to open a door via an Alexa-enabled smart lock. In the later attack, the adversary can command Alexa service to place a fake order on behalf of the HDVA device owner, and then the owner may suffer from the financial loss.

Due to the similarity of Amazon Alexa and Google Home, we mainly present the results of the former, which is more popular. All the findings can be applied to both of them unless explicitly specified.

4.1.2 Our Approach: VSButton

To provide user authentication and enhance the security of HDVA, we present Virtual Security Button (VSButton), a system to provide second-factor user authentication for HDVA users. VSButton works by detecting the nearby user motions via Wi-Fi signal variation. If VSButton detects user motions nearby, VSButton will activate HDVA. Otherwise, HDVA is inactive and will not take any voice command. The intuition behind our design is that most HDVAs require plug-in power supplies and they are deployed in relatively fixed locations inside a room. If the user can physically present near HDVA (e.g, within 1 meter) in the room, then the user should be treated as an authentic user. Therefore, VSButton can thwart adversaries who cannot enter the user’s room to activate the HDVA.

4.1.3 Challenges and Solutions

There are two technical challenges that VSButton should deal with.

First, VSButton detects user motions by monitoring Wi-Fi signal variation, it is challenging

to efficiently extract the Wi-Fi signal variations and eliminate the noises caused by environments (e.g., air flow). To address this challenge, VSButton leverages multiple modules (including PCA module, median and EMA filter module, and Butterworth filter module) to remove the Wi-Fi signal variations as well as reduce the Wi-Fi signal dimensions to accelerate the signal processing.

Second, it is challenging to design motions detection algorithm which can resist environmental changes (e.g., temperature variations, furniture relocation). To address the issue, VSButton employs a real-time outlier detection method to recognize outlier from the processed Wi-Fi signal. The algorithm will automatically shift the baseline (which represents there is no user motions), and therefore, it can dynamically adapt to the environmental changes.

4.1.4 Comparison with Prior Art

The comparison between VSButton and the previous solutions are summarized in Table 4.1. We have two key observations. First, VSButton is the first solution that does not depend on the Biometrics of the users. Second, VSButton is the only solution that does not need a wearable device, data training, and extra hardware purchase.

Table 4.1: Comparison between the previous solutions and VSButton.

Solutions	**Device-free Biometric-based**		**Device-aid Biometric-based**		**VSButton**
Papers	Kunz et al. [77]	Meng et al. [86]	Zhang et al. [121]	Feng et al. [56]	Our paper
Biometrics	Voice	Mouth motions	Mouth motions	Skin vibration	None
Need wearable devices?	No	Yes	Yes	Yes	No
Need data training?	Yes	No	No	No	No
Need extra hardware purchase?	No	Yes	Yes	Yes	No

We also note that motion detection can be finished by other approaches (e.g., using a camera [42] or a radar [67]). Compared with these approaches, VSButton does not require any extra hardware purchase. It leverages the Wi-Fi signals (emitted from COTS home Wi-Fi router) to finish the motion detection task.

4.2 Related Work

Several second-factor user authentication solutions have been proposed to address the problem. They can be roughly classified into two categories: (1) device-free biometric-based solution and

(2) wearable-aid biometric-based solution. They are elaborated below.

• **Device-free Biometric-based.** The device-free solution does not require users to carry any device to finish the user authentication. The authors in [77] proposed the approach to authenticate users by recognizing the users' voice. The voice authentication based solution has three major limitations. First, it requires the users to have voice training process before using HDVA devices, so the occasional home visitors (e.g., the users' friends) cannot command HDVA immediately. Second, users' voices may vary with their ages, illness, or tiredness. Third, the human voice is vulnerable to replay attacks.

Meng et al. [86] present WiVo, which uses Wi-Fi signals to recognize the mouth motions of the users and extract the unique features form both voice and Wi-Fi signals. Then, WiVo calculates the consistency between the two different types of signals to determine whether the voice commands are generated from the authentic users. This solution has two limitations. First, it requires users to purchase Wi-Fi signal antennas. Second, the high accuracy of mouth motions recognition is hard to be ensured as many factors may have an impact on the recognition accuracy (e.g., the location of the speaker, the Wi-Fi variation caused by motions from other people).

• **Wearable-aid Biometric-based.** The wearable-aid solution requires users to wear a special device to finish the user authentication. The approach in [121] requires the user to use the smartphone as a Doppler radar to detect the unique articulatory gesture of the user to help user authentication. Feng et al. [56] develops a proprietary wearable device which collects the skin vibration signals of users. The collected vibration signals are then continuously matched with the voice signals received by HDVA devices. The wearable-based solution has one major limitation. That is, it forces users to carry a device to finish the authentication, and therefore, it significantly decreases user convenience.

4.3 Virtual Security Button (VSButton)

We propose an access control mechanism which is based on physical presence to Alexa devices or other devices/services that require the detection of physical presence. It not only addresses V2, but also makes Alexa's authentication to be two-factor instead of current single factor (i.e., V1). We name this mechanism as virtual security button (VSButton), because whether physical presence is

detected is like whether a virtual button is pushed. The access to a device/service with VSButton is not allowed when the virtual button is not in a push state (i.e., physical presence is not detected). As a result, this mechanism enables Alexa devices to prevent fraudulent voice commands which are delivered when no persons are nearby them.

The mechanism detects physical presence based on the Wi-Fi technology. It results in negligible overhead on the Alexa device/service, because it reuses the user's existing home Wi-Fi network and needs negligible change on how the user requests Alexa services. It does the detection by monitoring the channel state information (CSI) of the channel used by the home Wi-Fi network. The CSI changes represent that some human motions happen nearby the Alexa device. Once any human motion is detected, the Alexa device is activated to accept voice commands. Therefore, the user just needs to make a motion (e.g., waving a hand for 0.2 meters) to activate the device before speaking his/her voice commands.

We believe that detecting human motions based on Wi-Fi signals is a practical yet low-cost solution approach due to two reasons. First, home Wi-Fi networks are commonly deployed, so no extra deployment cost is needed. Second, only a software upgrade is required for the Alexa devices, since all of them have been equipped with Wi-Fi. Before presenting our VSButton design, we introduce the CSI primer and the CSI-based human motion detection, which is based on the multi-path/multi-reflection effects.

4.3.1 CSI Primer

It is commonly used to characterize the channel state properties of Wi-Fi signals. Current Wi-Fi standards like IEEE 802.11n/ac rely on the Orthogonal Frequency Division Multiplexing (OFDM) technique, which divides a channel into multiple subcarriers, and uses the Multiple-Input Multiple-Output (MIMO) technology to boost speed. Each CSI value represents a subcarrier's channel quality (i.e., channel frequency response) for each input-output channel.

The mathematical definition of the CSI value is presented below. Let $\mathbf{x}_i$ be the N_T dimensional transmitted signal and $\mathbf{y}_i$ be the N_R dimensional received signal in subcarrier number i. For each subcarrier i, the CSI information $\mathbf{H}_i$ can be obtained based on the following equation:

$$\mathbf{y}_i = \mathbf{H}_i \mathbf{x}_i + \mathbf{n}_i, \tag{4.1}$$

where $\mathbf{n}_i$ represents noise vector. Without considering the noise, it can be known from Equation (4.1) that the matrix $\mathbf{H}_i$ transfers the input signal vector $\mathbf{x}_i$ to be output signal vector $\mathbf{y}_i$. It gives us the intuition about why the channel information is encoded in (and hence represented by) the matrix $\mathbf{H}_i$. Commodity Wi-Fi devices can record thousands of CSI values per second from numerous OFDM subcarriers, so even for subtle CSI variations caused by a motion, the CSI values can still provide descriptive information to us.

4.3.2 CSI-based Human Motion Detection

We detect whether there are any human motions and identify whether they happen inside a house/room by leveraging the multi-path and multi-reflection effects on CSI, respectively.

• **Multi-path Effect for Human Motions Detection.** The multi-path effect refers to the signal propagation phenomenon that a wireless signal reaches a receiving antenna along two or more paths. As shown in Figure 4.1, the receiver (RX) receives multiple copies from a common signal along multiple paths, the line-of-sight (LoS) path, one reflection from the human at location A, and one reflection from the furniture. Different lengths of the paths along which the Wi-Fi signals are sent result in phase changes of the signals, thereby leading to various CSI values. Therefore, human motions can change the paths of Wi-Fi signals and further make CSI values to change. For example, when the human moves from location A to location B, the new signal reflection path is being substituted for the original one. This move can thus distort the CSI values observed at the receiver.

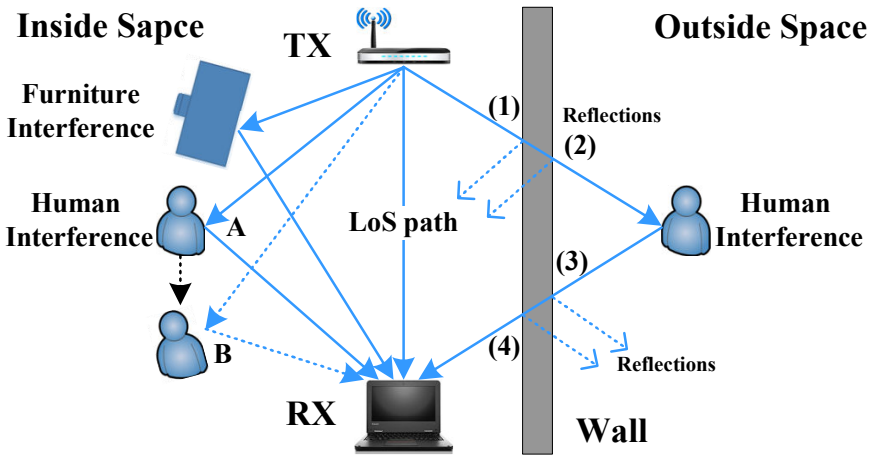


Figure 4.1: An illustration of multi-path and multi-reflection effects.

• **Multi-reflection Effect for Identifying Where the Motions are.** The multi-reflection effect means that a wireless signal may be reflected by multiple objects and it can cause the receiver to receive multiple copies of signals from different reflections. When we consider human motions inside and outside a room/house, they can be differentiated in terms of variation degrees of CSI values due to multiple different reflections. As shown in Figure 4.1, when a Wi-Fi signal is reflected by the human body outside the wall, the signal arriving at the receiver along this path should have experienced four reflections: (1), (2), (3), and (4). They happen since the signal traverses different media, from air to wall and from wall to air. Such multi-reflection effect causes the signal received by the receiver to suffer from a serious attenuation. Our results show that outside motions lead to only a small variation of CSI values, compared with a significant variation caused by inside human motions. The variation degrees of CSI values can thus be leveraged to identify the human motions occurring inside and outside the wall.

4.3.3 VSButton Design

In this section, we introduce our VSButton design as shown in Figure 4.2. It resides at the Alexa device and monitors human motions by its collected CSI values of the data packets received from the Wi-Fi access point (AP). Based on the CSI variations of the wireless channel between the device and the AP, VSButton can determine whether any human motion happens inside or not. Note that, in order to keep collecting CSI values over time, the Alexa device can send ICMP packets to the AP at a constant rate (e.g., 200 ICMP messages/second in our experiments) and then keep receiving the packets of ICMP reply messages from the AP.

The detection of inside human motions in the VSButton mainly consists of two phases, *CSI Processing Phase* and *Outlier Detection Phase*. When receiving CSI values, the former eliminates noises from them so that the CSI variation patterns caused by human motions can be augmented. Hence, VSButton can accurately differentiate indoor motions from the cases of no motions and outdoor motions. Based on the output of the first phase, the latter relies on a real-time outlier detection method to detect the CSI patterns of the human motions inside a room/house. The outlier detection method can work over non-stationary CSI data streams, so VSButton can be adaptive to environmental change (e.g., the CSI variation caused by temperature change or furniture relocation). In the following, we present the details of each phase and then give an example of identifying

indoor human motions.

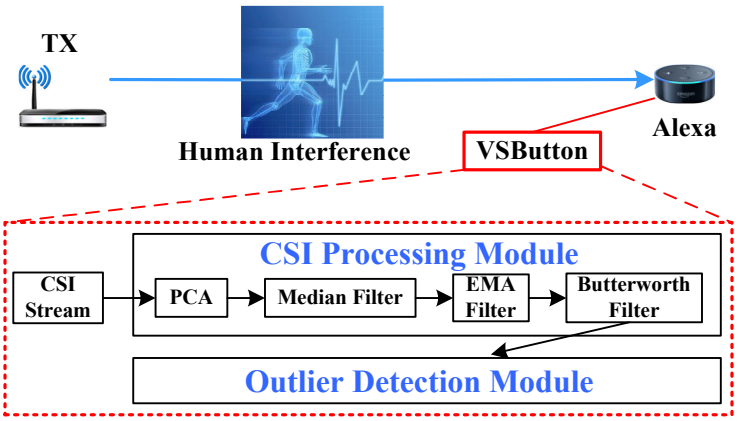


Figure 4.2: VSButton design.

4.3.4 CSI Processing Phase

This phase consists of three modules: principal component analysis (PCA) [68], median and exponential moving average (EMA) filter [37, 65], and Butterworth low-pass filter [46]. We first use the PCA module to reduce the dimensions of CSI values by removing those uncorrelated to motions detection. The last two modules are used to eliminate bursty noise and spikes, and filter out high-frequency noise in CSI stream values, respectively.

Figure 4.3 gives an example of the comparison between the original CSI over time and the CSI which has been processed by those three modules. It is shown that most of the noises in the original CSI are removed so that the processed one can give us more accurate information for motions detection. We elaborate the details of each module below.

• **PCA module.** We employ the PCA to remove uncorrelated noisy information from the collected CSI by recognizing the subcarriers which have strong correlations with motions detection. The PCA is usually used to choose the most representative principal components from all CSI time series. It can accelerate the subsequent signal processing because the collected CSI may contain too much noisy information. Take an Intel Link 5300 Wi-Fi adapter, which has N_T transmit antennas and N_R receive antennas, as an example. Each transmit-receive channel has 30 subcarriers, so there are $N_T \times N_R \times 30$ CSI streams to be generated.

Since the collected CSI information can be represented by a high dimensional matrix, the PCA

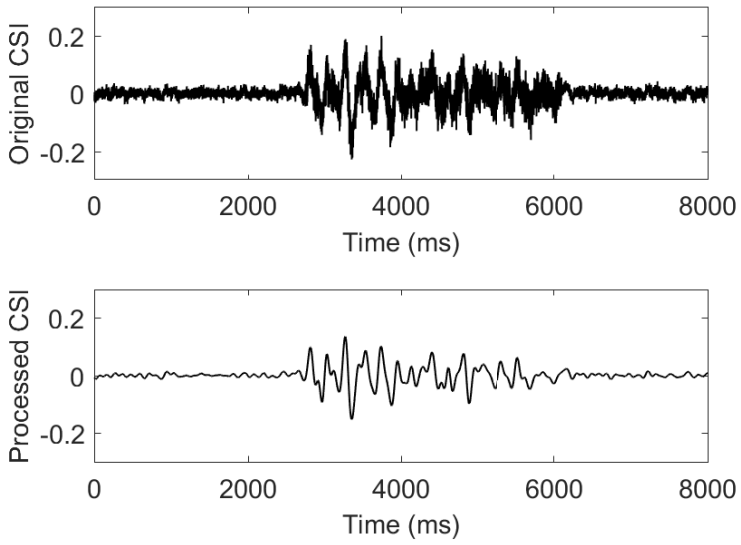


Figure 4.3: Comparison between original/processed CSI over time.

can be done by the following five components: 1) data centralization, 2) calculation of covariance matrix, 3) calculation of eigenvalues and eigenvectors, 4) selection of main eigenvalues, and 5) data reconstruction. The PCA details can be found in [30]. In our experiments, it is observed that the first four components almost show the most significant changes in CSI streams but the first one is too sensitive to signal noise. As a result, we keep only the second, the third, and the fourth components for further analysis.

• **Median and EMA Filter Module.** We next use a combination of a median filter and an EMA filter to eliminate bursty noise and spikes in CSI stream values. They happen because commodity Wi-Fi interface cards may have a slightly unstable transmission power level and also be affected by dynamic channel conditions (e.g., air flow, humidity, etc.). The median filter is often used to remove noise from the signal. Its main idea is to run through the signal entry by entry while replacing each entry with the median of neighboring entries. It can smooth out short-term fluctuations and highlight long-term trends. Note that the number of neighboring entries (i.e., moving window size) is a configurable parameter. We also adopt the EMA filter to smooth CSI values. It applies weighting factors which decrease exponentially to each older CSI value. The window size of EMA is a configurable parameter.

• **Butterworth Filter Module.** We finally apply the Butterworth low-pass filter to filter out high frequency CSI, since human motions cannot be generated too fast. Specifically, it is observed that

the CSI variations caused by human motions mainly happen in the low frequency domain (i.e., less than 100 Hz). Given a proper cut-off frequency, 100 Hz, high frequency noise can be removed by the filter.

4.3.5 Outlier Detection Phase

We detect human motions using a real-time hyper-ellipsoidal outlier detection method over non-stationary CSI data streams [90]. It improves accuracy over the typical moving average method, which detects an anomaly based on a threshold of the distance between current value and the average of old values, from two aspects. First, it employs a new distance metric, i.e., *Mahalanobis distance*, which is more accurate. Second, it exploits exponential moving average (EMA) to update the previous sample mean.

This detection method can be mathematically described as follows. Let $X_k = \{x_1, \cdots, x_k\}$ be the first k samples of CSI readings. Each sample is a $d \times 1$ vector, where d is the number of chosen components. The sample mean m_k and sample covariance S_k are given by

$$m_k = \frac{1}{k}\sum_{i=1}^{k} x_i, \;\; S_k = \frac{1}{k-1}\sum_{i=1}^{k}(x_i - m_k)(x_i - m_k)^T. \tag{4.2}$$

The Mahalanobis distance of a sample reading r from the X_k is defined as

$$D(r, X_k) = \sqrt{(r - m_k)^T S_k^{-1}(r - m_k)}. \tag{4.3}$$

By using Mahalanobis distance, we consider the reading r as an anomaly if $D(r, X_k) > t$, where t is a threshold parameter and needs to be carefully selected according to the experiments. All of the points bounded by $D(r, X_k) \leq t$ are considered as normal readings.

When we apply it to the non-stationary CSI streams, the sample mean is updated by

$$m_{k+1} = \alpha m_k + (1 - \alpha) x_{k+1}, \tag{4.4}$$

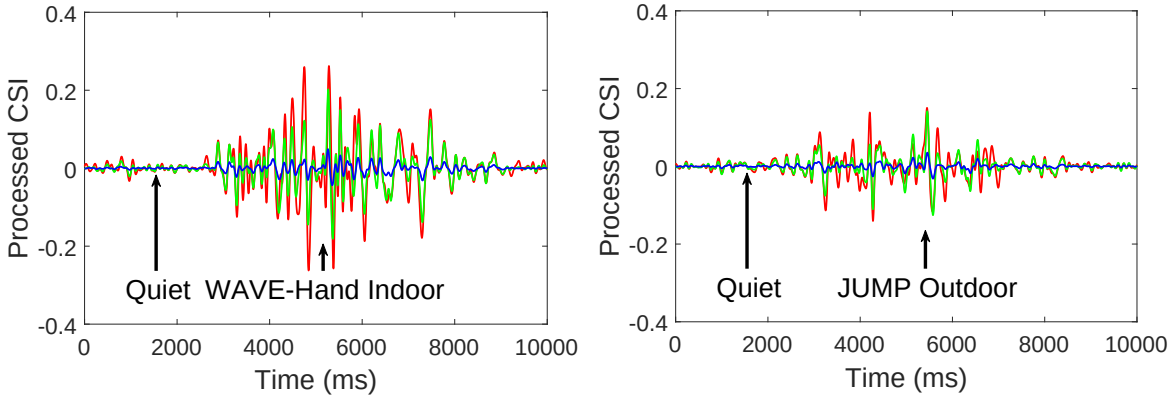
where $\alpha \in (0, 1)$ denotes a forgetting factor. The closer the receiving of a CSI value reading is, the larger weight it has to determine the next sample mean.

For the motions detection, we avoid false detections by specifying a threshold for the number of consecutive anomalous CSI values (10 is used in our experiments). It is because the noises may occasionally lead to some anomaly detections. However, human motions can make anomalous

CSI value readings to last for a long period of time (e.g., consecutive 10 readings). Note that the proposed motions detection method can work over non-stationary CSI data streams, so VSButton can be adaptive to environmental change.

4.3.6 An Example of Identifying Indoor Motions

We show that the CSI values of indoor and outdoor motions can be clearly differentiated so that the indoor motions can be properly detected. Figures 4.4a and 4.4b plot the processed CSI values respectively for a small indoor motion (i.e., waving a hand) in one laboratory room and a large outdoor motion (i.e., jumping). It is observed that the peak CSI values are 0.27 and 0.14, respectively. It shows that even the small indoor motion can lead to the CSI variations which are much larger than those of the strong outdoor motion. We believe that with proper parameter configurations, VSButton is able to correctly identify indoor motions and then activate an Alexa device to accept voice commands. More experimental results will be given in Section 4.4.



(a) Indoor motion CSI variation

(b) Outdoor motion CSI variation

Figure 4.4: Comparison between indoor and outdoor CSI variations.

4.4 Implementation and Evaluation

In this section, we present the implementation of our VSButton prototype and the evaluation results of both laboratory and real-world settings.

4.4.1 Prototype Implementation

Our VSButton prototype is based on commercial off-the-shelf (COTS) devices, as shown in Figure 4.5. The TX device is a Netgear R63000v2 Wi-Fi router, which can be considered as a home Wi-Fi AP. It is employed as the transmitter for the packets used by the Alexa device to collect CSI over time. The AP is set to the 802.11n mode [22], because the Alexa devices have not supported 802.11ac yet [23]. Since the Alexa devices are not open to development, we implement the motions detection module on a laptop, which is tagged to be RX. The Alexa device, Echo Dot, can then get motions detection result from the laptop. The RX device is a Lenovo X200 laptop equipped with an Intel Link 5300 Wi-Fi adapter, which is able to collect CSI values using the tool developed by the work [63]. To emulate the Alexa device's access control, the module controls MicroBot Push [24] to turn on/off the Alexa device's microphone through a smartphone, once any status change of access control is detected by the module based on motions detection.

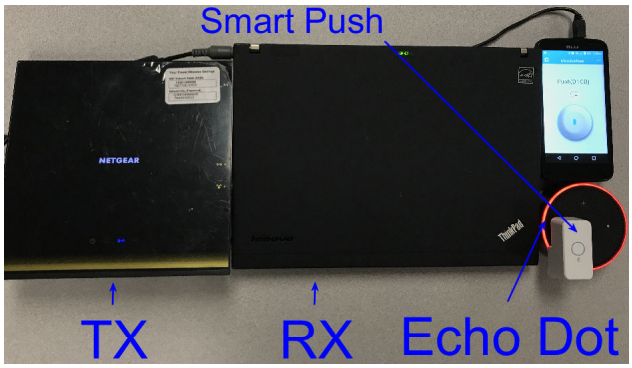


Figure 4.5: VSButton prototype.

Note that in our current VSButton prototype, there are a wireless router, a laptop, a smartphone, and a MicroBot. Seemingly, the deployment cost is not small. However, most of people have deployed a wireless router at home for Internet access and the others' functions can be integrated to the Alexa devices based on only software upgrades.

4.4.2 Evaluation

We next introduce our experimental settings and evaluate the performance of our VSButton prototype in three space settings: *square room*, *rectangle room*, and *two-bedroom apartment*. The performance refers to whether three cases, no motion, indoor motion, and outdoor motion, can

be correctly identified. We recruit six volunteers to participate in the experiments. They are required to do three motions including waving a hand (WAVE-HAND), sitting down and standing up (SIT-DOWN-STAND-UP), and jumping (JUMP, 0.5m), inside and outside a room. They represent three degrees of human motions, weak, medium, and strong, respectively. Note that we examine the Mahalanobis distance for each measurement and see whether indoor motions can be clearly detected or not.

