INTEGRATING SECURITY AND PRIVACY PROTECTION INTO A MOBILITY-CENTRIC INTERNET ARCHITECTURE

by

XIRUO LIU

A dissertation submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
In partial fulfillment of the requirements
For the degree of
Doctor of Philosophy
Graduate Program in Electrical and Computer Engineering
Written under the direction of
Wade Trappe
And approved by

New Brunswick, New Jersey
MAY, 2016

ABSTRACT OF THE DISSERTATION

Integrating Security and Privacy Protection into a Mobility-Centric Internet Architecture

By XIRUO LIU

Dissertation Director:
Wade Trappe

The Internet is a well-noted technological success that has significantly impacted the dissemination of information and brought human society closer together than it ever had been. While many of the initial design choices associated with the Internet led to its successful rise to prominence, the Internet was not designed to face many of the challenges that have emerged in the modern era in which people access information while on the move from anywhere, at any time. Notably, one of the primary hurdles challenging the continued success of the Internet is the security of the communications crossing the Internet.

In order to address the challenges facing the evolution of the Internet, several clean-slate future Internet architectures have been proposed, each attempting to address certain aspects in which the Internet needs to evolve, and each with varying advantages. Across all of these different architectures, there remains a need to examine and address fundamental aspects related to ensuring the security of these architectures. This thesis examines several of the key security challenges facing several of the emerging future Internet designs, and specifically explores aspects related to securing the MobilityFirst future Internet design. In particular, one of the core contributions of this thesis is a thorough exploration of aspects related to securing new naming services intended to support more dynamic associations between users, their names and their network addresses. This thesis provides a thorough exploration of the protocol-level security challenges facing the administration of new name resolution services that separate names from network addresses, and further examines the possibility of using such a name resolution service as a mechanism to apply access control in the future Internet. A further contribution of this thesis is the exploration of security services for mobile ad hoc networks and the Internet of Things, which represent two important and emerging network modalities that will become part of the future Internet.

Acknowledgements

How Time flies! As I near the end of my studies, I have the illusion that this is still the first year of my PhD studies. Now, though, at the end of the PhD journey, everything over the past six years seems so clear in my mind, and I am extremely grateful for all of the support and help I have received in this journey.

First and foremost, I would like to express my deepest gratitude to my advisor Prof. Wade Trappe for his continuing guidance and absolute support over my five years of research at WINLAB. He is not just the advisor of my research, he is also a great friend who has shared his experience and advice, and given me suggestions on matters outside of school (e.g. how to raise a child). He inspired and encouraged me through those hard times. Also, I am grateful to Prof. Yanyong Zhang, who has given me a lot of valuable advice on both graduate studies and career development over the past few years.

I want to thank all of my colleagues here at Wireless Information Network Laboratory (WINLAB). WINLAB is a big family and has a lot of excellent faculty members, staffs and students. I am fortunate to be surrounded by these wonderful people. We have enjoyed our work, as well as many rich and colourful life experiences, together over the past years. This will become my most cherished memory from graduate school, which I will cherish for the rest of my life.

Outside of Rutgers University, I would like to extend my gratitude to my mentor Meiyuan Zhao and collaborators Jianqing Zhang, Jesse Walker at Intel Labs. My summer internship at Intel Labs was one of the best research experiences that I had during graduate school. I benefited a lot from the collaboration, and from them I gained a clear idea about what my future career will be.

Last but not least, I would like to thank my family. My parents Zhibing Liu and Lin Zhang supported me completely, no matter what happened and no matter what decision I made. The encouragement and the logistics from them made this thesis possible. My lovely daughter Annie Liu Yuan joined our family in the middle of my PhD journey. She is the apple of my eye and the driving force that propels me. Also, my husband Huawei Yuan has accompanied me through this long journey, and has provided me both patience and encouragement.

In summary, I would like to give sincere thanks to all of those fantastic people that I have met, and thank them for their support, help, encouragement and companionship.

Dedication

To my parents Zhibing Liu and Lin Zhang for their endless love and continuous support. Also to my daughter Annie Liu Yuan and my husband Huawei Yuan for both the joy and troubles they brought.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The Internet was initially launched almost fifty years ago and has since evolved from a small-scale academic testbed to a large-scale commercial network that spans the world. Since the late 1990s, the Internet has been growing at an explosive rate. In the early 1990s, only 1% of the data being transmitted through telecommunication channels was carried over the Internet. Now the Internet carries more than 97% of all telecommunication information [67]. The Internet has revolutionized the way people live as it provides rich and abundant services to facilitate people's daily activities, as well as enabling interactions between the virtual world and the reality. In 2005, only 16% of the world's population was using the Internet. This number climbed to 40% in 2015, with 78% of the population in the developed world using the Internet and 32% of the population in the developing world using the Internet [124].

The core of the current Internet - the Internet Protocol Suite - was standardized in 1982. This protocol stack underpins the current Internet and involves several communication protocols that are widely used today such as the Transmission Control Protocol (TCP), the Internet Protocol (IP) and the User Datagram Protocol (UDP). Only a few core protocols such as TCP, IP and UDP, form the foundation of the network service layer to provide routing and data delivery in a reliable, connection-oriented, end-to-end fashion or unreliable connectionless fashion. This design gives the Internet a narrow waist such that the simple and efficient network service layer connects heterogeneous hardware devices in the lower layer and a large variety of applications and services in the upper layers. The end-to-end TCP/IP protocol stack serves the fixed host/server model well, which has dominated the Internet since its inception.

However, today's Internet is quite different from its early stages, when the Internet Protocol

Suite was developed. We are facing a historic, large-scale shift from PC's to mobile computing and embedded devices. With the proliferation of mobile devices, e.g. smartphones, tablets, wearable devices, and vehicles, the amount of mobile data traffic is increasing rapidly at a remarkable rate. According to [40], global mobile data will increase 10-fold between 2014 and 2019 at a speed three times faster than how fixed IP traffic grows. Looking ahead another five years, the number of mobile devices and their traffic will inevitably surpass the fixed devices and all other Internet traffic. As most of the end hosts are switching from fixed hosts to mobile end points, the characteristics of *mobility* pose many architectural challenges to the current Internet, including scalability, performance, robustness and security. As billions of wireless end-points become “first class” Internet devices and move across the Internet, *mobility and wireless access* issues (link bandwidth/quality fluctuation, multiple radios, disconnections, etc) will exacerbate to a large scale.

In addition, technology has also changed significantly since TCP/IP was designed. Improvements in computing and storage, fiber infrastructure and radio techniques enable a lot of new services and applications, which in turn stimulate the Internet's development and tune the underlying network infrastructure. For example, over the past few decades, Internet usage has been evolving to be dominated by content distribution. Content delivery networks are expecting to carry over half of Internet traffic by 2019 [40]. However, the original Internet was not customized for content delivery and retrieval, and hence the classic end-to-end model is unable to meet the requirements from emerging content services. Particularly, mobile content sharing - a increasingly popular application category - is facing technical difficulties as the existing Internet architecture cannot provide enough support either for mobility or for content delivery.

The Internet of Things (IoT) is another representative example of new services that need to be supported. As the result of the recent trend of extending the boundary of the Internet to include a wide array of untraditional computing devices, the Internet of Things makes the connection of the real world with the virtual world tighter than ever before. However, connecting many stand-alone IoT systems through the Internet brings many challenges, such as mobility, scalability, naming, resource constraints, inter-operability, security and privacy. Further, information collected by the IoT may be exposed to much wider audience through the Internet, which raises the severe concerns about the security and privacy.

Last but not least, as mentioned previously, a severe concern of today's Internet is security and privacy. One critical issue facing the Internet is that its security and privacy were not taken into consideration when it was designed and thus are not the core of the current Internet infrastructure. People have now realized the importance of security after paying a high price for various cyber attacks, such as XcodeGhost [135], Solar Sunrise [60], Morris Worm [104], OpenSSL Heartbleed [91], Melissa Virus [55] and Moonlight Maze [111]. On the other hand, the rise of social media has brought people's attention to the privacy issue. While sharing personal life with friends on the social networks, user privacy may also be violated and exposed to the unintended public. Not limited to the application level, the privacy leakage may span all layers of the Internet. As a user moves across networks, his location information and mobility pattern may be collected by malicious adversaries.

All of the challenges brought by various emerging services/applications/techniques require a fundamental change to the current Internet. The research community has suggested several clean-slate architectures to build the future Internet [106, 118]. MobilityFirst [7], XIA [1] and NDN [8] are representatives of those efforts. In this thesis, I will focus on MobilityFirst, whose main objective is to support large-scale mobility and wireless access through an identifier-based network architecture as well as provide new services needed for emerging mobile Internet application scenarios.

1.2 MobilityFirst: A Mobility-Centric Architecture for the Future Internet

MobilityFirst is a future Internet architecture that aims at providing fundamental solutions to the predicament the current Internet is facing. The prime design goals that drive the MobilityFirst are *mobility* and *trustworthiness*, the two major trends of the Internet. In order to achieve these two design goals, MobilityFirst breaks down them into a list of specific design goals, including seamless host and mobility, no single root of trust, intentional data receipt, proportional robustness, content addressability and evolvability. These design goals come together to solve the challenges faced by the current Internet.

The MobilityFirst architecture, shown in Figure 1.1, includes a set of system concepts and

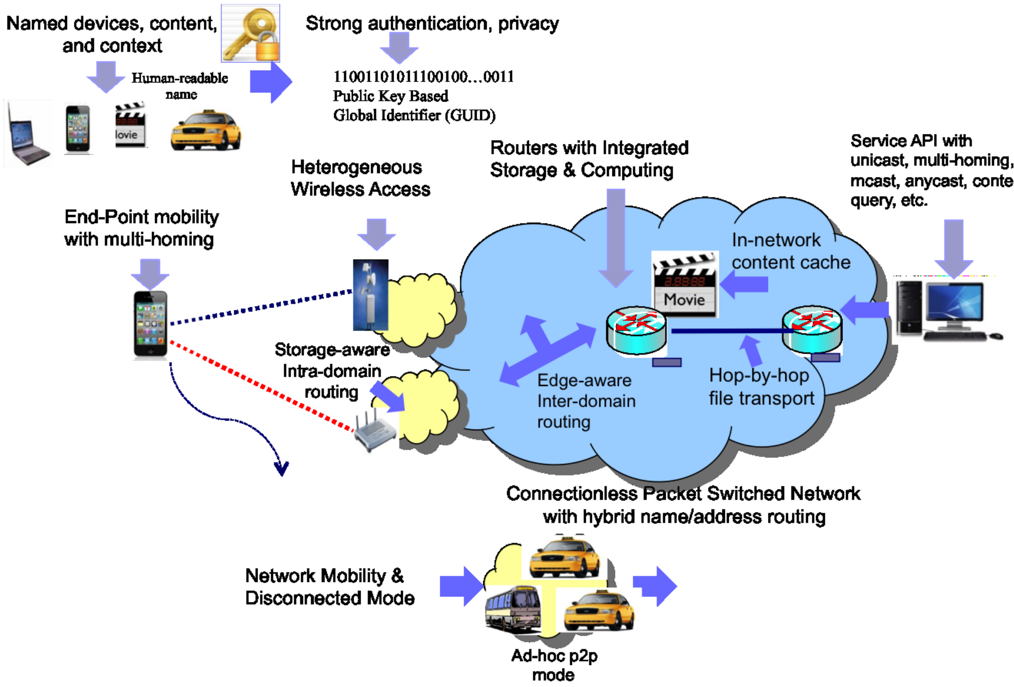

Named devices, content, and context
Human-readable name
Strong authentication, privacy
11001101011100100…0011
Public Key Based
Global Identifier (GUID)
Heterogeneous Wireless Access
Routers with Integrated Storage & Computing
Service API with unicast, multi-homing, mcast, anycast, content query, etc.
End-Point mobility with multi-homing
In-network content cache
Movie
Storage-aware Intra-domain routing
Edge-aware Inter-domain routing
Hop-by-hop file transport
Connectionless Packet Switched Network with hybrid name/address routing
Network Mobility & Disconnected Mode
Ad-hoc p2p mode

Figure 1.1: Major architecture features of the MobilityFirst network.

a suite of protocol elements. The major design features of this future Internet paradigm includes separation of names from addresses, security based on verifiable Globally Unique Identifier (GUID), name-based network service API, hybrid name/address based routing, storage-aware intra-domain routing and edge-aware inter-domain routing, mobile end-point with multi-homing, network Mobility/ad-hoc/disconnected modes, hop-by-hop transport and optional in-network computing services [110]. These architectural features can support or directly provide rich, efficient, reliable and trustworthy network services to network participants, such as end users, ISPs and services providers, as well as leave enough flexibility and evolvability to future service paradigms and demands.

Figure 1.2 shows that MobilityFirst introducs a new protocol stack and replaces the original “narrow waist” of the current Internet (i.e., TCP/IP) with a new name-based service layer, which consists of the *Name Certificate & Resolution Service (NCRS)* and the *Global Name Resolution Service (GNRS)*. This new service layer is centered on the concept of “flat” security-based GUIDs for network objects, a single abstraction that serves as both the network object’s identity and the public key. Various types of network entities, e.g., a smartphone, a person, a group of IoT devices, a piece of content or context, can obtain their globally unique identifiers

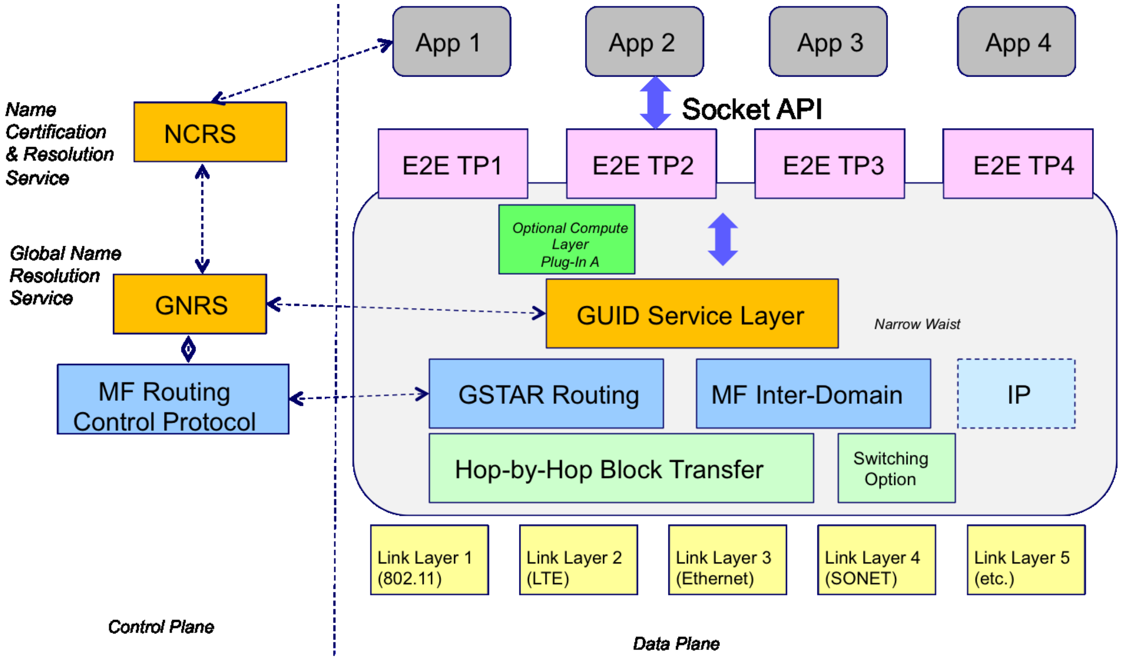


Figure 1.2: The protocol stack of the MobilityFirst architecture.

from the NCRS, which provides translation services between the human readable names and the public-key based globally unique identifiers. The GNRS provides a clean separation of the identifiers and the dynamic network address locators and supports on-the-fly binding of names to network addresses as needed for dynamic mobility. Therefore, the use of the GUID at the name-based service layer enables mobility-centric services at scale and builds up the foundation of a trustworthy network.

With the name-based service layer, the MobilityFirst architecture is able to relieve or at least mitigate many pains today's Internet is suffering. In the MobilityFirst network, due to the separation of the name and the network locator, users may request content by the content name directly, without bothering with the current network address. The routers handle the content request by querying the GNRS for a list of the up-to-date content storage locations in a timely fashion and then fetching the content from the nearest storage location [138]. This approach also handles content mobility well with support from the on-the-fly/late binding of names to addresses and the hop-by-hop storage-aware transport protocol MFTP [123]. The GUID, which is a public key, allows MobilityFirst to integrate trustworthiness as one of the basic properties of the architecture. Deriving the identifiers in a cryptographic approach makes many security measures, e.g., authentication, accountability and encryption, intrinsic to the MobilityFirst network thus facilitating further protection against various attacks.

1.3 Problem Statement

Different from the current Internet, MobilityFirst has taken security into consideration and treats it as one of the fundamental building blocks of the network's infrastructure. Specifically, the security mechanisms in the MobilityFirst networks are centered on the GUID and its associated name-based services (NCRS and GNRS), which provide the essential name resolution service and facilitate other network services and solutions, including routing, transport layer protocol, multi-homing, IoT solution, etc. Although the design of the GUID and the name-based service layer establishes a foundation for the MobilityFirst architecture to deploy its protocol stack with security properties, adopting the GUID alone cannot provide security and privacy with acceptable strength. The security measures that have been developed in this thesis take full advantage of the cryptographic characteristics of the GUID need to be designed and integrated into the MobilityFirst protocol suite and services.

Due to the importance of the GNRS in the context of MobilityFirst, the first and foremost important task is to secure the GNRS itself. Then, through the secured GNRS, security protections may be extended to other components of the network by integrating security guarantees provided by the GNRS and the specific security measures of individual network components. Therefore, starting from the GNRS, the work in this thesis first examines the GNRS protocol and integrates security measures to protect the GNRS against malicious attacks and exploits. Then, integration of regulation, in the form of access control at the GNRS, is also investigated and explored so that the GNRS can provide privacy-enhanced and fine-grained name resolution service to facilitate end users as well as other network components.

Today's Internet is much more complicated than it was back in the 1980s, when the transition from the ARPANET to the modern Internet began. At the inception of the Internet, it was purely a packet switching network. But now the Internet becomes a broad concept and is a huge mixture of many types of networks, e.g., cellular networks, vehicular networks, Wi-Fi networks, mobile ad hoc networks, wireless sensor networks, fiber networks, etc. As a future Internet architecture, the MobilityFirst architecture should take the characteristics of these heterogeneous networks into consideration and be able to accommodate their special needs. Hence, it is essential to understand those special networks and make sure the MobilityFirst infrastructure can also provide secure support to them. Therefore, we investigated two special networks, mobile ad hoc networks (MANET) and the Internet of things (IoT) systems. In MANETs, security concerns and network failures caused by the dynamics of the network topology become big obstacles for the MANET applications. Thus, we explored an overlay tunneling technique to improve the network resilience and robustness. On the other hand, we paid special attention to the Internet of Things because its applications, such as smart vehicles, smart home, smart city, etc, have been becoming more and more popular and reshaping people's daily lives. However, we still lack a unified solution to seamlessly connect the IoT to the MobilityFirst infrastructure. Hence, we also studied how to integrate IoT seamlessly and securely into the MobilityFirst architecture with the focus on bridging the gap of naming requirements between the global MobilityFirst network and the local IoT system.

1.4 Thesis Organization

This thesis explores security and privacy issues in the MobilityFirst network with the focus on the GNRS. It will also introduce future work on two specific scenarios. Overall, the thesis will be organized as follows:

Chapter 2 first introduces the GNRS and the NCRS. As the GNRS is one of the fundamentals for MobilityFirst, I take a deep look into the GNRS protocol suite, including the update protocol, the query protocol and the IP hole protocol. The security concerns and potential security weaknesses are investigated. Five attacks are identified and corresponding solutions are proposed. With the GNRS being secured, other network services/protocols may take full advantage of the naming service to enhance their performance and functionalities.

Chapter 3 explores the integration of more advanced functionalities into the GNRS. Access control as one of the most basic regulation method is examined. With access control, end users may be able to define their own policies at the GNRS to protect their privacy by only allowing desired network entities to contact them. In this chapter, a general access control scheme is introduced. As an example, I will focus on spatio-temporal access control mechanisms, where the access control policy is specified for particular location and time. There are some previous research results in this area but none of them consider verifying the spatio-temporal information, which thus could open a door for exploits. Our work fills this gap and provides secure spatio-temporal access control solutions to the GNRS. Moreover, a previously untouched special spatio-temporal access control scenario involving state transition is investigated and integrated into the GNRS. With our rich and fine-grained access control schemes, the GNRS empower end-users to express their needs and concerns in terms of access control policies.

Chapter 4 first investigates the challenges MANETs are facing, especially the security problems and network failures caused by the highly dynamic network topology. Then we presented and compared the related work, most of which struggle with the trade off between security protection and the overhead. Hence, we introduced an overlay tunneling scheme operating at the IP layer as a policy tool to mitigate the security threats as well as network failures in ad hoc networks. Our approach is a light weight touch since it does not change any underlying network protocol. Therefore it is flexible and applicable in a wide variety of network topologies and settings. Also, two optimizations are proposed to improve the performance if additional information is available. For example, if real time end-to-end traffic information is available, periodic re-evaluation technique is introduced to allow the scheme restore network operations in a timely fashion. Moreover, we present a trust assessment mechanism as another optional optimization, which may facilitate not only our overlay tunneling scheme but also other network protocols/functionalities, such as the routing protocol and access control.

Chapter 5 focuses on integrating the Internet of Things to the global MobilityFirst Infrastructure. In this chapter, we first compare two legacy IoT architectures and explain they cannot satisfy the current demands well. Then we define a unified framework supporting the Internet of Things based on the MobilityFirst infrastructure, featuring a IoT middleware design to bridge the gaps among heterogeneous IoT hardware, underlying network infrastructure and IoT applications. Naming and security challenges special to the IoT are discussed in Section 5.5.1. To resolve those challenges, we introduce a customized IoT Name Resolution Service

(IoT-NRS) at the IoT middleware. Moreover, to be consistent with the concept of GUID in the global Internett context, which serves as both the identifier and the public key, we explore a delegation-based key provision protocol in the local IoT context which establishes a symmetric membership key shared between a IoT device and a local group authority as the membership enrolment credential. Such a membership key identifies the unique IoT device in the local group as well as enables security protections, where functional keys (e.g., session key, attestation key, etc) may be derived from this membership key. With our unified MobilityFirst-based IoT architecture, various IoT applications can work independently or together seamlessly at scale, with proper security and privacy protection provided by the underlying MobilityFirst network and the IoT middleware.