• **Experimental Settings.** In all experiments, the RX (the laptop with an Echo dot) sends 200 ICMP `Echo Request` messages per second to the TX (the Wi-Fi router) so that it can keep collecting CSI over time from the ICMP `Echo Reply` messages sent by the TX. A CSI stream with the sampling rate of 200 values per second can thus be used for motions detection. Each ICMP message size is only 84 bytes, so network bandwidth is low.

The window sizes of the median and EMA filters are set to 9 and 15, respectively. Our experimental results show that these two numbers are large enough for the filters to remove noise. The cut-off frequency [46] for the Butterworth filter is set to $\omega_c = \frac{2\pi \times 100}{200} = 1\pi$ rad/s[1], because human motions lead to only low-frequency CSI variations, which are typically less than $f = 100$ Hz. In the outlier detection module, we set the forgetting factor α to be 0.98. It means that we give a larger weight to the recent CSI value readings. The experimental settings are summarized in Table 4.2.

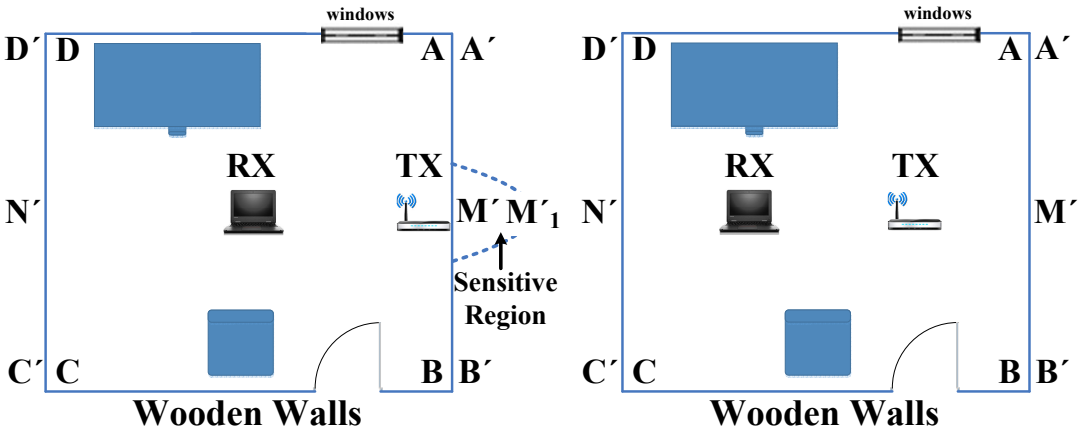
Table 4.2: Experiment settings

Parameters	**Values**
Sampling rate of CSI values	200
Median filter window size	9
EMA filter windows size	15
Cut-off frequency of Butterworth filtering (rad/s)	1π
Forgetting factor α	0.98

• **A Square Lab Room.** We deploy the VSButton in a wooden square room and evaluate it with two deployment configurations as shown in Figure 4.6a and Figure 4.6b. In the first configuration, the laptop with an Echo dot (RX) is placed at the center of the room and the Wi-Fi router (TX) is located at the edge. In the second configuration, the RX and the TX are placed between Locations

[1] rad/s is the unit of rotational speed (angular velocity)

N' and M' to divide the distance into three equidistant portions. In the experiments, the six participants do the aforementioned three motions (i.e., WAVE-HAND, SIT-DOWN-STAND-UP, and JUMP) at four indoor locations (A, B, C, and D) and six outdoor locations (A', B', C', D', M', and N').



(a) Square room: Configuration 1.

(b) Square room: Configuration 2.

Figure 4.6: Comparison of indoor and outdoor CSI variations.

(1) Configuration 1. The Mahalanobis distances we measured are summarized in Table 4.3. Note that each number of the indoor results is the minimum value of all the numbers measured among the participants, whereas that of the outdoor results is the maximum. This way can easily show whether indoor and outdoor motions can be clearly differentiated based on the Mahalanobis distances or not. We observe that all the indoor motions can be differentiated from no-motion cases and outdoor motions at all the locations except Location M'. Some indoor motions, such as the WAVE-HAND motion at Location D, have smaller distances than the motions, JUMP and SIT-DOWN-STAND-UP, at Location M'. The main reason is that that location is very close to where the Wi-Fi router is deployed. As a result, the router shall not be deployed at the location close to the wall next to outdoor space.

(2) Configuration 2. The results of this configuration are summarized in Table 4.4. We observe that the distance of each indoor motion is higher than the maximum distance (i.e., 0.241 from JUMP at Location M') of all the outdoor motions. It represents that VSButton can activate the Alexa service only due to indoor motions with this configuration.

• **A Rectangle Lab Room.** We now evaluate the prototype in a rectangle room with brick walls.

Table 4.3: Mahalanobis distance measured in a square room with Configuration 1.

Square Room	**Indoor Locations**				**Outdoor Locations**					
locations	A	B	C	D	A'	B'	C'	D'	M'	N'
WAVE-HAND	0.218	0.213	0.195	**0.191**	0.104	0.101	0.079	0.083	0.156	0.121
SIT-DOWN-STAND-UP	0.277	0.271	0.258	0.253	0.118	0.113	0.088	0.092	**0.238**	0.139
JUMP	0.392	0.391	0.371	0.366	0.132	0.128	0.099	0.103	**0.373**	0.165
DO NOTHING	0.026	0.021	0.027	0.024	0.023	0.027	0.028	0.023	0.020	0.023

Table 4.4: Mahalanobis Distance measured in a square room with Configuration 2.

Square Room	**Indoor locations**				**Outdoor locations**					
locations	A	B	C	D	A'	B'	C'	D'	M'	N'
WAVE-HAND	**0.312**	0.315	0.401	0.409	0.041	0.043	0.049	0.051	0.092	0.063
SIT-DOWN-STAND-UP	0.345	0.349	0.423	0.430	0.060	0.062	0.069	0.071	0.121	0.089
JUMP	0.401	0.407	0.451	0.459	0.069	0.071	0.084	0.086	**0.241**	0.099
DO NOTHING	0.025	0.021	0.022	0.024	0.028	0.026	0.021	0.022	0.023	0.025

The RX and the TX are placed between Locations N' and M' to divide the distance into three equidistant portions, as shown in Figure 4.7. Table 4.5 summarizes the measurement results. Note that the distance at each indoor location is the minimum value of all the numbers measured among the participants, whereas that at each outdoor location is the maximum.

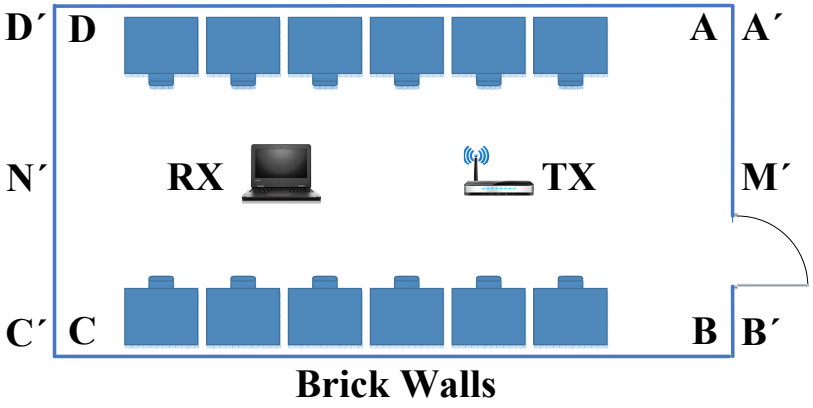


Figure 4.7: Rectangle room configuration.

There are two findings. First, the result is similar to that of the square room with the same configuration (i.e., Configuration 2). The minimum Mahalanobis distance (i.e., 0.147 from WAVE-HAND at Location A) among all indoor motions is higher than the maximum distance (i.e., 0.042 from JUMP at Location M') of all outdoor motions. Their difference is as large as 0.105. Second, that difference is higher than that (i.e., 0.071) observed in the square room with the same configuration. The main reason is that Wi-Fi signals are unable to penetrate the brick walls as easy as

Table 4.5: Mahalanobis distance measured in a rectangle room.

Rectangle Room	**Indoor locations**				**Outdoor locations**					
locations	A	B	C	D	A'	B'	C'	D'	M'	N'
WAVE-HAND	**0.147**	0.150	0.180	0.183	0.020	0.022	0.025	0.027	0.035	0.030
SIT-DOWN-STAND-UP	0.181	0.184	0.216	0.217	0.024	0.026	0.028	0.029	0.039	0.033
JUMP	0.254	0.255	0.287	0.288	0.029	0.029	0.032	0.033	**0.042**	0.035
DO NOTHING	0.022	0.021	0.022	0.027	0.028	0.026	0.021	0.022	0.020	0.025

wooden walls.

There are two lessons learned. First, the wall materials can influence the performance of VS-Button. The harder the wall materials are, the better performance the VSButton can get. Our experiment results show that VSButton can be applied to two kinds of wall materials, wood and brick. Second, users should not deploy VSButton at the location which is close to outdoor space.

• **An Apartment with Two-bedroom and One-bathroom.** We also evaluate VSButton in a $75m^2$ apartment with two bedrooms and one bathroom, as shown in Figure 4.8. We deploy the Wi-Fi router (TX) and the VSButton prototype (RX) at the center of the apartment and the living room, respectively. In the following, we first do parameter calibration to determine the threshold which differentiates indoor motions from the others, and then examine the result of a 100-minute experiment.

(1) Parameters Calibration. Before deploying the VSButton in a real-world scenario, we need to perform parameter calibration to determine a proper threshold t for the outlier motion detection module. It is because the threshold can change with different environments.

The calibration process includes two major steps. First, the Alexa owner chooses an indoor location to deploy his/her Alexa device and then determines which area is allowed to enable the device with human motions. At that location, s(he) does the smallest indoor motion (e.g., waving a hand) and collects its minimum Mahalanobis distance value. Second, the owner finds all the outdoor locations which are not allowed to enable the Alexa device. S(he) does the strongest outdoor motion (e.g., jumping) and collects its maximum Mahalanobis distance value. We then set the threshold t to be the half of the difference between the above two distance values. Note that the whole calibration process can be done within 5 minutes and only one-time calibration is needed for the initial deployment of the Alexa devices. Our solution does not require manual re-calibration, but only initial calibration. It is because all the parameters in Table 4.2 are fixed and optimized

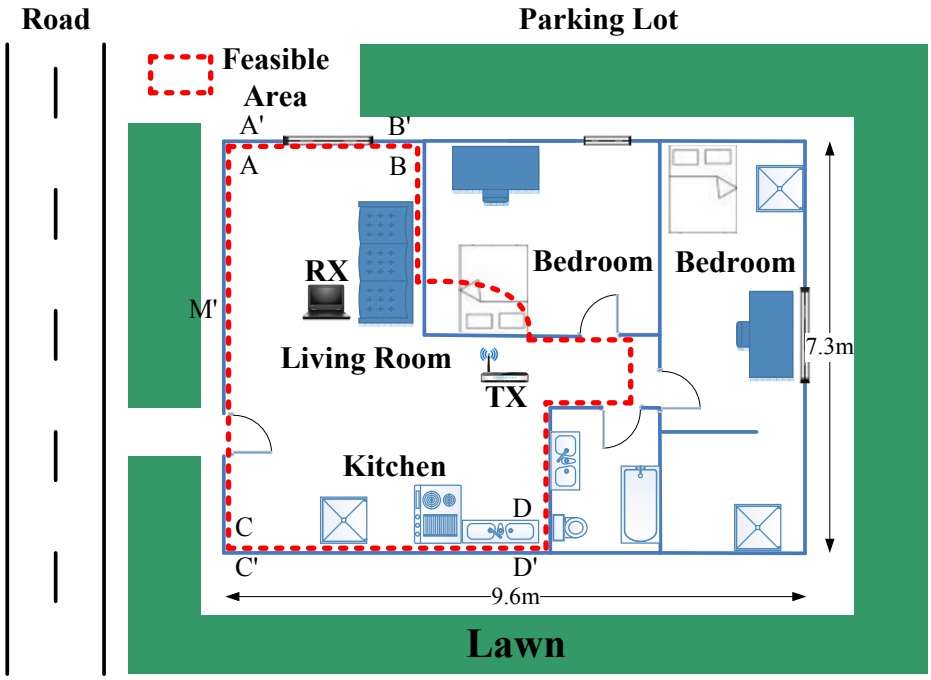


Figure 4.8: VSButton Deployment in an apartment with two bedrooms and one bathroom.

in our design, but only the threshold t and the CSI baselines need to be adapted to environmental changes. Specifically, the threshold t is affected only by wall materials, so it requires only initial calibration if the wall is not altered. The CSI baselines, which are used to indicate the conditions of no indoor motions, are automatically, dynamically adapted to different environments (e.g., a new object is deployed nearby).

The parameter calibration for the apartment is conducted as follows. We perform the WAVE-HAND motion at four indoor locations, A, B, C, and D, and the JUMP motion at five outdoor locations, A', B', C', D', and M'. We collect the Mahalanobis distance values of different cases, as shown in Table 4.5. The threshold t is set to 0.1, which can differentiate indoor and outdoor locations.

(2) 100-minute Experiment. In a 100-minute experiment, we observe that the Alexa device can be activated by the WAVE-HAND motion which happens in the area surrounded by the red dash line (shown in Figure 4.8). It is not activated by those three outdoor motions made by our participants. Note that we conduct the parameter calibration process when there is only one user in this apartment. However, in this 100-minute experiment, there are more than one participants in this apartment. Only one participant is allowed to wave his/her hands and the others sit on the sofa,

stand in the living room, or stay in the kitchen without making any motions. It shows that the VSButton system is not sensitive to new indoor objects (i.e., participants).

4.5 Discussions

We next discuss some limitations of VSButton.

• **Motions not from Humans.** In our current prototype, we do not consider the motions which are not from humans. For example, the VSButton may activate the HDVA service due to the jump of a pet when the owner is not around. In our future work, we plan to develop a pet-immune VSButton system.

• **Wi-Fi Hijacking Attack.** The attacker may compromise the Wi-Fi router and control the transmission power to cause a large Wi-Fi signal variance which activates VSButton. However, this attack may not be very practical since it requires a strong attack assumption. The adversary has to obtain the administrator username and password of the victim's Wi-Fi router, and further control the router's transmission power.

• **Tradeoff between Security and Convenience.** VSButton's resistance to malicious outdoor motions is a tradeoff between security and user convenience. To accommodate a variety of user demands and use environment, it is configurable by our design. For example, by increasing the threshold t, VSButton has a strong resistance to outdoor motions whereas users have a shorter communication range with their HDVA devices.

• **Physical Invasive Attack.** Our threat model assumes that the adversaries are from the outside space and cannot physically break into the room. This is because if the adversary can invade the victim's house, s(he) is able to do many things much eviller than attacking the HDVA.

Chapter 5

BFastPay: Supporting Fast Bitcoin Payment

5.1 Introduction

5.1.1 Background and Motivation

Bitcoin as a decentralized payment solution is increasingly gaining recognition and acceptance. For example, it has been accepted by many famous retailers and service providers such as Microsoft [11] and Samsung [17]. Nevertheless, Bitcoin suffers from a key problem. In practical applications, the payee needs to wait for 6 block confirmations (ave. 60 mins) for validating a Bitcoin transaction to defend against the potential double-spending attack launched by the payer. A shorter waiting time increases the risk of the double-spending attack in which the payer spends the same Bitcoin more than once and the payee loses the commodities/services without receiving the Bitcoin payment. The one-hour waiting time has seriously impeded the wide adoption of Bitcoin payment services because many businesses (e.g., vending machines) expect a much shorter waiting time. This problem is one of the most fundamental open problems of Bitcoin. In the past decade, researchers have devoted great efforts to address the problem, but no perfect solutions have been developed yet. We are thus motivated to seek for alternative approaches to address this critical problem.

5.1.2 Problem Statement

In this paper, we aim to develop a new Bitcoin payment protocol that satisfies the following requirements.

1. **Mainly using Bitcoin.** The solution should enable users to use Bitcoin as the major payment cryptocurrency instead of requiring users to adopt other cryptocurrencies.

2. **Short waiting time.** The time required for the payee to validate a Bitcoin payment should be short while still defending against the double-spending attacks.

3. **Decentralization.** The solution should preserve Bitcoin's decentralization feature: no reliance on any centralized trusted third party is required.

4. **Low-cost.** The extra operation cost should be low.

5.1.3 Our Approach: BFastPay

To support fast payment in the Bitcoin network, we propose an *inter-blockchain escrow* approach (i.e., BFastPay) in this paper. BFastPay is a general approach that can be deployed on any programmable smart contract (PSC)-supported blockchain platform (e.g., Ethereum [8], EOSIO [6]). BFastPay is called an inter-blockchain (or cross-blockchain) escrow approach since the security deposit (i.e., collateral) is escrowed on another PSC-supported blockchain. BFastPay is designed based on two key insights. First, BFastPay employs a decentralized smart contract called BFPayArbitrator to host the payer's security deposit and fulfill the role of a trusted payment arbitrator which guarantees that a payee always receives the payment even if attacks occur. Note that BFastPay preserves the *decentralization* feature if the underlying PSC-supported blockchain employs a decentralized consensus algorithm. Second, BFastPay takes advantage of the fast consensus property of emerging PSC-supported blockchains (e.g., EOSIO blockchain only needs less than 1 second to validate a transaction [6]) to reduce the waiting time of the Bitcoin transaction. More specifically, BFastPay works as follows. At first, a payer escrows sufficient security deposit into BFPayArbitrator. While the payer submits a Bitcoin transaction to the Bitcoin network, (s)he simultaneously submits a Bitcoin fast payment request (BFPayReq) message to BFPayArbitrator. The BFPayReq message contains all information that BFPayArbitrator needs to make the arbitration if a payment dispute arises later. Once the BFPayReq transaction is successfully validated in the PSC-supported blockchain, the payee can deliver commodities/services to the payer. Hence, the waiting time is reduced to the time needed to validate a transaction on the PSC-supported blockchain. If there is a payment dispute later, BFastPay allows both parties to submit evidence to prove that they are the honest parties. If the payee successfully proves that (s)he does not receive Bitcoin payment, BFPayArbitrator pays the payee using the security deposit. Otherwise, the security deposit still belongs to the payer.

5.1.4 Challenges and Solutions

The major technical challenge is that it is hard for BFPayArbitrator to recognize the dishonest party in a payment arbitration because BFPayArbitrator cannot access Bitcoin blockchain (i.e., the inter-blockchain transaction validation is hard). In a payment arbitration, both the payer and the payee have incentives to upload fake evidence to BFPayArbitrator. The payer may submit fake evidence to cheat BFPayArbitrator that the Bitcoin has already paid, while the payee may also submit fake evidence to cheat BFPayArbitrator that no Bitcoin payment is received. However, BFPayArbitrator is unable to distinguish which party is dishonest without accessing the canonical Bitcoin blockchain to obtain the ground truth. To address this challenge, we develop a Bitcoin proof-of-work (PoW)-based payment arbitration mechanism for BFPayArbitrator to identify the dishonest party. The key idea is that our PoW-based arbitration mechanism enables the honest party (either the payer or the payee) to generate a valid proof from the Bitcoin subchain, whereas the dishonest party cannot. Hence, the Bitcoin miners *automatically and unconsciously* help the honest party to generate a valid proof to win the payment arbitration. The dishonest party has to defeat all Bitcoin miners in the mining race to win the payment arbitration, so it is hard for the dishonest party to win the payment arbitration.

5.1.5 Comparison with Prior Art

We compare Ethereum-based BFastPay and EOSIO-based BFastPay with the classic escrow-based solution: Lightning Network [101]. Lightning Network is representative because most escrow-based solutions exploit a similar mechanism with Lightning Network. Lightning Network provides users with fast payment services by establishing some off-chain payment channels. Specifically, two parties (e.g., a payer and a payee) first open a secure payment channel by depositing some Bitcoin to a 2-of-2 multi-signature Bitcoin address. The parties interact directly to make payments by adjusting the respective ownership shares of the deposited fund. In cases where no direct payment channel exists between two parties, parties have to rely on intermediate peers to route transactions. The comparison results between BFastPay and Lightning Network are summarized in Table 5.1. In the table, Lightning Network is called *intra-blockchain escrow* approach because the security deposit is escrowed on the Bitcoin blockchain itself.

Table 5.1: The comparison between the intra-blockchain escrow approach (Lightning Network) and the inter-blockchain escrow approach (BFastPay) (●: yes, ◐: partial, ○: no).

Approaches	**Intra-blockchain escrow**	**Inter-blockchain escrow**	
Protocols	Lightning Network	Ethereum-based BFastPay	EOSIO-based BFastPay
Required waiting time	<1 second	≈ 3 mins	<1 second
Extra operation cost	<$0.01/tx	< $0.34/tx	$0/tx
Time required for refuel/reuse security deposit	60 mins	95 mins	95 mins
Decentralization is preserved?	◐	●	◐
Mainly using Bitcoin?	●	●	●
No requirements on the payment channel?	○	●	●
Routing-free?	○	●	●
Not only support micropayments?	○	●	●

Remarks. We have three remarks about Table 5.1.

1. Both Lightning Network and EOSIO-based BFastPay reduce decentralization. Because Lightning Network introduces payment hubs [10] and EOSIO is a permissioned blockchain in which the miners need permission to join [6].

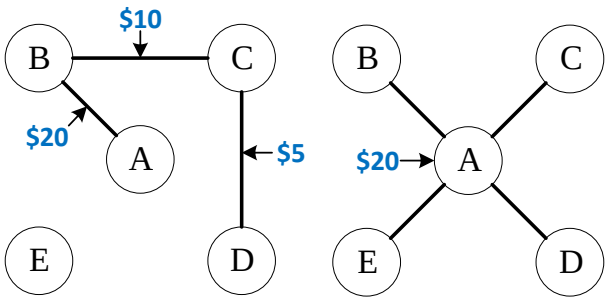
2. Both Lightning Network and EOSIO-based BFastPay can reduce waiting time to be less than 1 second. The key reason is that both of them trade decentralization for fast transaction validation.

3. Lightning Network directly uses the security deposit for fast transactions. If the security deposit is insufficient, the payer needs to trigger a Bitcoin transaction to *refuel* the security deposit, which takes 60 mins on average. In contrast, BFastPay does not directly use the security deposit for fast transactions. If there are no attacks, the security deposit will be free automatically after 95 mins[1].

Advantages. Compared with Lightning Network, BFastPay has several advantages.

1. **BFastPay has no requirements on the payment channel.** In contrast, Lightning Network has requirements on the payment channel. It requires establishing a payment channel or a routing path between the payer and the payee before using it. Figure 5.1a shows an example. Because there are no payment routing paths between A and E, the Lightning Network service is unavailable between them.

[1]The choice of 95 mins is explained in Section 5.8.2.



(a) Lightning Network (b) BFastPay

Figure 5.1: (a) In Lightning Network, the escrowed fund is associated with a payment channel. The maximum allowed transaction amount between A and D does not exceed $\texttt{min}\{20, 10, 5\} = \$5$. (b) In BFastPay, the escrowed fund is associated with a party. The maximum allowed transaction amount between A and any party is \$20.

2. **BFastPay is routing-free.** In contrast, Lightning Network requires cooperative intermediate peers to help to route transactions. Figure 5.1 shows an example. In Lightning Network, the escrowed fund is associated with a payment channel between two parties, so a payer can only use the escrowed fund for fast payment for *only one payee* (paying for other payees should rely on routing). However, in BFastPay, the escrowed fund is associated with the payer. The payer can use it to pay for *any party without routing*.

3. **BFastPay does not only support micropayments.** Lightning Network only supports micropayments, whereas BFastPay can support any payment with an amount less than the escrowed fund. Figure 5.1 shows an example. Suppose that there are three established payment channels: (1) between the party A and the party B (\$20 escrowed), (2) between the party B and the party C (\$10 escrowed), and (3) between the party C and the party D (\$5 escrowed). If the party A wants to send a fast Bitcoin payment to the party D via the routing path A→B→C→D, then the maximum allowed amount does not exceed $\texttt{min}\{10, 20, 5\} = \$5$. Generally speaking, the longer the path, the smaller the transaction amount allowed. Thus, Lightning Network suffers from a high probability of routing failure. A recent study [87, 9] shows that ***anyone who uses Lightning Network to transfer Bitcoin over \$5.5 has 50% chance of routing failure***.

Disadvantages. There are several disadvantages of BFastPay over Lightning Network. First, BFastPay requires payers to exchange other tokens (e.g., ETH, EOS) to be the security deposit.

Moreover, BFastPay requires the payer agent and the payee agent to have access to the Internet for a short period of time (i.e., during the arbitration time window). Last, BFastPay requires a slightly longer time to reuse/refuel the security deposit.

Summary. BFastPay and Lightning Network are competing approaches. Each approach has its appropriate application scenarios. For instance, Lightning Network is more suitable for micropayments, while BFastPay is more suitable for relatively large payments.

5.2 Related Work

We omit the related work on solutions [101, 53, 54, 108, 70, 100] that support fast Bitcoin payment since they have been covered in the previous section. In this section, we review two projects that work on inter/cross-blockchain transaction validation: BTC Relay project [2] and Summa project [21]. (1) BTC Relay project lets relayers relay Bitcoin blockchain headers to the Ethereum blockchain. The Bitcoin blockchain headers can be used by Ethereum smart contracts to validate Bitcoin transactions based on simple payment verification (SPV) [91] method. BTC Relay has two limitations. First, because of the lack of relayers, BTC Relay only works intermittently and hence it is not reliable. Second, it is very expensive to store the entire Bitcoin blockchain headers in Ethereum blockchain. Compared with BTC relay, BFastPay is a much more reliable and low-cost solution. (2) The Summa solution validates a Bitcoin transaction in an Ethereum smart contract only checking the total PoW carried by a proof. However, this solution is not sufficiently secure because an adversary with a small portion of hash power can fabricate a long enough block header chain by extending the mining time. In contrast, based on the novel PoW-based arbitration mechanism, BFastPay can support a much higher level of confidence for inter/cross-blockchain Bitcoin transaction validation.

5.3 Preliminaries

This section introduces the preliminaries of the Bitcoin blockchain and programmable smart contract (PSC)-supported blockchains.

Bitcoin Blockchain. The Bitcoin blockchain [91] is a shared public ledger on which all Bitcoin transactions are recorded. Numerous Bitcoin transactions are put into a new block and appended

to the blockchain in chronological order. When a block that contains a new Bitcoin transaction has been appended to the Bitcoin blockchain, this transaction has one block confirmation. When a subsequent block is appended to the blockchain, the number of block confirmation for this transaction is increased by one [4]. The current practice for accepting a secure Bitcoin transaction is: waiting for such transaction to have 6 block confirmations. ***Note that 6 block confirmations are based on an assumption that adversaries do not control more than 10% of the global hash power of the Bitcoin network and a double-spending probability of less than 0.1% is acceptable*** **[91].** Bitcoin network refers to the collection of nodes (e.g., miners, wallets) running the Bitcoin P2P protocol. The Bitcoin network uses a PoW-based method for reaching a consensus between different miners. The miner who can solve the hash-based PoW puzzle wins the right to produce a new block for the blockchain. A Bitcoin block header is 80 bytes containing information of (1) previous block hash field, (2) Merkle root field, (3) nonce field, etc. In the mining process, the miners try to find a nonce such that the mined block header meets the current Bitcoin network difficulty target, i.e., $\texttt{Hash}(\texttt{block header}) \leq \texttt{BTC\_diff\_target}$.

PSC-Supported Blockchains. Nowadays, blockchain technology is evolving beyond just supporting a cryptocurrency. Some emerging blockchains (e.g., Ethereum, EOSIO) support rich programmable smart contract functionalities. A smart contract model typically consists of program code (run on the blockchain), a storage file (stored on the blockchain), and an account balance (recorded on the blockchain). A user can deploy a smart contract by posting a transaction to the blockchain. A user can send a message (via a transaction) to a smart contract to trigger its function execution. All content of the blockchain, smart contracts, and transactions is publicly visible. The smart contract can partially fulfill the role of a *trusted third party*. After auditing from involved users and validating on the PSC-supported blockchain, the smart contract code is immutable, and all code executes exactly according to how it was programmed. Hence, the smart contract can support the program-controlled fund transfer. The fund managed by the smart contract is represented in the form of *token*. For example, Ethereum blockchain uses ETH token and EOSIO blockchain uses EOS token.

As shown in Table 5.2, *different blockchains exploit different consensus mechanisms, resulting in the different time needed to validate a transaction.* Compared with Bitcoin, many recently developed PSC-supported blockchains have a shorter transaction validation time. For instance,

as analyzed in [13], Ethereum needs about 12 confirmations (about 3 mins) to achieve a similar degree of security as 6 confirmations (about 60 mins) on Bitcoin blockchain.

Table 5.2: The consensus mechanisms and transaction validation time of different PSC-supported blockchains.

Blockchain type	**Blockchain**	**Consensus mechanism**	**Ave. tx validation time**
Non-PSC-supported blockchain	Bitcoin	Hash-based proof-of-work (PoW) protocol	$\approx$ 60 mins
PSC-supported blockchain	Ethereum	Modified "Greedy Heaviest Observed Subtree" (GHOST) protocol [8]	$\approx$ 3 mins
	EOS	Asynchronous Byzantine Fault Tolerance (aBFT) protocol [6]	$<$ 1 second
	Stellar [20]	Federated Byzantine Agreement (FBA) protocol [20]	$<$ 5 seconds
	Cardano [74]	Ouroboros Proof-of-Stake (PoS) protocol [74]	$<$ 5 mins
	NEO [12]	Delegated Byzantine Fault Tolerant (dBFT) protocol [12]	$\approx$ 15 seconds

5.4 Threat Model and Assumptions

Threat Model. The security threat mainly comes from the payer or the payee. (1) **Payer.** The payer may double-spend the Bitcoin and hence no Bitcoin payment will be received by the payee. Additionally, the payer attempts to win the payment arbitration to avoid releasing the security deposit to the payee. If the payer successfully double-spends the Bitcoin and wins the arbitration, then the payee loses the commodities/services without receiving any payment. This attack is a double-spending attack. (2) **Payee.** The payee is considered as a *semi-benign* entity. The payee may still raise a dispute and hope to win the arbitration even if (s)he has received the Bitcoin payment. If the payee wins the arbitration, then (s)he can receive payments twice. This attack is a double-payment attack. We do not consider the risk that the payee refuses to deliver commodities/services to the payer even if the payer correctly finishes the payment phase required by BFastPay because such risk exists in any payment method. Therefore, handling such risk is out of the scope of this paper. We note that this problem has been studied in [61].

Assumptions. We have the following assumptions. (1) Both the payer and payee are rational and they would defend for their own interests (e.g., the payer/payee would take actions to thwart any double-payment/double-spending attempt). (2) Both the payer and the payee can control a portion of the global Bitcoin hash power to launch attacks. However, either payer or payee cannot control more than 50% of the global hash power for Bitcoin mining. We assume that the remaining hash power is controlled by honest miners, who work together to extend the longest Bitcoin blockchain

as stipulated by the Bitcoin protocol. (3) The smart contract platforms adopted by BFastPay are secure. We admit that some existing smart contract platforms have security vulnerabilities. However, developers have devoted great efforts to fix the known vulnerabilities and bring them into real-world applications. (4) Both the payer and payee have fairly reliable Internet connections during BFastPay service.

5.5 BFastPay Overview

In this section, we first briefly describe BFastPay flowchart and then clarify how to use the security deposit in BFastPay.

5.5.1 BFastPay Flowchart

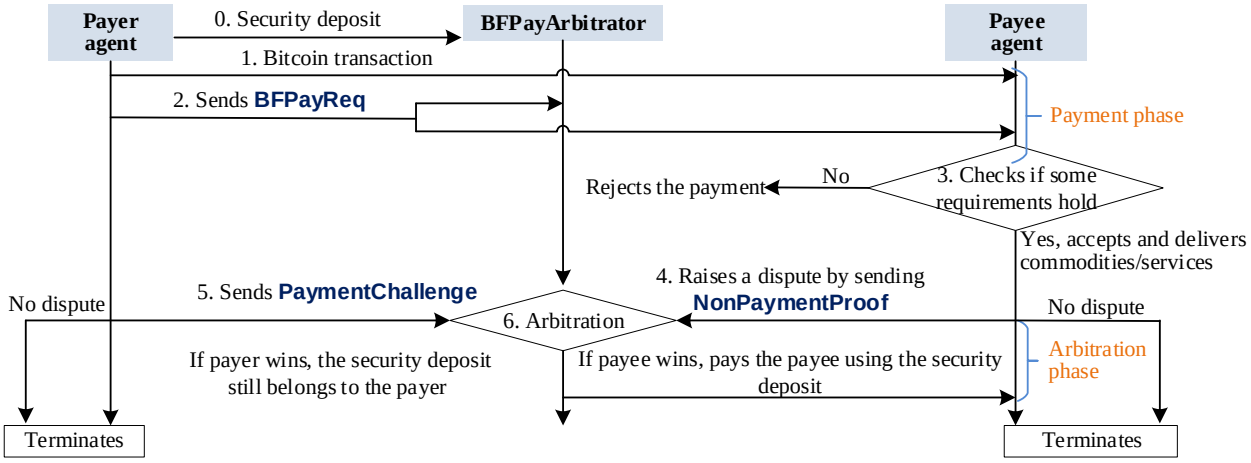


Figure 5.2: The flowchart of BFastPay.

Figure 5.2 depicts BFastPay flowchart, which consists of two phases: the payment phase and the arbitration phase.

Payment Phase. There are three steps (steps 1-3 in Figure 5.2) in this phase.

1. Before using the BFastPay service, the payer agent should first escrow sufficient security deposit to BFPayArbitrator which is deployed on a PSC-supported blockchain. The security deposit is added to BFPayArbitrator via a transaction sending from the payer's PSC-supported blockchain account address `Payer_addr` to BFPayArbitrator. Then, the payer agent can send a Bitcoin transaction to the payee.

2. While the payer agent broadcasts the Bitcoin transaction to the Bitcoin network, the payer agent simultaneously submits a BFPayReq message to BFPayArbitrator. The BFPayReq message consists of (1) the Bitcoin payment information, (2) the Bitcoin blockchain information, (3) the PSC-supported blockchain information, and (4) the transaction amount. Table 5.3 summaries all of the information carried in the BFPayReq message. Note that the payer needs to refer to the public sources (e.g., [3]) to get the real-time conversion rate between bitcoin and the PSC-supported blockchain token to compute the amount of token with a matching value. BFastPay uses the conversion rate when the Bitcoin transaction occurs. Future conversion rate fluctuations do not affect the fairness of the transaction.

3. The payee agent receives the BFPayReq message and checks if the following two requirements hold. (1) Each field of the BFPayReq message is correct[2]. (2) The payer's available security deposit should not be less than the escrowed Bitcoin transaction amount[3]. If the payee agent finds any field in the BFPayReq message is incorrect or the security deposit is insufficient, the payee rejects the Bitcoin transaction. Otherwise, the payee waits for BFPayReq to be validated by the PSC-supported blockchain before accepting the Bitcoin transaction and delivering commodities/services to the payer. The waiting time is thus reduced to be the time needed to validate a transaction on the PSC-supported blockchain.

Arbitration Phase. If the payee agent successfully receives the Bitcoin payment later and does not raise a dispute during a pre-defined arbitration time window, then the BFastPay service life cycle finishes and terminates. Otherwise, BFastPay allows the payee agent to raise a dispute by sending NonPaymentProof to BFPayArbitrator. Then, the BFastPay arbitration phase is activated. ***Note that there are no attacks in the vast majority of real-world Bitcoin payment cases, so the arbitration phase is rarely activated.*** There are three steps (steps 4-6 in Figure 5.2) in the arbitration phase.

[2]To check the correctness of BFPayReq message, the payee can refer to the trust sources and make a field-to-filed comparison. More specifically, the payment information (`BTC_TxID`, `BTC_Tx_time`) and blockchain information (`BTC_diff_target`, `Block_hash`) is public on the Bitcoin Blockchain. The correct PSC-supported blockchain information (`Payer_addr`, `Payee_addr`) is also known by the payee. In addition, the transaction amount (`Token_amount`) can be computed by the payee because the real-time conversion rate is publicly accessible by the payee.

[3]The adopted check mechanism is exactly the same as the mechanism used in the cash-based payments: after the payer gives cash to the payee, the payee must check and ensure that (1) it is not the fake cash and (2) the amount of cash is sufficient before accepting it.

Table 5.3: The information in BFPayReq message.

Element type	Field name	Description
Payment info	`BTC_TxID`	The Bitcoin transaction ID
	`BTC_Tx_time`	The Bitcoin transaction time
Blockchain info	`BTC_diff_target`	Current Bitcoin network difficulty target
	`Block_hash`	The hash of the latest Bitcoin block header that does not include the escrowed Bitcoin tx
PSC-supported blockchain info	`Payer_addr`	The payer's PSC-supported blockchain account address
	`Payee_addr`	The payee's PSC-supported blockchain account address
Transaction amount	`Token_amount`	The amount of token needs to transfer to the payee if a double-spending attack occurs

1. To raise a dispute, the payee agent needs to generate NonPaymentProof and submits it to BFPayArbitrator for arbitration within a pre-defined arbitration time window.

2. The payer agent examines BFPayArbitrator to check if there is a NonPaymentProof message received by BFPayArbitrator during the arbitration time window. If the payer finds that NonPaymentProof message is received, BFPayArbitrator allows the payer to generate PaymentChallenge and to send it to BFPayArbitrator within a pre-defined rebuttal time window.

3. BFPayArbitrator arbitrates the dispute based on the received NonPaymentProof and PaymentChallenge. If the payee wins the dispute, BFPayArbitrator pays the payee using the payer's security deposit. Otherwise, the security deposit still belongs to the payer. After the arbitration process, the BFastPay service finishes and terminates.

The detailed arbitration mechanism is further described in Section 5.6. Note that no manual operations are needed in using BFastPay. The payer agent and payee agent (e.g., smartphone) run BFastPay software to automatically finish the required operations. The software modules of BFastPay are introduced in Section 5.8.1.

5.5.2 Security Deposit Clarification

The security deposit in BFPayArbitrator has two states: free and frozen. Consider that the payer has a security deposit worth S_1 dollars and a BFastPay Bitcoin payment has a transaction amount worth S_2 dollars, where $S_1 > S_2$. Then, during the service life cycle of BFastPay, BFPayArbitrator freezes S_2 dollars. The remaining $(S_1 - S_2)$ dollars are free. After the termination of a BFastPay fast payment service, if the frozen security deposit does not release to the payee, then it will be free again. The free deposit can be used for concurrent or future BFastPay fast payment services. Therefore, if both parties are honest (the vast majority of cases), the payer can enjoy a one-time deposit and permanent BFastPay fast payment services. *Note that the payer can withdraw free security deposit to his own account at any time.*

5.6 BFastPay Arbitration

In this section, we describe (1) how to design NonPaymentProof and PaymentChallenge, (2) how to check the validity of NonPaymentProof and PaymentChallenge, and (3) the detailed PoW-based arbitration mechanism used in BFPayArbitrator.

5.6.1 NonPaymentProof Design and Validation

In an arbitration, to convince BFPayArbitrator that the escrowed Bitcoin transaction is not included in the Bitcoin blockchain, the payee needs to submit NonPaymentProof to BFPayArbitrator. NonPaymentProof contains a number of linked block headers (named *block header proof*). The Block header proof here is used to prove that sufficient PoW has been done to extend the Bitcoin blockchain, in which the escrowed Bitcoin transaction is not included. Figure 5.3 illustrates the structure of the block header proof. The number of the linked block headers in the block header proof is defined as the length of NonPaymentProof (denoted as n_1).

A valid NonPaymentProof should satisfy the following four requirements (R1)-(R4).

- (**R1**) The block headers meet the current Bitcoin difficulty target: $\texttt{Hash}(\texttt{block header No.}\, i) \leq \texttt{BTC\_diff\_target}$, for $i = 1, \cdots, n_1$.
- (**R2**) The block headers indeed form a linked blockchain:

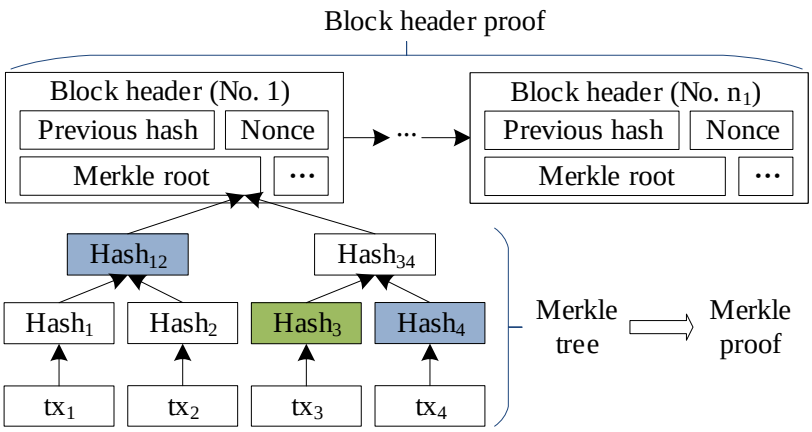


Figure 5.3: The structures of the block header proof and the Merkle proof.

$\texttt{Hash}(\texttt{block header No.}\ i) =$ the previous hash field in block header No. $i+1$, for $i = 1, \cdots, n_1 - 1$.

- (**R3**) The block header chain indeed extends from the latest Bitcoin block when the escrowed Bitcoin transaction occurs: the previous hash field in the first block header equals `Block_hash`.

- (**R4**) The escrowed Bitcoin transaction is not included in the first block header in NonPaymentProof.

On receipt of NonPaymentProof, BFPayArbitrator can check whether the requirements (R1)-(R3) are satisfied or not, but it cannot check whether the fourth requirement (R4) is satisfied or not (it is hard to prove non-inclusion of a transaction. For example, the classic *Merkle proof* [91] can prove inclusion but not non-inclusion). To prevent the payee from submitting NonPaymentProof which does not satisfy the requirement (R4), BFastPay allows the payer to reveal such cheating behavior of the payee by submitting PaymentChallenge (by using the Merkle proof) to prove that the escrowed Bitcoin transaction is included in the first block header of NonPaymentProof. If there is no such PaymentChallenge received and the requirements (R1)-(R3) are satisfied, then NonPaymentProof is treated to be valid. Otherwise, it is invalid.

5.6.2 PaymentChallenge Design and Validation

On receipt of NonPaymentProof, BFPayArbitrator checks if one of the requirements from (R1)-(R3) is not satisfied. If yes, BFPayArbitrator directly lets the payer win the arbitration without any requirements for PaymentChallenge. If no, the payer needs to submit PaymentChallenge to BFPayArbitrator. There are two cases as described below.

Case 1: NonPaymentProof Satisfies Requirement (R4). In this case, PaymentChallenge consists of two components: the block header proof and the Merkle proof. (1) *Block Header Proof.* The block header proof in PaymentChallenge is used to prove that sufficient PoW has been done to extend the Bitcoin blockchain. Figure 5.3 depicts the structure of a block header proof. The number of the linked block headers in the block header proof is defined as the length of PaymentChallenge (denoted by n_2). A valid block header proof should satisfy the requirements (R1)-(R3) as stated in Section 5.6.1. (2) *Merkle Proof.* The Merkle proof is used to prove that the escrowed Bitcoin transaction is indeed included in the first block header of the block header proof. The Merkle proof is generated from a Merkle tree, as shown in Figure 5.3. The Merkle tree has a number of leaf nodes at the bottom of the tree containing hashes of each transaction in a Bitcoin block. An intermediate node in the tree is the hash of its two children. Finally, a single root node (called a Merkle root) can be obtained. How to generate and validate the Merkle proof can be illustrated by the following example. As shown in Figure 5.3, in order to validate the inclusion of $\texttt{tx}_3$, the payer only needs to provide a Merkle proof, which consists of two parts: (1) the Merkle branch [$\texttt{Hash}_4$, $\texttt{Hash}_{12}$], and (2) the branch position [right, left]. To validate the Merkle proof, BFPayArbitrator computes

$$\texttt{Hash}_3 = \texttt{Hash}(\texttt{tx}_3),$$

$$\texttt{Hash}_{34} = \texttt{Hash}(\texttt{Hash}_3||\texttt{Hash}_4), (\texttt{Hash}_4 \texttt{ on } \texttt{right}),$$

$$\texttt{Hash}_{1234} = \texttt{Hash}(\texttt{Hash}_{12}||\texttt{hash}_{34}), (\texttt{Hash}_{12} \texttt{ on } \texttt{left}),$$

where $||$ represents concatenation. If $\texttt{Hash}_{1234}$ equals the Merkle root of the first block header, then $\texttt{tx}_3$ is indeed included in the block header. If the Merkle proof in PaymentChallenge can successfully prove the inclusion of the escrowed Bitcoin transaction, then it is valid. Otherwise, the Merkle proof is considered to be invalid. Note that PaymentChallenge is valid if both its block header proof and Merkle proof are valid. Otherwise, it is invalid.

Case 2: NonPaymentProof Does not Satisfy Requirement (R4). In this case, PaymentChallenge only contains the Merkle proof, which is submitted to BFPayArbitrator to prove that NonPayment-Proof does not satisfy requirement (R4). In other words, the Merkle proof is used to prove that the escrowed Bitcoin transaction is indeed included in the first block header of NonPaymentProof. The inclusion-proof process is exactly the same as described above. If the Merkle proof successfully proves that NonPaymentProof does not satisfy the requirement (R4), then PaymentChallenge is valid. Otherwise, it is invalid.

5.6.3 PoW-based Arbitration Mechanism

We next introduce the arbitration window and rebuttal time window settings. Then, we describe the PoW-based arbitration mechanism. Last, the strategy which can ensure the honest party to win is illustrated.

Arbitration and Rebuttal Time Window Settings. Let T_c denote the elapsed time since the Bitcoin transaction is broadcast to the Bitcoin network. The arbitration time window is set to be $[T_c - \varepsilon, T_c]$ and the rebuttal time window is set to be $[T_c, T_c + \varepsilon]$. We set ε to be 5 mins in BFastPay. To simplify our theoretical analysis later, we treat that both the payee and the payer submit their evidence (i.e., NonPaymentProof and PaymentChallenge) for arbitration at the same time point T_c. The adjustable parameter T_c is also called the mining *competition time period*.

Arbitration Mechanism. The key mechanism in the payment arbitration is: ***If both NonPayment-Proof and PaymentChallenge are valid, the winner is the party who submits a block header proof that carries more PoW.*** The precise and complete logic of the arbitration mechanism is illustrated as follows.

- Case 1 (NonPaymentProof does not satisfy the requirements (R1)-(R3)): the payer wins.
- Case 2 (NonPaymentProof satisfies the requirements (R1)-(R3) but no PaymentChallenge is received): the payee wins.
- Case 3 (NonPaymentProof satisfies the requirements (R1)-(R3) and PaymentChallenge with only Merkle proof is received): if PaymentChallenge is valid, then the payer wins. Otherwise, the payee wins.

- Case 4 (NonPaymentProof satisfies the requirements (R1)-(R3) and PaymentChallenge with both the block header proof and the Merkle proof is received): if PaymentChallenge is invalid, then the payee wins. Otherwise, the winner is the party who submits a block header proof that carries more PoW. In the case of a tie, BFPayArbitrator lets the payee win. Mathematically, recall that the block header lengths of NonPaymentProof and PaymentChalleng are denoted as n_1 and n_2, respectively. A simplified description is: if $n_1 \geq n_2$, then the payee wins; otherwise, the payer wins.