Finally, the thesis concludes by summarizing the contributions of the research conducted, and identifies several security challenges associated with the future of the Internet, particularly as we move towards an increasingly connected Internet involving many heterogeneous devices and many different network modalities.

Chapter 2

GNRS security

2.1 Overview

A recent trend in clean-slate network design has been to separate the role of identifiers from network locators. An essential component to such a separation is the ability to resolve names into network addresses. One challenge facing name resolution is securing the name resolution service. This chapter examines the security of a clean-slate name resolution service suitable for mobile networking. We begin with a high-level threat analysis, and identify several types of attacks that may be used against name resolution services. We then present secure protocols that together form a secure GNRS. Specifically, we present a secure update protocol that allows users to update their network addresses as they migrate and that includes several checkpoints that prevents spoofing, collusion, stale identifiers and false identifier announcements. Since the primary function behind the GNRS is to respond to address-lookup queries, we also present a secure query protocol. Finally, we address the security risks associated with IP holes that can arise in the GNRS.

2.2 Background

The separation of identifiers from network locators is an important aspect underlying many future Internet architectures. By separating identifiers from the underlying addresses needed to locate and route to a network object, identifier-based networking [33, 98, 105, 113] supports simplified session management, multi-homing and device/user mobility. The separation between these two attributes makes it possible to avoid implicit or explicit binding of sources and destinations to the network's actual topology. In particular, the use of identifiers allows programmers to route to mobile devices based on an identifier like the International Mobile

Subscriber Identity (IMSI) number rather than on an IP address, and this simple architectural change leads to simpler networking primitives that allows for easier development when there is a high level of dynamism in the underlying network, as occurs in mobile services.

One challenge in designing an identifier-based protocol stack is the task of translating an identifier into a network address. When a user or application contacts the networking subsystem with an identifier that names a communicating entity (e.g. an intended recipient), this identifier must be translated into a network address that is then provided to the user/application and which could subsequently be used by various network services, such as routing. As part of the MobilityFirst [7] clean-slate architecture, a two-tier approach to name resolution was outlined (as shown in Figure 2.1).

Each network entity in MobilityFirst has at least three attributes: a user-level descriptor (i.e. a human readable name, such as Jim Smith's laptop), a network-level identifier (called a globally unique identifier, or GUID for short) and a *route*-able topological address (referred to as a network address, or NA for short). In MobilityFirst's two-tier approach to name resolution, a **Name Certificate & Resolution Service (NCRS)** is used to translate user-level descriptors into GUIDs, while a **Global Name Resolution Service (GNRS)** provides the mapping between GUIDs and the corresponding NAs.

Recently, as part of the MobilityFirst project, the design of a fast global name resolution service was presented [131]. The work in [131], however, focused on the balance between scalability and latency, with the goal of mitigating update/query latency, while also ensuring system consistency and supporting incremental deployment. The benefits of such a resolution service, however, are undermined when subjected to security threats, and consequently any clean-slate name resolution service should have security in mind as one of its founding principles. In this chapter we revisit the name resolution service presented in [131], and examine the security risks that name resolution services might face in an adversarial setting. We begin the chapter in Section 2.3 by presenting an overview of the two-tier approach to name resolution employed in MobilityFirst. We then proceed with a high-level threat analysis in Section 2.4, and then use this threat analysis to motivate the design of several secure protocols in Section 2.5 that ensure the secure operation of a global name resolution service. Throughout the chapter, our presentation is IP-centric to facilitate seamless transition from the current Internet architecture,

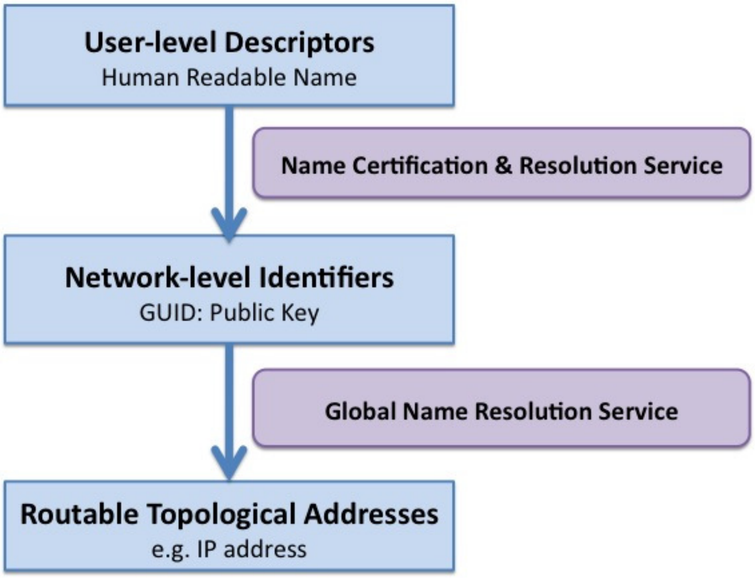

User-level Descriptors
Human Readable Name
Name Certification & Resolution Service
Network-level Identifiers
GUID: Public Key
Global Name Resolution Service
Routable Topological Addresses
e.g. IP address

Figure 2.1: Overview of the two-tier name resolution involved in MobilityFirst.

and we provide a detailed walk-through behind each step, explaining why specific details were included with an emphasis on addressing potential security exploits. Finally, we conclude the chapter in Section 2.6.

2.3 Two-tier Name Resolution

The motivation behind the separation of human readable names from GUIDs in the MobilityFirst project is that the NCRS and GNRS operate at different time-scales: the translation between a human-readable name and a GUID is not likely to change as rapidly as the translation between a GUID and a corresponding network attachment point (particularly in highly mobile settings). Further, MobilityFirst employs a flat, location-independent identifier space where public keys are used as GUIDs (as in AIP [26], HIP [98], and ROFL [34]), and each end host, such as laptops, mobile phones, servers and virtual machines, can have a GUID. Each GUID can associate with one or more network addresses (NAs) that it attaches to. As an example, the NAs of a multi-homed laptop might include the NA of its 3G service provider as well as the NA of its WiFi network. The roles of the NCRS and GNRS are depicted in Figure 2.2.

The NCRS is involved in two main functions: (1) it provides a translation between a human-readable name and the corresponding GUID; and (2) it acts as a certificate authority in distributing the GUIDs, which are themselves public keys and hence must be appropriately packaged for use. In short, the NCRS must provide services analogous to those provided by a Public Key Infrastructure (PKI). In MobilityFirst, users may either generate their own public keys (for use as GUIDs) and submit these to the NCRS for registration, or may contact the NCRS to acquire public keys (for use as GUIDs). Consequently, the NCRS publishes approved cryptographic suites, object categories and object description formats, as well as allows users to self-certify themselves. Following standard certificate formats such as those discussed in [120, 127], users may generate their own certificates and then submit them to the NCRS, who engages in a challenge-response protocol to verify that the submitter possesses the corresponding private key. In the context of MobilityFirst, the NCRS certificates follow the general format of X.509 certificates, and include the following fields: version, serial number, signature algorithm, validity, public key, object category code, object description (human readable keywords) and appropriate signatures. The object description field is useful in the context of global networking, as it allows for the description of network roles/functions (such as whether a device is a border gateway router for a particular Autonomous System (AS)). In this chapter, we make a generic assumption that a network entity's locator consists of an AS number as its network address, as well as a local address within that network/AS. Hence we use AS number and NA interchangeably in this chapter.

Since the GUID's correspondence to public keys, we propose the use of elliptic curve cryptography (ECC) [64] and note that the curve P-256 of FIPS PUB 186-3 DSS is appropriate for our purposes [85]. In particular, when choosing a cryptographic suite, both security strength and efficiency need to be considered. According to the National Security Agency (NSA), elliptic curve public key cryptography using the 256-bit prime modulus elliptic curve, as specified in FIPS 186-3, and the SHA-256 hash function provide adequate protection for classified information up to the SECRET level [15]. Moreover, ECC P-256 has cryptanalytic security strength comparable to RSA with a key size of 3072 bits, yet ECC P-256 requires a key size of only 256 bits, which translates into lower computational costs and higher efficiency [129]. Lastly, it is widely accepted that ECC is a strong and more efficient approach to public key cryptography that is suitable for resource-constrained environments, such as the mobile networks that we are concerned in this chapter. Finally, we note the NCRS performs conventional tasks associated

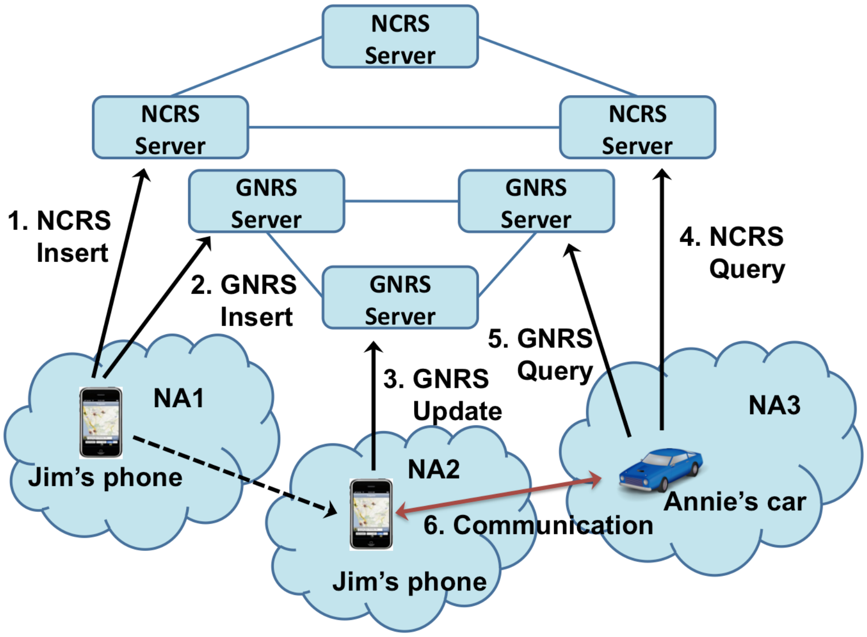


Figure 2.2: The MobilityFirst approach to name resolution involves the Name Certificate & Resolution Service (NCRS) and the Global Name Resolution Service (GNRS). The NCRS serves the role of a certificate registration service that associates human-readable names to GUIDs, while the GNRS translates GUIDs into network addresses

with a certificate authority, such as the insertion, update and revocation of the GUID/digital certificates. Since these mechanisms are a straight-forward application of well-known PKI techniques, we will not focus on their specification in the remainder of the chapter.

We now provide a high-level description of the functions performed by the GNRS. To perform the mapping service for a given GUID, the GNRS applies $K (K > 1)$ hashing functions onto it to produce a list of K locators, which are IP addresses in today's Internet, and stores the GUID→NA mapping in the ASs that announce those network addresses. By doing so, the GNRS spreads the GUID→NA mappings amongst ASs, such that an AS will host mappings of other ASs, as well as have its mappings hosted by others. This design leverages the routing infrastructure to reach the hosting AS in a single overlay hop; it does not require a home agent, unlike mobile IP and existing cellular networks. There are three types of events in GNRS: insert, update and query. When a user A (with a GUID) joins a network for the first time, it sends an insert message to the GNRS (a distributed system) reporting mapping information of its GUID and network address. When A moves to another network, it sends an update message to the GNRS to report the new mapping. If another user B wants to communicate with A, it sends a query message to the GNRS asking for A's network address before their communication, and the GNRS will return A's mapping to B [131].

2.4 GNRS Security Concerns

Although the design of the NCRS and GNRS are effective in managing translations between different identifiers, these two separate systems may themselves be the target for attacks and exploits. Securing the design of a certificate authority, which is essentially the role played by the NCRS, involves well-known techniques and we consequently refer the reader to [23, 128, 130] for a survey of such mechanisms. The GNRS, however, is a new network service that manages very dynamic GUID-to-NA mappings. In particular, the lack of hierarchies in GNRS implies that techniques used to secure the inherently hierarchical Domain Name System (DNS) [37] cannot be leveraged to secure GNRS. GNRS is fundamentally distributed and tied to the functioning of the underlying network, and hence warrants its own security investigation.

Identifying and understanding potential security threats is a necessary starting point for designing a secure network architecture. We now present a security analysis for GNRS, and highlight a collection of threats and risks that may be faced. The threats that GNRS might face include:

- Unauthorized Access: an intruder gains access or gathers information from a resource it is not entitled to. As a consequence, an adversary may examine, remove or even modify confidential information.

- Masquerading: an intruder is able to mimic an authorized user or network process. As a result, the intruder may forge signatures, or impersonate a source address.

- Information Modification: unauthorized or malicious information is injected into the network or its resources. This threat can cause service failures, false information transmissions or network operations, etc.

- Message Manipulation: an adversary manipulates the message exchange process between network entities. Such manipulation may involve replay, rerouting, misrouting and deletion of messages.

- False Traffic Insertion: Insertion of false traffic into the network can place an additional burden upon the network and consume precious network resources. The consequence might be an increase in delay and performance degradation for network services and applications.

Consequently, major risks associated with these security threats are identified as below:

- Illegitimate Resource Consumption: an unauthorized user gains access to resources it is not allowed to and consumes networks resources. Threats such as unauthorized access, masquerading, information modification, message manipulation can lead to this risk.
- Pilfering of Service: an adversary is able to use service that it does not have the privilege of usage. This risk might be a result of unauthorized access, masquerading, information modification or message manipulation.
- Denial of Service: this risk makes network service or resource unavailable to its intended users. This is a serious risk that may result from unauthorized access, masquerading or message manipulation.

As summarized earlier, GNRS has three basic operations: insert, update and query. Among them, GNRS insert and update are quite similar, with insert being a first-time update. Therefore, in examining security, our attention will be restricted to the update and query operations in the remainder of this chapter.

The basic process involved in GNRS update is shown in Figure 2.3. Suppose user A belongs to network NA1 and needs to update its mapping ($GUID_A, NA1$). A sends an update message to its local router who will then forward the message to the border gateway router. The border gateway router hashes A's $GUID_A$ to get the mapping's storage location X in network NA2, which is an IP address, and then sends the mapping contained in the update message to X. X stores the mapping and waits for other users in the (global) network to query. In GNRS's design, to improve performance and efficiency, the border gateway router performs K different predefined hash functions to find K different mapping storage locations. As a result, there are K mapping replicas in the global network. When another user B queries A's mapping, it can

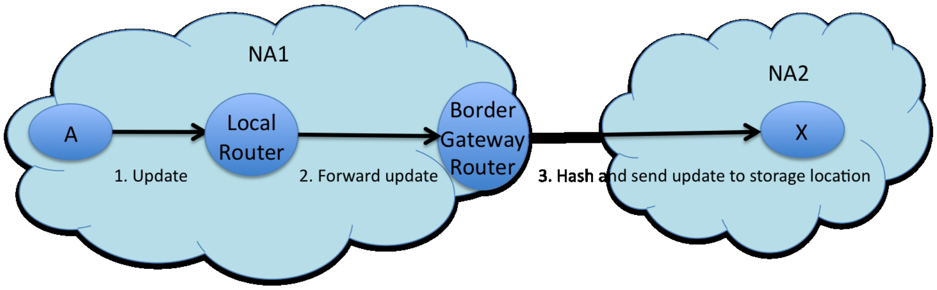


Figure 2.3: GNRS Update Scheme.

receive the mapping from the nearest replica and therefore lower the query delay. A benefit of distributing multiple mapping replicas is that it prevents single-point-of-failure risks, and thus enhances the reliability and robustness of GNRS.

Unfortunately, if the user sends the mapping message $(GUID, NA)$ to the local router and follows the simple procedure just outlined, the update procedure cannot be guaranteed to proceed correctly. In particular, attackers might come from anywhere in the network and attack cooperatively. We now outline five possible attacks against GNRS.

1. The GUID spoofing attack is a masquerading threat, where a malicious user A claims another user B's GUID and attempts to associate it with A's own network address NA_A, by announcing the mapping $(GUID_B, NA_A)$. The consequence of this attack is a denial of service as it can cause traffic directed for B to be directed to A's network address.

2. The stale mapping attack is a message manipulation attack involving a malicious GNRS server. In this attack, if a device moves and issues an update, the malicious GNRS server can purposely ignore the update and claim it still has the most recent mapping. Perhaps worse, a GNRS server can selectively choose which (possibly stale) mapping to give out during queries. The result is a denial of service.

3. The third potential attack, false announcement attack, is an information modification attack that results in illegitimate resource consumption. User A, which is in network NA1, claims its $GUID_A$ binds to a different network address, $(GUID_A, NA2)$. Thus A can direct its traffic to network NA2, which causes NA2 network resources to be consumed. To protect against the third type of attack, one can let the network, such as the local router and/or border gateway router, sign update messages to the GNRS after it verifies whether the user belongs to

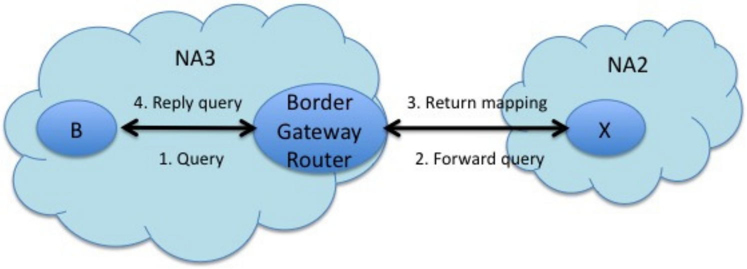


Figure 2.4: GNRS Query Scheme.

it. However, this is not enough because it cannot prevent the fourth attack, the collusion attack.

4. The collusion attack is an example of an information modification attack in which a malicious user, its network and the location where the mapping is stored collude with each other. The objective behind the malicious collusion is to allow for a fake mapping involving a false network address to pass the verification and become stored in the storage place.

In the GNRS query scheme, assume user B is in network NA3 and wants to query A's mapping. B first sends out the query request message to its border gateway router. The router runs the K predefined hash functions on A's GUID as used in A's update process to find the nearest replica storage location X (found by examining the K hash results). The border gateway router forwards the query request message to X. X returns the mapping to B's border gateway router, who will then reply to B's query with the mapping information. Figure 2.4 shows the query scheme. Interestingly, we note that by securing the update scheme, GNRS query itself becomes less-open to exploitation since all of the mappings would have been correctly verified prior to insertion into GNRS.

5. The final attack we mention is a family of information manipulation attacks that we collectively label as BGP churn exploits. Problems may also arise due to network topology changes, such as BGP churn [51]. Although hashing is used to find the mapping storage location in GNRS, the hash result might fall into an IP hole (i.e. an IP address that is not announced by any AS). For this reason, the IP hole protocol was proposed to solve this problem in [131]. The scheme involves rehashing the hash result if the result falls into IP hole until a valid IP address is found. If it still falls into an IP hole after M attempts at rehashing, then the scheme chooses a deputy AS who announces the IP address that has the minimum IP distance to the current hash value. In this scheme, BGP churn, such as announcing a new IP prefix or withdrawing an

IP prefix, might cause security weaknesses. Take withdrawing the IP prefix for example, and suppose network NA2 wants to withdraw its IP prefix. This will make the IP prefix of NA2 become an IP hole and the mappings stored in NA2 become orphan mappings. Therefore, before withdrawing IP prefix, NA2 runs IP hole protocol to find locations to insert orphan mappings and moves the mappings there. Without a secure scheme, a malicious user can insert many fake orphan mappings to a target network to exhaust that network's storage resources, while also causing the GNRS to report false mappings to queries. We will specifically address this issue in section 2.5.3.

2.5 Secure GNRS Protocol

We now explore a secure GNRS service that is targeted at addressing the security concerns outlined in Section 2.4. As a starting point, we assume that the NCRS plays the role of a PKI and a clock synchronization system is available. In our design, the public keys used are equivalent to the user's identifier the GUID, and thus we use these terms interchangeably. A 256-bit long GUID is sufficient to keep birthday attacks infeasible. If the desired probability of random collision is kept as low as 10^{-6}, the pool of available GUID will be about 4.8×10^{35} [127], which is more than large enough for GUID assignment for the foreseeable future.

2.5.1 Securing GNRS Update

The objective behind GNRS update is to allow users to update their network address as they migrate across the global network. The main network components involved in the GNRS update scheme are the user, the local router, the border gateway router and the DHCP server. Here, the DHCP server provides verification of a user's IP address [49]. The update protocol details are shown in Figure 2.5. Within network NA1, A is a user who wants to inform GNRS of its new mapping $(GUID_A, NA1)$, L is A's local router, G is the border gateway router for NA1, D is the DHCP server with which G can verify A's network address, and X is the storage location in network NA2 responsible for keeping track of the mapping. There are eight messages exchanged during the update process:

1. A → L: $\{GUID_A, [N_a, GUID_L, (GUID_A, NA1, T_A, E_A)_{A^{-1}}]_L\}$

There are two timestamps involved in the message exchange: the mapping generation time T_A (corresponds to when the new mapping was instantiated) and the expiration time E_A (which corresponds to when this new mapping will expire). These timestamps are included with the GUID and network address, and necessitates that A must update or renew its mapping before expiration. This protects against stale mapping attacks, which were mentioned in Section 2.4. Since the mapping has an expiration time, the freshness of the mapping is guaranteed: users who query A's mapping can check these timestamps to ensure the mapping is not stale. The use of timestamps for freshness, however, introduces a tradeoff between security and efficiency. If E_A is too large, then it is possible that a prior mapping will still be valid after user A moves to another network and updates its new mapping with a new generation time and expiration time. This is problematic in the case of a malicious mapping storage location X. An untrustworthy X may give users an old mapping (that has not expired) instead of the up-to-date mapping. On the other hand, if E_A is too small, then the likelihood of a successful attack is low, the impact minimal, but the overhead will be large because A must renew its mapping frequently.