Strategy of the Honest Party. To ensure the honest party to win the arbitration, BFastPay leverages the following strategy. There are two cases.

- Case 1 (the escrowed Bitcoin transaction is not included in the Bitcoin blockchain): in this case, the payee is the honest party who can directly truncate a segment of the block header chain from the Bitcoin blockchain to generate a long valid NonPaymentProof (the truncated block header chain segment starts with the block header in which the previous hash field equals `Block_hash` (see Table 5.3) and ends with the latest block header). If the payer adopts the same strategy as the payee, the payer cannot get a valid PaymentChallenge simply because the escrowed Bitcoin transaction is not included in the Bitcoin blockchain so that a valid Merkle proof of inclusion cannot be generated by the payer. Therefore, the dishonest party (i.e., the payer) has to fabricate PaymentChallenge.

- Case 2 (the escrowed Bitcoin transaction is included in the Bitcoin blockchain): in this case, the payer is the honest party who can directly truncate a segment of block header chain from the Bitcoin blockchain to generate a long valid PaymentChallenge, whereas the dishonest party (i.e., the payee) has to fabricate NonPaymentProof.

Why the Honest Party can Win? The honest miners always work to extend the Bitcoin blockchain from which the honest party can truncate a segment of block header chain to generate a long valid NonPaymentProof (in case the payee is honest) or a long valid PaymentChallenge (in case the payer is honest). Hence, by extending the Bitcoin blockchain, the honest miners actually help the honest party to generate what is desirable (i.e., either NonPaymentProof or PaymentChallenge). Recall the above PoW-based arbitration rule, the dishonest party has to defeat the honest miners in

the block generation (i.e., mining) race in order to win. In a nutshell, the reason why the honest party can win the arbitration is: ***Because honest miners (with a large portion of hash power) always help the honest party, the dishonest party (with a small portion of hash power) is hard to win the Bitcoin block generating race (i.e., mining race) in the long run.*** Note that the miners help the honest party *unconsciously and automatically*. The coordination with honest miners is never required. The detailed security analysis is presented in Section 5.7.

Handling Transaction Delay. The above design is based on the fact that the escrowed Bitcoin transaction sets a sufficient transaction fee to ensure that it is mined in the very first block after it is broadcasted to the Bitcoin network. For a Bitcoin transactions with a low transaction fee, it may wait for several blocks to be included in the Bitcoin blockchain, resulting in the transaction delay issue. Theoretically, there is no transaction delay issue if the transaction fee is set to be sufficient high because the mining priority of miners is determined by the transaction fee.

The following method can be used to handle the transaction delay issue. Consider that the transaction fee is too low and the escrowed Bitcoin transaction is mined after n' blocks since it is broadcasted. If BFPayArbitrator receives NonPaymentProof from the payee, then the payer can send a PaymentChallenge in the same way. The only difference is that the length of PaymentChallenge is counted as $n_1 - n'$, which means that the first n' blocks are not counted in the PoW. As mentioned above, the miners automatically help the payer to extend the length of PaymentChallenge, so the payer can still win the arbitration in the long run. In this case, BFastPay just needs to increase the parameter T_c to ensure the payer to win.

5.7 Security Analysis

In this section, we first analyze how BFastPay defends the hash-based double-spending attack and double-payment attack. Then, we analyze how BFastPay defends some other possible attacks.

Zero-Sum Game. An attack in BFastPay is a zero-sum game between the payer and the payee. In a double-spending or a double-payment attack, the gain or loss of one party is exactly balanced by the loss or gain of the other party. Therefore, the payer and the payee have a conflict of interests and no collusion attacks are possible. When one party attempts to launch an attack, the other party will try to prevent the attack to defend for his/her own interests.

Mathematical Notations. The payer-controlled and the payee-controlled hash power are represented as αH and βH, respectively, where H is the global hash power for mining Bitcoin. The remaining hash power (denoted as γH) is controlled by the honest miners. It follows that $\alpha + \beta + \gamma = 1$.

5.7.1 Defending Double-spending Attack

In the double-spending attack, the adversary is the payer who has successfully double spent the Bitcoin and tries to win the payment arbitration. The payee is the defender, who attempts to win the arbitration and receive the payment from BFPayArbitrator.

Payer (Adversary). To maximize the probability of winning the arbitration, the payer needs to submit a PaymentChallenge as long as possible. The canonical Bitcoin blockchain extended by the honest miners cannot be used to generate a valid PaymentChallenge since the escrowed Bitcoin transaction is not included. This means that the payer has to fabricate a valid PaymentChallenge by investing his/her controlled hash power. Therefore, PaymentChallenge is generated with hash power αH.

Payee (Defender). To maximize the probability of winning the arbitration, the payee needs to submit a valid NonPaymentProof as long as possible. Note that the honest miners will extend the canonical Bitcoin blockchain and a valid NonPaymentProof can be directly generated from the canonical Bitcoin blockchain. Therefore, the payee does not need to generate NonPaymentProof by only relying on his/her controlled hash power. To get a longer NonPaymentProof at the arbitration time, the payee should invest his/her controlled hash power to work together with the honest miners to extend the canonical Bitcoin blockchain. Therefore, NonPaymentProof is generated with hash power $(\beta + \gamma)H = (1 - \alpha)H$.

Attack Success Rate Analysis. The attack success rate is equivalent to the probability that the adversary successfully fabricates an alternative chain (mined with αH hash power) longer than the honest Bitcoin blockchain (mined with $(\beta + \gamma)H$ hash power) in the competition time period T_c. Because $\alpha < \beta + \gamma$, the probability decreases exponentially with an increasing T_c. We now analyze the probability that the adversary (with αH hash power) wins the mining race against the miners (with $(1 - \alpha)H$ hash power) in T_c time. The number of blocks expected to be mined by the Bitcoin miners in a certain time can be modeled as *Poisson distribution* [103]. Suppose that the honest

miners produce one block per 10 mins on average, the probability of mining exactly k_1 blocks in T_c mins is given by

$$P_1(k_1, T_c) = e^{-\frac{T_c}{10}} \frac{(\frac{T_c}{10})^{k_1}}{k_1!}, \tag{5.1}$$

where $e = 2.71828\cdots$ is the base of the natural logarithm. Likewise, the probability of mining exactly k_2 blocks in T_c mins for the adversary is given by

$$P_2(k_2, T_c) = e^{-\frac{\alpha T_c}{10(1-\alpha)}} \frac{(\frac{\alpha T_c}{10(1-\alpha)})^{k_2}}{k_2!}. \tag{5.2}$$

The double-spending attack occurs if the adversary produces an alternative blockchain longer than the honest blockchain. Table 5.4 enumerates all the cases for double-spending attacks and their probability. By summing up all cases, the probability of double-spending attack P_{ds} can be computed by

$$P_{ds} = \sum_{k_1=0}^{+\infty} [P_1(k_1, T_c) \sum_{k_2=k_1+1}^{+\infty} P_2(k_2, T_c)], \tag{5.3}$$

where $P_1(k_1, T_c)$ and $P_2(k_2, T_c)$ are defined in Equation (5.1) and Equation (5.2), respectively.

Table 5.4: Double-spending attacks cases and their probabilities.

Case num.	**Case description**	**Double-spending probability**
0	honest miners mine 0 block; adversary mines $\geq$1 blocks	$P_1(0, T_c) \sum_{k_2=1}^{+\infty} P_2(k_2, T_c)$
1	honest miners mine 1 block; adversary mines $\geq$2 blocks	$P_1(1, T_c) \sum_{k_2=2}^{+\infty} P_2(k_2, T_c)$
2	honest miners mine 2 block; adversary mines $\geq$3 blocks	$P_1(2, T_c) \sum_{k_2=3}^{+\infty} P_2(k_2, T_c)$
...	...	...

Figure 5.4 plots the probability of double-spending P_{ds} as a function of T_c when $\alpha = \beta = 0.1$, $\gamma = 0.8$. It shows that P_{ds} is decreasing exponentially with an increasing T_c. Accordingly, the double-spending probability P_{ds} can be reduced to sufficiently small by increasing the parameter T_c in BFastPay. For example, if $T_c = 95$ mins, then it holds that $P_{ds} = 0.096\% < 0.1\%$.

5.7.2 Defending Double-payment Attack

In the double-payment attack, the payee is the adversary who has successfully received the Bitcoin payment and tries to receive a second payment from BFPayArbitrator. The payer is the defender,

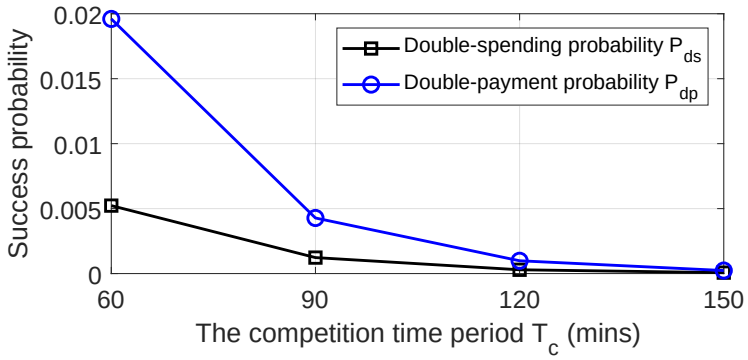


Figure 5.4: The success probability of double-spending/payment as a function of T_c ($\alpha = \beta = 0.1$, $\gamma = 0.8$).

Table 5.5: Cost for different operations in Ethereum-based BFastPay.

Operations	Pre-operations	Ethereum tx (BFPayReq)	Ethereum tx (NonPaymentProof)	Ethereum tx (PaymentChallenge[1])	Ethereum tx (PaymentChallenge[2])	Ethereum tx (send deposit)
Cost (gas)	2.46×10^6	2.78×10^5	1.04×10^5	1.32×10^5	8.08×10^4	6.60×10^4
Cost fee in ETH	4.9×10^{-3}	5.56×10^{-4}	2.08×10^{-4}	2.54×10^{-4}	1.62×10^{-4}	1.32×10^{-4}
Cost in $ (300$/ETH)	$1.47	$0.167	$0.0624	$0.0762	$0.0486	$0.0396
1: PaymentChallenge contains both the Merkle proof and the block header proof. 2: PaymentChallenge contains only the Merkle proof.						

who wants to win the arbitration and prevent the payee from receiving a second payment from BFPayArbitrator. In the double-payment attack, NonPaymentProof is generated with hash power βH and PaymentChallenge is generated with hash power $(\alpha+\gamma)H = (1-\beta)H$. By the similar analysis as P_{ds}, the probability of double-payment attack P_{dp} is given by

$$P_{dp} = \sum_{k_1=0}^{+\infty} [P_1(k_1, T_c) \sum_{k_2=k_1}^{+\infty} P_2(k_2, T_c)], \tag{5.4}$$

where $P_1(k_1, T_c)$ and $P_2(k_2, T_c)$ are defined in Equation (5.1) and Equation (5.2), respectively. The sole difference in computing p_{dp} and p_{ds} is caused by the arbitration mechanism in dealing with the tie: if the lengths of NonPaymentProof and PaymentChallenge equal, then BFPayArbitrator lets the payee win. Figure 5.4 plots the probability of double-payment P_{dp} as a function of T_c. As expected, P_{dp} is decreasing exponentially with an increasing T_c. Thus, the double-payment probability P_{dp} can be reduced to sufficiently small by increasing the parameter T_c.

5.7.3 Defending Other Attacks

Fake BFPayReq Attack. In BFastPay, the payer may submit a fake BFPayReq message to BFPayArbitrator to launch a variety of attacks. For example, the payer can send BFPayReq with

`Token_amount` $= 0$ to BFPayArbitrator, where `Token_amount` specifies the amount of token that should be paid to the payee if Bitcoin transaction is not included in the Bitcoin blockchain. This indicates that the payee will receive no payment if the Bitcoin is double spent by the payer. To resist the fake BFPayReq message attack, the payee must check the correctness of BFPayReq message. If BFPayReq message is correct, the payee can accept the Bitcoin payment and deliver commodities/services to the payer. Otherwise, the payee rejects the Bitcoin payment.

Impersonation Attack. An adversary may impersonate either the payer or the payee to launch attacks. For example, the adversary can impersonate the payee to raise a dispute by sending a fake NonPaymentProof to BFPayArbitrator. This impersonation attack can be defended by the access control provided by BFPayArbitrator. BFPayArbitrator stores the account addresses of both payer and payee (see Table 5.3), so only NonPaymentProof sent from the payee's account address and PaymentChallenge sent from the payer's account address can be accepted by BFPayArbitrator.

Segment Replay Attack. In a segment replay attack, the adversary may replay a segment of Bitcoin blockchain to generate NonPaymentProof or PaymentChallenge. For example, in a payment dispute, the payee may truncate any segment from Bitcoin blockchain to generate a long NonPaymentProof and try to send it to BFPayArbitrator to win the payment arbitration. This attack cannot succeed because NonPaymentProof or PaymentChallenge generated by the above way cannot meet the requirement (R3), so they are invalid. The party who sends an invalid NonPaymentProof or an invalid PaymentChallenge will lose the arbitration immediately.

Pre-mining Attack. The adversary with less hash power is hard to win the arbitration in a long competition time period T_c, but the adversary may fabricate NonPaymentProof or PaymentChallenge ahead of time to make them long enough to win the later arbitration. To defend the pre-mining attacks, BFPayArbitrator requires valid NonPaymentProof and PaymentChallenge to extend from the latest Bitcoin block when the Bitcoin transaction occurs (via checking the requirement (R3)). The latest block hash will be updated whenever a new Bitcoin block is generated, so the block hash (at the time when the escrowed Bitcoin transaction occurs) cannot be known by the adversary in advance. Therefore, the adversary cannot pre-mine NonPaymentProof or PaymentChallenge to launch the attack.

5.8 Evaluation of Cost

BFastPay is a general approach that can be deployed on any PSC-supported blockchain platform. We instantiate BFastPay on top of two popular PSC-supported blockchains (i.e., Ethereum and EOSIO) and then we evaluate their operation cost.

5.8.1 Implementation

BFastPay Modules. Figure 5.5 shows the modules in BFastPay prototype. The payer agent consists of the BFPayReqGen module and the PaymentChallengeGen module. The payee agent contains the BFPayReqCheck module and the NonPaymentProofGen module. All of the four modules connect to both the Bitcoin network and the Ethereum/EOSIO network to listen/access the needed information. The four modules work as follows. (1) The BFPayReqGen module generates the BFPayReq message and sends it to BFPayArbitrator whenever there are requests from the payer. All of the information in the BFPayReq message is publicly accessible. (2) The EscrowCheck module can automatically check if BFPayReq sent by the payer agent is correct or not by comparing it with the ground truth accessible from the public sources. (3) The NonPaymentProofGen module helps the honest payee to generate NonPaymentProof from the Bitcoin blockchain. (4) The PaymentChallengeGen module generates PaymentChallenge by truncating a segment of the block header chain from Bitcoin blockchain. Some implementation details are skipped due to the lack of space.

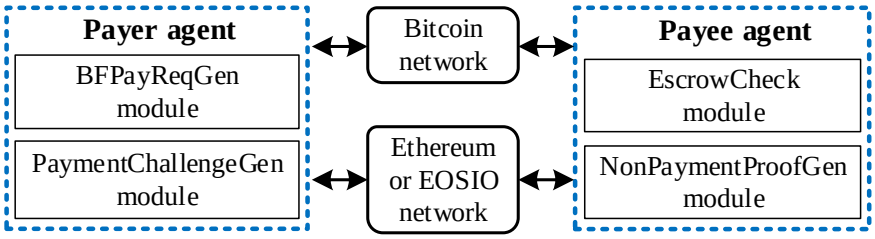


Figure 5.5: BFastPay modules.

Ethereum and EOSIO-based BFPayArbitrator. Ethereum-based BFPayArbitrator is implemented by Solidity [18, 19]. The browser-based compiler and IDE called Remix [15] is applied to develop BFPayArbitrator. We use the Ethereum test network Rinkeby [16] to test BFPayArbitrator. EOSIO-based BFPayArbitrator is developed by EOSIO C++ in the EOS Contract Development

Toolkit (CDT) 1.3.1 [7].

5.8.2 Evaluation

In this section, the experiment settings are first introduced. Then, we evaluate the operation cost of Ethereum-based BFastPay and EOSIO-based BFastPay.

Experiment Settings. In the experiments, the competition time T_c is set to be 95 mins (i.e., the security deposit will be frozen for 95 mins and then be free again). If $T_c = 95$ mins, then $P_{ds} = 0.096\% < 0.1\%$ according to Equation (5.3). That is, the double-spending probability in BFastPay is less than 0.1% in the presence of an adversary with 10% of the global hash power, Therefore, BFastPay achieves a comparable security level as the 6-confirmation-waiting approach against the double-spending attacks[4].

Ethereum-based BFastPay Evaluation. The Ethereum smart contract supported functions have "gas" cost depending on how many computational steps and storage space it requires. The cost is computed as (costed gas)×(gas price). In the experiments, the gas price is set to be 2 gwei, where 1 gwei$= 10^{-9}$ ETH. The cost of Ethereum-based BFastPay may come from five parts: (1) Pre-operations (including deploy contract operation and add security deposit operation), (2) Ethereum tx (send BFPayReq), (3) Ethereum tx (send NonPaymentProof), (4) Ethereum tx (send PaymentChallenge), and (5)Ethereum tx (transfer security deposit to the payee).

Table 5.6 summarizes the average cost for each part. The fee for pre-operations is a one-time cost, so we do not consider them in calculating the operation cost per fast Bitcoin transaction. There are many use cases of BFastPay, resulting in different operation costs. We consider the two most common use cases to evaluate the operation cost of BFastPay for a Bitcoin transaction.

- **Case 1** (no dispute arises): In this case, the cost only comes from Ethereum tx (send BFPayReq). The operation cost per Bitcoin transaction is 5.56×10^{-4} ETH (or \$0.167, 300 \$/ETH[5]).

- **Case 2** (dispute arises): In this case, we consider the payee sends out NonPaymentProof and the payer sends out PaymentChallenge. If the payer wins, the cost comes from (1) Ethereum

[4]Note that the double-spending probability in the 6-confirmation-waiting approach is also less than 0.1% in the presence of an adversary with 10% of the global hash power [91].

[5]The price is from Sep. 2020.

Table 5.6: Cost for different operations in Ethereum-based BFastPay.

Operations	Pre-operations	Ethereum tx (BFPayReq)	Ethereum tx (NonPaymentProof)	Ethereum tx (PaymentChallenge[1])	Ethereum tx (PaymentChallenge[2])	Ethereum tx (send deposit)
Cost (gas)	2.46×10^6	2.78×10^5	1.04×10^5	1.32×10^5	8.08×10^4	6.60×10^4
Cost fee in ETH	4.9×10^{-3}	5.56×10^{-4}	2.08×10^{-4}	2.54×10^{-4}	1.62×10^{-4}	1.32×10^{-4}
Cost in $ (300$/ETH)	$1.47	$0.167	$0.0624	$0.0762	$0.0486	$0.0396
1: PaymentChallenge contains both the Merkle proof and the block header proof. 2: PaymentChallenge contains only the Merkle proof.						

Table 5.7: EOS token needed to stake for different operations in EOSIO-based BFastPay.

Operations	Pre-operations	EOSIO tx (BFPayReq)	EOSIO tx (NonPaymentProof)	EOSIO tx (PaymentChallenge[1])	EOSIO tx (PaymentChallenge[2])	EOSIO tx (send deposit)
EOS needed	3.6	0.5	1.1	1.6	0.8	0.1
1: PaymentChallenge contains both the Merkle proof and the block header proof. 2: PaymentChallenge contains only the Merkle proof.						

tx (send BFPayReq), (2) Ethereum tx (send NonPaymentProof), and (3) Ethereum tx (send PaymentChallenge). The total operation cost per Bitcoin transaction is 1.02×10^{-3} ETH (or $0.306, 300 $/ETH). If the payee wins, the cost additionally includes (4) Ethereum tx (transfer security deposit). The total operation cost per Bitcoin transaction is 1.12×10^{-3} ETH (or $0.336, 300 $/ETH).

In practice, case 1 (no dispute arises) is more frequent than case 2 (dispute arises) because there are no attacks in the vast majority of real-world Bitcoin payments. In summary, the operation cost of Ethereum-based BFastPay is low.

EOSIO-based BFastPay Evaluation. The EOSIO blockchain allocates blockchain resources based on the amount of EOS token staked. We set the use frequency of BFastPay service to be up to 10 times per day. Table 5.7 summarizes the EOS token needed to stake for different operations. It shows that the user needs to stake at most 7.7 EOS (or $38.5, 5$/EOS) to use 10 times of BFastPay service per day no matter whether there is a dispute or not during the BFastPay service life cycle. Because the deployer can revoke the smart contract to reclaim the staked EOS token later, we treat the EOSIO-based BFastPay service to be free of charge.

5.9 Discussions

We discuss two issues below.

Why not directly use other tokens? Because Bitcoin has dominated in practical usage [1], our

goal is to develop a solution to support fast Bitcoin payment while keeping Bitcoin as the major payment currency. *Note that in the vast majority of cases (both parties are honest and never launch attacks), the payer can enjoy a one-time deposit and permanent BFastPay fast payment services.* Thus, BFastPay is different from the solution that requires users to exchange Bitcoin to other tokens every time before using the fast payment.

How to mitigate the online requirement issue? After the payment, the payer agent and payee agent can also delegate the arbitration operations to an always-online cloud server. This solution can mitigate the online requirement issue.

Chapter 6

SecEQP: Secure kNN Queries Processing Scheme

6.1 Introduction

6.1.1 Background and Motivation

In location-based services, a user sends his/her current location to a location service provider, and the service provider then responds the user with the query results (such as the top five nearest restaurants). For lower cost, higher performance, and better flexibility, location service providers often host their geospatial data on public clouds. However, in this service model, security and privacy are major concerns as public clouds are typically not fully trusted. The confidential geospatial data and querier location information may be leaked or inferred by the cloud service providers. These storage clouds may have financial incentives (e.g., delivering advertisements to users) to collect or infer their customer sensitive information by analyzing the stored data and user queries. Moreover, these public storage clouds may be compromised and all of the stored information is further leaked by hackers. For example, it is reported that Dropbox is hacked and more than 68 million Dropbox account information is now for sale on the DarkNet marketplace [5].

In this paper, we focus on secure k nearest neighbor (SkNN) query. The location-based kNN search is one of the most widely used location-based services. The state-of-the-art solutions are either not sufficiently efficient or non-strong-provable-secure to perform the location-based kNN searches over the encrypted geospatial data on cloud. Therefore, it is crucial to develop a scheme that provides strong provable security against the untrusted clouds, while still preserving the cloud's ability to efficiently perform location-based kNN queries over the encrypted geospatial data.

6.1.2 Problem Formulation

• **Threat Model.** We consider a service model which consists of a data owner, a cloud, and multiple users. The data owner will store the geospatial data on the cloud. The cloud will serve the users' location-based queries. The adversary we consider is the cloud, which is assumed to be honest-but-curious. More specifically, the cloud provides reliable data and query services as the protocol specification, but it is curious about data it stores and queries it receives. Therefore, to protect data privacy and users' location privacy, the data owner needs to encrypt data before outsourcing and the data users need to encrypt the queries before submitting to the cloud.

• **Geospatial Data.** We consider that the data owner stores geospatial data items. Each data item consists of spatial information (e.g., the location of a restaurant) and non-spatial information (e.g., the rating of a restaurant). Data items can be represented and indexed by their spatial information. Formally, they are represented by points $p_1, \cdots, p_n$ in the two-dimensional geographical space.

• **Approximate kNN.** The secure kNN problem is modeled as how the cloud finds the top-k nearest points of $q \in U$ given by a user, as well as provides both the data owner and the user with the security guarantee. It should be ensured that the honest-but-curious cloud cannot deduce any useful information from the data it stores. Meanwhile, when the data user submits its current location to the cloud to launch a kNN query, the honest-but-curious cloud cannot learn the data user's location. In SecEQP, we use the Euclidean distance as the distance metric. To reduce query latency, SecEQP does not aim to discover strict accurate results but acceptable approximate results (e.g., the error is limited to 10%). Note that an approximate answer of kNN with a small error is still very useful in some use scenarios. For example, a user wants to find the top five nearest restaurants within 1 km for lunch. SecEQP may return five nearest restaurants within 1.1 km. The approximate results can still help the user to find a nearby restaurant (s)he likes.

6.1.3 Our Approach: SecEQP

We propose SecEQP scheme to address the aforementioned secure kNN problem. The service model and design goals of SecEQP scheme are elaborated below.

• **Service Model.** The proposed SecEQP service model is depicted in Figure 6.1. In SecEQP scheme, data owner delegates the query service to authenticated data users by sharing the secret

keys with them. Each Geospatial data item hosted by the data owner consists of location information (spatial attributes) and other information (non-spatial attributes). In order to preserve the ability to query and retrieve the data efficiently, the data owner extracts the spatial attributes of each data item and builds a secure index and then encrypts the entire data items by using the shared keys. Because queries are processed on the secure index, the data items can be encrypted by any encryption algorithms including the standard encryption algorithms with strongest security assurance (e.g., AES). Each secure index item should contain the identifier information (i.e., a pointer) to record the association between the secure index item and the encrypted data item. Afterward, the data owner outsources both the secure index items and the encrypted data items to the powerful cloud, which provides both storage and search services. After the cloud receives the secure index and encrypted data items, the authorized data users can use the shared keys to generate valid search tokens and search for the corresponding SkNN results.

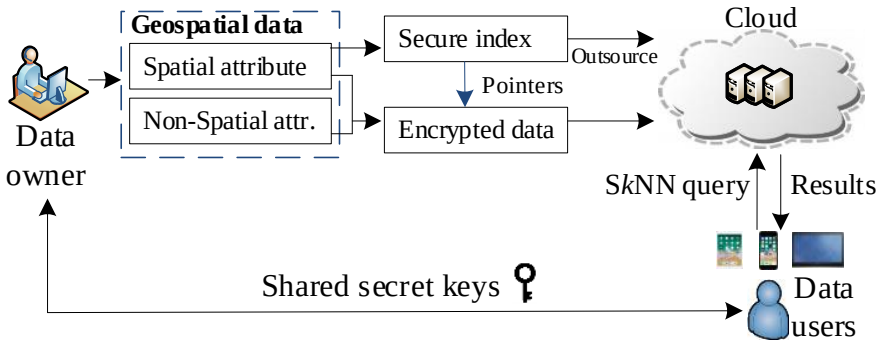


Figure 6.1: SecEQP Service Model.

• **Design Goals.** There are three design goals: security, efficiency, and accuracy, which are described in detail as follows.

- *Security.* SecEQP should preserve the following three types of privacy. (1) Data privacy: from the encrypted data items, the adversary cannot reveal any useful information about the data. (2) Index privacy: from the secure index, the adversary cannot learn any useful information about the spatial information of the data items. (3) Token privacy: from the encrypted search token, the adversary cannot infer any information about the query point's location.

- *Efficiency.* SecEQP should satisfy two types of efficiency requirements. (1) Low query

latency: the data user can get the result within a reasonable amount of time. (2) Low interaction: the protocol should be non-interactive, or it just requires a small constant number of interactions between the data user and the cloud server.

- *Accuracy.* Let o_i be the ith nearest point returned by SecEQP scheme, and let o_i^* be the ground truth, i.e., the actual ith nearest point. We can compute their distances between the query point q, denoted as $\|q, o_i\|$ and $\|q, o_i^*\|$, respectively. SecEQP should keep that $\|o_i, q\|$ is as close as possible to $\|o_i^*, q\|$ (for all $i = 1, \cdots, k$). A formal definition of accuracy metric can be found in Section 6.6.

6.1.4 Challenges and Solutions

There are three technical challenges SecEQP shall deal with.

• **C1:** *How to achieve strong data privacy while still supporting efficient kNN query processing?* To measure the proximity of two encrypted points, the straightforward approach is to compute their distance over the encrypted data. A dilemma arises: on one hand, to ensure low query latency requires the data is weakly encrypted (e.g., using order-preserving encryption); on the other hand, if strong encryption (e.g., fully homomorphic encryption (FHE) [59]) is used, the query latency will be prohibitively long. To address the dilemma, we propose the projection-based space encoding method to build a secure index. In SecEQP, the geospatial data can be formally encrypted by standard encryption methods to achieve strong data privacy (e.g., CPA-secure [71]). The secure index enables SecEQP to circumvent heavy computation over encrypted data while still supports secure kNN query processing.

• **C2:** *How to design a secure index for sublinear query latency while preserving the index privacy?* Building a secure index is not enough for secure kNN query processing. Without any index optimization, the cloud may linearly scan each encrypted data item in the database to evaluate its distance with the queried location. The linear query latency is prohibitively slow for a large dataset (e.g., a million locations are stored in the cloud). To tackle this challenge, we first propose the prefix-free encoding technique to turn the kNN query processing problem to be the keywords query problem. Then, we exploit the indistinguishable Bloom filter (IBF) tree data structure for the secure index building, which can ensure the protocol to be secure and sublinear.

• **C3:** *How to develop effective strategies to improve the result accuracy of SecEQP?* SecEQP can only return approximate query results. How to satisfy the high query result accuracy demands is not an easy task. In order to solve this problem, we leverage the observed successive inclusion property to develop an effective strategy to improve the result accuracy.

6.1.5 Comparison with Prior Art

We compare our proposed SecEQP with other six state-of-the-art SkNN schemes [117, 66, 119, 55, 112, 118] based on features that a secure kNN scheme is expected to satisfy, such as the support of strong security (i.e., the data privacy and users' location privacy will not be disclosed or inferred), the support of sublinear query processing time (i.e., the query running time is in $O(k \log n)$), etc. The results are summarized in Table 6.1. Among these features, the two important ones are the support of strong security and the support of sublinear query processing time. The major limitation for most of the previous secure kNN schemes is that it is hard to achieve both of them simultaneously. Wang et al.[112] proposed a secure kNN scheme based on order-preserving encryption (OPE) [32], which is a deterministic encryption scheme whose encryption function preserves numerical ordering of the plaintexts. A similar method called distance-recoverable encryption (DRE) is leveraged in [117] and [66] to support secure kNN search. The DRE enables anyone to recover the distance between two points by running a function over their encrypted data. The OPE and DRE are two cases of property-preserving encryptions, which only provide weak privacy protection. They are vulnerable to various serious attacks, as analyzed in [92]. Elmehdwi et al. [55] proposed a novel protocol over encrypted data based on a twin-cloud model [45] and Paillier cryptosystem [98]. This protocol employs too many heavy cryptographic operations, so its query latency is too long, rendering it impractical for large datasets. The private information retrieval (PIR)-based schemes [119] mainly consider how to protect query privacy but not data privacy. Besides, the inefficiency of PIR significantly increases the total search time. Yao et al. [118] designs a solution that can support secure nearest neighbor search by exploiting Voronoi diagram [95] for space partition. Voronoi-based schemes require each data user to download and maintain a copy of the large-size index locally for query processing, which seriously impedes its real-world applications. Besides, the generation of order-k Voronoi diagram for kNN is very computational intensive, as analyzed in [49].

Table 6.1: The comparison among previous schemes and SecEQP.

Features	Wong et al. [117]	Hu et al. [66]	Yi et al. [119]	Elmehdwi et al. [55]	Wang et al.[112]	Yao et al. [118]	**SecEQP**
Strong security	×	×	×	√	×	√	√
Sublinear query latency	×	√	×	×	√	√	√
Result accuracy	Accurate	Accurate	Accurate	Accurate	Accurate	Accurate	`Approximate`$^{\#}$
kNN or 1NN	k	k	k	k	k	1	k
High-dimensional data	√	√	×	√	√	×	×*
No local index	√	×	√	√	√	×	√
Single server	√	√	√	×	√	√	√
Rounds of interaction	1	$O(\log n)$	1	1	1	2	1

#: SecEQP can achieve high result accuracy by well-developed strategies.

∗: Handling data with dimensionality more than two is not required for location-based services.

Different from the previous works, SecEQP scheme can support the two most important features (i.e., strong security and sublinear query processing time). However, we would like to point out that SecEQP still has two downsides: (1) returning approximate results to a query instead of accurate ones and (2) only supporting 2-dimensional data, which may not fit all use scenarios (e.g., requiring strict accurate results or 3-dimensional data).

6.2 Related Work

The related work can be classified into five categories, which are introduced as follows.

• **Location Obfuscation Approach.** Schemes based on location obfuscation [89], and data transformation [73, 117] do not use strong standard encryption algorithm. Therefore, they suffer from weak privacy.

• **Private Information Retrieval Approach.** The Private Information Retrieval (PIR)-based solutions [119] mainly consider protecting query privacy but not data privacy. Besides, PIR-based solutions suffer from long query latency for large-scale dataset.

• **Fully Homomorphic Encryption Approach.** Fully homomorphic encryption (FHE) [59] enables cloud to perform kNN computation directly over the encrypted data. However, current FHE solutions still lack efficiency.

• **Property-preserving Encryption Approach.** Distance-recoverable encryption (DRE)-based schemes [117, 66] and Order-preserving encryption (OPE)-based SkNN schemes [117, 112] achieve weak security, as analyzed in [92].

• **Voronoi Diagram Approach.** Voronoi-based scheme [118] requires each data user to download

and maintain a copy of the large-size index locally for query processing, which seriously impedes its real-world applications.

6.3 Space Encoding

In this section, we propose the space encoding technique, which can be used to build a secure index for secure kNN query processing. In the following, we first introduce our customized primitive projection function. Then, we introduce how to encode/stipulate a searching region with infinite space by a single primitive projection function and how to encode/stipulate a searching region with finite space by projection function composition (i.e., multiple primitive projection functions). Moreover, we will introduce how to perform proximity testing between two locations by using the generated codes.

6.3.1 Projection Function Introduction

The projection function is defined as follows.

Definition 1 *(**Primitive Projection Function**) The primitive projection function $h : \mathbb{R}^2 \to \mathbb{Z}$ maps a two-dimensional vector $\vec{q}$ to an integer,*

$$h(\vec{q}) = \lfloor \frac{\vec{a} \cdot \vec{q} + b}{d} \rfloor, \tag{6.1}$$

where $\vec{a} = (\theta, r)$ denotes a two-dimensional vector in polar coordinate form, where the angle θ is chosen uniformly from the range $[0, 2\pi)$ and the radius $r = 1$. The parameter b is chosen uniformly from the range $[0, d)$.

• **Geometric Interpretation.** The primitive projection function has a simple geometric interpretation. As shown in Figure 6.2, suppose that $\vec{a}$ crosses the origin and its slope is identical with the straight line in the figure. So the projection of a point q is a point A onto the line $\vec{a}$. By viewing the vector along $\vec{a}$ as a new coordinate axis, A can be represented by its distance from the origin, i.e., $A = \vec{a} \cdot \vec{q}$. The point B is also on the line by shifting A a distance of b. Then, the straight line is divided by discrete intervals of length d. The projected value is the ID of the interval containing B. The farthest bound that B can reach is C, where $C = \vec{a} \cdot \vec{q} + d$, i.e., $B \in [A, C)$ along the line.

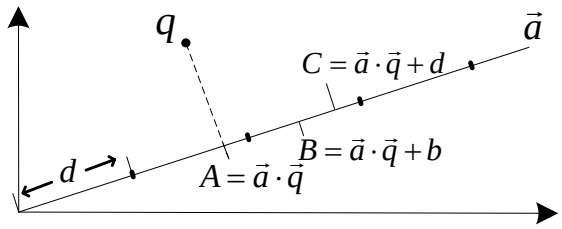


Figure 6.2: Geometric illustration of the primitive projection function.

• **Comparison with LSH.** The primitive projection function in Equation (6.1) has a similar form with locality sensitive hashing (LSH) defined in [52], where the LSH is defined to map a high-dimensional data to an integer. The parameter $\vec{a}$ is a high-dimensional vector with entries chosen independently from a p-stable distribution. The traditional usage of LSH is to reduce the dimensionality of high-dimensional data for accelerating similarity search without security considerations. Different from traditional usage, SecEQP exploits multiple primitive projection functions to project two-dimensional data to high-dimensional data (i.e., the data has a high-dimensional vector representation) for secure kNN search.

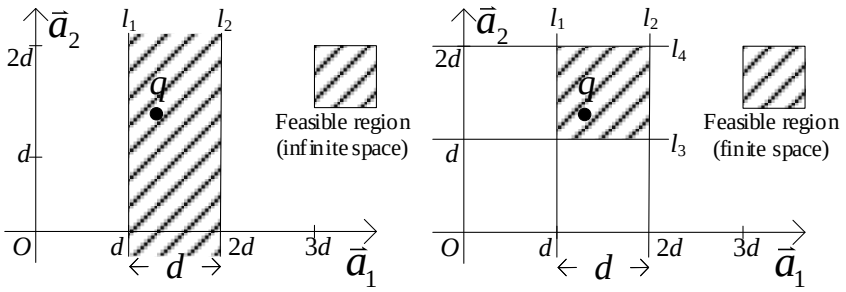
6.3.2 Space Encoding via a Single Primitive Projection Function

We now illustrate how to use a single primitive projection function to encode an infinite geometric region.

Definition 2 *(**Feasible Region**) Given a point q in the two-dimensional space and a projection function h, we define the feasible region of $\vec{q}$ with respect to the projection function h as consisting of all the possible points $\vec{p}$, such that $h(\vec{q}) = h(\vec{p})$, denoted as* $\mathbf{FR}(h(\vec{q}))$.

Figure 6.3a shows an example to illustrate the feasible region with respect to a single primitive projection function. Consider a projection function $h(\vec{q}) = \lfloor \frac{\vec{a}_1 \cdot \vec{q} + b}{d} \rfloor$ with $b = 0$. Given a point $q \in \mathbb{R}^2$, suppose that $d \leq \vec{a}_1 \cdot \vec{q} < 2d$, so we have $h(\vec{q}) = 1$. As shown in Figure 6.3a, for any point in the shadowed area will be projected to be 1. Therefore, the feasible region of q is an infinite region between two parallel lines l_1 and l_2. The distance of l_1 and l_2 is exactly d. In more general cases, the properties of the feasible region are identified by the following Theorem.

Theorem 1 *Given a point $q \in \mathbb{R}^2$, the feasible region of q is between two parallel lines l_1 and l_2 that are perpendicular to $\vec{a}$, as shown in Figure 6.3a. Define the width of the feasible region wid as*



(a) The feasible region with respect to a single primitive projection function.

(b) The feasible region with respect to two orthogonal projection functions.

Figure 6.3: Two examples to illustrate the feasible region.

the distance of l_1 and l_2, we have wid $= d$, which is independent of the location of q and the choice of b.

The projected code value enables us to test whether two points p, q are in the same infinite d-width space by checking $h(\vec{q}) \stackrel{?}{=} h(\vec{p})$. For example, as shown in Figure 6.3a, if $h(\vec{q}) = h(\vec{p})$, then point q must locate in the feasible region (shadowed area). Note that the proximity testing by using a single projected code is not accurate because the encoded space is infinite. Two far away points may have the same projected code.

6.3.3 Projection Function Composition Introduction

We use two kinds of compositions: AND-composition and OR-composition, which are defined as follows.

Definition 3 *(**AND-composition** and **OR-composition**)*

- *AND-composition: Consider there are v projection functions $h_1, \cdots, h_v$. A new composite projection function g can be constructed as the AND-composition of them, denoted as $g = AND(h_1, \cdots, h_v)$.* Equal criterion*: given any two points q and p, $g(\vec{q}) = g(\vec{p})$ if and only if $h_i(\vec{q}) = h_i(\vec{p})$ for all $i \in [v]$, where $[v]$ denotes the set $\{1, \cdots, v\}$.*

- *OR-composition: Consider there are t projection functions $h_1, \cdots, h_t$. A new composite projection function g can be constructed as the OR-composition of them, denoted as $g = OR(h_1, \cdots, h_t)$.* Equal criterion*: given any two points q and p, $g(\vec{q}) = g(\vec{p})$ if and only if $h_i(\vec{q}) = h_i(\vec{p})$ for at least one $i \in [t]$, where $[t]$ denotes the set $\{1, \cdots, t\}$.*

The outputs of a composite projection function can be represented in many ways (e.g., it can be represented by a hierarchical table, as shown in Table 6.2). We give the following example to illustrate how to determine if two composite projection functions equal or not.

• **An Example to Illustrate Equal Criterion.** Table 6.2 shows an example to illustrate equal criterion for the composite projection function. Consider there are 4 primitive projection functions $h_{1,1,1}, h_{1,1,2}, h_{1,2,1}, h_{1,2,2}$. Suppose that $g_{1,1}$ is constructed by AND-composition of $h_{1,1,1}, h_{1,1,2}$ denoted as $g_{1,1} = AND(h_{1,1,1}, h_{1,1,2})$. Likewise, suppose that $g_{1,2}$ is AND-composition of $h_{1,2,1}, h_{1,2,2}$ denoted as $g_{1,2} = AND(h_{1,2,1}, h_{1,2,2})$. Let f_1 be constructed by OR-composition of $g_{1,1}, g_{1,2}$, i.e., $f_1 = OR(g_{1,1}, g_{1,2})$. In the example, given two points q and p, since $h_{1,1,1}(\vec{q}) = h_{1,1,1}(\vec{p}), h_{1,1,2}(\vec{q}) = h_{1,1,2}(\vec{p})$, we have $g_{1,1}(\vec{q}) = g_{1,1}(\vec{p})$. Because $h_{1,2,1}(\vec{q}) \neq h_{1,2,1}(\vec{p})$, we have $g_{1,2}(\vec{q}) \neq g_{1,2}(\vec{p})$. Moreover, it holds that $f_1(\vec{q}) = f_1(\vec{p})$, because either $g_{1,1}(\vec{q}) = g_{1,1}(\vec{p})$ or $g_{1,2}(\vec{q}) = g_{1,2}(\vec{p})$ will lead to $f_1(\vec{q}) = f_1(\vec{p})$.

Table 6.2: An example to illustrate equal criterion.

point q	$f_1(\vec{q}) = OR(g_{1,1}(\vec{q}), g_{1,2}(\vec{q}))$			
	$g_{1,1}(\vec{q}) = AND(h_{1,1,1}(\vec{q}), h_{1,1,2}(\vec{q}))$		$g_{1,2}(\vec{q}) = AND(h_{1,2,1}(\vec{q}), h_{1,2,2}(\vec{q}))$	
	$h_{1,1,1}(\vec{q}) = 1$	$h_{1,1,2}(\vec{q}) = 2$	$h_{1,2,1}(\vec{q}) = 1$	$h_{1,2,2}(\vec{q}) = 2$
point p	$f_1(\vec{p}) = OR(g_{1,1}(\vec{p}), g_{1,2}(\vec{p}))$			
	$g_{1,1}(\vec{p}) = AND(h_{1,1,1}(\vec{p}), h_{1,1,2}(\vec{p}))$		$g_{1,2}(\vec{p}) = AND(h_{1,2,1}(\vec{p}), h_{1,2,1}(\vec{p}))$	
	$h_{1,1,1}(\vec{p}) = 1$	$h_{1,1,2}(\vec{p}) = 2$	$h_{1,2,1}(\vec{p}) = 3$	$h_{1,2,2}(\vec{p}) = 4$

6.3.4 Space Encoding via Projection Function Composition

In this section, we illustrate how to use projection function composition to encode a finite space.

• **Space Encoding by only AND-composition.** Given a point $q \in \mathbb{R}^2$, we now study the feasible region of q with respect to a projection function g, where g is AND-composition of v primitive projection functions. Taking the simplest case $v = 2$ as an example, let

$$g(\vec{q}) = AND(h_1(\vec{q}), h_2(\vec{q})), \tag{6.2}$$

where $h_1(\vec{q}) = \lfloor \frac{\vec{a}_1 \cdot \vec{q} + b_1}{d} \rfloor$ and $h_2(\vec{q}) = \lfloor \frac{\vec{a}_2 \cdot \vec{q} + b_2}{d} \rfloor$, $b_1 = b_2 = 0$, $\vec{a}_1$ and $\vec{a}_2$ are orthogonal vectors

(i.e., $\vec{a}_1 \perp \vec{a}_2$), $d \leq \vec{a}_1 \cdot \vec{q} < 2d$, and $d \leq \vec{a}_1 \cdot \vec{q} < 2d$. As shown in Figure 6.3b, the feasible region of q with respect to h_1 is an infinite region between l_1 and l_2. Likewise, the feasible region of q with respect to h_2 is an infinite region between l_3 and l_4. Therefore, the feasible region of q with respect to $g = AND(h_1, h_2)$ is the intersection region (i.e., a d-width square), as shown in Figure 6.3b (shadowed area).

• **Space Encoding by first AND-composition and then OR-composition.** Given a point $q \in \mathbb{R}^2$, where is the feasible region of q with respect to a projection function which is constructed by first AND-composition and then OR-composition? We consider there are two projection functions $g_1(\vec{q})$ and $g_2(\vec{q})$, which are constructed by AND-composition in the same way as Equation (6.2). Let $f = OR(g_1, g_2)$. Because each of $g_1(\vec{q})$ and $g_2(\vec{q})$ specifies a square feasible region of q as shown in Figure 6.3b. Therefore, the feasible region of q with respect to f is the union of two d-width square feasible regions. For instance, Figure 6.4 shows feasible regions generated by the union of three square feasible regions with different choices of parameter d.

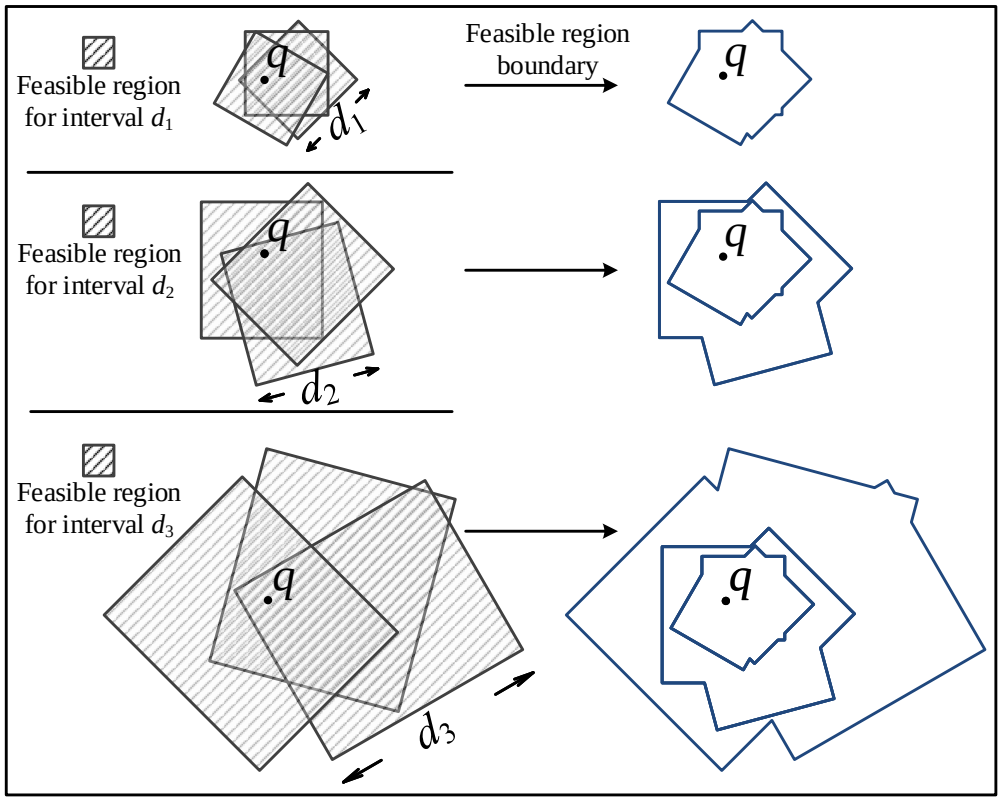


Figure 6.4: An example to illustrate successive inclusion property ($\mathbf{FR}(f_1(q)) \subset \mathbf{FR}(f_2(q)) \subset \mathbf{FR}(f_3(q))$).