In this step, $(GUID_A, NA1, T_A, E_A)$ is signed by A's private key A^{-1} to protect against a GUID spoofing attack. An adversary cannot claim to be A and launch a GUID spoofing attack because only A knows its private key. Other users can verify the signature since they know A's public key $GUID_A$. Also a nonce N_a is added in the update message to prevent replays. $GUID_L$ is added for receiver verification, and the message is encrypted by L's public key so that only L can read the message. The update message also contains $GUID_A$ so that the receiver L can verify the sender.

2. L → G: $\{GUID_L, [N_l, GUID_A, GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1}}]_{L^{-1}}\}_G$

   L first verifies whether the mapping $(GUID_A, NA1)$ is correct. This is the first checkpoint to protect against the false announcement attack mentioned in Section 2.4. If the mapping is correct, then L forwards A's mapping to the border gateway router G for NA1. Similar to message 1, a nonce N_l is added and message 2 is signed by L's private key and subsequently encrypted by receiver G's public key.

3. G → D: $\{GUID_G, [N_g, GUID_A, (GUID_A, NA1, T_A, E_A)_{A^{-1}}]_D\}$

   The border gateway router G is the second checkpoint to prevent the false announcement attack. In particular, this step is intended to handle cases where A and L collude with each other. To do this, G sends message 3 to the DHCP server D to verify A's network address.

4. D → G: $\{N_g + 1, (GUID_A, NA1, T'_A, E'_A)_{D^{-1}}\}_G$

   After receiving G's query, D replies with A's current network address and its valid time period (T'_A, E'_A). Also, $N_g + 1$ is included and serves as the ACK corresponding to the nonce N_g.

5. G → L: $\{N_l + 1\}_{G^{-1}}$

   If the reply from D matches A's update, G will return an ACK $(N_l + 1)$ to L implying the acceptance of A's update, which has been forwarded by L.

6. L → A: $\{N_a + 1\}_{L^{-1}}$

   L returns $(N_a + 1)$ to A, informing A of the successful update to A's mapping.

7. G → X: $\{GUID_G, \{[GUID_A, N'_g, (GUID_A, NA1, T_A, E_A)_{A^{-1}, G^{-1}}]_{G^{-1}}\}_x\}$

   G hashes A's GUID by using K predefined hash functions to obtain the IP addresses of K different storage locations responsible for storing A's mapping. Before sending A's mapping to any storage location X, both G and A sign the mapping $(GUID_A, NA1, T_A, E_A)$ so that other users can verify whether this mapping has been checked by network NA1. This prevents a collusion involving A, L, G, and X working together to create a fake mapping, such as $(GUID_A, NA3, T_A, E_A)_{A^{-1}}$, which would have otherwise passed the verification and been stored in X. Since G signs the mapping, and since a border gateway router is only responsible for its own domain, any user who would have queried $GUID_A$ can check whether G is actually NA3's border gateway router by querying the NCRS, which provides the user a description of G's responsibilities.

8. X → G: $\{N'_g + 1\}_{X^{-1}}$

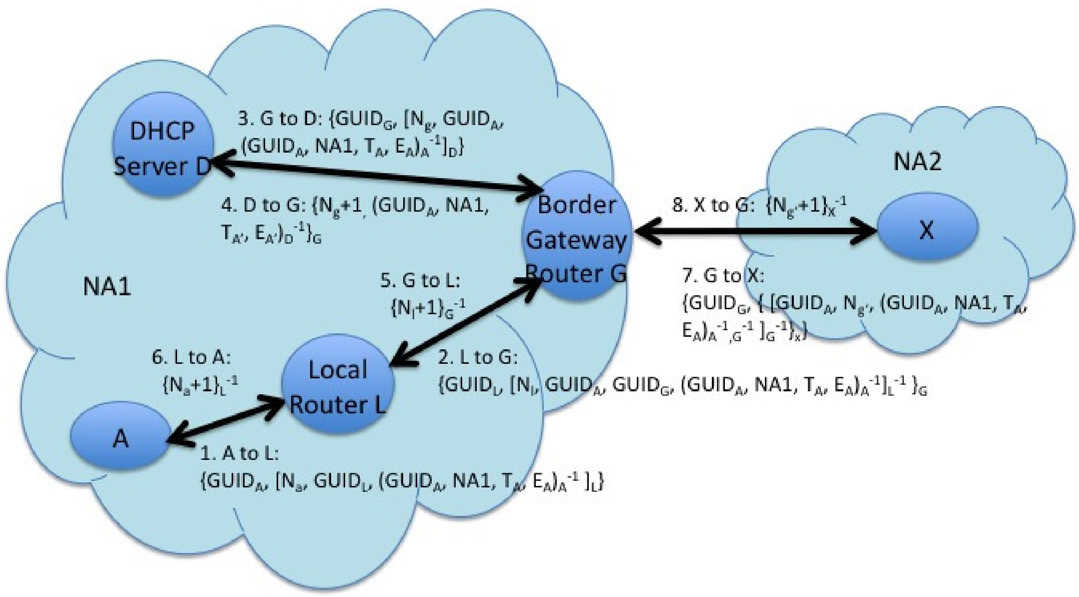

DHCP Server D
3. G to D: $\{GUID_G, [N_g, GUID_A, (GUID_A, NA1, T_A, E_A)_{A^{-1}}]_D\}$
4. D to G: $\{N_g+1, (GUID_A, NA1, T_{A'}, E_{A'})_{D^{-1}}\}_G$
Border Gateway Router G
8. X to G: $\{N_{g'}+1\}_{X^{-1}}$
NA2
X
NA1
5. G to L: $\{N_l+1\}_{G^{-1}}$
7. G to X: $\{GUID_G, \{[GUID_A, N_{g'}, (GUID_A, NA1, T_A, E_A)_{A^{-1}, G^{-1}}]_{G^{-1}}\}_X\}$
6. L to A: $\{N_a+1\}_{L^{-1}}$
Local Router L
2. L to G: $\{GUID_L, [N_l, GUID_A, GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1}}]_{L^{-1}}\}_G$
A
1. A to L: $\{GUID_A, [N_a, GUID_L, (GUID_A, NA1, T_A, E_A)_{A^{-1}}]_L\}$

Figure 2.5: GNRS Update Protocol.

X returns an ACK to G indicating the mapping is accepted and the update process is finished. Finally, A's mapping is stored at X as $\{GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1}, G^{-1}}\}$.

2.5.2 Securing GNRS Query

GNRS query provides a means by which a user can acquire the network address corresponding to a known GUID. In the GNRS query, the involved network components are the user making the query, the border gateway router, and the mapping storage location. The query protocol is shown in Figure 2.6. To start, assume user B, who is in network NA3, wants to query the GNRS to find A's mapping. Suppose that S is the border gateway router for NA3, NA2 is mapping storage location that is the closet to NA3 among the K replicas identified by the K hash functions, and in NA2 the mapping is stored at X. In our secure GNRS query protocol, there are only four messages exchanged during the query process:

1. B → S: $\{GUID_B, [GUID_S, N_b, (GUID_A, GUID_B, T_B)_{B^{-1}}]_S\}$

   B sends S the query for A's GUID, $GUID_A$, together with its own identifier $GUID_B$ and timestamp T_B. The request is first signed by B's private key B^{-1}, then the receiver S's GUID is added for verification purposes and a nonce N_b is added to prevent replay

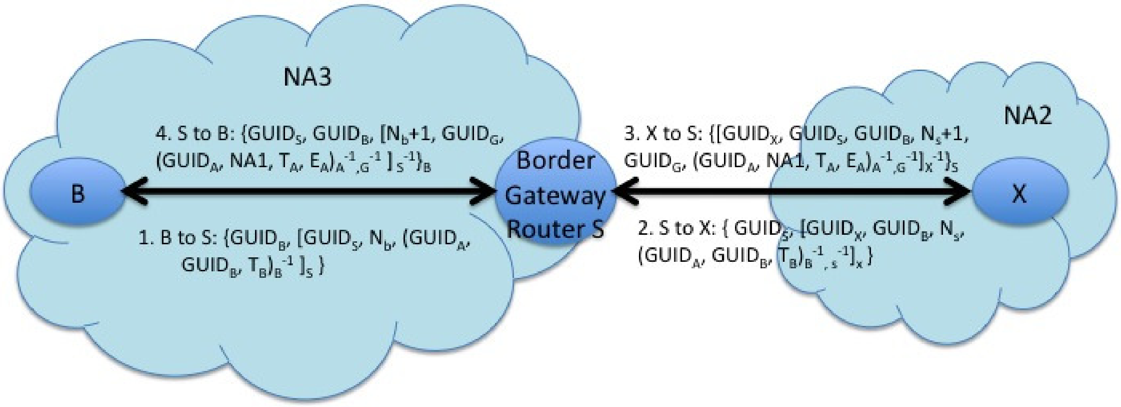

NA3
4. S to B: {$GUID_S$, $GUID_B$, [N_b+1, $GUID_G$, ($GUID_A$, NA1, T_A, E_A)$_{A^{-1},G^{-1}}$]$_{S^{-1}}$}$_B$
B
Border Gateway Router S
1. B to S: {$GUID_B$, [$GUID_S$, N_b, ($GUID_A$, $GUID_B$, T_B)$_{B^{-1}}$]$_S$ }
3. X to S: {[$GUID_X$, $GUID_S$, $GUID_B$, N_s+1, $GUID_G$, ($GUID_A$, NA1, T_A, E_A)$_{A^{-1},G^{-1}}$]$_{X^{-1}}$}$_S$
NA2
X
2. S to X: { $GUID_S$, [$GUID_X$, $GUID_B$, N_s, ($GUID_A$, $GUID_B$, T_B)$_{B^{-1},S^{-1}}$]$_X$ }

Figure 2.6: GNRS Query Protocol.

attacks. The message is finally encrypted by S's public key.

2. S → X: $\{GUID_S, [GUID_X, GUID_B, N_s, (GUID_A, GUID_B, T_B)_{B^{-1},S^{-1}}]_x\}$

   S verifies the request and then performs the same K predefined hash functions (as used in the GNRS update process) on $GUID_A$. S chooses the nearest one from the K hash results, which is X in NA2 for this example, and forwards the query request to X.

3. X → S: $\{[GUID_X, GUID_S, GUID_B, N_s + 1, GUID_G,$
   $(GUID_A, NA1, T_A, E_A)_{A^{-1},G^{-1}}]_{X^{-1}}\}_S$

   X verifies the request and returns A's mapping $\{GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1},G^{-1}}\}$ to S.

4. S → B: $\{GUID_S, GUID_B, [N_b{+}1, GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1},G^{-1}}]_{S^{-1}}\}_B$

   S forwards A's mapping to B. B can check the mapping by verifying the timestamps and the signatures of A and G. Now the query process is finished.

2.5.3 Securing IP Hole Protocol

Hashing is used in the GNRS update and query process to find mapping storage locations. Due to the fragmentation of IP address space, the hashing result could possibly fall into an IP hole, which is not announced by any AS. As a result, in such an event, a valid IP address cannot be found to store the mapping. In this case, the IP hole protocol [131] is performed to find the mapping storage location. Basically, the IP hole protocol involves continuous hashing until a valid mapping storage address is found. The high-level details are summarized below:

1. If the hash result falls into an IP hole, rehash (up to M times) the hashing result until a valid IP address IP_x is found. IP_x is the mapping storage location.
2. If it still falls into an IP hole after M rehash attempts, choose a deputy AS who announces the IP address IP_x that has the minimum IP distance to the current hashing value. Assume A and B are two n-bits IP addresses, the IP distance between A and B is defined as:

$$IP_{dist}(A,B) = \sum_{i=0}^{n-1} |A_i - B_i| \times 2^i$$

The IP hole protocol can be used to handle changes in prefixes (analogous to the changes in prefixes associated with BGP churn), which directly influences GNRS update and queries. Prefix change announcements either can be due to an AS withdrawing a previously announced prefix, or due to an AS announcing a new prefix.

Orphan Mapping Insert

The Orphan mapping insert step addresses the problem of a prefix being withdrawn by an AS. In Figure 2.7, user A's GNRS mapping $(GUID_A, NA1)$ is stored at X in NA2, while Y is in network NA4. Suppose NA2 wants to withdraw its IP prefix, which will make its current IP prefix become an IP hole, and hence A's GNRS mapping will become an orphan mapping since others will not be able to reach storage location X associated with this mapping. To solve this problem, before withdrawing its IP prefix, NA2 must run the IP hole protocol and move A's mapping to another storage location. Orphan Mapping Insert, involves:

1. X → Y: $\{GUID_X, [R, GUID_Y, (N_X, GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1},G^{-1}})_{X^{-1}}]_Y\}$

   The IP hole protocol is called to find a place to insert A's mapping. First, X applies the appropriate hash function to $GUID_A$. If this hash result falls into an IP hole or NA2 itself, X will hash the hash result again (following the IP hole protocol) until it finds a valid IP address or a deputy IP_x. For the sake of discussion, let us assume that this new location corresponds to Y, which is located in network NA4, in Figure 2.7. Suppose

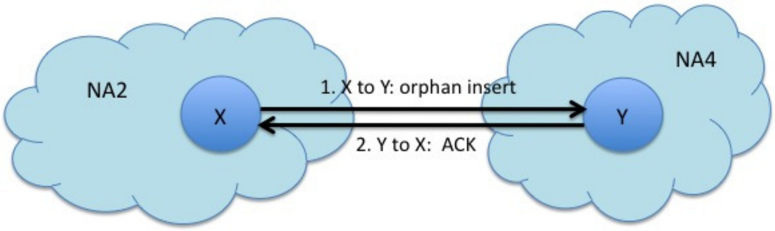


Figure 2.7: Orphan Mapping Insert.

the amount of times needed to rehash $GUID_A$ is R (we note $R \leq M$, where M is the maximum number of rehashes allowed in the IP hole protocol). After finding Y, X signs A's mapping $\{GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1},G^{-1}}\}$ together with the nonce N_X. Then X includes R and the receiver Y's identifier $GUID_Y$, and sends the message to Y after encrypting with Y's public key. R is added for the convenience of Y's verification in the next step.

2. Y → X: $\{N_X + 1\}_{Y^{-1}}$

   Y should not simply accept the orphan mapping insert message from X without verification – otherwise, attackers may take advantage of the lack of verification to insert fake mappings. For example, a malicious user E in network NA5 may create fake mappings and claim it is the original legitimate storage place for these mappings. The attack would proceed by E then asking NA4 to accept these mappings under the pretense that NA5 is withdrawing its IP prefix. If NA4 accepts all those fake mappings blindly, its own network resources might be exhausted and become victim to a false announcement attack, with the consequence being resource consumption and denial of service. Thus, before accepting the mapping, Y also runs the IP hole protocol on the GUID in the mapping to verify whether the sender X was a legitimate storage location for this mapping. If the insert message passes the verification, Y stores the mapping and replies with an ACK $(N_X + 1)$ to X. Otherwise, Y will reject the insert.

Mapping Migration

The Mapping Migration process happens when an AS claims a new IP prefix. The newly announced prefix might have previously been an IP hole that is now being claimed by the AS.

As a result, during the process of GNRS update or insert, yet before the prefix announcement, users in the global network will have called the IP hole protocol and avoided storing GNRS mappings at locations with this particular IP prefix. However, after the prefix announcement, this IP prefix is no longer an IP hole and therefore will receive queries from other users for mappings that it does not have. The Mapping Migration scheme, shown in Figure 2.8, is designed to solve this problem.

Assume A's GNRS mapping is currently stored in X within NA2 after performing the IP hole protocol. Now assume that network NA6 announces a new IP prefix and that storage location Z is in NA6. Later Z receives a GNRS query for A's mapping, but it does not have the requested mapping due to the original IP hole status prior to the new prefix announcement. To remedy this, Z performs the following procedures to fulfill the query service request:

1. Z → X: $\{GUID_Z, [GUID_X, (N_Z, GUID_A)_{Z^{-1}}]_X\}$

   After receiving a query for A's mapping, which it does not have, Z checks whether it should have the mapping by hashing $GUID_A$ or calling the IP hole protocol. During the continuous hashing (up to M times), if the hash result falls into Z's newly announced prefix before it falls into another valid IP address, which should be the current mapping storage location X for A's mapping, then Z is responsible for A's mapping and should start the mapping migration process immediately. Consequently, Z sends out a migration request to X. The request contains A's GUID and a nonce N_Z, which are signed by Z's private key and encrypted by X's public key. Z can determine X by following through with the IP Hole Protocol, and querying the NCRS for X's certificate.

2. X → Z: $\{GUID_X, [N_Z + 1, GUID_Z,$
   $(GUID_G, (GUID_A, NA1, T_A, E_A)_{A^{-1}, G^{-1}}, N_X)_{X^{-1}}]_Z\}$

   X performs the IP hole protocol to verify whether the migration request is legitimate. If one of the hashing results matches Z and the number of required hashes needed to arrive at Z's address is less than the number of hashing attempts needed to arrive at X's address, then X will accept the request and send A's mapping to Z.

3. Z → X: $\{N_X + 1\}_{Z^{-1}}$

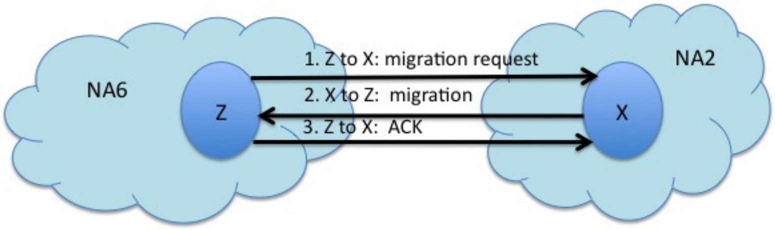


Figure 2.8: Mapping Migration.

Z returns an ACK $(N_X + 1)$ to inform X that it received the mapping successfully. This indicates the end of the mapping migration process.

2.6 Conclusion

Many clean-slate network architectures promote the separation of identifiers from routable addresses. Such a separation promises to better facilitate mobile networking if a fast global name resolution service can be developed. Although preliminary evidence suggests that fast global name resolution is possible for flat (non-hierarchical) addresses, one critical challenge facing such a service is the assurance that this name resolution service operates in a trustworthy manner in presence of a variety of security threats. This chapter has examined the security of such a name resolution service. We started by presenting a high-level threat analysis, where several attack categories were identified. Using the threat analysis as a foundation, we presented the design of several secure protocols that would constitute a secure global name resolution service. A secure update protocol that allows users to update their network addresses as they migrate was presented and includes several checkpoints that allows the protocol to prevent spoofing, collusion, stale identifiers and false identifier announcements. Since the primary function behind a name resolution service is to respond to address-lookup queries, we also presented a secure query protocol. Finally, we noted that IP holes represent a possible attack vector, and consequently devised a protocol that addresses security risks associated with IP holes that arise in a global name resolution service.

Chapter 3

GNRS access control

3.1 Overview

The lack of access control and regulation in the current Internet has resulted in many security and privacy problems. To prevent unauthorized access to protected information, integrating access control into future Internet design is crucial. In this chapter, a suite of access control mechanisms that are well-suited for the mobile Internet are introduced into the MobilityFirst infrastructure. The emphasis of the proposed methods is on supporting new spatio-temporal access control, which can be a powerful new paradigm for security in mobile systems. We choose to enforce the proposed access control schemes at the GNRS due to its importance to the MobilityFirst network: it is the first line of defence as one first needs to contact the GNRS for target's network address before it can reach the target.

3.2 Background

The current Internet was not developed with security and regulation as its core principals and as a result there have been many security and privacy breaches throughout the Internet's history. Many of these security flaws resulted from a lack of regulation to important information. Due to the lack of access regulations, anyone in the network can query various network services to lookup a wide variety of information, which may be exploited to conduct malicious attacks. A typical case where the improper exposure of user information leads to an attack is the *email-bomb* style of Denial-of-Service (DoS) attack. In this attack, the attacker first obtains the victim's email address and then abuses the network by sending a huge volume of email (spam) to the victim's email address in an attempt to exhaust the resources of the host victim's email server. This attack succeeds as a result of the adversary having easy access to the victim's email address, and because the server is unable to refuse incoming traffic. If the Internet, instead, had regulated access to the user's relevant information (such as name, organization, email and address) in the first place, then this attack would have been less likely to occur.

With the increased need to improve security, there has been an effort to fix the security design flaws in the Internet. However, the solutions developed so far are basically patches added to the Internet protocol stack, and only target specific network components. For example, firewalls and filters are designed for application layer (layer 7 in the OSI model) services to prevent illegitimate access to the user data; the Internet Protocol Security (IPsec) suite secures Internet Protocol (IP) communication at the Internet Layer (layer 3 in the OSI model) while the Transport Layer Security (TLS) suite operates at the transport layer to protect communication traffic. The fragmented security and regulation solutions make the Internet architecture chaotic and do not provide a complete and consistent protection against a wide variety of attacks. The inconsistency of current security methods and lack of a system-wide solution that regulates access to certain information causes the Internet to be open to attacks regardless of the patches applied. As a result, developing clean-slate future Internet architectures that apply regulation is desirable. [1, 7, 8] are representative future Internet architecture projects that include security in their design principles, and are integrating security as an essential functionality of the future Internet. In this chapter, we have the philosophical viewpoint that if we can restrict adversarial access to information needed to launch an attack, whatever that attack may be, then we increase the difficulty for attacks to be launched. Thus, we complement existing security methods and will ultimately arrive at a better and more trustworthy future Internet architecture.

Another trend for improving the future Internet is how *mobility* is handled. In the traditional Internet architecture, which was originally designed for communications between static end hosts, the IP address has a dual-role, serving as both the identifier and the locator of a network entity. This conflating usage of the IP address in the current TCP/IP stack makes it unable to reflect the actual network attachment point or the physical location of a mobile device while also keeping the identity of the mobile device unchanged. Some efforts, such as [107, 119], have attempted to address this issue in the TCP/IP stack. Unfortunately, these patches to the Internet architecture are inefficient and suffer from scalability, complexity and security concerns.

Introducing an independent name space to separate the identifiers of network objects and the locators is an alternative Internet mobility solution as proposed in [1,7,26,34,98]. For example, in the MobilityFirst future Internet architecture [7], a network entity is allowed to self-certify its identifier, the globally unique identifier (GUID), and the Global Name Resolution Service (GNRS) provides a mapping between GUIDs and network addresses. In addition to supporting mobility, the approach of separating the identifier and the locator also facilitates multi-homing, multicast and authentication. However, the shortcoming of the current identity-based communication is that it lacks any enforcement of how the GNRS responds to queries. The end user is unable to protect its information (the mapping of identifier and locator) from unfavourable access and therefore cannot prevent unwanted incoming traffic.