In analogy to the single primitive projection function-based space encoding, the composite projection function code values enable us to perform proximity testing over a finite two-dimensional space. More concretely, we can test whether a point p is in the feasible region $\mathbf{FR}(f(\vec{q}))$ of q by checking $f(\vec{q}) \stackrel{?}{=} f(\vec{p})$. The proximity testing over a finite space can get a much more accurate result than proximity testing over an infinite space.

6.4 kNN Protocol for Plaintext Domain

In this section, we describe how to process kNN queries in plaintext domain (i.e., no data encryption is enforced) and then elaborate on how to transform it to secure kNN protocol in Section 6.5.

Our kNN protocol design is developed on the top of an essential property: successive inclusion property. It can be used to generate a series of gradually enlarged feasible regions for a point. The cloud can search from the smallest feasible region to the largest one and gradually find the k nearest points. In the following, we will introduce the successive inclusion property, present our kNN protocol design, and discuss two critical parameters used in our protocol.

• **Successive Inclusion Property.** We construct three projection functions f_1, f_2, f_3, each of which is constructed by first AND-composition of two primitive projection functions (as in Equation (6.2)) and then three OR-composition. In the construction of f_1, f_2, and f_3, three parameters d_1, d_2, and d_3 are used to generate the corresponding primitive projection function, respectively. Suppose that $d_1 < d_2 < d_3$, Figure 6.4 shows an example of the feasible regions of q with respect to f_1, f_2, and f_3, respectively. It is shown in Figure 6.4 that $\mathbf{FR}(f_1(\vec{q})) \subset \mathbf{FR}(f_2(\vec{q})) \subset \mathbf{FR}(f_3(\vec{q}))$. In general, consider there is a series of composite projection function $f_1, \cdots, f_L$ (with the same first AND-composition and then OR-composite patterns), which are constructed by a series of interval lengths $(d_1, \cdots, d_L)$ (with $d_1 < \cdots < d_L$), respectively. If the gap between two successive values in $d_1, \cdots, d_L$ is sufficiently large, it holds that

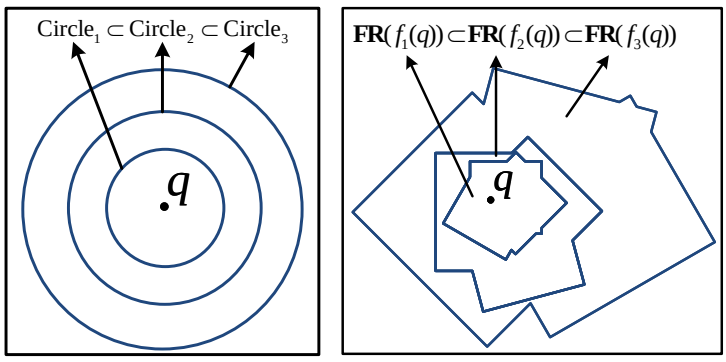
$$\mathbf{FR}(f_1(\vec{q})) \subset \mathbf{FR}(f_2(\vec{q})) \subset \cdots \subset \mathbf{FR}(f_L(\vec{q})). \tag{6.3}$$

We call the property exhibited in Formula (6.3) as successive inclusion property.

6.4.1 *k*NN Protocol Design

We next elaborate on our *k*NN protocol high-level design rationale and present its main algorithms in detail.

• **High-level Design Rationale.** Consider the service model as depicted in Figure 6.1. The data owner hosts a dataset of n data items in plaintext, then (s)he extracts the spatial attributes, denoted as $p_1, \cdots, p_n$, to build an index for *k*NN search. The data owner chooses a series of composite projection functions with successive increasing interval lengths $(d_1, \cdots, d_L)$. Given composite projection functions and a data point p_i, the data owner computes a series of feasible regions with successive increasing interval lengths $(d_1, \cdots, d_L)$. Each feasible region is represented by its composite projection function codes, which is outsourced to the cloud to serve as the index. For a query point q, the data user computes a series of the above chosen composite projection functions outputs, and send the codes to cloud for results. Upon reception of the query from the data user, the cloud evaluates the proximity of q and p_i by comparing whether their corresponding composite projection function output codes equal. The cloud searches from the smallest feasible region of q to the largest one until k points are found. Figure 6.5b shows an example of three feasible regions of q that satisfy the successive inclusion property (i.e., $\mathbf{FR}(f_1(\vec{q})) \subset \mathbf{FR}(f_2(\vec{q})) \subset \mathbf{FR}(f_3(\vec{q}))$). Each feasible region is generated by first two AND-composition and then three OR-composition, as shown in Figure 6.4. For the query point q, the cloud searches from the smallest feasible region $\mathbf{FR}(f_1(\vec{q}))$ to the largest feasible region $\mathbf{FR}(f_3(\vec{q}))$ to gradually find the closest points.



(a) Circle-based accurate *k*NN search process.

(b) Projection-based approximate *k*NN search process.

Figure 6.5: Comparison between accurate and approximate *k*NN search process.

• ***k*NN Protocol in Detail.** The *k*NN protocol design involves in choosing a group of composite projection functions. We first define the following mathematical notations to facilitate our description. Then, we describe *k*NN protocol in detail.

Definition 4 ***(Projection Function Family)***

- $\mathscr{H}_{1,1}^{d_i}$*: We define* $\mathscr{H}_{1,1}^{d_i}$ *to be the primitive projection function family which contains all of primitive projection functions generated by Equation (6.1) (with* $d = d_i$*). Let* $h \leftarrow \mathscr{H}_{1,1}^{d_i}$ *be the process of randomly sampling a projection function* h *from* $\mathscr{H}_{1,1}^{d_i}$*, where the randomness comes from the random choices of the vector* $\vec{a}$ *and* b *in Equation (6.1).*

- $\mathscr{H}_{v,1}^{d_i}$ *: We define* $\mathscr{H}_{v,1}^{d_i}$ *to be AND-composite projection function family which contains all of composite projection functions generated by the AND-composition of* v *randomly chosen primitive projection functions* $h_1, \cdots, h_v$*, where* $h_i \leftarrow \mathscr{H}_{1,1}^{d_i}$ *for all* $i \in [v]$*. Let* $g \leftarrow \mathscr{H}_{v,1}^{d_i}$ *be the process of randomly sampling a composite projection function* g *from* $\mathscr{H}_{v,1}^{d_i}$*.*

- $\mathscr{H}_{v,t}^{d_i}$*: We define* $\mathscr{H}_{v,t}^{d_i}$ *to be the Or-composite projection function family which contains all of composite projection functions generated by the OR-composition of* t *randomly chosen AND-composite projection functions* $g_1, \cdots, g_t$*, where* $g_i \leftarrow \mathscr{H}_{v,1}^{d_i}$ *for all* $i \in [t]$*. Let* $f \leftarrow \mathscr{H}_{v,t}^{d_i}$ *be the process of randomly sampling a composite projection function* f *from* $\mathscr{H}_{v,t}^{d_i}$*.*

With the above notations, the proposed *k*NN protocol is described as follows. First, the data owner setups several global parameters including v, t, and L successive increasing interval lengths $(d_1, \cdots d_L)$. Second, the data owner invokes Algorithm 6.1 (`Index-Building`) to compute and store the projection function output values of each point in the index matrix $\mathbb{I}'$. Afterward, the index $\mathbb{I}'$ is sent to the cloud for storage. Then, the data user calls the Algorithm 6.2 (`Token-Generation`) to compute and store the projection function output values of the query point in the token array $\mathbb{T}'$. Recall that whether two points are in the same feasible region or not can be deduced by comparing their projection function output values, the cloud calls Algorithm 6.3 (`Query-Processing`) to check whether $p_i (i \in [n])$ is in the smallest feasible region $\mathbf{FR}(f_1(\vec{q}))$ of query point q via checking $f_1(\vec{q}) \stackrel{?}{=} f_1(\vec{p}_i)$ (i.e., $\mathbb{T}'(1) \stackrel{?}{=} \mathbb{I}'(i,1)$) (step 3 in Algorithm 6.3). According to the successive inclusion property, the cloud searches from the smallest feasible region of q (i.e., $\mathbf{FR}(f_1(\vec{q}))$) to the largest one (i.e., $\mathbf{FR}(f_L(\vec{q}))$) and stops until at least k distinct points

are found. Suppose that the search stops when $k'(k \geq k)$ points are found, the data user computes their accurate distance to the query point q and sorts them to figure out the top-k closest points as the query results.

Algorithm 6.1: Index-Building

Input: $v, t, L, (d_1, \cdots, d_L), p_1, \cdots, p_n$
Output: $\mathbb{I}'$
for $(i = 1; i \leq L; i++)$ **do**
  $f_i \leftarrow \mathscr{H}_{v,t}^{d_i}$;
for $(i = 1; i \leq n; i++)$ **do**
  for $(j = 1; j \leq L; j++)$ **do**
    compute $\mathbb{I}'(i, j) = f_j(p_i)$;
    /* $\mathbb{I}'(i, j)$ represents the composite projection function output values for data point p_i with $d = d_j$ */

Algorithm 6.2: Token-Generation

Input: q and $f_1, \cdots, f_L$
Output: $\mathbb{T}'$
for $(j = 1; j \leq L; j++)$ **do**
  compute $\mathbb{T}'(j) = f_j(q)$;
  /* $\mathbb{T}'(j)$ represents the composite projection function output values for query point q with $d = d_j$ */

6.4.2 Analysis of kNN Protocol Parameters

In our kNN protocol, there are two critical parameters: v (the number of AND-composition) and t (the number of OR-composition). We discuss how they influence the performance of kNN protocol as follows.

- **Geometric Analysis of v.** Setting up a larger v implies an improvement of the proximity measurement precision. In order to precisely measure the proximity between two points in two-dimensional space, it is desirable that points p and q are projected to more random directions on the two-dimensional plane and then compare their projection function output values. As a result, a larger v implies an improvement of the proximity measurement precision.

Algorithm 6.3: Query-Processing

Input: k, L, $\mathbb{I}'$, $\mathbb{T}'$ and $p_1, \cdots, p_n$
Output: $\mathbb{R}'$
/* $\mathbb{R}'$ represents the set of returned points */
Initialization: $\mathbb{R}' = Null$; $i = j = 1$; $result\_num = 0$;
while $(result\_num < k$ && $j \leq L)$ **do**
  if (Is_equal$(\mathbb{T}'(j), \mathbb{I}'(i,j)) ==$ True) && $(p_i \notin \mathbb{R}')$ /* search for the data point p_i in $\mathbf{FR}(f_j(\vec{q}))$ */
  then
    $\mathbb{R}' = \mathbb{R}' \cup p_j$, $result\_num++$;
  if $(i == n)$ /* if $\mathbf{FR}(f_j(\vec{q}))$ have been searched, then search in $\mathbf{FR}(f_{j+1}(\vec{q}))$ */
  then
    $j++$, $i = 1$;
  else
    $i++$;
    /* search for the next data point p_{i+1} in $\mathbf{FR}(f_j(\vec{q}))$ */

- **Geometric Analysis of t.** Setting up a larger t implies an improvement of the result accuracy. For the query point q, if the cloud searches from a series of concentric circle regions (centered at the point q), then the cloud always gets the accurate results. Figure 6.5a shows an example of the ideal accurate kNN search process. For the query point q, the cloud first searches from the smallest circle $\texttt{Circle}_1$ to the largest circle $\texttt{Circle}_3$ to gradually find the closest points. In comparison, Figure 6.5b shows the projection-based approximate kNN search process. In order to increase the result accuracy, it is desirable that the feasible regions are close to circles with point q at the center. Figure 6.6 shows that increasing the number of OR-composition t can make the feasible region closer to a circle. Therefore, a larger t implies an improvement of the result accuracy.

• **An Optimization in Projection Function Generation.** In the AND-composite function g generation, SecEQP chooses t random primitive projection functions with t random directions for a point to project. In order to better measure the proximity of two points p, q in the space, it is expected that p, q are projected in many different directions over the space. The difference between two directions $\vec{a}_1$ and $\vec{a}_2$ can be represented by their angle, denoted as $\widehat{\vec{a}_1, \vec{a}_2}$. Random projection direction choices may lead to many pairs of similar directions (e.g., $\widehat{\vec{a}_1, \vec{a}_2}$ is very small). To in-

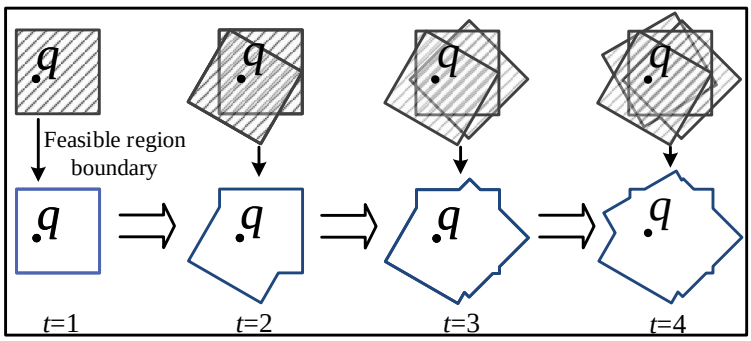


Figure 6.6: The feasible region is getting closer to a circle by increasing the number of OR-composition t.

crease the difference between projected directions, we refine our direction choice process by first choosing a random direction $\vec{a}_1 = (\theta, 1)$, and then choose the remaining $v-1$ vectors to equally divide the space. For example, if there are three chosen directions $\vec{a}_1, \vec{a}_2$, and $\vec{a}_3$, an optimized solution is to keep $(\widehat{\vec{a}_1, \vec{a}_2}) = (\widehat{\vec{a}_2, \vec{a}_3}) = \pi/3$.

6.5 Transforming *k*NN to Secure *k*NN

In this section, we describe how to transform the above *k*NN protocol to be a secure and sublinear protocol. Our approach is to leverage searchable symmetric encryption (SSE) for keyword query [50]. It allows data users to have secure keyword query processing on the cloud. In order to harness SSE, there are three technical issues need to be addressed.

- *How to generate keywords?* Our solution is to design a prefix-free encoding method (§ 6.5.1) to encode the projection function output values to generate keywords.
- *How to perform search operations over the index?* Our solution is to do the operation transformation (§ 6.5.2) which transforms the `Is_equal` evaluation in Algorithm 6.3 to be `Is_exist` evaluation.
- *How to build a secure index?* Our solution is to use the indistinguishable Bloom filter (IBF) tree based secure index (§ 6.5.3) which provides the sublinear search time as well as a strong security guarantee.

In the following, we will elaborate on our remedies to the above technical issues in detail and finally present how to apply them to the SkNN protocol (i.e., SecEQP) design (§ 6.5.4).

6.5.1 Prefix-free Encoding

In Algorithm 6.3 (Query-Processing), for two points p and q, in order to know whether p locates in the feasible region $\mathbf{FR}(f_i(\vec{q}))$ of q, we need to evaluate the logic expression

$$\texttt{Is\_equal}(f_i(\vec{q}), f_i(\vec{p})), \tag{6.4}$$

where $f_i = OR(g_{i,1}, \cdots, g_{i,t})$. Then, the logic expression (6.4) can be translated to

$$\texttt{Is\_equal}(g_{i,1}(\vec{q}), g_{i,1}(\vec{p})) \vee \cdots \vee \texttt{Is\_equal}(g_{i,t}(\vec{q}), g_{i,t}(p)). \tag{6.5}$$

Consider $g_{i,j} = AND(h_{i,j,1}, \cdots, h_{i,j,v})$, we let $str(g_{i,j}(\vec{q})) = h_{i,j,1}(\vec{q}))||\cdots||h_{i,j,v}(\vec{q}))$, where "||" denotes the string concatenation. In order to circumvent `Is_equal` evaluation, we construct two sets by prefix encoding as

$$\begin{aligned} Q_i &= \{i||1||str(g_{i,1}(\vec{q})), \cdots, i||t||str(g_{i,t}(\vec{q}))\}, \\ P_i &= \{i||1||str(g_{i,1}(\vec{p})), \cdots, i||t||str(g_{i,t}(\vec{p}))\}. \end{aligned} \tag{6.6}$$

For each component in the coding, we reserve a fix number of bit to ensure that the code is prefix-free. The prefix-free encoding ensures if one element in the set Q_i equals to another element in set P_i, then their each coding component must equal to each other. For example, suppose that $h_{1,1,1}(\vec{q}) = 1, h_{1,1,2}(\vec{q}) = 11, h_{1,1,1}(\vec{p}) = 11$, and $h_{1,1,2}(\vec{p}) = 1$, a direct encoding will lead to $1||1||str(g_{i,j}(\vec{q})) =$ "1" + "1" + "11" + "1" = "11111" and $1||1||str(g_{i,j}(\vec{p}))$ = "1" + "1" + "1" + "11" = "11111". This leads to $str(g_{i,j}(\vec{q})) = str(g_{i,j}(\vec{p}))$ despite $g_{i,j}(\vec{q}) \neq g_{i,j}(\vec{p})$. However, if we fix 2 digits to encode each component, then we have $01||01||str(g_{i,j}(\vec{q}))$ = "01" + "01" + "11" + "01" = "01011101" and $01||01||str(g_{i,j}(\vec{p}))$ = "01" + "01" + "01" + "11" = "01010111", so $str(g_{i,j}(\vec{q})) \neq str(g_{i,j}(\vec{p}))$. Therefore, prefix-free encoding preserves the equal relationship after coding. In the above example, we choose 2 digits for each coding component. However, in real applications, the data owner should choose a number which is not less than the maximum number of digits for each coding component.

6.5.2 Operation Transformation

With the prefix-free encoding, the following Theorem holds immediately.

Theorem 2 *Logic expression (6.4) and (6.5) are* True $\Longleftrightarrow Q_i \cap P_i \neq \emptyset$*, where "*$\Longleftrightarrow$*" denotes logical equivalence.*

Let us reuse the settings in Table 6.2 as an example. According to prefix-free encoding described in Equation (6.6), we have $Q_1 = \{01010102, 01020102\}$ and $P_1 = \{01010102, 01020304\}$, where we fix 2 bits for each component in coding. Because $Q_1 \cap P_1 = \{01010102\} \neq \emptyset$, we have $f_1(\vec{q}) = f_1(\vec{p})$. Based on Theorem 2, we can employ `Is_exist` evaluation to replace `Is_equal` evaluation. That is, we can know whether query point p is located in the feasible region $\mathbf{FR}(f_i(\vec{q}))$ of q by checking $Q_i \cap P_i \stackrel{?}{=} \emptyset$. In order to check $Q_i \cap P_i \stackrel{?}{=} \emptyset$, we can traverse every element in Q_i and then test whether it exists in P_i.

6.5.3 Indistinguishable Bloom Filter Tree based Secure Index

The secure index used in SecEQP is built based on a data structure called indistinguishable Bloom filter (IBF) tree. In the following, we will provide the primer of indistinguishable bloom filter and then introduce how to construct an IBF tree for the secure index. Finally, we will discuss why IBF-based index is secure and efficient.

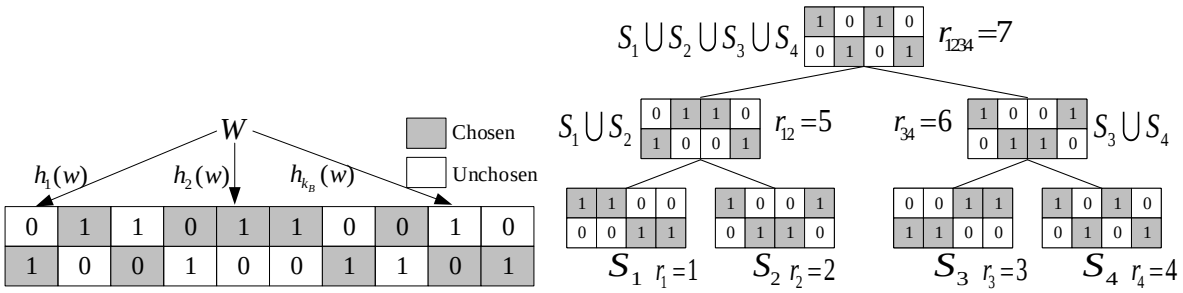
• **Indistinguishable Bloom Filter.** The indistinguishable Bloom filter (IBF) is a data structure that is extended from Bloom filter [43]. It can be used to test whether an element is a member of a set or not. IBF is defined as follows.

Definition 5 *(IBF [81]) An IBF is an array B of m twins,* k_B *different hash functions* h_1*,* h_2*,* $\cdots$*,* h_{k_B}*, and a random oracle H. Each twin consists of two cells where each cell stores either 0 or 1 and the two cells should be different. The two cells in a twin are named as 0-cell and 1-cell, respectively. For each twin, the oracle H determines which cell is chosen in a random fashion. For every twin, the chosen cell is initialized to 0 and the unchosen cell is set to 1. Given one keyword w, we hash it to* k_B *twins* $B[h_1(w)]$*,* $B[h_2(w)]$*,* $\cdots$*,* $B[h_{k_B}(w)]$*, and for each of these* k_B *twins, we set its chosen cell to 1 and the unchosen cell to 0.*

Figure 6.7a shows an example of IBF. Let us describe how to embed a keyword w_i into an IBF. We assume that the data owner and data users share $k_B + 1$ secret keys $K_1, \cdots, K_{k_B+1}$. We construct k_B hash functions using the keyed hash message authentication code (HMAC), where $h_i(\cdot) = \texttt{HMAC}_{K_i}(\cdot) \mod m$, for $i \in [k_B]$. We construct another hash function as $h_{i+1}(\cdot) = \texttt{HMAC}_{K_B+1}(\cdot)$. The random oracle is instantiated as $H(\cdot) = \texttt{SHA1}(\cdot) \mod 2$. An IBF can be viewed as a two-dimensional array B with two rows and m columns. Let $B[i][j]$ be the value in the ith row and jth column of the IBF B. To embed a keyword w_i into the IBF B, we set

$$\begin{aligned} B[H(h_{k_B+1}(h_j(w_i)) \oplus r_B)][h_j(w_i)] &= 1, \\ B[1 - H(h_{k_B+1}(h_j(w_i)) \oplus r_B)][h_j(w_i)] &= 0, \end{aligned} \tag{6.7}$$

for all $j \in [k_B]$, where r_B is a random number associated with IBF B. To test whether a keyword w_i is in the IBF B, we just need to compute the corresponding hashes and test whether the positions indicated by these hashes are all 1. If all positions are 1, then w_i is in the IBF, otherwise not.



(a) An simplified example of indistinguishable Bloom filter.

(b) An simplified example of indistinguishable Bloom filter tree.

Figure 6.7: Indistinguishable Bloom filter and indistinguishable Bloom filter tree examples.

• **Indistinguishable Bloom Filter Tree.** IBFs can be organized into a binary tree structure to achieve sublinear search time. Figure 6.7b shows an example of IBF tree. An IBF tree is constructed as follows. Suppose that B_v is the father IBF of two children IBFs: B_l (left child) and B_r (right child), then B_v is constructed as follows: for each $i \in [m]$, the value of B_v's ith twin is the logical OR of B_l's ith twin and B_r's ith twin. That is

$$\begin{aligned} &B_v[H(h_{k_B+1}(i) \oplus r_{B_v})][i] = \\ &B_l[H(h_{k_B+1}(i) \oplus r_{B_l})][i] \vee B_r[H(h_{k_B+1}(i) \oplus r_{B_r})][i]. \end{aligned} \tag{6.8}$$

By this way, the IBF tree can be constructed from a number of leaf nodes until there is one root node. As shown in Figure 6.7b, if B_l is an IBF representing set S_1 and B_r is an IBF representing set

S_2, then we have that B_v is an IBF representing set $S_1 \cup S_2$ while the random numbers, r_1, r_2, and r_{12}, for S_1, S_2, and $S_1 \cup S_2$ are 1, 2, and 5, respectively. More examples and the illustration about how to build a Bloom filter tree can be found in [60, 82]. This property enables us to perform a binary search from the root IBF in the tree to the leaf IBF to test whether a keyword is embedded in a leaf IBF in $O(\log(n))$ time.