The goal of our work is to eliminate open access to a wide array of information useful for launching attacks and prevent unauthorized access to data/services by the future Internet architecture without incorporating additional patches. We aim to maintain support for large-scale mobility, and thus we have chosen the MobilityFirst architecture [7] as the basis for our access control design. The MobilityFirst architecture adopts identifier and locator separation [131] to support mobility and has already integrated several security features in its core design as discussed in the previous chapter. As a complement to the current security enforcement mechanisms [94] in MobilityFirst, we have developed a suite of access regulations that operate at the GNRS. Our regulation mechanisms support spatio-temporal access control (STAC), in which the accessibility to the protected resource/object depends on the location and time characteristics of the subject. Though our access control (AC) scheme is designed in the context of the GNRS service of the MobilityFirst architecture, it is general and can be applied to other similar network services, such as the name resolution of TorIP in Nebula [9,92] and in XIA [1,63].

This chapter is organized as follows: the complete access control scheme is presented in Section 3.3. I focus on spatio-temporal access control in Section 3.4, and define a token that supports implementation of flexible regulation policies. Finally, the chapter is concluded in Section 3.5.

figure/basic-eps-converted-to.pdf

Figure 3.1: Basic GNRS Access Control Scheme.

3.3 Access Control at the Name Resolution Service

The GNRS supports mobility by enabling a host to inform others the change of its location as well as to query others' locators. However, this service currently does not protect the mapping from being queried by illegitimate users. The fact that neither the GNRS nor the mapping owner have control over the querier may result in information/privacy leakage as well as many severe consequences. For example, malicious users can obtain the mapping and use it to launch DoS attacks, or may frequently query a specific user to track his/her behavior. Therefore, enforcing access control *at the query* would be a powerful tool as it can provide the mapping owner the ability of choosing who it is willing to communicate with. With AC incorporated into the GNRS, users can protect their location information contained in the GNRS mappings against unauthorized disclosure, while at the same time ensure the mapping's accessibility to legitimate users. In addition, integration of access control can support advanced services and fine-grained functionalities, such as allowing the mapping owner to decide when and where it is reachable.

As a starting point for AC, the GNRS mapping owner first defines a policy of who will be allowed to obtain its mapping and then submits the policy to the GNRS. In our AC scheme, authorized users can be identified by their identities directly, or the attributes obtained from Name Certificate & Resolution Service (NCRS), or via spatio-temporal attributes in special application scenarios. The GNRS server controls the distribution of the GNRS mapping according to the mapping policy's restrictions. The scheme is shown in Figure 3.1.

By allowing the policy to be flexibly specified, the GNRS AC design can support a variety of different access control schemes so as to cover a variety of applications. In a basic case, where we want only a single host to access the mapping, the policy has the simple format of $(GUID_B, T_B, E_B)_{A^{-1}}$. $GUID_B$ is the GUID of user B, to whom the mapping owner A is only willing to grant its mapping. Two timestamps, policy start time T_B (the time when the policy begins) and policy expiration time E_B (the time when the policy will expire), define the policy validity period. Lastly, the policy is signed by A's private key A^{-1}. In the identity-based case, the policy can be a *whitelist* or a *blacklist*, or even a combination. A defines its whitelist/blacklist following the format

$$\{(W, GUID_1, T_1, E_1), (B, GUID_2, T_2, E_2), \cdots\}$$

and submits it to the GNRS server. In each cell, **W** stands for the whitelist component and **B** for blacklist. The GUID is the identifier of the user to whom the policy applies, and policy start time T and policy expiration time E are also included. For a more complicated situation, where the subjects of the regulation are not defined by identifiers but by attributes, an attribute-based AC scheme would utilize the NCRS to map between attributes and GUIDs [94].

In order to incorporate the small pieces of regulation designs stated above into a single well-organized AC scheme, we propose the use of eXtensible Access Control Markup Language (XACML) as the policy language and processing model. XACML is flexible and suitable for a variety of application environments. The XACML context handler insulates the core language from the application environment [17] and provides conversion for the access requests and authorization decisions between the native format in the application environment (e.g. the GNRS API in our scheme) and the XACML canonical form. Although XACML is primarily an Attribute Based Access Control system (ABAC), both the basic GNRS AC and the identity-based GNRS AC can be considered as a specialization of the attribute-based GNRS AC and therefore able to be implemented by XACML. Here we provide a simple example of a *policy rule* where only users belonging to the Rutgers University domain can read the GNRS mapping. The expression of the *Target AttributeValue* is

```
<AttributeValue DataType="urn:oasis:names:tc:
xacml:1.0:data-type:x500Name">OU=Rutgers
University,OU=ECE,O=RU,C=US</AttributeValue>
```

The *Resource* is defined as

```
"http://www.w3.org/2001/XMLSchema#anyURI">
urn:gnrs:schemas:mapping
```

while the *"Read" action* is expressed as

`"http://www.w3.org/2001/XMLSchema#string">Read`

3.4 Spatio-temporal Access Control via GNRS Regulation

Mobility and location information is supporting many new types of applications, and it is possible to use location as a means to define new security services. Following this trend, we have outline two spatio-temporal access control (STAC) schemes in the GNRS that support security by regulating a mapping query based on the querier's location and time characteristics.

3.4.1 General STAC

The first scenario we discuss is where the GNRS mapping is only open to users in a designated location during a specific time period. As an example of this AC application, one could grant access permission to certain information/services only to users in an office building during normal office hours. Previous research work [36,38,109] has outlined solutions for this type of STAC, yet these works do not have any security mechanisms to ensure the correctness of spatio-temporal (ST) information in terms of ST verification, or to securely submit ST information to the appropriate entities. We have defined a secure STAC scheme that fills this gap and provides secure and efficient STAC for the GNRS service. Figure 3.2 shows a typical network setting for our scheme.

1. A→S: $\{(GUID_A, NA1, T_A, E_A)_{A^{-1}}, (L, T_P, E_P)_{A^{-1}}, P_{flag}\}$

   User A in network NA1 sends a GNRS update message to the GNRS server S following the standard GNRS protocols defined in [94]. The message carries the original GNRS update, $\{(GUID_A, NA1, T_A, E_A)_{A^{-1}}\}$, as well as a ST policy $(L, T_P, E_P)_{A^{-1}}$, which contains a location constraint L (such as GPS coordinates) and two time constraints (policy start time T_P and policy expiration time E_P). With this policy, only users at location L during time interval $< T_P, E_P >$ are eligible to obtain this GNRS mapping. Both the mapping and the policy are signed by A's private key to protect against a GUID spoofing attack [94]. The policy flag P_{flag} indicates the mapping status after the policy expires

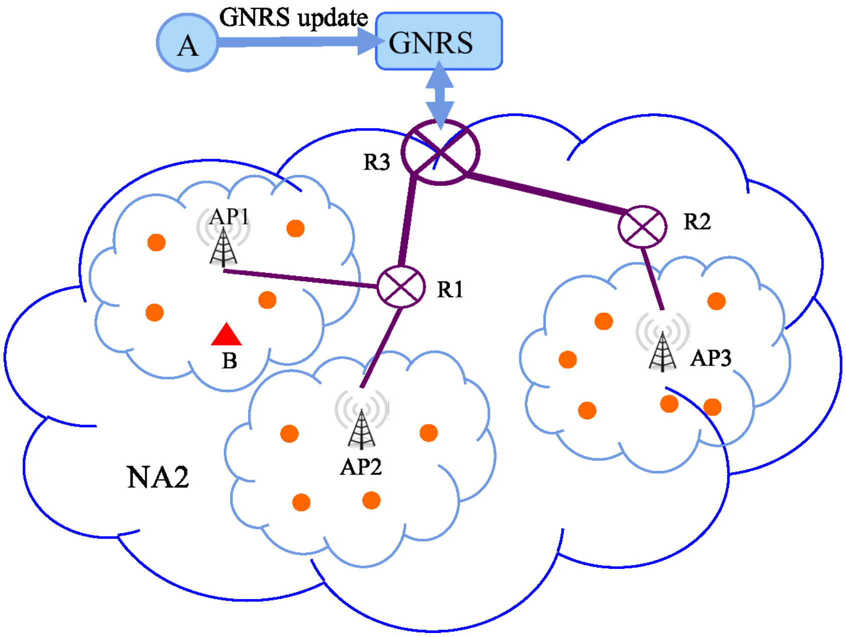


Figure 3.2: Spatio-temporal Access Control Topology: user A updates its GNRS mapping to the distributed GNRS system and then a mobile user B in a different network queries the GNRS for A's mapping.

and before the mapping expires. If P_{flag} is set to 1, then the mapping will become available to all other users after the policy expires; otherwise, no one can access the mapping after the policy expiration.

2. B→S: $\{GUID_B, [GUID_S, N_1, (GUID_A, GUID_B)_{B^{-1}}]_S\}$

   User B (represented by the triangle in Figure 3.2), who is currently in network NA2, wants to communicate with A so that it sends a query request to the GNRS server S following the standard GNRS query protocol.

3. S→B: $\{GUID_S, (GUID_B, N_2, L_Q, T_Q)_{S^{-1}}\}_B$

   Before evaluating the query request, S sends a ST information request to B asking for its current location. In the request message, $GUID_B$, flag L_Q and T_Q indicate that B's location and time information is being requested. N_2 is a nonce to prevent a replay attack. The message is first signed by S's private key to ensure information integrity and to express the sender's identity. Then $GUID_S$ is added for sender verification and finally the message is encrypted by B's public key to guarantee information confidentiality.

4. B→AP1:

   $\{GUID_B, [GUID_{AP1}, (GUID_B, N_2 + 1, L_B, T_B)_{B^{-1}}]_{AP1}\}$

   B can obtain its ST information from a proper localization algorithm/service, such as Global Positioning System (GPS), and generate a ST response for S in the format of $(GUID_B, N_2 + 1, L_B, T_B)_{B^{-1}}$, in which L_B is B's current location information, T_B is the current timestamp. In order to prevent B from lying about its ST information, we propose a ST verification mechanism where B's response is verified by its access point (AP). Assume B's AP is AP1 in figure 3.2, B sends its ST response to AP1 for verification.

5. AP1→S: $\{GUID_{AP1}, [GUID_S, GUID_B, (GUID_B, N_2 + 1, L_B, T_B)_{B^{-1}, AP1^{-1}}]_S\}$

   Here we assume there is a trustworthy localization service accepted by all network entities so that AP1 can verify B's ST information. If the information is correct, AP1 also signs the ST response and then forwards it to the GNRS server S.

6. S→B: $\{(GUID_A, NA1, T_A, E_A)_{A^{-1}, NA1^{-1}}, (N_1 + 1)\}_B$

   After receiving message 5 from AP1, S submits the ST information for evaluation. If the access decision is "permit", S sends A's mapping to B.

3.4.2 STAC with State Transitions

Another STAC scenario we are interested in is where accessibility to the mapping does not only depend on the current location and time, but also on the mobile user's previous spatio-temporal characteristics. In other words, we want a *stateful* form of access control, where *state* is defined as being at a specific location in a specific time interval. To the best of our knowledge, none of the current STAC schemes, e.g. [36, 38, 109], support AC based on state transitions in a large-scale mobile network very well because there is a large burden for distributed servers to maintain state records for a large number of users. In order to achieve scalability and efficiency, we examined the problem from a security point of view and have devised a token-based approach that is scalable and handles the state verification process:

1. A→S: $\{(GUID_A, NA1, T_A, E_A)_{A^{-1}}, [(L1, T_{P1}, E_{P1}), (L2, T_{P2}, E_{P2})]_{A^{-1}}, P_{flag}\}$

   The difference between this application scenario and the previous one in section 3.4.1 is that now the policy involves two states: the *current state* and the *previous state*. Each state is represented by a location L and a time interval (T, E) as $< L, T, E >$. Note that the timestamps for the two states should satisfy the relation of $T_{P1} < E_{P1} < T_{P2} < E_{P2}$. Both the current state and the previous state should be satisfied to gain the access to A's mapping. The policy flag P_{flag} is the same as in section 3.4.1.

2. B→S: $\{GUID_B, [GUID_S, N_1, (GUID_A, GUID_B)_{B^{-1}}]_S\}$

   A mobile user B in network NA2, who wants to communicate with A, sends a GNRS query request to the GNRS server S.

3. S→B: $\{GUID_S, (GUID_B, N_2, L_Q, T_Q, TK_Q)_S\}_{B^{-1}}$

   After receiving a query request, S sends a ST attribute query to B, in which L_Q and T_Q indicates that the current location and time information are being requested ,and TK_Q is a flag informing B that it needs to present a valid token that contains B's verified previous and current states. The *token* serves as a proof of the user's previous state. The details of the token definition and use rules are specified in Section 3.4.3.

4. B→AP1: $\{GUID_B, [GUID_{AP1}, (GUID_B, N_2 + 1, L_B, T_B, TK_B)_{B^{-1}}]_{AP1}\}$

   As B is a mobile user, it presents its current valid token to its access point AP1 for ST authentication. L_B and T_B are B's current ST attributes while TK_B is the token B current holds. The authentication procedures are similar to that in section 3.4.1.

5. AP1→S: $\{GUID_{AP1}, [GUID_S, GUID_B, (GUID_B, N_2{+}1, L_B, T_B, TK_B)_{B^{-1}, AP1^{-1}}]_S\}$

   AP1 verifies the token TK_B and the ST attributes submitted by B through the same localization algorithm that B used. If B's submission is correct, AP1 also signs the attribute query response and forwards it to S.

6. S→B: $\{(GUID_A, NA1, T_A, E_A)_{A^{-1}, NA1^{-1}}, (N_1 + 1), TK'_B\}_B$

   When S receives B's attribute response (forwarded by AP1), it verifies the signatures of B and AP1, and at the same time may also need to verify the token TK_B either through the

NCRS or a chain of keys depending on which method is used to generate the ST token. The token generation will be discussed in section 3.4.3. After verification, S submits B's ST attributes and the token for evaluation. If the access request is approved, S sends A's mapping to B together with a new token TK'_B for B's future use. B stores TK'_B as a proof of its moving path represented by the states in this token.

3.4.3 Token Format

The *token* is an object that carries security information and the privileges to perform certain operations. The ST token used in Section 3.4.2 contains the users' identifiers, states involved, token timestamps, all of which should be properly encrypted to allow relevant objects to be verified. In our GNRS service, as the user may move between the networks and therefore contact a different GNRS server (the one that is nearest to the user's current location), in order to query for a GNRS mapping the token issued by one GNRS server needs to be verifiable by all other GNRS servers.

Intuitively, following the methology used in [101, 133], we first considered a token format

$$\{GUID_S, (GUID_B, S_P, S_C, T_T, E_T)_{S^{-1}}\}.$$

$(GUID_B, S_P, S_C, T_T, E_T)$ carries the information about the ST states for user B whose identifier is $GUID_B$. S_P and S_C are B's previous state and current state respectively, both of which should contain ST information. T_T and E_T are two token timestamps: token generation time T_T and token expiration time E_T. The token is encrypted by the private key of the issuer, GNRS server S, whose identifier $GUID_S$ is also added for issuer verification. When B presents this token to other network entities, the verifier can check whether the token is issued by a legitimate GNRS server from the NCRS by query S's object attributes with $GUID_S$.

However, this token generation method places additional traffic burden onto both the NCRS and the GNRS because GNRS servers, who want to verify the token, must make NCRS queries to check the token issuer's identity. To reduce the overhead, we devised a different token format: $\{Seq, GUID_B, S_P, S_C, T_T, E_T\}_K$. In this format, Seq is a sequence number for the token, which is encrypted by a shared key K submitted by the GNRS mapping owner. However, this key shared by the mapping owner and all the relevant GNRS servers should be chosen carefully to avoid key manipulation attacks while keeping overhead minimal. First, the key is updated periodically to prevent staleness. Second, in order to reduce the overhead of updating the shared key, we use a hash chain to generate a sequence of keys from a seed for one-time use [86]. In addition to the mapping and the policy, user A's GNRS update also contains a hash function and a seed for the shared keys' generation: $\{(GUID_A, NA1, T_A, E_A)_{A^{-1}}, (L, T_P, E_P)_{A^{-1}}, (H, K_n, T_K, N)_{A^{-1}}\}$. Here H is a hash function, K_n is the seed, N is the number of shared keys that are to be generated and T_K is the period for updating a new key. The chain of one-time keys generated by the hash function is $K_n \rightarrow K_{n-1} \rightarrow \cdots \rightarrow K_1 \rightarrow K_0$, where $K_i = H(K_{i+1})$ for $0 \leq i \leq n-1$ and K_0 is the first key to use. Because of the one-way property of the hashing, it is not possible for an attacker to figure out the current key K_i, even if he is able to obtain the previous key K_{i-1}. With the hash function and the seed, all GNRS servers involved can calculate the subsequent keys themselves without synchronizing and communicating with each other. As a result, it is unnecessary to exchange and update the shared keys except when delivering the key initialization information at the beginning, and this fact reduces the potential for spoofing threats and the communication overhead.

Compared to the first token generation method, the second one does not need any extra communications to verify the shared keys. Once the GNRS mapping owner decides the hash function and the seed, the GNRS servers who receive the update can calculate all the following keys.

3.4.4 Token Usage Discussion

Containing both the previous state and the current state in the token can prevent malicious token manipulation, and further allows for inferring the complete state transition path. Thus, our ST token design is quite flexiable and can accommodate many special policies and provide fine-grained functionalities. The features of the ST token design includes the following:

- ▷ Satisfy the requirement for multiple previous states.
- ▷ Robust to potential attacks, e.g. token reuse attack.
- ▷ Be able to express some special policies.

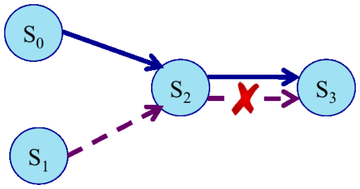


Figure 3.3: A policy that requires having been in previous state (S_0) in order to access to the next state (S_3).

We now present four policy examples in the following section to show the advantage of the token approach.

- ***Policy Example 1***

In the first example, there are two initial states S_0 and S_1, both of which can reach state S_2. The policy requires that state S_3 can only be reached from state S_0 through state S_2, as shown in Figure 3.3.

This policy is represented using the second token generation method. The involved tokens for $GUID_1$ and $GUID_2$ are listed below.

- Current state S_0: $TK_0 = (Seq_1, GUID_1, S_0, S_0, T_0, E_0)$
- Current state S_1: $TK_1 = (Seq_2, GUID_2, S_1, S_1, T_1, E_1)$
- Current state S_2, previous state S_0:

$$TK_2 = (Seq_3, GUID_1, S_0, S_2, T_2, E_2)$$

- Current state S_2, previous state S_1:

$$TK_2' = (Seq_4, GUID_2, S_1, S_2, T_2', E_2')$$

According to the policy, in order to transit to state S_3, the user must present token TK_2 at state S_2, which indicates its current state is S_2 and previous state is S_0. So only $GUID_1$ with TK_2 can enter S_3. This example shows how the token method accommodates a policy that requires both a previous state and the current state. Along with all the previous tokens a user has, since history information is carried in the token, the complete state transition path can be

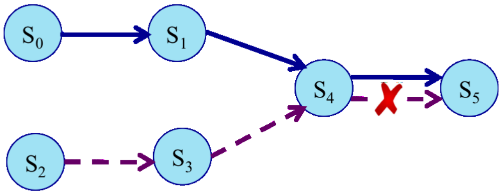


Figure 3.4: A policy with the requirement of being at S_0 and then S_1 before transiting from S_4 to S_5 is accepted.

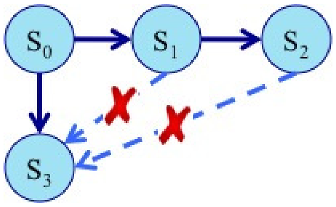


Figure 3.5: A policy that only allows one to enter S_3 from S_0 but not from the paths $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$ or $S_0 \rightarrow S_1 \rightarrow S_3$, which pose a weakness that could lead to a token reuse attack.

determined so that it is possible to express a policy involving two or more previous states as in the second example.

- ***Policy Example 2***

In Figure 3.4, this policy requires subjects to satisfy the requirements for the current state and *two* previous states.

There are two possible transition paths in Figure 3.4: $S_0 \rightarrow S_1 \rightarrow S_4$ and $S_2 \rightarrow S_3 \rightarrow S_4$. The policy only allows the first path to enter state S_5. Users who are at state S_4 and transited from S_2 and S_3 cannot access S_5. Under this policy, a user needs to present two tokens to enter S_5: $(Seq_1, GUID, S_0, S_1, T_1, E_1)$ and $(Seq_2, GUID, S_1, S_4, T_4, E_4)$. A problem may arise if we allow a single user to have multiple valid tokens simultaneously. The system may be open to a *token reuse attack* in which an attacker uses a previous token, which has not expired yet, to enter a state which he is not allowed to enter. The token reuse attack against the second policy example is shown in the next example in Figure 3.5.

- ***Policy Example 3***

Assume token TK_0, TK_1 and TK_2 correspond to current states S_0, S_1 and S_2 respectively. The third policy example states that S_3 can only be entered from S_0 but not from S_1 or S_2.

In other words, this policy only allows transition from state S_0 to S_3, but neither the path $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$ nor $S_0 \rightarrow S_1 \rightarrow S_3$ are allowed. However, if the user is allowed to have multiple valid tokens at the same time, an evil attacker may obtain TK_0, TK_1 and TK_2, all of which are valid tokens, if he starts from S_0, goes through S_1 and reaches S_2. Then the attacker requests to enter S_3 at state S_2 by only presenting TK_0 and hiding TK_1 and TK_2 from the system. As the token TK_0 has not expired yet, the system may fail because it is unable to differentiate the real current state of the attacker.

In order to protect against the token reuse attack mentioned above, we propose and compare two solutions. One easy solution uses a key associated with each state and updates the key frequently. More specifically, the way of obtaining the key is restricted to the designated location and time interval and the valid duration of the key is very small. For example, in Figure 3.5 the user needs to be in a specific ST state S_0 to obtain the key promptly before entering S_3. However, this method will fail if the malicious user has a colluding partner that obtains the key at the specific location during the specific time and then immediately sends the key to the attacker, who then uses the key together with the token TK_0 to enter S_3.

The failure of the previous method leads us to a solution where token records are kept at GNRS servers as shown in Figure 3.6. The basic idea is to keep track of the querier's states by maintaining a log so that once the user leaves a state and enters a new state, it cannot lie about its current state by presenting the token of the old state and hiding the token of the new state. In this scheme, for each GNRS querier, if the GNRS revokes a previous old token or issues a new token, it needs to update the querier's log to keep track of its state transition path. As the token is a per-mapping use, the GNRS server who initiates the token change should inform other GNRS servers, who also hold the GNRS mapping that the token is meant for, about the log update. For efficiency, the log only needs to contain the token's sequence number.

Comparing the two solutions against the token reuse attack, the first solution is open to a collusion attack and is therefore unreliable. The second solution is more robust, though this is at the cost of having an increased overhead of maintaining and updating the token log. Since the number of mapping storage locations is usually not large (e.g. roughly 3 to 5 copies per mapping) and the log only keeps the token's sequence number, the cost is acceptable.

- ***Policy Example 4***

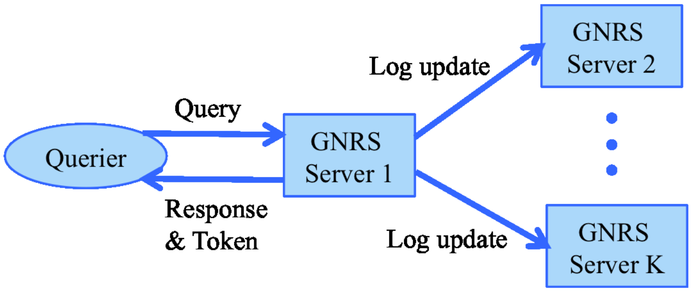


Figure 3.6: A solution to the token reuse attack uses a log to keep track of the issued tokens.

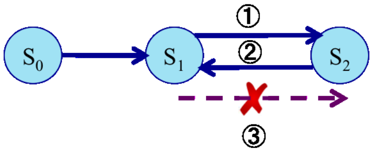


Figure 3.7: A policy that restricts the number of times one may enter S_2 from S_1 to one, thus avoiding reusing a one-time ticket.

The last policy example is shown in Figure 3.7. Here, our token scheme can prevent falling into a two-state loop. In practice, this policy can describe a one-time use ticket efficiently as re-entry into a specific state is restricted. Assume the policy allows the state transition in step 1 from state S_1 to S_2 and then in step 2 from S_2 back to S_1, but does not allow entering S_2 from S_1 again. When the user first enters S_1 from S_0, it holds token $TK_1 = (Seq, GUID, S_0, S_1, T_1, E_1)$ and presents TK_1 when it wants to transit to S_2. Then in state S_2, the current token this user holds becomes $TK_2 = (Seq, GUID, S_1, S_2, T_2, E_2)$, which can be used to return to S_1. However, once the user returns back to S_1, it can no longer enter S_2 because its current token becomes $(Seq, GUID, S_2, S_1, T_3, E_3)$, which indicates the previous state before S_1 is S_2. Therefore, our scheme is able to express the policy of avoiding a two-state loop.

3.5 Conclusion

Many future Internet architectures are being designed to include new forms of name resolution, and the associated name resolution servers represent a powerful architectural component that can support security. In particular, it is possible to deploy access control mechanisms at a name resolution service so as to regulate access to network address information, which could otherwise be used by an adversary to launch network attacks or could be queried to glean information useful for subverting privacy. As the best of our knowledge, the access control scheme suite discussed in this chapter is the first one to explore the deployment of access control within a name resolution service for large-scale mobile networks, and especially explore two example STAC schemes that ensure the correctness and freshness of the ST information through verification, authentication and the use of tokens.

Chapter 4

Overlay Tunneling as a Policy Tool

4.1 Introduction

Ad hoc networks and mobile ad hoc networks (MANET) have found applications in many areas, ranging from environmental monitoring, personal area and home networking, animal tracking, disaster recovery, and tactical operations. As an example, ad hoc networks built from small sensors deployed in a target area have been successfully used for remote sensing and monitoring environmental parameters, such as temperature, humidity and pollution indicators. Mobile ad hoc networks have also found application in tactical scenarios, and more recently in the form of vehicular networks in which devices installed on vehicles form ad hoc networks to deliver important driving and safety information. Therefore, mobile ad hoc networks are an important component of the global Internet and the MobilityFirst architecture should provide sufficient support for the MANET.

Unfortunately, the requirements that the ad hoc network be highly adaptive, as well as security vulnerabilities brought on by the intrinsic characteristics of the MANET, are still among the biggest concerns regarding ad hoc networks. As ad hoc networks involve a collection of wireless nodes without any support from a fixed infrastructure or centralized management, it is usually hard and expensive to maintain the MANET, causing it to be less reliable than fixed networks. Generally, the MANET networks work in an unattended fashion with very little or even no maintenance, and thus if there is a failure or a network topology change, it is often infeasible to repair failed devices and/or reconfigure the network– in spite of this being a primary objective behind ad hoc protocol design! Also, for large-scale systems, maintenance becomes too expensive to perform. Even worse, the high rate of topological changes due to mobility and device failures in a MANET makes these systems inherently less stable. The underlying network-layer protocols may be unable to accommodate the dynamic changes associated with mobility, and thus might not be able to adequately adjust the network configuration at the time of deployment to a suitable new network topology that can sustain user demands.

Further, in most of the MANET use cases, the devices are resource constrained in terms of bandwidth, computation capability, storage, and power. This characteristic not only contributes to the high rate of network topology changes (as devices stop working due to power exhaustion) but also opens doors for exploits and attacks. Those devices with limited resources are vulnerable to attacks because it is often impossible to equip them with strong security measures, such as access to certificate authorities and the associated asymmetric cryptographic algorithms and sophisticated security protocols. Further, the unattended nature of the network's management makes protecting the network more challenging because devices can be easily captured or compromised. With the increasingly broad usages of MANET, weaknesses associated with the network's stability and security aspects may lead to severe consequences.

To improve adaptability and security in a MANET, there has been an extensive effort by the research community, ranging from the physical layer all the way up to the application layer. For example, many efforts have been focused on the network layer as most of the popular ad hoc routing protocols, such as OLSR [41], AODV [108] and DSR [76], lack security measures. Attacks targeted at routing protocols have been identified, such as routing table overflow attack, routing cache poisoning attacks, wormhole attack, blackhole attack and Byzantine attacks [45, 95, 134]. To defend the MANET against these attacks, a variety of protocols with security measured integrated into the protocol have been proposed. Security-aware ad hoc routing protocol (SAR) [136] protects against blackhole attacks. Packet leash protocol [70] is designed to prevent wormhole attack. ARAN [114] defends against impersonation and repudiation attacks. SEAD [69] prevents modification attacks. SEAR [88] provides routing message authenticity to protect against attacks targeting at the AODV protocol.

Unfortunately, there is no panacea for MANETs that protects them against all of the recognized threats without introducing significant overhead or sacrificing performance. Those security-enhanced protocols integrate security measures to basic ad hoc routing schemes to prevent varying attacks at the cost of efficiency and/or complexity. As for improving the adaptability to highly dynamic topology changes, a few previous research works examine the problem at different levels of the network protocol stack, mainly in the MAC layer and network layer. As an example, [75] improves the adaptability of DSR routing protocol by predicting link status used by the link caching scheme. Unfortunately, because of the dynamism and heterogeneity of MANET networks, there is no optimum network protocol stack solution for all. A good choice of the protocol stack and well-tuned network configurations for one particular MANET scenario may not work in other scenarios, or even as the network evolves over time or experiences attacks against its topological configuration.

In this chapter, we propose an alternate view for protecting ad hoc networks. Specifically, we propose a communication planning approach, involving the use of overlay tunneling in this chapter, as a backup/complementary tool for rescuing a network as it experiences network topology changes and attacks. Our approach provides a policy tool that has situational awareness and takes actions correspondingly to mitigate severe consequences of network failures and attacks. Through the use of monitoring, our overlay approach continually monitors the network status and if it detects a change (such as an attack, a failure or a topology change) that causes performance degradation, our tool tunnels the traffic following a predefined (policy) strategy to mitigate the situation. Our mechanism has the following features:

- **Lightweight**: it applies a lightweight touch that complements the existing MANET network protocol stack without altering or redesigning the underlying protocols. As it does not add any additional security measures to the protocol stack and is only adopted when necessary, protocol overhead can be reduced significantly.
- **Flexible**: the overlay approach sits on top of the network layer and does not place any specific requirements on the underlying network architecture or the routing protocol. Hence, it can work with arbitrary network layer protocols, and its utility is not restricted to any particular type of attack.
- **Fast Response**: when a part of the network fails, the overlay tunneling can react faster than the underlying routing protocol, which needs time to converge.
- **Extensible**: this approach can serve as a policy tool, which has a basic version and two

optional optimizations if additional information is available. Here we define the *policy tool* as that all nodes in a network will follow a security policy that dictates how they respond to different network conditions/threats.

- **Tunable**: the parameters of the tunneling tool are tunable so that the tool is able to adapt to the dynamics of the network.

We note that our use of overlay tunneling has promising applications. For example, in a tactical MANET, once the devices are deployed in the battlefield, it is impossible to repair the network when there is a failure or a device is compromised. Normally, the performance of the network at this time will degrade and may even cause the network to cease to meet minimal working conditions. With our tunneling method, the traffic can be tunneled to other parts of the network to avoid the source of the performance degradation. As a result, with only a minimal overhead, performance can be improved. Another use case for our tunneling tool is mesh networks that provide private peer-to-peer communications without support from an Internet connection or cellular coverage, such as Firechat [2]. Users may take advantage of our tunneling technique to avoid the government supervision and achieve private communications. For example, if users have certain knowledge about the supervision checkpoints, which can be considered as attack zones in the context of this chapter, they may use our tunneling tool to avoid the undesired traffic checking.

4.2 Related Work

Overlays and tunnels are widely used in both the Internet and in MANETs. Overlays are useful for deploying virtual networks without altering the underlying network infrastructure, and tunneling is a powerful tool to build overlay networks or set up customized virtual paths over existing networks. A wide variety of tunneling protocols and overlay networks have been developed to provide advanced features and services on top of the traditional Internet. For example, the Generic Routing Encapsulation (GRE) protocol [53] runs on top of IP and encapsulates several network layer protocols within virtual point-to-point links. Secure Shell (SSH) [137] is another widely used protocol which initiates and maintains shell sessions on remote machines with encrypted tunnels. The two-layer IP-in-IP overlay network X-Bone [126] provides network management, including dynamic resource discovery, automated deployment as well as network monitoring. Other well-known overlay systems include commercial virtual private networks (VPN) [115], Mbone [52] and 6bone [54].

Beyond providing advanced functionalities to the Internet and applications, the overlay philosophy has also been employed to handle network failures and defend against security threats. For example, Resilient Overlay Networks (RON) [25] focuses on network robustness and provides an overlay architecture for distributed Internet applications that supports rapid detection and recovery from path outages and performance degradation. CenterTrack [122] is an overlay scheme to identify and track flood-type denial-of-service (DoS) attacks in high-speed IP-based networks. SOS [79] is another overlay tunneling architecture, which aims at preventing DoS attacks proactively by intensive filtering, randomness and anonymization.

In MANETs, overlay tunneling is also frequently employed both on the attack and defense side. For example, two colluding attackers can launch wormholes attack [71, 80], one of the most severe threats to routing protocols in MANETs, by employing tunnels. On the other hand, overlay tunneling also enables performance improvement and advanced functionalities in MANET. Many efficient multicast schemes in MANET are proposed using overlays, such as [61, 62, 81]. As another example, in [132], bi-directional tunneling allows a vehicle network design involving network mobility management.

Our work introduces a tunneling technique over IP as a light-weight touch for benevolent nodes in a MANET to address network threats. The basic idea is to create a new *"phantom source"* to replace the real source, and therefore make it difficult for an adversary to target a specific source-destination path. With tunneling, it is possible for messages from a source to be sent out on a directed route for a certain amount of hops before being multicast flooded or sent via a unicast shortest path to the destination.

4.3 Overview of Our Tunneling Scheme

4.3.1 Tunneling Methology

Our tunneling scheme is designed as a policy tool for MANETs that serves as a map between situational awareness and deciding which security tools to apply. That being said, all nodes in a network will follow a security policy that dictates how they respond to different network conditions/threats. Our policy tool enforcement operates as an overlay. In that context, overlay tunneling sits on top of the IP layer and can provide an alternative path to avoid potential threats. With dynamic information being obtained from real-time traffic monitoring or other resources (e.g., trust assessment, as discussed in Section 4.3.4), communication sources may be able to detect an on-going attack and then engage in overlay tunneling to protect against the attack. Within a policy framework for protecting a network, nodes may use overlay tunneling reactively or proactively based on the needs and their network situational awarenesses.

In our tunneling approach, the source of a communication chooses a node, known as the tunnel agent, to detour the traffic flow when it believes there might be an attack or network abnormality. Assuming there is an attacker who has some knowledge of the flow path between the source and destination or even just happens to be on the shortest path from the source to the destination, tunneling the traffic to a tunnel agent which is unknown to the attacker may allow the traffic to avoid the attack and therefore benefit the performance. Or, if there is a failure in a part of the network, tunneling the traffic appropriately can send the data flow through unaffected areas and hence avoid the consequences of a failure.

In the following discussion, we will focus on the situation where there is an attack in the network. The scheme is illustrated in Figure 4.1. Suppose there is traffic between the source node src and the destination node $dest$ along the shortest path $src \rightarrow 1 \rightarrow 2 \rightarrow dest$ highlighted in green. After the network has been in operation for a while, a compromised node 2 engages in a blackhole attack that involves dropping data packets that pass through it. When this attack is detected (discussed in Section 4.3.2), the source node src starts tunneling the flow to a selected tunnel agent (for example, node 5 depicted by an orange circle) to avoid the attacker. Though there are certain costs with tunneling, such as increased delay, or increased hops for the routing path, the benefits of avoiding attacks outweigh the risks of running near an adversary. If the tunnel agent is properly chosen, the performance, such as packet delivery rate, will be much better compared to hitting an attacker. However, if the chosen tunnel agent still directs the traffic toward an attacker, then the tunneling fails and yields essentially no benefit. The tunneling is only turned on at the source node when necessary, such as when sufficient performance degradation is observed, and hence the costs of using the tunnel is mitigated.

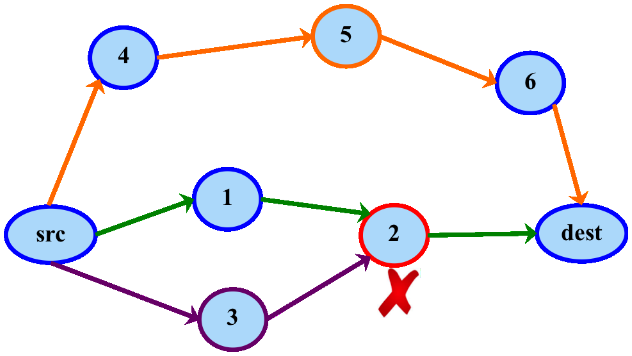


Figure 4.1: Example of a single tunnel scheme in a wireless ad hoc network in the presence of one attacker (node 2).

In order to choose a good tunnel agent, certain knowledge is needed to evaluate the tunnel agent candidates. For example, knowing the behavior of other nodes is very useful for differentiating between malicious nodes and benevolent nodes. The knowledge of the topology and nodes' mobility pattern helps the source in choosing a shorter tunnel path. If we have all the information aforementioned, it is trivial to determine the tunnel agent. Unfortunately, more often than not, not all the information is available and hence it is difficult for the source to clearly identify attackers and their locations. As a result, we adopt a *best effort* approach: if there is any side information differentiating between the benevolence of nodes in the network, we can use it as a reference to assist in choosing the tunnel agent; otherwise, we choose the tunnel agent randomly from the candidate set of tunnel agent (excluding the source and the destination) for simplicity. How to evaluate the trustworthiness of the nodes and provide additional information for determining the tunnel agent will be discussed in detail in section 4.3.4.

4.3.2 Network Anomaly Detection

As a network anomaly event initiates the starting of the tunneling procedure at the source, it is necessary to define what constitutes a network anomaly and then monitor network operations so that any relevant network anomaly event can be detected and fed back to the source in a timely fashion to launch tunneling. Generally, a network anomaly [73, 125] refers to a dramatic change of network performance (usually means a performance degradation rather than an improvement) or disruption of normal network operations.

Detecting the interruption of network operations is specified and handled by the corresponding network protocols, and therefore is out of the scope of this work. With respect to the network performance, it is tied closely to the type of the network and specific applications. The network performance analysis and evaluation are usually complicated as the network is a dynamic system influenced by a large number of aspects, such as traffic pattern, mobility pattern, network size, network connectivity, topological rate of change, link capacity, fraction of unidirectional links, fraction and frequency of sleeping nodes [27, 43, 66]. End-to-end delay, throughput, packet delivery rate, overhead, energy consumption, error rate, jitter, out of order delivery as well as security attributes, such as confidentiality, authenticity, integrity, availability and robustness, are the typical factors to consider when evaluating network performance [32, 44]. Different applications may weight these factors in different ways according to their own interests. For example, delay sensitive applications, e.g. video conferencing and streaming, may focus on delay and jitter more than energy consumption and overhead. Tactical networks may care more about error and security attributes. As a result, the definition of network performance is application-oriented and specific for the network type and functionalities.

With a specification for network performance in place, the corresponding interested network statistics are monitored and the source of the data flow is informed once an abnormal event is observed. Typically, the network anomaly might be either due to network failures or security-related issues. Network failure detection and analysis have been widely studied and many approaches, such as artificial intelligence, state machine modeling and machine learning, have been proposed [125]. As to security threats in MANETs, many efforts have analyzed the attacks [45, 95, 134] and a large variety of methods to detect various attacks are now available [72, 84, 103]. There are several different ways to classify the attacks. Based on attack means, attacks can be classified as passive attacks and active attacks. Passive attacks, such as eavesdropping, traffic monitoring and analysis, generally do not disrupt the normal operations of the communication so that they are hard to notice. On the other hand, active attacks, such as blackhole attack, routing cache poisoning attack and Byzantine attack, actually destroy either data packets delivery or control message transmission and hence are more likely to be identified. Another threat classification can be based on attack domain and categorizes attacks as external attacks (launched by nodes not belonging to the network) and internal attacks (launched by compromised nodes) [134].

For our purpose, any appropriate approach can be adopted individually or collectively to detect the anomalous events in the MANET that affect the end-to-end traffic performance and thereby emit a signaling for the tunneling tool. No matter whether the network anomaly is due to a failure or an attack, as long as the occurrence causing performance degradation can be detected and the source node is informed correspondingly, the tunneling mode can be turned on at the source to improve the performance.

4.3.3 Re-evaluation Technique

A ***re**-evaluation* technique is integrated into the tunneling tool for performance optimization. In the case where the real-time traffic information is available, the single tunnel with re-evaluation can benefit from the dynamic monitoring and tune the tunnel to adapt to the environment. The goal behind this is to tunnel the traffic to *another* tunnel agent if the performance degradation does not exhibit sufficient improvement, or if it experiences subsequent degradation. For example, the delivery performance of network traffic is evaluated periodically to provide an end-to-end feedback to the source, who will then make a tunnel decision if the performance does not meet its expectation (e.g. as specified by a policy at the source node). Similar approaches have also been employed in other works, such as [84], where the training data is updated at regular time intervals.

One challenge for choosing the tunnel agent is the lack of sufficient information, which may suggest that choosing a random choice is the best solution. As a result of the randomness, though, the tunnel agent chosen might be unable to direct the traffic away from attackers. Therefore, we must be able to adapt and choose another tunnel agent. In our work, we adopt a *try-and-see* fashion to choose the tunnel agent at the source side. The evaluation and tunneling process will be performed periodically to prevent attacks promptly and improve the traffic delivery. More specifically, the network performance evaluation result, specifically the end-to-end feedback of the on-going traffic, is sent to the source periodically so that the source node may evaluate the choice of the tunnel agent. If the performance meets the expectation, the source continues to use the same tunnel agent; otherwise, the source node repeats the random process to choose another node as the new tunnel agent. As in figure 4.1, the first random choice of a tunnel agent might be node 3 (marked with a purple circle), which fails as it still leads the traffic toward the compromised node 2. Then after a short period of time, the re-evaluation process repeats and informs the source node the failure in the first stage so that the source node src chooses another node 5 highlighted by orange color as the new tunnel agent in the second stage, detouring the traffic successfully away from the attacker.

This re-evaluation can help to detect new anomalous events in the network promptly, as well as remedy poor tunnel agent choices. The frequency of re-evaluation also influences the effectiveness of tunneling scheme. The more frequent this re-evaluation process is performed, the faster the source reacts to attacks and network fluctuation.

4.3.4 Trust Assessment

As another optimization besides re-evaluation, trust assessment provides an optional extension to further improve the tunneling performance. It collects information about the trustworthiness of the nodes in the network and provides side information to the source for refining the candidate set of tunnel agent. Trust plays an essential role in the networking and is becoming increasingly important as it is relevant to security and privacy issues due to the proliferation of ubiquitous networks (include Internet, wireless sensor networks, Internet of Things, etc.), which bridges virtual world and real physical world as well as connect people and various devices. For example, to enjoy a video streaming service, the user needs to trust the content provider to provide legitimate content as well as the ISP to deliver the video. In peer-to-peer (P2P) systems, the crucial premise of a successful operation is the trust between two peers.

Recent research explores applications of trust assessment in many different fields. [21] investigates trust management in P2P networks and presents a scalable decentralized trust model. [30] provides an honesty checking algorithm to build the reputation systems for filtering out the inconsistent recommender's behaviors in P2P systems. SourceRank [31] explores searching challenges in web databases and proposes a trust-aware approach to facilitate the search. In MANETs, security-aware adaptive DSR (SADSR) [56] protects against source route modification and route cache poisoning by introducing trust comparison into route selection. However, how to evaluate trust and assign trust score to each node, which is then used to calculate the route trust value, is not discussed in SADSR. As a summary, though the importance of trust has been recognized and trust assessment has a promising future of improving networks and applications performance, how to evaluate trust and building proper trust model are challenging because of the subjectivity and reciprocity associated with trust quantification.