The security intuition behind IBF tree-based index is that the positions of 0-cell and 1-cell are determined by the random oracle H, so an IBF tree is a completely random data structure consists of an equal number of 1s and 0s. Moreover, a node-specific random number r_B (see Equation (6.7)) is adopted while each IBF node is generated. With this design, even if there are two points at the same location, their IBF nodes are likely to be different unless their random numbers are equivalent. This approach thus prevents the cloud from inferring the projected values or the closeness of locations in geospatial database by analyzing their IBF nodes in the IBF tree-based index. Therefore, intuitively, the IBF tree-based index can achieve index privacy (the formal index indistinguishability proof is elaborated in Step 2 of Theorem 3).

6.5.4 SkNN Protocol (SecEQP) Design

We next introduce the SecEQP design which employs the aforementioned prefix-free encoding, operation transformation, and IBF-tree based security index techniques.

- *Index-Building.* For each data point in the database, the data owner computes its projection function output values (exactly the same process as described in Algorithm 6.1). Then, the data owner employs the prefix-encoding (according to the method described in Equation (6.6)) to generate a set of codes for each point. The set of codes are grouped by a series of sets P_i, for $i \in [L]$. Each point's codes are embedded into a distinct IBF (in the way as described by Equation (6.7)). All IBFs generated by data points now serve as the leaf nodes to construct a balanced IBF tree (in the way as described by Equation (6.8)). Each IBF node in the IBF tree is associated with a random number as shown in Equation (6.7) and Equation (6.8). The IBF tree along with the random number for each IBF node in the tree serve as the secure index, which is outsourced and stored in the cloud.

- *Token-Generation.* Given the query point q, the data user computes projection function

values and employs the prefix-encoding to generate a set of codes for each point (according to the method described in Equation (6.6)). The set of codes are grouped by a series of sets Q_i, for $i \in [L]$. The set of codes serve as a series of keywords. Note that the keywords in Q_{i_1} are put before keywords in Q_{i_2} if $i_1 < i_2$. For a keyword w_i, the data user computes k_B locations $h_j(w_i)$, for $j \in [k_B]$. For each location $h_j(w_i)$, the data user computes hash $h_{K_B+1}(h_j(w_i))$. The search token t_{w_i} of keyword w_i is a k_B-pair of hashes and locations: $\{h_{K_B+1}(h_j(w_i), h_j(w_i)\}$, for $j \in [k_B]$. The data user generates search tokens in the above way for all keywords in Q_i ($i \in [L]$) and sends these search tokens to the cloud for results. Because these hash functions are one-way, it is hard for the cloud to deduce the useful information of the query point by viewing these search tokens.

- *Query-processing.* On receipt of a search token t_{w_i} for keyword w_i from the data user, the cloud performs the query processing, which is described as follows. Let $t_{w_i}[j]$ denote the jth ordered pair in t_{w_i}, i.e., $t_{w_i}[j] = \{h_{K_B+1}(h_j(w_i), h_j(w_i))\}$. Let $t_{w_i}[j].f$ and $t_{w_i}[j].s$ be the first and second hash in $t_{w_i}[j]$, respectively. For an IBF B that the cloud checks against t_{w_i}, if there exist $j \in [k_B]$ such that $B[H(t_{w_i}[j].f \oplus r_B)][t_{w_i}[j].s] = 0$, then t_{w_i} does not match any of the items embedded in the IBF. If the cloud determines that t_{w_i} does not match the IBF B, the query processing terminates. Otherwise, the cloud processes t_{w_i} against the left and right children of the IBF B. The search begins from the root IBF until the cloud reaches the leaf IBF and get the corresponding encrypted data item. The cloud searches from the first keyword-generated tokens to the last keyword-generated tokens and stops until at least k distinct IBF leaf nodes are found. Last, the cloud returns the corresponding encrypted data item to the data user for further processing.

6.5.5 Security Analysis

In this section, we first describe the adopted security model and related notations. Then, we define leakage functions and perform security proof for SecEQP.

• *Secure Model and Notations.* We adopt the widely used adaptive indistinguishability under chosen-keyword attack (IND-CKA) secure model [50]. Let $\mathbf{D} = \{d_1, \cdots, d_n\}$ denote the set of data items. Let $\mathbb{I}$ and $\mathbf{T}$ denote the index and search token, respectively. Suppose that SecEQP employes

a CPA-secure encryption scheme [71] to encrypt each data items.

• *Leakage Functions.* Before we carry out the formal security proof, we introduce two leakage functions. (1) $\mathscr{L}_1(\mathbb{I},\mathbf{D})$: Given the index $\mathbb{I}$ and dataset $\mathbf{D}$, this function outputs the size of each IBF m, the number of data items n in $\mathbf{D}$, the data item identifiers $ID = (id_1,\cdots,id_n)$, and the size of each encrypted data item. (2) $\mathscr{L}_2(\mathbb{I},\mathbf{D},q_i)$: This function takes as input the index $\mathbb{I}$, the set of data items $\mathbf{D}$, and a query q_i. It outputs two types of information: the *search pattern*, which is the information about whether the same search was performed before or not, and the *access pattern*, which is the information about which data item identifiers that match query q_i.

Theorem 3 *SecEQP scheme is adaptive IND-CKA $(\mathscr{L}_1,\mathscr{L}_2)$-secure in the random oracle model.*

Proof: In the proof, we first describe a simulator $\mathscr{S}$ that can simulate a view $A_v^* = (\mathbb{I}^*,\mathbf{T}^*,\mathbf{c}^*)$ with the help of information accessible in the leakage functions $\mathscr{L}_1$ and $\mathscr{L}_2$. Next, we show that a probabilistic polynomial-time (PPT) adversary cannot distinguish between the simulated view $A_v^* = (\mathbb{I}^*,\mathbf{T}^*,\mathbf{c}^*)$ and the real adversary view $A_v = (\mathbb{I},\mathbf{T},\mathbf{c})$.

• Step (1): Simulate $\mathbf{c}^*$ (which captures the requirement for data privacy). To simulate the encrypted data items $D = \{d_1,\cdots,d_n\}$, the simulator first learns the value n and the size of each encrypted data item from the leakage function $\mathscr{L}_1$. Then, the simulator generates the simulated ciphertext with randomly selected plaintext and the known CPA-secure encryption algorithm. The simulator needs to ensure that the simulated ciphertext has the same size as the real ciphertext. Because the CPA-secure encryption algorithm achieves ciphertext indistinguishability, a PPT adversary cannot distinguish the simulated ciphertext with the real ciphertext.

• Step (2): Simulate $\mathbb{I}^*$ (which captures the requirement for index privacy). To simulate the IBF tree T, the simulator $\mathscr{S}$ constructs an identically structured IBF tree first. Then, for each node v in T, the simulator $\mathscr{S}$ sets up an IBF B_v with the same size as in the IBF in the index $\mathbb{I}$. Note that the simulator $\mathscr{S}$ can learn the IBF size from the leakage function $\mathscr{L}_1$. In the ith twin of B_v, the simulator $\mathscr{S}$ stores either 0 at $B_v[0][i]$ and 1 at $B_v[1][i]$, or vice versa. For each twin, how to assign 0-cell and 1-cell is decided by fairly tossing a coin. Next, for each IBF node, the simulator $\mathscr{S}$ generates a random number to associate with it. Finally, the simulator $\mathscr{S}$ outputs the IBF tree T and its associated random number as the simulated index $\mathbb{I}^*$ to the adversary. The simulated index $\mathbb{I}^*$ has exactly the same structure with the real index $\mathbb{I}$. The IBF nodes in either $\mathbb{I}^*$ or $\mathbb{I}$ have

the same size and equally distributed 0-cell and 1-cell. Hence, a PPT adversary cannot distinguish between the simulated index $\mathbb{I}^*$ and the real index $\mathbb{I}$.

• Step (3): Simulate $\mathbf{T}^*$ (which captures the requirement for token privacy). Suppose that the simulator $\mathscr{S}$ receives a query q_i. From the leakage function $\mathscr{L}_2$, the simulator $\mathscr{S}$ knows whether this query has been searched before or not. If it has been searched before, the simulator $\mathscr{S}$ outputs the previous searched token t_{q_i} to the adversary. Otherwise, the simulator $\mathscr{S}$ generates a new search token t_{q_i} as follows. The search token for a query is the set of k_B-pair of hashes and locations. Because the simulator $\mathscr{S}$ can learn access pattern from the leakage function $\mathscr{L}_2$, the simulator $\mathscr{S}$ knows which leaf IBF node in the index matches the search token t_{q_i}. For the leaf IBF node that matches the search token t_{q_i}, the simulator $\mathscr{S}$ can program the bit output by the random oracle $H(\cdot)$ to select k_B-pair of hashes and locations and ensure that the selected k_B-pair of hashes and locations match the leaf IBF node v. For the leaf IBF node that does not match the search token t_{q_i}, the simulator $\mathscr{S}$ is able to ensure that the simulated search token does not match the IBF node by programming the bit output by the random oracle. By this way, the simulator $\mathscr{S}$ can output the generated k_B-pair of hashes and locations as the simulated search token $\mathbf{T}^*$. Since the search token is k_B-pair of hashes and locations which are produced by the random hash functions, the simulated search token $\mathbf{T}^*$ is indistinguishable from the real search token $\mathbf{T}$ by a PPT adversary.

In summary, the simulated view $A_v^* = (\mathbb{I}^*, \mathbf{T}^*, \mathbf{c}^*)$ and the real view $A_v = (\mathbb{I}, \mathbf{T}, \mathbf{c})$ are indistinguishable by a PPT adversary. Therefore, SecEQP scheme is adaptive IND-CKA $(\mathscr{L}_1, \mathscr{L}_2)$-secure in the random oracle model.

■

6.6 Performance Evaluation

In this section, we first introduce parameter settings, datasets, performance metrics, and implementation. Then, we evaluate the performance of SecEQP and compare it with other two schemes (Elmehdwi et al. [55] and Yao et al. [118]) with the strong security assurance. Last, we describe a strategy to improve the result accuracy to meet a variety of location service demands.

6.6.1 Parameters Settings

Table 6.4 summarizes the default parameter settings in the experiment. Among these parameters, the choices of $(d_1, \cdots, d_L)$ are not straightforward, because they affect the size of the feasible region. If they are set to be too small, too few or even no points are inside the feasible region. if they are set to be too large, then too many points are inside the feasible region, this would make the post-processing inefficient. Therefore, the choices of $(d_1, \cdots, d_L)$ should depend on the distribution of data in the dataset. Since the data owner holds all data in plaintext, (s)he has the knowledge of the data distribution to choose appropriate $(d_1, \cdots, d_L)$. Accordingly, we design a parameter training algorithm (run by the data owner) for choosing appropriate $(d_1, \cdots, d_L)$ in the following.

• **Parameter Training Algorithm.** The parameter training algorithm is described in Algorithm 6.4. On input a port of randomly sampled data from the dataset, the algorithm outputs the choices of $(d_1, \cdots, d_L)$. The number of sampled inputs depends on the processing ability of the data owner. The notations used in the algorithm is explained in Table 6.3. The functionality of `Rand_Sample` is to randomly sample a point from the geographical space. Figure 6.8 shows a group of 50 random sampled points in the state of New York and California by using `Rand_Sample`. Given an array and its length, the function `Range` returns the maximum item minus the minimum item in the array. The functionality of `Compute_num` is described as follows: given a random query point q' (denoted as $(Q\_x[i], Q\_y[i])$) and sampled dataset, the function first samples $f' \leftarrow \mathscr{H}_{v,t}^{d'}$, which specifies a feasible region $\mathbf{FR}(f'(q'))$. Then, the function counts how many points in the sampled dataset locate in the same feasible region $\mathbf{FR}(f'(q'))$ and returns the number. The design rationale of the parameters training algorithm is that it uses a port of sampled data to estimate the number of points in the returned result for the entire dataset and adjust $d' = \mathtt{max}\{d_i\} = d_L$ to be an appropriate value to ensure that appropriate number of points can be returned in the search. The adjusting process starts from a very large d' and then makes big step size decreasing and then small step size increasing iteratively to get the appropriate value. Afterward, d' is divided to be L pieces uniformly and then $(d_1, \cdots, d_L)$ are derived.

Algorithm 6.4: Parameters-Training

Input: $v, t, n, k, L, U, Sample\_x[\,], Sample\_y[\,], Sample\_num, Query\_num, Repeat\_num, low, high, Big\_step, Small\_step$

Output: $(d_1, \cdots, d_L)$

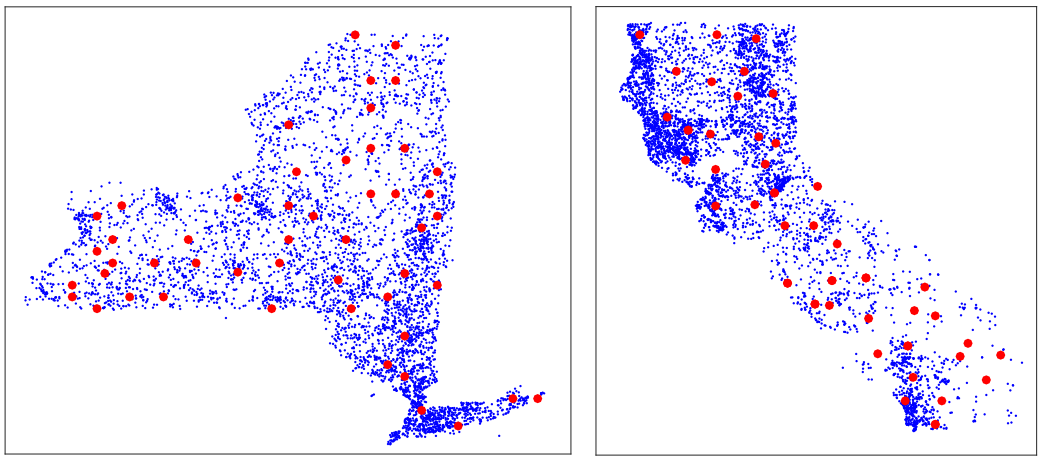
for $(i = 1; i < Query\_num + 1; i++)$ **do**
  $(Q\_x[i], Q\_y[i]) = \texttt{Rand\_Sample}(U)$;
$Range_x = \texttt{Range}(Q\_x, Query\_num)$;
$Range_y = \texttt{Range}(Q\_y, Query\_num)$;
$d' = (Range_x^2 + Range_y^2)^{1/2}$;
$Expect\_num = \frac{k}{n} \times Sample\_num$;
$Estimate\_num = 0$;
while $(Estimate\_num < low \times Expect\_num \,||\, Estimate\_num > high \times Expect\_num)$ **do**
  $sum = 0$;
  for $(i = 1; i < Query\_num + 1; i++)$ **do**
    for $(j = 1; j < Repeat\_num + 1; j++)$ **do**
      Point_num$[i][j] = \texttt{Compute\_num}(Q\_x[i], Q\_y[i], Sample\_x, Sample\_y, Sample\_num, v, t, d')$;
      $sum = sum +$ **Point_num**$[i][j]$;
  $Expect\_num = \frac{sum}{Query\_num \times Repeat\_num}$;
  if $Expect\_num > high \times Expect\_num$ **then**
    $d' = \frac{d'}{Big\_step}$;
  if $Expect\_num < low \times Expect\_num$ **then**
    $d' = d' \times Small\_step$;
for $(i = 1; i < L + 1; i++)$ **do**
  $d_i = \frac{d'}{L} \times i$;

Table 6.3: Notations used in Algorithm 6.4.

Notations	Meanings	Default Values
$Sample\_x[\,]$	the array contains X coordinates of the sampled data	–
$Sample\_y[\,]$	the array contains Y coordinates of the sampled data	–
$Sample\_num$	the number of sampled data	20000
$Query\_num$	the number of random queries	50
$Repeat\_num$	the number of repeat times	50
low and $high$	the lower bound and higher bound	5 and 8
Big_step	the big step size and	5
$Small\_step$	the small step size	2

6.6.2 Datasets, Metrics, and Implementation

• **Datasets.** (1) **NY** is a real-world dataset contains 1 million spatial data in the state of New York (NY) from OpenStreetMap Project [14], which collects geographical data from volunteered mobile device carriers. (2) **CA** is a real-world dataset contains 1 million spatial data in California (CA) from OpenStreetMap Project. (3) **UF** is a synthetic dataset contains 1 million spatial data generated from uniform (UF) distribution. More specifically, each data is denoted as (X_{UF}, Y_{UF}), where $X_{UF} \sim U[0, 10^9]$ and $Y_{UF} \sim U[0, 10^9]$. Figure 6.8 exhibits the data distribution of CA and NY by randomly drawing 5000 points from the corresponding dataset.



(a) New York (NY) (b) California (CA)

Figure 6.8: The data distribution of two real-world datasets (big and red points are 50 randomly sampled points).

• **Metrics.** Three metrics are used: (1) query latency, (2) query result accuracy, and (3) query cost. The query latency is defined as the time for the cloud to respond to an SkNN query. The result accuracy of an SkNN query can be reflected by *Overall Approximation Ratio* (OAR) [109], which is defined as $\frac{1}{k}\sum_{i=1}^{k}\frac{\|o_i,q\|}{\|o_i^*,q\|}$, where q is the query point, o_i is the ith nearest point in the search results and o_i^* is the ground truth (i.e., the actual ith nearest point in the dataset). Theoretically, the high result accuracy means OAR should be close to 1. The query cost consists of the communication cost and the size of the secure index maintained in the cloud.

• **Implementation.** The SecEQP implementations are achieved by C++. We carry out the experiments on a cluster node (serves as the cloud) equipped with 128 GB RAM and two 2.5Ghz 10-core Intel Xeon E5-2670v2 CPU. In our experiments, unless otherwise stated, when we vary

the value of one parameter in concern, we keep all other parameters at their default values, which are displayed in Table 6.4.

Table 6.4: Parameter settings.

Notations	Meanings	Default Values
n	the number of points	100,000
k	the number of nearest points required in a query	50
m	the number of twins in the root node of IBF tree	$10Ltn$
k_B	the number of hash functions in an IBF	7
v	the number of AND-composition	6
t	the number of OR-composition	6
L	the number of interval lengths	5

6.6.3 Experimental Results

We first evaluate the performance of SecEQP in terms of query latency, result accuracy, and query cost. Then we compare SecEQP with other two schemes (Elmehdwi et al. [55] and Yao et al. [118]) with the strong security assurance.

• **Query Latency.** The query latency as a function of n (i.e., the number of points in a dataset) and k (i.e., the number of nearest points required in a query) is shown in Figure 6.9 and Figure 6.10. It can be observed that the query latency grows sublinear with n and a slightly faster than linear with k. While $k = 50$, the query latency for a dataset contains 1 million points with is less than 50 msec.

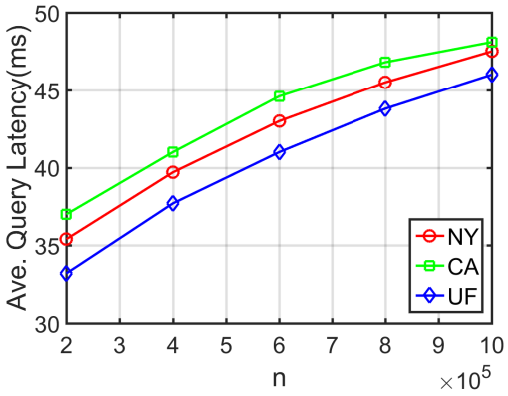


Figure 6.9: SecEQP query latency by varying the parameter n.

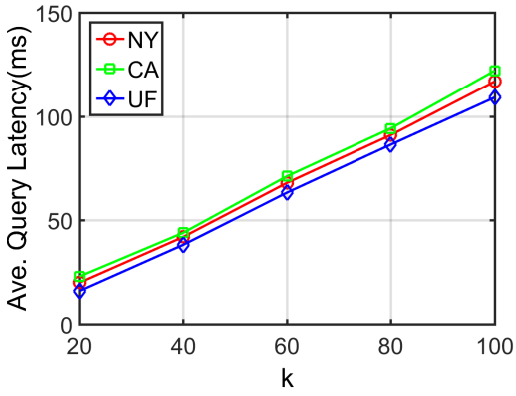


Figure 6.10: SecEQP query latency by varying the parameter k.

• **Result Accuracy.** Let n' be the total number of inserted items in the root node of the IBF. Figure 6.11 and 6.12 exhibit OAR as a function of v and t, respectively. The OAR is monotonically decreasing with increasing v and t. Hence, increasing v and t can improve the result accuracy. As shown in Figure 6.11, if $v = 6$ and $t = 6$, the average OAR is about 1.3. This means that the average distance between the queried point and returned results is 1.3 times longer than the ground truth. Note that a strategy is developed to further improve the result accuracy (i.e., OAR can be improved to be 1.1) in Section 6.6.4.

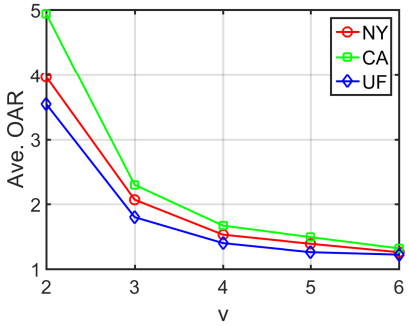


Figure 6.11: SecEQP OAR by varying the parameter v.

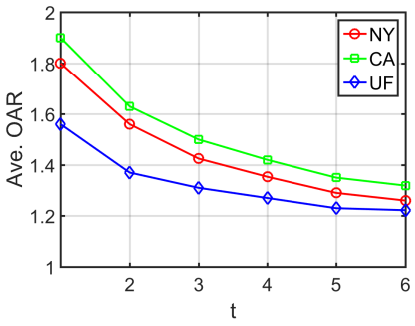


Figure 6.12: SecEQP OAR by varying the parameter t.

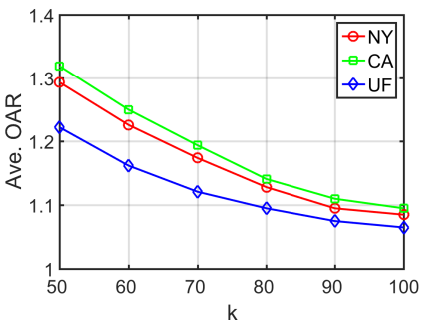


Figure 6.13: SecEQP OAR by varying the parameter k.

• **Query Cost.** For the query cost, we consider the communication volume and the size of the secure index which helps to accelerate the query processing in the cloud. The communication volume consists of the transmission of the search token and the encrypted data items. Since the size of encrypted data items is independent of the adopted SecEQP scheme, we only consider the communication volume of the search token. Table 6.5 shows the token size in an SkNN query for different schemes. The token size of SecEQP is computed based on the parameter settings in Table 6.4. It is shown that all of the schemes have a constant token size for an SkNN query. The search token only takes 6.93 KB, indicating a very small communication volume. For index size, SecEQP employs the existing IBF tree compression algorithms [81] to compress the index size. The index size varies with dataset sizes. While the dataset contains 10^4, 10^5, and 10^6 points, the index will take 0.44 GB, 3.13 GB, and 15.8 GB, respectively.

• **Comparison with Other Schemes.** We compare SecEQP with two schemes with strong security assurance. The experiment results for query latency are average over the three datasets. The results are summarized in Table 6.5. We have five observations.