Here we propose a trust assessment scheme to evaluate the trustworthiness of nodes in the MANET. The scheme calculates trust score for each node in the network and distributes the trust scores across the network to build a trust graph, which may be used by many network entities, policies and protocols. The definition of a node's trust score contains four basic trust aspects as follows:

- **Integrity**: consists of integrity in both the control plane and data plane. In the control plane, integrity refers to securing routing protocols to avoid misrouting, replay, black hole, etc; in data plane, integrity means preventing message manipulation and tampering.
- **Capability**: refers to the ability to handle packets and indicators may include router process speed and available storage.
- **Link quality**: usually refers to bandwidth fluctuations and packet delivery ratios.
- **Confidentiality**: indicates whether a proper cryptographic scheme is used to protect against spoofing.

The aspects of the trust definition outlined above may be adjusted according to specific network topology and application requirements. Every node in the network observes behaviours of every direct neighbours and gives a score for each trust aspect, known as the *aspect score* $s_{i,j,k}(t)$, (i.e. the score of kth aspect for node i's neighbour j at time t). The *transient score* $s_{i,j}(t)$ for node i's neighbour node j at time t is the weighted average of all the four aspect score as shown in Equation 4.1. Depending on the application requirements, different trust aspects may have varying significance, and therefore are assigned different weights α_k. The transient score will be updated and distributed throughout the network periodically. Then each node in the network will receive trust reports about other nodes in terms of the transient score. With respect to all the received transient trust scores associated with a target node, the highest M scores and lowest M scores are excluded in order to prevent malicious score manipulation. M should be set appropriately according to the network size and density. Then the *transient* *overall score* $s_j(t)$ of a target node j at time t is the average of all the received transient score after removing M highest and M lowest scores. A node may simply use the transient overall score of the current time period as a reference while communicating with other nodes or performing network operations. Further, as building trust is an accumulative process over time, exponential moving average (EMA) [74] would be a more accurate method to calculate the trust score for a particular node over time. Hence, we define *trust score* $S_j(t)$ as the EMA of current and previous transient overall score for node j as in Equation 4.2. β is the coefficient representing the degree of weighting decrease and is a constant between 0 and 1.

$$s_{i,j}(t) = \sum_{k=0}^{k=3} \alpha_k s_{i,j,k}(t) \tag{4.1}$$

$$S_j(t) = \beta s_j(t) + (1-\beta) s_j(t-1) \tag{4.2}$$

Basically, each node monitors and rates its neighbors' behaviors and then distributes the ratings throughout the network periodically. The ratings may be carried by the routing control messages (e.g., in a reserved field) or flooded to the whole network independently. After receiving the rating reports from other nodes, each node can compute the trust score and thus build a trust graph of the whole network. The trust graph may be provided to other applications as reference or assist network protocols to obtain better performance. For example, a node can adopt a policy of *"drop all the packets from nodes with trust score lower than a threshold"*. Access control may be specified in terms of a trust score: *"do not allow nodes with trust score lower than a threshold to contact me"*. Ad hoc routing protocols may be improved by considering the node's trustworthiness when calculating the routing table. The simplest application is to exclude nodes with trust score lower than a threshold from the routing graph so that the node can avoid routing through the suspicious nodes. Another more sophisticated method is to use a weighted summation of the link weight and the trust score instead of only link weight while computing the routing table. In this way, those paths that are more trustworthy will be preferred when choosing the *"shortest"* routing path.

In the context of our tunneling scheme, the trust assessment results can be used to refine the candidate set of tunnel agents. The source node may simply exclude all suspicious nodes with trust score lower than a threshold from the candidate set and therefore do not choose them as the tunnel agent, though the actual routes might still go through suspicious nodes. Furthermore, those suspicious nodes can be completely avoided by removing them from the routing graph with the cost of modifying underlying routing algorithm as aforementioned.

4.4 Tunnel algorithm

As a starting point, we assume an ideal network anomaly detection algorithm exists, whereby packet delivery rates are monitored for all network flows and trust scores (obtained from trust assessment in the improved case) for all nodes in the network. An end-to-end traffic feedback information is assumed to be available so that we can make tunneling decisions by comparing real-time packet delivery rates with a threshold. Every flow transmission is divided into several sessions, each of which sends M packets. The source evaluates the performance in terms of end-to-end packet delivery rate provided at the beginning of a new session. If the performance in the previous transmission session meets the expectation (a pre-defined threshold), the source continues to send the next M packets without any adjustment. However, if the performance falls below the threshold, it indicates that the transmission path is being attacked in the previous session, and therefore the source chooses another tunnel agent to send the next M packets in the new session.

Features: The distinguishing features of this algorithm are *single* tunneling and ***re**-evaluation*. Only one tunnel agent is chosen to build a tunneling path each time the source decides to tunnel traffic. The comparison between the packet delivery rate and the pre-defined threshold repeats at the beginning of each session that involves transmitting M packets. This periodic re-evaluation allows the source to detect abnormal situations and restore the traffic in time. The smaller the parameter M is, the faster the source reacts to attacks, though at the expense of having information that is potentially less stable.

Packet Manipulation: The tunneling algorithm operates at the IP layer. When the source decides to tunnel the flow, it encapsulates the original IP packet as the payload of an IP-within-IP packet and adds a new IP header with the IP protocol number set to 4 and the destination address set to the tunnel agent. When the wrapped IP-within-IP packet arrives at the tunnel agent, the agent checks the IP protocol number. If the IP protocol number is 4, meaning the packet is a tunnel packet, the tunnel agent removes the IP-within-IP header and checks the original IP header wrapped inside the IP-within-IP packet. Then the agent forwards the unencapsulated packet to the designated destination.

Choice of Tunnel Agent: Choosing a good tunnel agent is important as it affects the performance in the next session of M packets. We employ a try-and-see fashion that the re-evaluation technique is used to tune the traffic. The source randomly chooses a node from the candidate set as the tunnel agent. Introducing *randomness* into choosing the tunnel agent makes it more difficult for an attacker to track a specific traffic flow and launch an attack against the flow. However, on the other hand, the randomly chosen tunnel agent may still lead to traffic going through the attacker or its vicinity. So at the beginning of the next session, when the comparison repeats, if it turns out that the choice of the agent is not good (i.e. the packet delivery rate still does not meet an expected target level), the source repeats the random selection process again to choose the tunnel agent.

Algorithm 1 and algorithm 2 describe how to set up a single tunnel with re-evaluation at the source side and how to handle the tunneling packet at the tunnel agent. Algorithm 1 shows the pseudo code at the source side.

The pseudo code for how the tunnel agent handles the encapsulated packets is shown in algorithm 2.

Employing Trust Assessment: Since the trust scores of all nodes are available from trust assessment, the source first sets a trust threshold and filters out itself and the destination as well as suspicious nodes, whose trust score is below the threshold, from the set containing all the nodes in the network. Here we assume that the lower the trust score is, the worse the node is in terms of trust attributes defined in section 4.3.4. This filtering results in a refined set of tunnel agent candidates. Furthermore, we may even expose the routing table for the refining process of the candidate set so that only nodes on the path which does not go through suspicious nodes will be included in the candidate set.

By adopting this algorithm, once an attacker comes in, the source will be able to detect the attack at the beginning of the next session of M packets by observing a drop of the packet

Algorithm 1 Algorithm of single tunnel with re-evaluation at the source node

```
                                ▷ %comment: initialize packet delivery rate to 1%
$deliveryrate \leftarrow 1$
while ( there is an IP packet to send ) do
  ▷ %comment: evaluate the performance of previous session ($M$ packets) at the beginning
  of each new session%
    if (a new session) then
        if (packet delivery rate in previous session $\leq Threshold$) then
            choose a new $TunnelAgent$ randomly from the candidate set
        end if
    end if
    encapsulate the IP packet into an IP-within-IP packet
    send the IP-within-IP packet to the $TunnelAgent$
end while
```

delivery rate. Then the source starts by choosing a tunnel agent and tunneling the traffic immediately. If the choice of the tunnel agent is good, the routing path of the flow can avoid the attacker and the packet delivery rate will increase. However, if the randomly chosen tunnel agent still unfortunately leads to the routing path going through the attacker, the packet delivery rate in this session will remain low and result in changing the tunnel agent at the beginning of the next session. After several tries, the traffic may eventually be able to avoid the attacker with a good probability.

4.5 Simulation results and analysis

4.5.1 Simulation and attack setup

Our simulation uses the discrete-event network simulator NS-3 [10], which is a popular, free software package primarily for research and educational use. We choose OLSR [41] as the underlying routing protocol for the ad hoc network. We use attacks as anomaly events that destroy the normal network traffic. An attack module was introduced into NS-3 for generating internal attacks, and involves choreographing compromised nodes in the simulation that launch

Algorithm 2 Algorithm of single tunnel with re-evaluation at the tunnel agent

```
while ( receive an IP packet ) do
    if ( my address == destination address ) then
        ▷ %comment: if the destination of the packet is self, check whether it is a IP-within-IP packet%
        if ( protocol number == 4 ) then
            ▷ %comment: remove the IP-within-IP header and forward the packet to original destination%
            remove the IP-within-IP header
            check IP header and send the IP packet to destination address
        else
            handle the packet as normal IP packet
        end if
    else
        continue to forward the packet
    end if
end while
```

the attack.

In our simulation, we choose to launch a black hole attack as it can directly influence the packet delivery rate, which acts as the instigator for turning on single tunneling mechanism. Strictly speaking, there are two types blackhole attacks: one exploits the routing protocol by advertising a forged route to a destination in order to intercept packets; the other one directly drops the intercepted control and/or data packets entirely or selectively without any forwarding. In order to simplify the network anomaly detection as it is not the focus of this work, per-flow based information of end-to-end packet delivery rate is used as the indicator of whether an attack is on-going. Therefore, we set the black hole attack to only drops data packets because it can be reflected directly by the packet delivery rate. Our particular black hole attack does not drop control packets as it is normally easy to detect if the compromised node only drops control packets. For example, the malicious node will be automatically excluded from other nodes' routing table if it drops all the control packets. In such a case, the negative impacts from the compromised nodes on the benevolent nodes will be reduced to a minimum. However, because the underlying network is assumed to be an IP network, where the IP protocol transmits packets in a best-effort manner without any quality guarantee, it is hard to discover the misbehaviour if the malicious node drops the packets selectively, namely only drops data packets but not control packet. Then, in this case the end-to-end packet delivery rate is able to identify the attacks as well as path quality.

In the rest of the chapter, we present simulation results for two different network topologies. Section 4.5.2 presents how the single tunnel scheme works in a static scenario, while section 4.5.3 introduces randomness and mobility to the network deployment and provides comparison between three different schemes (i.e., no tunnel, single tunnel with re-evaluation, single tunnel with re-evaluation and trust assessment).

4.5.2 Static scenario

We built a static, grid-based wireless network containing 36 nodes as shown in figure 4.2. Also, the packet delivery rate in each session is measured in the simulation to provide a real-time feedback to the source.

In the first test, the source node, node 7, begins to send a UDP flow of 2000 packets to

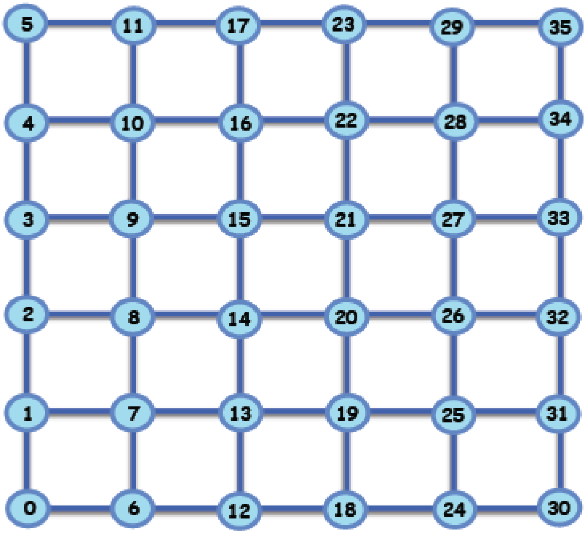


Figure 4.2: Grid network topology of single tunnel with re-evaluation simulation.

the destination node 28 with interval 0.1 seconds from time 20s, while an attacker, node 14, is present and drops all of the UDP packets during the whole simulation period. The re-evaluation parameter M set at the source represents how many packets each session transmits. We compare the performance when M is set to 100 and 200 respectively. The original OLSR routing path from the source to the destination is $7 \rightarrow 14 \rightarrow 21 \rightarrow 28$. Because the attacker node 14 drops all the UDP packets, the packet delivery rate in the first session is 0 as shown in figure 4.4. At the beginning of the second session, the source node compares the packet delivery rate in the first session with a threshold 0.8, which results in launching tunneling traffic. The first tunnel agent is chosen randomly, e.g., node 29. This choice avoids the attacker with the tunneling path $7 \rightarrow 8 \rightarrow 15 \rightarrow 22 \rightarrow 29 \rightarrow 28$. As a result, the packet delivery rate in the second session increases to 0.99 and then keeps higher than the threshold in the following sessions. Note that the first try of choosing tunnel agent in both simulations succeeds as it drives the traffic away from the attacker and pulls up the packet delivery rate, so the source node keeps using the same agent to tunnel the flow. Also, from the comparison of the two different settings of re-evaluation parameter M, we can see that the smaller M is, the faster the source reacts to the attack. Therefore, when $M = 100$, the performance is better than the case where $M = 200$.

The second simulation shows how the source adapts to an attacker coming in after the flow starts, as shown in figure 4.3. The source node 7 sends 2000 UDP packets to destination node

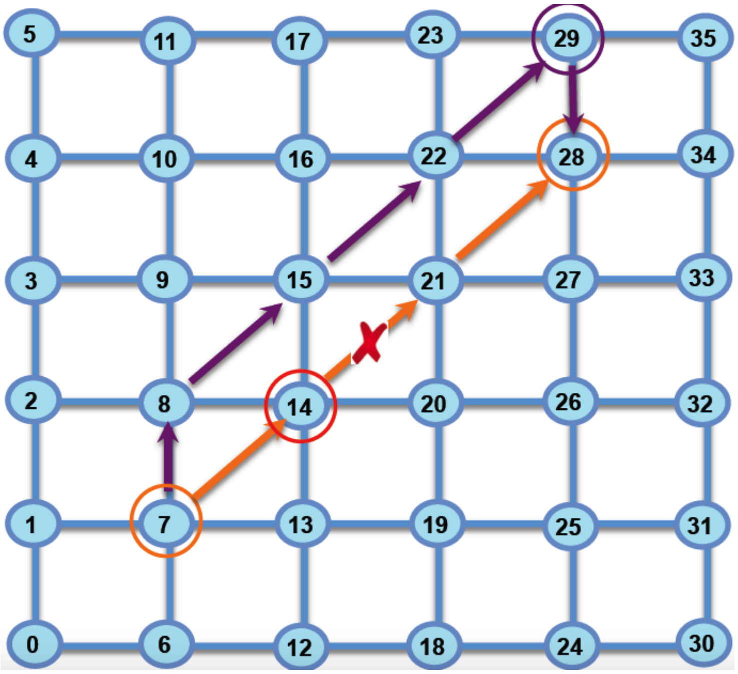


Figure 4.3: Single tunnel with re-evaluation in a grid network topology with one attacker: source node is 7, destination node is 28, attacker is node 14.

28 from time 20s to time 220s with a packet interval of 0.1s along the path 7 → 14 → 21 → 28. We also compare two different settings of the re-evaluation parameter M: $M = 200$ so that there are 10 sessions and $M = 100$ so that there are 20 sessions. Node 14 with a red circle in figure 4.3 launches a blackhole attack, which involves dropping all UDP packets from time 56s to time 100s. In the case of $M = 200$, as a result of the attack beginning, the packet delivery rate in the third session drops to 0 as shown in figure 4.5. Then, at the beginning of the fourth session, the source finds that the packet delivery rate of the previous session falls below the threshold 0.8 and therefore chooses node 29 as the tunnel agent for this session, which successfully avoids the attacker and pulls the packet delivery rate above the threshold 0.8. In the following session, the source will not change the tunnel agent as the current one works well. In the case of $M = 100$, the source detects the attack faster than the previous case and start the tunneling sooner.

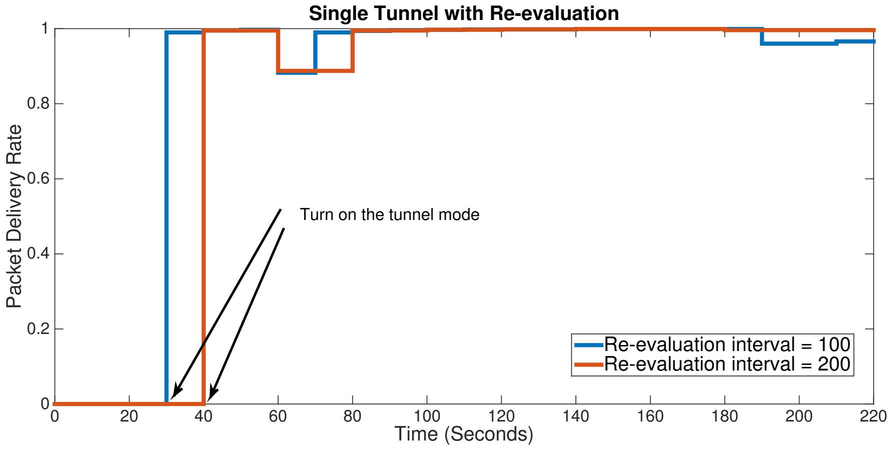


Figure 4.4: Single tunnel with re-evaluation while the attacker presents all the time: the source node turns on tunnel mode at time 30s and 40 when M is 100 and 200 respectively.

4.5.3 Mobile scenario

Section $4.5.2$ illustrates the mechanism of the single tunnel in a static scenario. Next we conduct simulations in a network setting which introduces randomness and mobility. 36 nodes are deployed in a square area randomly and move according to the *"RandomWalk2dMobilityModel"*, which is a $2D$ random walk mobility model in NS-3. The mobility pattern of this model is often recognized as a Brownian motion model. Each node moves towards a random direction at a random speed over a specified distance or a fixed amount of time. If a node hits one of the predefined boundaries of a rectangle, it rebounds on the boundary with a reflexive angle and speed. [10]

Three data flows were transmitted concurrently and each sends 2000 UDP packets. These three source-destination pairs, as well as attackers, were chosen randomly. Thus, we send 6000 packets in total and will calculate the number of received packets by three flows in total to evaluate our tunnel schemes. We compare three schemes discussed so far: transmission without tunneling, the single tunnel with re-evaluation and the single tunnel with re-evaluation $\&$ trust assessment. For the transmission without tunneling, we simply send the data flow in the presence of attackers. The single tunnel with re-evaluation begins to send three data flows as normal, i.e. without tunneling, and monitors the packet delivery rate for each flow. Once the packet delivery rate of a flow falls below the threshold, it turns on the single tunnel

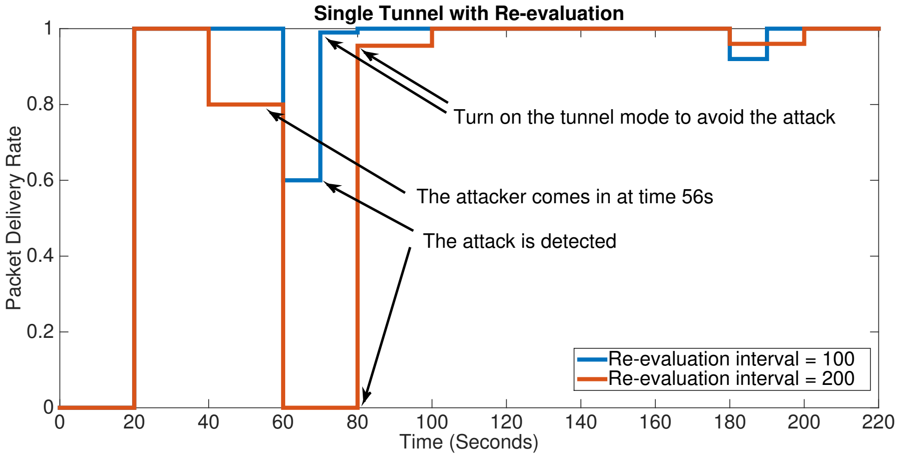


Figure 4.5: Single tunnel with re-evaluation: the data flow starts at time 20s, the attacker comes in at time 56s and leaves at time 100s.

with re-evaluation mode. The re-evaluation parameter M is also set to 100. As for the single tunnel with re-evaluation & trust assessment, every setting is same as the single tunnel with re-evaluation except the candidate set of tunnel agents. Here the trust assessment mechanism is implemented by introducing a trust module to NS-3 to refine the candidate set such that compromised nodes who have trust score lower than a threshold will be removed.

We conducted simulations under different attacker settings and used the "RandomWalk2dMobilityModel" with different speed settings. Namely, the simulations were performed in the presence of one attacker, two attackers, three attackers and four attackers respectively. Also, in NS-3, users may specify *"Speed"* attribute as a random/fixed variable used to decide the speed (m/s) in "RandomWalk2dMobilityModel" model. We tuned the "Speed" attribute and performed simulations with different speed settings. Also, as randomness plays a significant role in choosing source, destination, attackers and tunnel agents, the result might also deviate from the theoretical derivation in a single run of the simulation. Hence, we conducted five runs for each attacker and speed combination to remove undesirable randomness. The simulation results of the number of packets received are shown in figure 4.6, while figure 4.7 shows the average end-to-end delay for the packets received successfully.

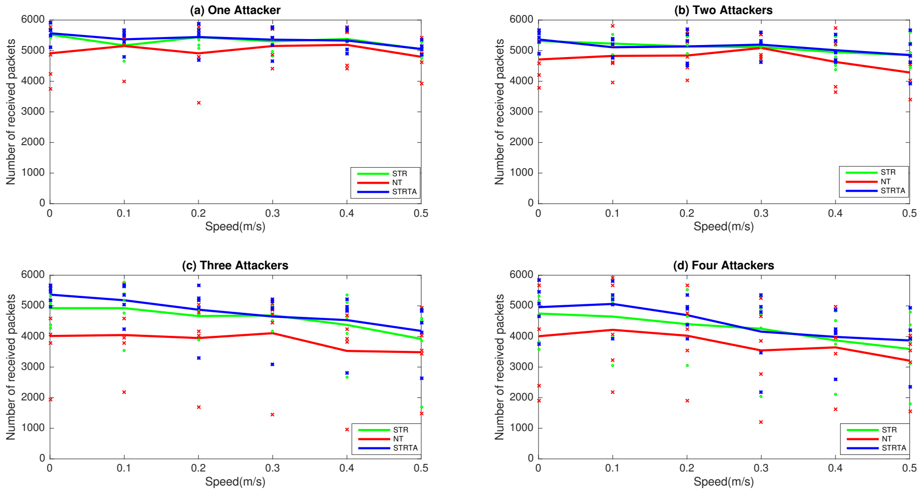


Figure 4.6: Packet delivery performance comparison: comparing the number of received packets between no tunnel (NT), single tunnel with re-evaluation (STR) and single tunnel with re-evaluation & trust assessment (STRTA).

4.5.4 Analysis

The simulation results of static network topology in section 4.5.2 proves that the tunnel scheme works as the design intended. The single tunnel with re-evaluation method allows the source to adapt to the attacks with a small amount of delay. The source is able to detect the attack from the observed drop in the packet delivery rate with a predictable delay relating to the session length. However, as randomness is introduced in the process of choosing tunnel agent, how fast the source can restore the traffic is not deterministic – it may be the case that the underlay beneath tunnel will go through attacking node, resulting in tunnel agent change in the next session. Therefore, this tunneling method can be considered as a *best effort* and light-weight resolution to mitigate attacks.

In the wireless mobile scenarios examined in section 4.5.3, randomness and mobility are introduced to evaluate the proposed tunnel schemes from a more general perspective. Three data transmission schemes are compared: transmission without tunneling (NT), the single tunnel with re-evaluation (STR) and single tunnel with re-evaluation & trust assessment (STRTA). In figure 4.6 and figure 4.7, the solid lines are the average of five runs for a particular scheme, while the symbols (blue star, red cross and green circle) are the number of packets received

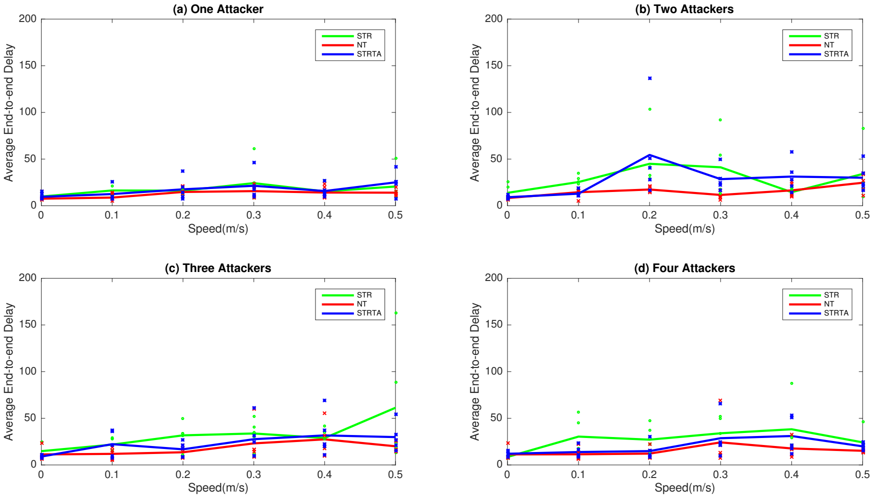


Figure 4.7: Delay comparison: comparing average end-to-end delay between no tunnel (NT), single tunnel with re-evaluation (STR) and single tunnel with re-evaluation & trust assessment (STRTA).

by three flows in total with respect to a combination of attacker and speed setting. The results show that, in general, the single tunnel with re-evaluation out performs transmission without tunneling, though in some specific cases it might be worse than without tunneling due to a poor choice of the tunnel agent. Introducing trust assessment can improve the performance in most simulation settings since it reduces the chance of hitting a compromised node. One observation is that introducing random movement may sometimes allow the traffic avoid the attacker automatically when the attacker moves away from the routing path. However, when the speed increases, the dynamic changes in the topology may also cause link failures and routing table re-computation. Also, when more attackers are present, the benefit of adopting tunneling technique also increases in terms of a higher packet delivery rate because it is harder for the source-destination pair to get rid of attackers by simply moving around randomly. However, the trade-off for more packets received successfully is increasing delay, as shown in figure 4.7. In most cases, both of the tunnel schemes (STR and STRTA) have a larger end-to-end delay than using no tunnel (NT). Hence, our overlay tunnel is a proper choice for delay tolerant applications but not necessarily delay-sensitive applications as it sacrifices a small amount of end-to-end delay for a better packet delivery rate.

As a summary, our approaches can mitigate the negative influence of network anomaly events, such as attacks and network failures, as long as the abnormality is detected.

4.6 Conclusion

As security is an essential feature that affects the successful operation of a MANET, there have been many studies devoted to security for MANETs at different layers. Most research done so far has focused on securing network protocols. Unfortunately, many of them struggled with the trade off between security protections and the overhead introduced to network protocols. Moreover, in those work, securing MANET requires protocol modification or even re-design. In this chapter, we proposed an alternative overlay approach, which can be used as a complement to other MANET security mechanisms, and involves the use of a policy tool at the source to enhance the network robustness and mitigate the performance degradation caused by attacks or network failures. Our approach is a lightweight tool that does not require any modification for the underlying network protocol. Furthermore, two optimizations, re-evaluation and trust assessment, are introduced to improve the effectiveness of the overlay tunneling. The three techniques we present are very flexible so that overlay tunneling, re-evaluation and trust assessment may be combined together or adopted independently to work with other network protocols/functionalities.

Chapter 5

MobilityFirst-based Internet of Things

5.1 Introduction

The initial concept and implementation of the Internet of Things (IoT) appeared as early as the 1980s and became popular in late 1990s [28]. Recent development in many relevant areas, including automation, wireless sensor networks, embedded systems and micro-electromechanical systems (MEMS), has accelerated the evolution of the Internet of Things (IoT) [29, 57]. Strong demand has led to the recent proliferation of IoT systems as IoT applications now exist in nearly every field and are playing an increasingly important role in our daily life [16], e.g., healthcare systems, building and home automation, environmental monitoring, infrastructure management, energy management and transportation systems. According to the Federal Trade Commission (FTC), the number of IoT devices outnumbered of the number of people in the workplace already [42], and the number of wireless devices connected to the Internet of Things will be about 26 billion by 2020 and will greatly outnumber hub devices (smartphones, tablets and PCs) [18].

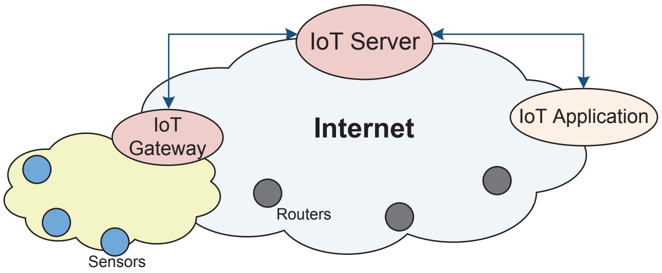
As a result of the recent trend of extending the boundary of the Internet to include a wide array of untraditional computing devices, the Internet of Things makes the connection between the real world and the virtual world tighter than ever before. However, connecting various stand-alone IoT systems through the Internet brings many challenges, such as scalability, naming, resource constraints, mobility, inter-operability, security and privacy. To address these challenges, new IoT solutions have been proposed that utilize clean slate future Internet architectures, such as XIA [1], NDN [8], Nebula [9], and MobilityFirst [7]. These approaches solve important problems such as scalability, mobility, naming, context, robustness, content retrieval, and evolvability. However, one major hurdle facing IoT is security and privacy. As defined in [42], "IoT" refers to the *connectivity between everyday objects and the Internet* and *the ability to exchange data between them*. As a result, potential security and privacy risks exist in a broad scope, ranging from the physical world to the Internet, and can be exploited to harm people. For example, a compromised IoT device may facilitate attacks on other systems. Unauthorized access may result in the leakage and misuse of personal information. A breach in the Internet may feed back to the physical world and create risks and threats to people's physical safety.

Although conventional security will address many problems, there are unique aspects associated with IoT security that necessitate a broad and holistic approach to secure IoT. The deployment of IoT devices is typically in an unattended manner, which facilitates the possibility of physical attacks. Low-end IoT devices are incapable of performing heavier, conventional cryptographic algorithms due to their constrained resources. A large quantity of IoT devices adopt wireless as a means to communicate, which is open to eavesdropping and jamming. In this chapter, we first investigate the current IoT solutions and their security and privacy threats. Then, we present a design for an IoT platform based on the MobilityFirst infrastructure. Specifically, we introduce an IoT middleware to manage IoT devices and data. Based upon our security and privacy analysis, security countermeasures are proposed to protect against those challenges.

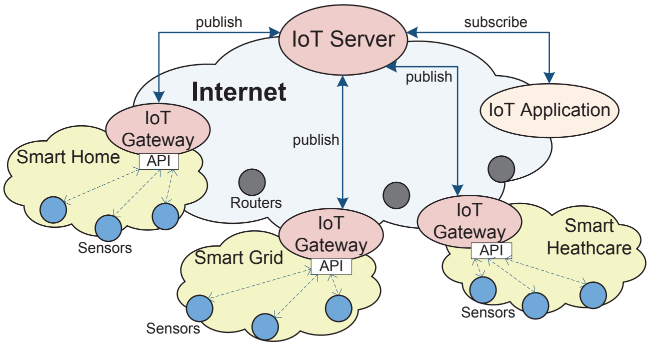
5.2 Survey of the Evolution of IoT Architectures

Though the IoT has a promising future, many current IoT infrastructures do not support applications seamlessly. Historically, a large quantity of stand-alone IoT systems were proprietary implementations, such as DF-1 [14], MelsecNet [6], Smart Distributed System (SDS) [12], BACnet [102], etc. These fragmented solutions were typically integrated vertically and characterized as "silo" solutions, as shown in Figure 5.1(a). A large amount of independent legacy IoT systems co-exist across the Internet. However, their "silo" nature conflicts with the open spirit of the Internet and hence introduces problems of inter-operability and service-level interaction, which limits the benefits of IoT systems and could impede large-scale IoT deployment.

As the traditional approach has many fundamental issues, overlay-based IoT architectures



(a)



(b)

Figure 5.1: Legacy IoT architectures: (a) Silo architecture of standalone IoT systems. (b) The overlay-based IoT architecture connects standalone IoT systems with the IoT server through standardized APIs on the IoT gateways.

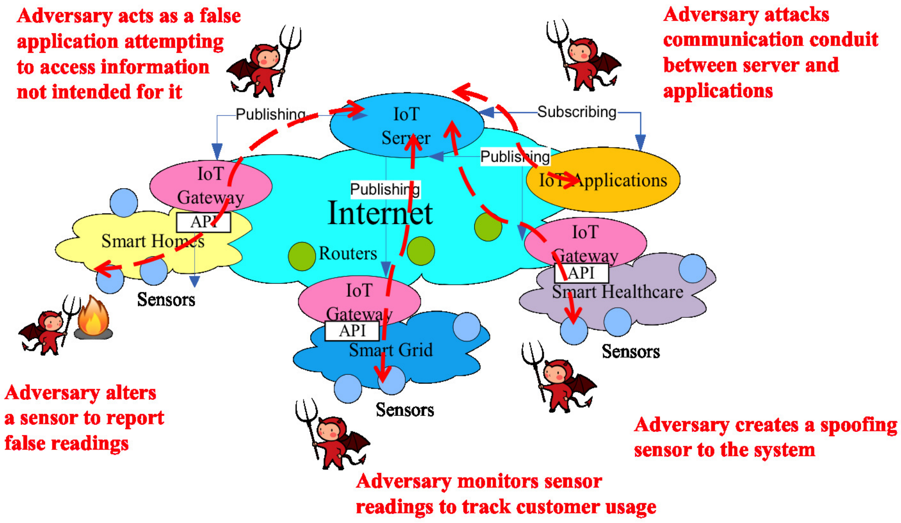
were proposed in an attempt to unify IoT solutions. Figure 5.1(b) shows this approach, where an overlay IoT forms a network on top of the current Internet. Standard control and data APIs were adopted to connect IoT devices and the Internet. Data from various devices in different IoT deployments is pushed to central IoT servers through IoT gateways across the Internet. IoT applications then directly subscribe to the IoT server. Low Throughput Networks (LTN) [19], Global Sensor Networks (GSN) [3, 22] and Constrained RESTful Environments (CORE) [116,117] are representatives of the standardization efforts involving the overlay-based IoT approach. This overlay approach, however, has several weaknesses that makes it not ideal for wide deployment. First of all, the merging of control and data forwarding at the central server is a bottleneck and causes scalability concerns. Secondly, the central IoT server is a single point failure and represents an ideal point of attack. Further, this solution does not seamlessly support mobility, which is a major requirement of the future Internet and a desirable characteristic for IoT applications.

Because of the increasing importance of the IoT, future Internet designs need to take IoT into consideration and support it effectively and efficiently. *NDN-based IoT* and *MobilityFirst-based IoT* are two representatives of future Internet based IoT systems as they capture the key features of the IoT as well as follow the current trends of the Internet. As *data* plays a pivotal role in the IoT, the NDN-based IoT is proposed to better support content retrieval. On the otherhand, MobilityFirst-based IoT builds upon the MobilityFirst network [87, 89], whose major goal is to support *large scale mobility*, and therefore is better able to handle mobile scenarios where the IoT devices are moving. One typical mobile IoT application scenario is vehicular networks, where sensors installed in the moving vehicles collect data and provide the data to the relevant applications through the underlying IoT infrastructure. In this chapter, we use the MobilityFirst network to support the IoT and present our IoT design in section 5.4. We note, though, that some of our security approaches are general and can be applied to other future Interent architecture based IoT system.

5.3 General Security Analysis of IoT Systems

The IoT extends the Internet to the physical world and thus poses many new security and privacy challenges. Some of the problems are due to the intrinsic characteristics of the IoT and its differences compared to traditional networks, while others arise as a result of the integration of the IoT and the Internet. As shown in Figure 5.2, various adversaries may come in at different points to attack IoT systems. To protect against those attacks, it is important to examine the security problems according to the information flows and potential adversarial points of control. Below, we outline four security and privacy problems:

- **Authentication and physical threats**: with a highly distributed nature, a large number of IoT devices, such as RFID tags and wireless sensors, are generally deployed in public areas without any protection, which makes it difficult to manage the devices and causes the devices to be vulnerable to physical attacks. For example, an illegitimate sensor may register itself claiming that it is at one location while it is actually at a different location. Or a sensor installed in a room monitoring the room temperature is moved



Adversary acts as a false application attempting to access information not intended for it

Adversary attacks communication conduit between server and applications

Adversary alters a sensor to report false readings

Adversary monitors sensor readings to track customer usage

Adversary creates a spoofing sensor to the system

Figure 5.2: Potential threats for the IoT systems.

to another room by a malicious person. This introduces the challenge of authenticating IoT devices, which involves recognizing the device and verifying its association with a correct topological address.

- **Integrity**: the unattended environment for IoT devices also makes data integrity a concern. Once deployed, most of these devices, if not all, operate in a self-supported manner. As with very limited maintenance or even no maintenance, tampering data is much easier of a task than in a supervised wired network. Besides, as a result of a natural loss of calibration or a deliberate perturbation of the measurement environment by an attacker, the data collected by IoT devices is quite likely to have low quality and might be corrupted at the environmental level. In short, IoT data may be noisy and easy to spoof and forge.

- **Confidentiality**: the communication method between devices and the gateway is primarily wireless, which results in confidentiality risks. For example, eavesdropping is a major concern in the wireless networks. Unfortunately, unlike many other wireless environments such as cellular networks and Wi-Fi networks, it is much more difficult for IoT networks to provide confidentiality for data transmission due to the resource-constrained nature of low-end devices, which are a large fraction of IoT devices. Different from typical devices in traditional wired and wireless networks, such as smartphones, tablets, PCs and routers, most of the devices in IoT networks are active sensors or passive RFID tags, which have very limited resources and capabilities. Constraints on power, computational capability, storage and other aspects of an IoT device introduce a high barrier for it to perform the necessary operations to achieve data confidentiality, such as through encryption and key management.

- **Privacy**: as an existing public concern for monitoring and interacting with the real world, the consequence of information leakage in local IoT networks becomes exacerbated when integrated into the global Internet. By connecting real world objects and information through the Internet, data may become accessible to various organizations and domains across the Internet, instead of only being revealed to a small group, which makes it more likely to be exposed to sophisticated malicious parties and therefore increases the probability of being exploited and attacked.

Conventional security and privacy techniques are not necessarily appropriate for the IoT due to the special characteristics of the IoT. The attractive prospect of IoT applications, as well as the strong needs of increasing public confidence about security and privacy issues, requires new and comprehensive solutions to not only protect local IoT devices but also the broader Internet aspect of the IoT. In the following sections, we will examine the problems aforementioned and explore security and privacy techniques to support the IoT infrastructure based on the MobilityFirst network.

5.4 MobilityFirst-based IoT Architecture

5.4.1 IoT Architecture Design based on MobilityFirst Infrastructure

As a robust and trustworthy mobility-centric architecture with abundant in-network services for the future Internet, MobilityFirst can address many challenges that today's IoT is facing, such as scalability, mobility, content retrieval, inter-operability, security and privacy, etc. Therefore, we propose a unified solution to support IoT based on MobilityFirst network architecture [90]. Our IoT platform consists of four basic components shown in Figure 5.3:

- **Devices**: a wide variety of devices, such as sensors, actuators and tags, that use embedded techniques to sense, communicate and/or interact with the external environments.

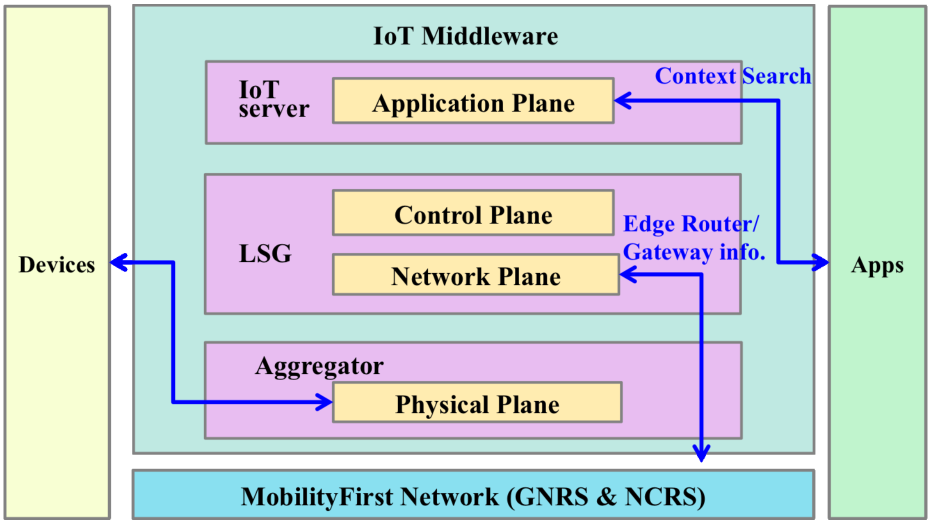


Figure 5.3: There are four basic components in the architecture of IoT in MobilityFirst: devices, IoT middleware, MobilityFirst network and applications.

- **MobilityFirst network**: MobilityFirst provides connectivity for different distributed IoT devices and applications. Due to the seamless mobility support of MobilityFirst, besides well-established IoT system, more dynamic IoT applications and systems can be developed on top of it.
- **IoT middleware**: MobilityFirst-based IoT middleware integrates three functional layers—*Aggregator*, *Local Service Gateway (LSG)*, and the *IoT server*. The aggregator supports sensor abstraction to hide the hardware specifics for sensors and presents a single interface to query and subscribe to the sensor data. LSG connects the local IoT system to the global Internet and provides necessary management, including naming resolution, key management and context processing services. The IoT server is a logically centralized server in the control plane that manages subscription memberships as well as provides a lookup service so that subscribers may query for the IoT data/services.
- **Applications**: end users who consume the IoT data and may feed back to the external environment through actuators.

Figure 5.4 explains how the IoT architecture works with the support of the MobilityFirst infrastructure. The aggregator collects data and then the LSG sends the aggregated data to the storage location. The raw data might be processed by the LSG for context refining/aggregating

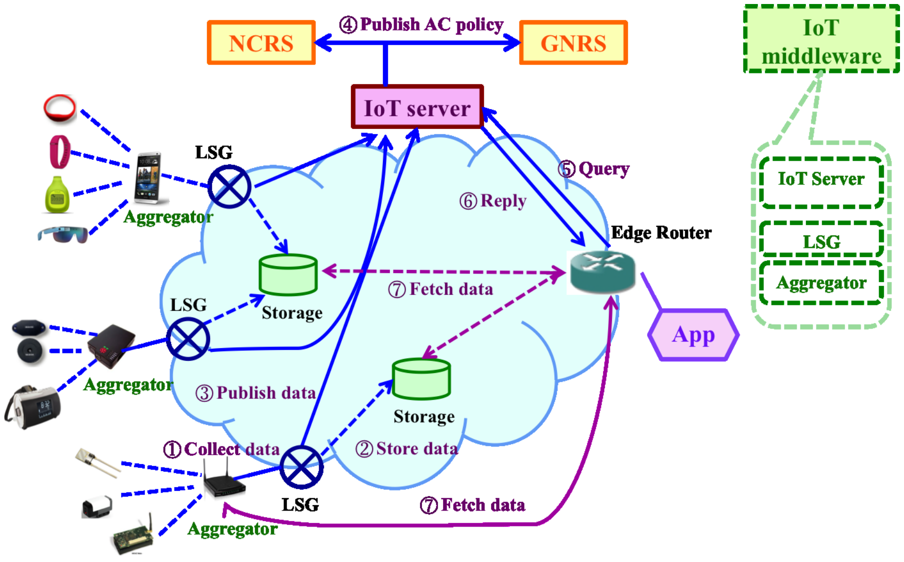


Figure 5.4: Unified IoT architecture based on MobilityFirst infrastructure.

purposes. Meanwhile, the LSG publishes the data information, which contains a data GUID, the access control policy and the storage location information (e.g., human readable names or network address), to the IoT server so that end users may query the IoT server about where to fetch the data. The IoT server may decide how to enforce the access control policy: either at itself or push it to the NCRS/GNRS. The data consumer, typically an application, first makes a query at the IoT server for data information through its edge router and then fetches the data from the storage location or the aggregator directly.