First, while $k = 1$ (i.e., 1NN), Yao et al. [118] has the shortest query latency over three database sizes (10^4, 10^5, 10^6), ranges from 5 ms to 12 ms, whereas SecEQP has a comparable query latency (i.e., from 9 ms to 31 ms). Second, while $k = 50$, SecEQP has the shortest query latency, ranges from 12 ms to 47 ms, which is not significantly increased with k, whereas Elmehdwi et al. [55] do (e.g., 14.3 sec→11.78 min). Note that Yao et al. [118] does not support the use scenarios while $k > 1$. Third, the average OAR for both Yao et al. [118] and Elmehdwi et al. [55] is 1, whereas SecEQP is about 1.3. Forth, all of three schemes do not create large tokens. The token size ranges from 8 bytes (Yao et al. [118]) to 6.9 KB (SecEQP). Fifth, SecEQP's index is the largest one compared with other schemes. In a dataset contains 10^6 points, SecEQP's index takes 15.8 GB, whereas Yao et al. [118] and Elmehdwi et al. [55] use 20.3 MB and 0 MB, respectively. There are two causes. First, Elmehdwi et al. [55] does not employ the index mechanism for the query acceleration. Second, SecEQP supports the use scenarios while $k > 1$; however, Yao et al. [118] does not.

Table 6.5: Compare SecEQP with other schemes (NA: not applicable).

Scheme	**Query latency**						**Query cost**				**Accuracy (OAR)**
	$k = 1$			$k = 50$			Communication volume	Index size			
	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^4$	$n = 10^5$	$n = 10^6$	(token size + data items size)	$n = 10^4$	$n = 10^5$	$n = 10^6$	
SecEQP	9 ms	21 ms	31 ms	12 ms	32 ms	47 ms	6.93 KB + data item size	0.44 GB	3.13 GB	15.8 GB	≈1.3
Yao et al. [118]	5 ms	9 ms	12 ms	Na	Na	Na	8 byte + data items size	14.8 MB	17.4 MB	20.3 MB	1
Elmehdwi et al. [55]	0.15 sec	1.44 sec	14.3 sec	0.12 min	1.18 min	11.78 min	16 byte + data items size	NA	NA	NA	1

6.6.4 Result Accuracy Improvement

In this section, we first illustrate the top nearest accuracy property and then describe how to use it to develop an effective strategy to improve the result accuracy.

• **Top Nearest Accuracy Property.** It is observed in the experiments that the closer the point, the less probability it is missed in the searching. We call this top nearest accuracy property. Theoretically, this property is caused by the successive inclusion property as exhibited in Formula (6.3). The following example is helpful for understanding. Suppose that the query processing stops after searching $\mathbf{FR}(f_5(q))$. For the points in $\mathbf{FR}(f_1(q))$, they are searched for 5 times; ... ; for the points in $\mathbf{FR}(f_5(q))$, they are searched for only 1 time. This repeated filtering process leads to top nearest

accuracy property. Remarkably, top nearest accuracy property is of practical significance since most of the decisions are made among the top nearest query results. For example, a customer may search for the 20NN restaurants, but it is highly likely that the customer decides to eat in the top 5 nearest restaurant, which is of very high probability to be the exact ground truth due to the top nearest accuracy property of SecEQP.

• **Improve Accuracy Strategy.** The top nearest accuracy property indicates a strategy to improve the result accuracy of SecEQP scheme. The strategy is that if we want to get kNN, we can query k'NN, where $k' > k$, and figure out the top-k nearest points as the query results. To evaluate this strategy, we conduct an experiment as follows. We query kNN, where $k = 50, \cdots, 100$, and select the top 50 nearest points as the final query results for 50NN. The experiment results are plotted in Figure 6.13. We observe that the result accuracy is improved by increasing k. If we let $k = 100$, the average OAR is less than 1.1, which demonstrates that this strategy is effective in improving the result accuracy.

Chapter 7

Conclusions

In this chapter, we provide a quick summary of our work. In this work, we address several security and efficiency challenges both in IoT devices and IoT infrastructures.

We have conducted two projects to address security and efficiency challenges in IoT devices. In the first project, we show that 29 of the 40 popular Wi-Fi IoT devices have no security protocols deployed, or contain problematic security implementations. By exploiting these vulnerabilities, adversaries can launch various cyberattacks against IoT device owners. The identified vulnerabilities stem from hardware/software limitations and/or imprudent security designs of the IoT devices. These limitations preclude the possibility of deploying security solutions on the devices themselves. Accordingly, this study has proposed an infrastructure-based solution, designated as SecWIR, for securing the IoT communications using COTS home Wi-Fi routers. Importantly, SecWIR provides the Wi-Fi IoT devices with full mainstream security protocol support without the need for any modifications of the existing IoT devices and IoT servers or the purchase of additional security hardware. It can enable IoT users to enjoy inexpensive, but still secure IoT devices without any substantial increase in the device access delay or the degradation of the non-IoT data service performance. As such, SecWIR plays a valuable role in paving the way for the further development and deployment of Wi-Fi smart home IoT technology. In the second project, we identify several security vulnerabilities of HDVAs by considering Amazon Alexa and Google Home as case studies. Surprisingly, the Alexa and Google Home services rely on only a weak single-factor authentication, which can be easily broken. To secure the HDVA services, we seek to propose an additional factor authentication, physical presence. An HDVA device can accept voice commands only when any person is physically present nearby. We thus design a solution called virtual security button (VSButton) to do the physical presence detection. We prototype and evaluate it on an Alexa device. Our experimental results show that VSButton can do accurate detection in both laboratory and real-world home settings. We hope our initial efforts can stimulate further research

on HDVA security.

We have conducted two projects to address security and efficiency challenges in IoT infrastructures. In the first project, We propose BFastPay, the first inter-blockchain and routing-free protocol, to support fast payment in the Bitcoin network. BFastPay can be built based on any PSC-supported blockchain, therefore, any further improvement on the transaction validation mechanism of the PSC-supported blockchains can directly lead to the improvement of BFastPay. Moreover, we develop a PoW-based arbitration mechanism to enable BFastPay to make fair payment arbitration in a payment dispute. Our comparative study shows that BFastPay and Lightning Network are competing approaches. We hope that our study can stimulate more future research endeavors on IoT Infrastructures like blockchain systems In the second project, we propose a novel SecEQP scheme that supports practical SkNN query processing over encrypted geospatial data in cloud computing. The key novelty of our scheme is in applying projection function composition to encode two-dimensional data to test the proximity of two points by only equality checking operations. We formulated the related theory and explained it via various illustrative graphic examples. We implemented and evaluated SecEQP scheme on both real-world and synthetic datasets. It is shown that SecEQP scheme can achieve strong security, high efficiency, and high result accuracy. We hope that our study will invite more research in the IoT infrastructures like cloud servers.

APPENDIX

My Publications

- **Xinyu Lei**, Guan-Hua Tu, Tian Xie and Sihan Wang. BFastPay: a Routing-free Protocol for Fast Payment in Bitcoin Network, ACM Conference on Data and Application Security and Privacy (**ACM CodaSpy'21**), 2021.
- **Xinyu Lei**, Guan-Hua Tu, Chi-Yu Li, Tian Xie, Mi Zhang. SecWIR: Securing Smart Home IoT Communications via Wi-Fi Routers with Embedded Intelligence, ACM International Conference on Mobile Systems, Applications, and Services (**ACM MobiSys'20**), 2020.
- **Xinyu Lei**, Guan-Hua Tu, Alex X. Liu, Tian Xie. Fast and Secure kNN Query Processing in Cloud Computing, IEEE Conference on Communications and Network Security (**IEEE CNS'20**), 2020.
- **Xinyu Lei**, Tian Xie, Guan-Hua Tu, Alex X. Liu. An Inter-blockchain Escrow Approach for Fast Bitcoin Payment (Extended Abstract), IEEE International Conference on Distributed Computing Systems (**IEEE ICDCS'20**), 2020.
- **Xinyu Lei**, Alex X. Liu, Rui Li, Guan-Hua, Tu. SecEQP: A Secure and Efficient Scheme for SkNN Query Problem over Encrypted Geodata on Cloud, IEEE International Conference on Data Engineering (**IEEE ICDE'19**), 2019.
- **Xinyu Lei**, Guan-Hua, Tu, Alex X. Liu, Chi-Yu Li, Tian Xie. The Insecurity of Home Digital Voice Assistants-Vulnerabilities, Attacks and Countermeasures, IEEE Conference on Communications and Network Security (**IEEE CNS'18**), 2018.
- **Xinyu Lei**, Alex X. Liu, Rui Li. Secure kNN Queries over Encrypted Data: Dimensionality is not Always a Curse (Extended Abstract), IEEE International Conference on Data Engineering (**IEEE ICDE'17**), 2017.
- Yiwen Hu, Sihan Wang, Guan-Hua Tu, Li Xiao, Tian Xie, **Xinyu Lei**, Chi-Yu Li. Security Threats from Bitcoin Wallet Smartphone Applications: Vulnerabilities, Attacks, and Countermeasures, ACM Conference on Data and Application Security and Privacy (**ACM CodaSpy'21**), 2021.
- Tian Xie, Sihan Wang, Guan-Hua Tu, Chi-Yu Li, **Xinyu Lei**. Exploring the Insecurity of Google Account Registration Protocol via Model Checking, IEEE Symposium Series on Computational Intelligence (**IEEE SSCI'19**), 2019.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Bitcoin dominance. https://coinmarketcap.com/charts/#dominance-percentage.

[2] Btc relay. http://btcrelay.org/.

[3] Coinmarketcap. https://coinmarketcap.com/.

[4] Confirmation. https://en.bitcoin.it/wiki/Confirmation.

[5] Dropbox hack. http://www.troyhunt.com/the-dropbox-hack-is-real/.

[6] Eos.io white paper. shorturl.at/mCF28.

[7] Eosio.cdt (contract development toolkit) is a suite of tools used to build eosio contracts. https://github.com/EOSIO/eosio.cdt.

[8] Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper.

[9] Lightning strikes, but select hubs dominate network funds. https://diar.co/volume-2-issue-25/.

[10] Mathematical proof that the lightning network cannot be a decentralized bitcoin scaling solution. https://bit.ly/2tg3sxe.

[11] Microsoft adds bitcoin payments for xbox games and mobile content. https://bit.ly/1vIvjLP.

[12] Neo white paper. http://docs.neo.org/en-us/whitepaper.html.

[13] On slow and fast block times. https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/.

[14] Openstreetmap. http://www.openstreetmap.org/.

[15] Remix. https://github.com/ethereum/remix.

[16] Rinkeby. https://rinkeby.etherscan.io/.

[17] Samsung stores in three baltic states accept cryptocurrencies. https://bit.ly/2LAPFHz.

[18] The solidity contract-oriented programming language. https://github.com/ethereum/solidity.

[19] Solidity documentation. https://solidity.readthedocs.io/en/v0.4.24/.

[20] The stellar consensus protocol: A federated model for internet-level consensus.

https://www.stellar.org/papers/stellar-consensus-protocol.pdf.

[21] summa project. https://bit.ly/3ymZfWX.

[22] Ieee std. 802.11n-2009: Enhancements for higher throughput. http://www.ieee802.org, 2009.

[23] Connect echo dot to wi-fi. https://amzn.to/2ViZSSZ, 2017.

[24] Microbot push. https://prota.info/, 2017.

[25] An overview of wireless protected access 2 (wpa2). https://www.lifewire.com/what-is-wpa2-818352, 2017.

[26] Control network traffic with iptables. https://linode.com/docs/security/firewalls/control-network-traffic-with-iptables/, 2018.

[27] Linux top command. https://www.lifewire.com/linux-top-command-2201163, 2018.

[28] Openwrt/lede project. https://openwrt.org/, 2018.

[29] Wi-fi security. https://www.wi-fi.org/discover-wi-fi/security, 2020.

[30] H. Abdi and L. J. Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[31] ActivityManager. https://bit.ly/2o2pulx, 2018.

[32] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 563–574, 2004.

[33] S. Aich, S. Chakraborty, M. Sain, H.-i. Lee, and H.-C. Kim. A review on benefits of iot integrated blockchain based supply chain management implementations across different sectors with case study. In *international conference on advanced communication technology (ICACT)*, pages 138–141, 2019.

[34] L. Alliance. Lorawan: Low power wide area network. https://lora-alliance.org/about-lorawan, 2020.

[35] A. Alshamsi and T. Saito. A technical comparison of ipsec and ssl. In *International Conference on Advanced Information Networking and Applications (AINA)*, pages 395–398, 2005.

[36] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *CoRR*, abs/1708.05044, 2017.

[37] G. R. Arce. *Nonlinear signal processing: a statistical approach*. John Wiley & Sons, 2005.

[38] ASN1. Asn1. http://luca.ntop.org/Teaching/Appunti/asn1.html, 2018.

[39] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with bitcoins. In *IEEE P2P Proceedings*, pages 1–5, 2013.

[40] T. Be'ery and A. Shulman. A perfect crime? only time will only time will tell. https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf, 2013.

[41] C. Benzaïd, A. Boulgheraif, F. Z. Dahmane, A. Al-Nemrat, and K. Zeraoulia. Intelligent detection of mac spoofing attack in 802.11 network. In *Proceedings of the 17th International Conference on Distributed Computing and Networking (ICDCN)*, pages 1–5, 2016.

[42] Z.-P. Bian, J. Hou, L.-P. Chau, and N. Magnenat-Thalmann. Fall detection based on body part tracking using a depth camera. *IEEE journal of biomedical and health informatics (IEEE J-BHI)*, pages 430–439, 2015.

[43] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 1970.

[44] R. Bonetto, N. Bui, V. Lakkundi, A. Olivereau, A. Serbanati, and M. Rossi. Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. In *IEEE international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*, pages 1–7, 2012.

[45] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In *Workshop on cryptography and security in clouds (WCSC)*, 2011.

[46] S. Butterworth. On the theory of filter amplifiers. *Wireless Engineer*, 7(6):536–541, 1930.

[47] Caextract. Mozilla ca certificate store in pem format. https://curl.haxx.se/docs/caextract.html, 2018.

[48] B. K. Chawla, O. Gupta, and B. Sawhney. A review on ipsec and ssl vpn. *International Journal of Scientific & Engineering Research (IJSER)*, pages 21–24, 2014.

[49] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino. Secure knn query processing in untrusted cloud environments. *IEEE Transactions on Knowledge and Data Engineering (IKDE)*, pages 2818–2831, 2014.

[50] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. pages 79–88, 2006.

[51] CVE. Beast attack. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3389, 2011.

[52] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry (SCG)*, pages 253–262, 2004.

[53] C. Decker and R. Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems (SSS)*, pages 3–18, 2015.

[54] A. Dmitrienko, D. Noack, and M. Yung. Secure wallet-assisted offline bitcoin payments with double-spender revocation. In *ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 520–531, 2017.

[55] Y. Elmehdwi, B. K. Samanthula, and W. Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *IEEE International Conference on Data Engineering (ICDE)*, pages 664–675, 2014.

[56] H. Feng, K. Fawaz, and K. G. Shin. Continuous authentication for voice assistants. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 343–355, 2017.

[57] E. Fernandes, J. Jung, and A. Prakash. Security analysis of emerging smart home applications. In *IEEE symposium on security and privacy (S&P)*, 2016.

[58] J. Fruhlinger. The mirai botnet explained: How teen scammers and cctv cameras almost brought down the internet. https://bit.ly/35Ttjwq, 2018.

[59] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the annual ACM symposium on Theory of computing (STOC)*, pages 169–178, 2009.

[60] E.-J. Goh et al. Secure indexes. *IACR Cryptology ePrint Archive*, page 216, 2003.

[61] S. Goldfeder, J. Bonneau, R. Gennaro, and A. Narayanan. Escrow protocols for cryptocurrencies: How to buy physical goods using bitcoin. In *International Conference on Financial Cryptography and Data Security (FC)*, pages 321–339. Springer, 2017.

[62] I. Hafeez, A. Y. Ding, L. Suomalainen, A. Kirichenko, and S. Tarkoma. Securebox: Toward safer and smarter iot networks. In *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking (CoNEXT Workshop)*, pages 55–60, 2016.

[63] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: Gathering 802.11 n traces with channel state information. *SIGCOMM Computer Communication Review*, 41(1):53–53, 2011.

[64] W. L. Harris and J. Wonglimpiyarat. Blockchain platform and future bank competition. *Foresight*, 2019.

[65] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

[66] H. Hu, J. Xu, C. Ren, and B. Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *IEEE International Conference on Data Engineering (ICDE)*, pages 601–612, 2011.

[67] B. Jokanovic, M. Amin, and F. Ahmad. Radar fall motion detection using deep learning. In *IEEE radar conference (RadarConf)*, pages 1–6, 2016.

[68] I. Jolliffe. Principal component analysis. In *International encyclopedia of statistical science (IESS)*, pages 1094–1096. 2011.

[69] R. Jones. Netperf. https://github.com/HewlettPackard/netperf, 2018.

[70] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in bitcoin. In *ACM conference on Computer and communications security (CCS)*, pages 906–917, 2012.

[71] J. Katz and Y. Lindell. Introduction to modern cryptography: principles and protocols. cryptography and network security, 2008.

[72] S. P. Kavalaris and E. Serrelis. Security issues of contemporary multimedia implementations: The case of sonos and sonosnet. In *International Conference in Information Security and Digital Forensics (ISDF)*, pages 63–74, 2014.

[73] A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD*, pages 239–257. 2007.

[74] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference (CRYPTO)*, 2017.

[75] Komando. Best apps to control your router. https://www.komando.com/apps/368384/best-apps-to-control-your-router/all, 2018.

[76] S. Kumar, Y. Hu, M. P. Andersen, R. A. Popa, and D. E. Culler. Jedi: Many-to-many end-to-end encryption and key delegation for iot. In *Network and Distributed Systems Symposium (NDSS)*, 2019.

[77] M. Kunz, K. Kasper, H. Reininger, M. Möbius, and J. Ohms. Continuous speaker verification in realtime. In *Proceedings of the Biometrics Special Interest Group (BIOSIG)*, pages 79–88, 2011.

[78] X. Lei, G.-H. Tu, C.-Y. Li, T. Xie, and M. Zhang. Secwir: securing smart home iot communications via wi-fi routers with embedded intelligence. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (Mobisys)*, pages 260–272, 2020.

[79] X. Lei, G.-H. Tu, A. X. Liu, C.-Y. Li, and T. Xie. The insecurity of home digital voice assistants-vulnerabilities, attacks and countermeasures. In *IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2018.

[80] X. Lei, G.-H. Tu, T. Xie, and S. Wang. Bfastpay: A routing-free protocol for fast payment in bitcoin network. In *Proceedings of the ACM Conference on Data and Application Security and Privacy (CodaSpy)*, pages 77–87, 2021.

[81] R. Li and A. X. Liu. Adaptively secure conjunctive query processing over encrypted data for cloud computing. In *IEEE International Conference on Data Engineering (ICDE)*, pages 697–708, 2017.

[82] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar. Fast range query processing with strong privacy protection for cloud computing. *Proceedings of the VLDB Endowment (PVLDB)*, pages 1953–1964, 2014.

[83] J. Lin, Z. Shen, A. Zhang, and Y. Chai. Blockchain and iot based food traceability for smart agriculture. In *Proceedings of the 3rd International Conference on Crowd Science and Engineering*, pages 1–6, 2018.

[84] R. Mazerik. Beast vs. crime attack. https://bit.ly/37hmC8b, 2013.

[85] Mbed. Delta dfcm-nnn40. https://os.mbed.com/components/Delta-DFCM-NNN40/, 2018.

[86] Y. Meng, Z. Wang, W. Zhang, P. Wu, H. Zhu, X. Liang, and Y. Liu. Wivo: Enhancing the security of voice control system via wireless signal in iot environment. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 81–90, 2018.

[87] S. Mercan, E. Erdin, and K. Akkaya. Improving transaction success rate via smart gateway selection in cryptocurrency payment channel networks. *arXiv preprint arXiv:2003.10877*, 2020.

[88] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[89] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the international conference on Very large data bases (VLDB)*, pages 763–774, 2006.

[90] M. Moshtaghi, C. Leckie, S. Karunasekera, J. C. Bezdek, S. Rajasegarar, and M. Palaniswami. Incremental elliptical boundary estimation for anomaly detection in wireless sensor networks. In *IEEE international conference on data mining (ICDM)*, pages 467–476, 2011.

[91] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[92] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 644–655, 2015.

[93] Netgear. Readycloud at netgear. http://readycloud.netgear.com/client/en/welcome.html, 2020.

[94] J. Obermaier and M. Hutle. Analyzing the security and privacy of cloud-based video surveillance systems. In *ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS Workshop)*, pages 22–28, 2016.

[95] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 2009.

[96] Openwrt. Package: openssl-util. https://openwrt.org/packages/pkgdata/openssl-util, 2018.

[97] Openwrt. Supported routers. https://openwrt.org/toh/start, 2018.

[98] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques (ICTACT)*, 1999.

[99] K. Paterson. On the security of rc4 in tls and wpa. http://www.isg.rhul.ac.uk/tls/, 2013.

[100] C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí. Double-spending prevention for bitcoin zero-confirmation transactions. *IACR Cryptology ePrint Archive*, 2017.

[101] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *https://lightning. network/lightning-network-paper.pdf*, 2016.

[102] D. Roethlisberger. Sslsplit. https://www.roe.ch/SSLsplit, 2018.

[103] M. Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[104] Z.-H. E. P. Salowey, J. and H. Tschofenig. Transport layer security (tls) session resumption without server-side state. https://tools.ietf.org/html/rfc5077, 2008.

[105] A. K. Simpson, F. Roesner, and T. Kohno. Securing vulnerable home iot devices with an in-hub security manager. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 551–556, 2017.

[106] SSLdump. Ssldump home page. http://ssldump.sourceforge.net/, 2018.

[107] Statista. Unit shipments of wi-fi enabled smart home devices worldwide from 2016 to 2020 (in millions). https://bit.ly/2EhpNj1, 2016.

[108] Takahashi and A. Otsuka. Secure offline payments in bitcoin. In *International Conference on Financial Cryptography and Data Security (FC)*, 2019.

[109] Y. Tao, K. Yi, C. Sheng, and P. Kalnis. Quality and efficiency in high dimensional nearest neighbor search. In *SIGMOD*, pages 563–576, 2009.

[110] Y. Tian, N. Zhang, Y.-H. Lin, X. Wang, B. Ur, X. Guo, and P. Tague. Smartauth: User-centered authorization for the internet of things. In *USENIX Security Symposium (USENIX Security)*, pages 361–378, 2017.

[111] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet x. 509 public key infrastructure (pki) proxy certificate profile. Technical report, 2004.

[112] B. Wang, Y. Hou, and M. Li. Practical and secure nearest neighbor search on encrypted large-scale data. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9, 2016.

[113] D. Watkins. Strategy analytics: Amazon, google to ship nearly 3 million digital voice assistant devices in 2017. https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2016/10/05/strategy-analytics-amazon-google-to-ship-nearly-3-million-digital-voice-assistant-devices-in-2017#.WQtiXeXyuUk, 2016.

[114] C. Weinschenk. Wi-fi installed base forecast to reach 17 billion by 2030, driven by the smart home. https://bit.ly/3A24qg3, 2019.

[115] Wolfssl. Wolfssl memory usage. https://www.wolfssl.com/docs/benchmarks/, 2018.

[116] D. Wong. Downgrade attack on tls 1.3 and vulnerabilities in major tls libraries. https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2019/february/downgrade-attack-on-tls-1.3-and-vulnerabilities-in-major-tls-libraries/, 2019.

[117] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD)*, pages 139–152, 2009.

[118] B. Yao, F. Li, and X. Xiao. Secure nearest neighbor revisited. In *IEEE international conference on data engineering (ICDE)*, pages 733–744, 2013.

[119] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan. Practical k nearest neighbor queries with location privacy. In *IEEE international conference on data engineering (ICDE)*, pages

640–651, 2014.

[120] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2015.

[121] L. Zhang, S. Tan, and J. Yang. Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 57–71, 2017.

[122] T. Zoller. Tls / sslv3 renegotiation vulnerability explained. http://www.g-sec.lu/practicaltls.pdf, 2011.