5.4.2 Protect IoT through Existing MobilityFirst Network Services

As shown in Figure 5.3, the MobilityFirst-based IoT architecture consists of four basic building blocks: *Devices*, *IoT middleware*, *MobilityFirst network* and *Applications*. Our goal is to provide clean and secure data to various applications/services in the upper layer and make their development/management easy. Thus, as illustrated in Figure 5.5, we integrate the security/privacy mechanisms into the network, the IoT middleware and the lower layer "devices", but not the application layer.

We have discussed security and privacy aspects of the MobilityFirst network in Chapter 2 and Chapter 3, with a focus on the name-based service layer. Those rich in-network services of MobilityFirst enable many powerful functionalities to protect against a wide variety of attacks.

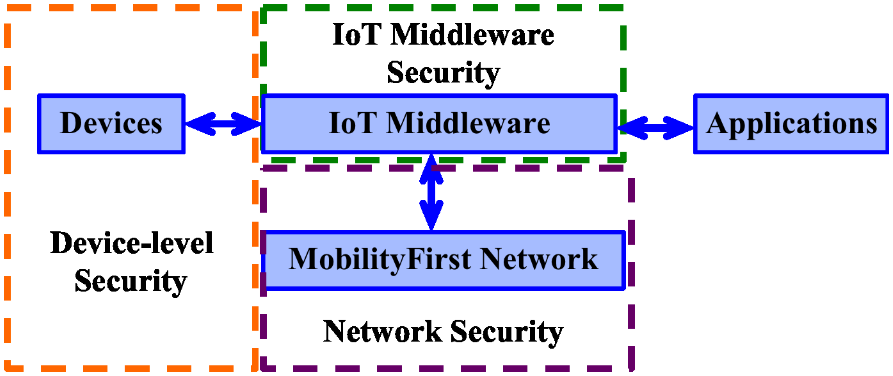


Figure 5.5: Security/privacy mechanisms are introduced into devices, IoT middleware and MobilityFirst network.

For example, the NCRS serves the role of a certificate registration center that associates human-readable names to certificates. This enables various security methods, such as encryption and authentication, to secure the data transmission above the IoT device layer. Together with the security schemes at the device level, which provide secure communication channels between IoT devices and the Internet, the confidentiality of the data flow from the bottom (IoT devices) to the top (applications) can be achieved.

The security mechanisms enabled by network services, i.e. the GNRS and the NCRS, build a shield against many attacks, such as false registration attacks and device misplacement attacks, which were mentioned in section 5.3. *False registration attack* refers to an adversary registering a device with a fake location. For example, in an IoT system (shown in Figure 5.6) crossing two domains (e.g. the Winlab domain and CoRE building domain), an adversary installs a temperature sensor $S?$ in Winlab room A103 and then tries to register it as a temperature sensor in CoRE building room 530. As the registration is fulfilled by the IoT name resolution service (IoT-NRS), discussed in section 5.5.1, the adversary provides a fake location attribute together with other information, such as manufacturer, brand, serial number, to the IoT name resolution service, which might sit in another domain. *Misplacement attack* is another form of attack involving manipulating device location by moving a device to another location (e.g., move sensor S3 in Winlab room A102 to CoRE building room 531 in Figure 5.6) without updating the new location. The common characteristic of these two attacks is *location forgery* and

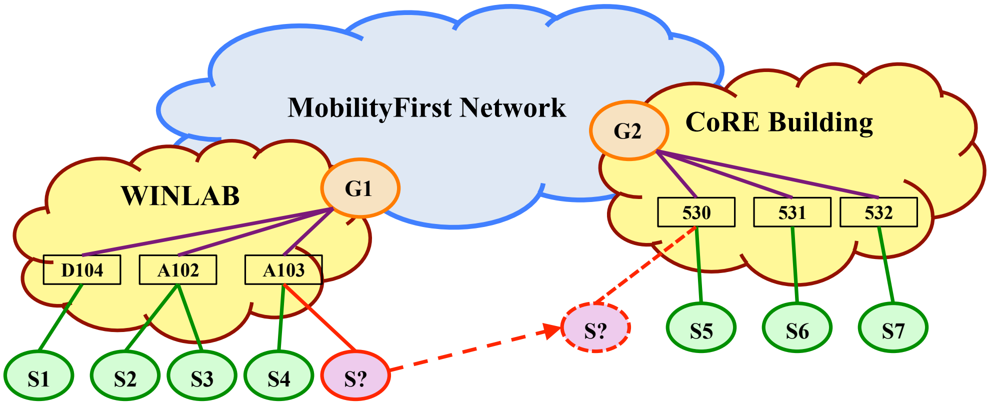


Figure 5.6: An example of false registration attack: a malicious sensor “S?” in Winlab room A103 claims that it is in CoRE building room 530.

is similar to *GNRS false announcement attack*, which is defined as a network object with identifier $GUID_1$ in network $NA1$ inserting or updating a GNRS mapping $< GUID_1, NA2 >$ with wrong network address to the GNRS. Therefore, we can apply the same philosophy of preventing GNRS false announcement attacks to location forgery attacks in the context of the IoT.

The local service gateway (LSG) plays an essential role in protecting against attacks associated with location forgery. As for the false registration attack, before the LSG forwards the registration request, it signs the request with its private key. The digital signature has two functions here: (1) guarantee the data integrity so that others can not forge the message; (2) provide a proof of the origin of the message. When the IoT-NRS receives the registration request, it checks the signature and the request content. If the location attribute contradicts with the origin proved by the LSG's signature, the registration request will be rejected; otherwise, the IoT-NRS will then proceed to register the sensor and assign a GUID. The functionalities of the IoT-NRS will be discussed in section 5.5.1. Take the IoT system in Figure 5.6 for example:

1. S? → G1: $\{[GUID_{M1}, (GUID_{M1}, Request, T_1)_{S?}]_{M1}\}$

   The malicious sensor $S?$ launches a false registration attack by sending a registration request to the IoT-NRS $M1$. The request claims that $S?$ is in CoRE building domain, while it is actually in Winlab domain.

2. G1 → M1: $\{GUID_{G1}, [GUID_{M1}, (GUID_{M1}, Request, T_1)_{S?}]_{M1}\}_{G1^{-1}}$

> The Winlab LSG $G1$ also signs the request with its private key to assure the origin of the request. When $M1$ receives the request forwarded by $G1$, it will reject the request since the request is not signed by $G2$.

The methodology to prevent misplacement attack is similar to the method discussed above. Not only the registration request but also other messages, including data message and membership renewal message, need to be signed properly by the LSG. Hence the error can be detected once the information of the origin sensor does not match its LSG's signature, such as changing the location without updating with the IoT-NRS.

Another significant contribution provided by network services is privacy protection, such as GNRS access control discussed in Chapter 3. The access control in the GNRS protects the data privacy, and also increases the difficulty of launching attacks by restricting adversarial access to information that is essential for launching an attack, whatever that attack might be. Access control enforced at the GNRS query is a powerful tool as it can provide the GNRS mapping owner, who is typically the data owner or a surrogate in the context of IoT, the ability of choosing who it is willing to communicate with. With the support of access control in the GNRS, IoT devices or data owners can protect the IoT data's location information contained in the GNRS mappings against unauthorized disclosure, while at the same time ensure the mapping's accessibility to legitimate subscribers or applications. In addition, GNRS access control can support advanced services, such as allowing the mapping owner to decide when and where it is reachable. These fine-grained functionalities provided by GNRS access control make it possible to specify detailed policies/regulations while distributing the data collected by the IoT devices.

Our MobilityFirst-based IoT architecture provides a unified solution to solve issues of interoperability as well as support mobility. Our design also can benefit from many built-in security features of the MobilityFirst network infrastructure. However, this architecture still has some unique security concerns that arise and are related to general security concerns aforementioned in section 5.3. Specifically, most of the security properties and advanced services are based on the concept of a GUID, which is essentially a public key. Unfortunately, low-end IoT devices, which account for a majority of IoT devices, cannot afford to perform expensive asymmetric cryptographic operations. This leads to key management challenges for our IoT solution. Therefore, the following sections will address these unique security concerns with the focus on key management.

5.5 IoT Middleware Security

The IoT platform in MobilityFirst introduces a new essential component, *IoT middleware* to handle the data processing and distribution in the data plane as well as system management in the control plane. The IoT middleware consists of three layers, *aggregator*, *local service gateway (LSG)* and *IoT server*, so that it can support a variety of functionalities, including managing the IoT devices, processing the raw data collected by the devices and then distributing the processed data through MobilityFirst network. This approach allows a clean separation of IoT systems from the underlying network so that the network is only responsible for transmitting the IoT data, just as it handles other network traffic. Therefore, integrating our IoT platform into MobilityFirst architecture has a lightweight touch that is easy to deploy.

In the IoT middleware architecture described in Figure 5.3 and Figure 5.4, the *aggregator* sitting at the bottom level is usually located near or at the gateway of the local network. This is the first stage of data processing as the aggregator collects raw data directly from a variety of heterogeneous IoT devices and filters out the redundant data. The introduction of the aggregator hides the complex heterogeneous hardware from upper layer applications and facilitates application development. The *LSG* level contains one *IoT Name Resolution Service (IoT-NRS)* and several other functional components depending on the system and application requirements. The top level is the *IoT server*, which is essentially a logically centralized but physical distributed database that provides lookup service for data consumers. The aggregator, the LSG and the IoT server form the middleware of MobilityFirst-based IoT and connect the various IoT devices with the upper layer applications.

Our security measures are primarily integrated in the LSG layer, while the access control policy is enforced at the IoT server, who might push it further to the GNRS. From the perspective of the data consumer, which is typically an application, to either fetch the processed data from the data storage location or get the raw data from the aggregator directly, it needs to know where to fetch the data. Therefore, the data consumer looks up the data storage location at the IoT server with its identity. On the other side, the IoT server evaluates the request with the access control policy defined by the IoT data owner. If the decision is “approve”, it returns the data location to the data consumer (possibly with a token depending on the specific access control policy). Then the consumer may the be able to fetch the data or setup a subscription. In the case where the IoT server pushes the enforcement of the access control policy to the GNRS, the GNRS will grant a token to the data consumer if the access decision is “approve”. Then the data consumer can present the token to the IoT server to request data information.

At the data producer side, efficient and secure IoT device and data management is not only necessary but crucial for providing accurate and fresh IoT data. Particularly, key and identifier management plays an essential role in identifying/verifying IoT devices as well as establishing/maintaining confidential communication channels in the local IoT systems. However, due to the intrinsic characteristics of IoT devices, especially low-end devices, we are facing key management challenges unique to IoT systems. To cope with those challenges, we integrated a IoT Name Resolution Service at the LSG to enhance the security, efficiency and inter-operability.

5.5.1 Overview of the Name Resolution Framework in MobilityFirst-based IoT

As discussed in Chapter 2, the GNRS and the NCRS center on the concept of the GUID, which identifies a network entity and facilitates network address related operations, such as routing, multicasting, multihoming, etc. Meanwhile, from the security point of view, the GUID is also the public key of a network entity, which enables a variety of security operations, e.g., encryption and authenticity verification. The dual role of the GUID as being the identifier and the public key brings built-in security protection, convenience and efficiency into the MobilityFirst network.

Within the scope of the IoT, however, the identifier application/assignment operation and relevant cryptographic computations may be beyond the capability of most IoT devices. We are not worried about those devices with strong computational capability and sufficient resources, such as smartphones, laptops and tablets. Instead, we focus on low-end IoT devices, such as

RFID tags and small sensors, which accounts for a majority of IoT devices. These low-end devices, unfortunately, have very limited computational capabilities and strict resource constraints in terms of storage, power, bandwidth, etc. Hence, they are unable to perform many heavy duty tasks, including the computation operations of public/private cryptography. To enable secure communication, we propose a lightweight approach by using symmetric keys to replace the GUID in the scope of local IoT systems. In other words, an IoT device's identification associated with membership within a local IoT system is associated with a symmetric membership key shared between the device and the local group authority, i.e., the LSG. Symmetric cryptography is chosen to make the membership key because it is lightweight and has much less computational requirements.

On the other side, in order to smoothly integrate the local IoT system into the global MobilityFirst network and take advantage of the rich network services provided by MobilityFirst, it is necessary to maintain use of the GUID in the scope of broader global network so that data consumers can contact IoT devices or retrieve/subscribe IoT data with the corresponding GUIDs. In order to reconcile the conflicts of the two different cryptographic schemes (used in two different scopes), integrating an IoT name resolution service (shown in Figure 5.7) in the middleware to handle naming-related issues is the best solution as it keeps a clean separation between the underlying network infrastructure and the IoT system. The IoT-NRS bridges the gap between the different cryptographic schemes used in the local IoT network and the global Internet. This design makes IoT systems flexible, extensible and easy to manage.

5.5.2 Major Functionalities of IoT Name Resolution Service

Our IoT Name Resolution Service (IoT-NRS) can be considered as a simplified embodiment of the MobilityFirst name-based service layer in the context of the local IoT systems. It assigns identifiers, establishes long-term keys (i.e., membership keys) and provides key/identifier related management. In the following sections, we envision the major functionalities the IoT-NRS should have.

- **GUID Assignment**: IoT-NRS serves as a surrogate and works together with the NCRS to assign GUIDs to network objects, such as IoT data, certain devices or a group of IoT

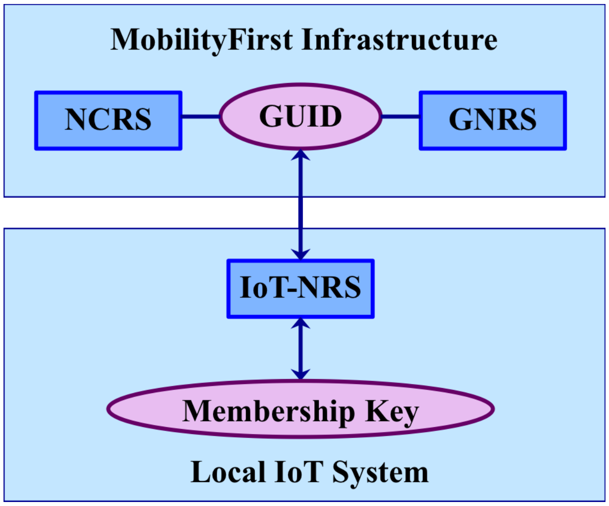


Figure 5.7: Three-tier name resolution framework of the MobilityFirst-based IoT architecture: NCRS, GNRS and IoT-NRS. The IoT-NRS connects the GUID used in global MobilityFirst network and the symmetric membership key used in the local IoT system.

device, to assist advanced network services in the global Internet. The relation between the GUID and the device/data might be *one-to-one*, *one-to-many* and *many-to-one*. From the application point of view, generally the data is of interest and one piece of data has one GUID, which is associated with a list of attributes describing the data object. With respect to devices, it is more often than not that a group of devices map to a single GUID. As an example, it is reasonable to assign one GUID to all of the temperature sensors in Winlab room D104. However, sometimes one device may obtain a GUID if necessary. For example, the device is the only device in its category or it is important for the application. Also, sometimes one sensor may associate with two or more GUIDs. For example, a multi-function sensor attached to Tom's mug may have $GUID1$ with attributes of location information and $GUID2$ associated with the mug's temperature.

- **Membership Credential Establishment**: IoT-NRS at the LSG may act as a group authority to establish membership credentials associated with the IoT device in the scope of the local IoT system. This long-term membership credential identifies the device in the local group. A proper symmetric cryptographic algorithm is chosen to generate such a credential (i.e. a symmetric key) so that low-end devices can afford to perform security operations. After establishing the membership credential, short-term keys (for example, session keys) and functional keys (for example, attestation key) may be derived from the membership key.

- **Name Translation between GUID and Membership Key**: Two directional name mapping between the GUID and the membership key connects the local IoT system to the global network so that IoT-related operations can be performed seamlessly.
- **Membership Management**: IoT-NRS should also provide membership management, where the membership is represented by the GUID and/or the membership key. Such management includes renewal and revocation operations. Both the GUID and the membership key have expiration times to guarantee key freshness and security strength. Therefore, renewing in a timely and efficient fashion is essential to keep the device/data "alive" in the system. On the other hand, revocation is needed to remove compromised/dysfunctional devices so that potential security/privacy threats can be reduced.

Figure 5.7 illustrates the flow of IoT-NRS's major functions. The most important function of the IoT-NRS is to serve as a registrar. During device discovery, when a new IoT device joins a local group, the IoT-NRS acts as the group authority and registers this new device. The registration establishes a long-term membership key with the new device using a proper key provisioning protocol, which depends on the device's capability, as well as associates the membership key with the device's attributes/identity, such as manufacturer, model, serial number, function, etc. In the global context, the IoT-NRS serves as a surrogate to apply and manage GUIDs for the low end IoT devices within the local group and the data they generated because these devices and data do not have the ability to run asymmetric cryptographic algorithms. After initial device discovery and registration, the IoT-NRS manages the identifiers and keys, and performs tasks that include renewal, verification and revocation.

5.6 Delegation-based Key Provisioning Protocol

5.6.1 Overview

In the scope of local IoT systems, it is critical to discover edge devices and establish long-term membership credentials between those devices and a group management authority (i.e., the IoT-NRS in our IoT architecture) in a lightweight fashion. In the device discovery process, enrolling a new device and establishing a secure membership credential between edge devices such as sensors and actuators can be difficult, as such devices can have heterogeneous and potentially limited computational capabilities. This property can limit interactions for certain protocols and further limit management of membership keys/credentials.

Therefore, we propose a delegation-based key provisioning protocol in this section to facilitate the device discovery process and establish the membership key for the newly added edge device. Specifically, in the context of our MobilityFirst-based IoT architecture, this protocol can serve as: (1) a grouping tool in the initial stage of setting up a local IoT network; (2) registration tool in the device discovery process when a new IoT device wants to join an existing IoT system. Our key provisioning protocol, though, is general and can have broad application. Other use cases will be explored in section 5.6.9.

Problem Description

To generalize our goal, we would like to establish *lightweight* symmetric key(s) between one or more edge IoT devices (referred to as a *Child* device), such as sensors, RFID tags and actuators, and a computing device (referred to as a *Guardian* device) within an IoT network. Here, we emphasize the lightweight nature of the output key and involved cryptographic operations associated with the Child device because of the intrinsic characteristics of the IoT systems as discussed in previous sections of this chapter.

Trust does not come from nowhere. To build a mutual trust between the Child device and the Guardian device, we need to extract the new trust relationship from an existing trust relationship. With that being said, we start from a pre-established trust relationship, with which we can attest the identity and integrity of the target Child. As a result of this reasoning, a third party is introduced into our protocol, known as the *Parent* device. The Parent, for example, can be the owner of the Child device or the manufacturer of the Child. Essentially, the Child and the Parent may have a pre-established trust relationship, in which the Child device stores therein a *parent key* that indicates the Parent's privilege on the Child.

Now the challenge rests with how to leverage a device's parent key and Parent trust establishment to delegate certain privileges to a new party, referred to herein as a Guardian, to enable the IoT edge device (Child) to establish a shared key securely with the new party, which thus acts as the edge device's Guardian. In other words, taking advantage of a pre-existing trust relationship between a Parent device and a Child device, we would like to yield a lightweight symmetric key shared between the Child device and the Guardian device.

5.6.2 System Model

Figure 5.8 describes the system model for the delegation-based key provisioning protocol, which consists of three participants: *Child*, *Guardian* and *Parent*. Assume that there is an IoT network, which may be any type of computing network including various computing devices that couple together, e.g., in a wired or wireless manner. Some devices may be continuously connected, while other devices may be occasionally connected. As illustrated in Figure 5.8, this IoT network includes a Child device and a Guardian device. A third device, a Parent device, can be either within or outside this IoT network. This Parent may be, but not necessarily, a relatively more capable computing device, such as a server computer, gateway device, router, personal computer, smartphone, or any other type of computing device. For example, the Parent may be a computing system of a manufacturer, such as the manufacturer of the Child device. The importance of the Parent arises from its existing trust relationship with the Child, from where we can extract and establish a new trust relationship.

The second protocol participant is the Child, which is an edge IoT device and may be implemented as a sensor, actuator, or so forth. To indicate its trust relationship and parenthood status with regard to the Child, the Parent may provision a *parent key* into a non-volatile storage of the Child device. This non-volatile storage may be a secure storage, accessible only in a trusted execution environment (TEE) [58,59] of the Child device. In some cases, this TEE may be implemented using separate hardware circuitry, such as a separate micro-controller from a main processor of the child device, e.g., a central processing unit (CPU).

As further illustrated in Figure 5.8, a Guardian device is also present in the network. In our MobilityFirst-based IoT architecture, the Guardian is the IoT-NRS of the local IoT network. In other embodiments, the Guardian may take the form of a gateway, smartphone, or consumer computer with an installed delegation protocol package. As an example, the Guardian may be a central authority for a domain (such as an entire IoT network or a portion thereof). The

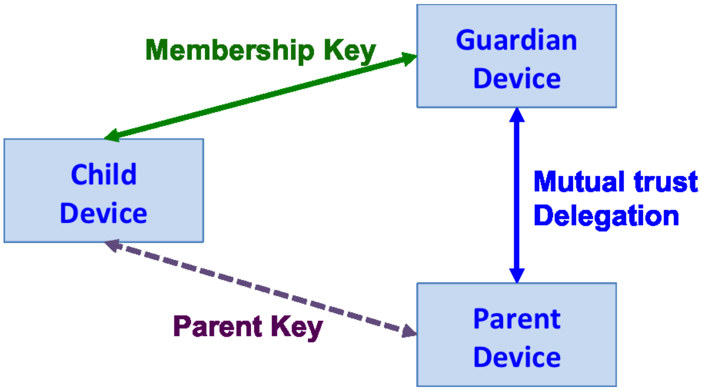


Figure 5.8: The block diagram illustrates the framework of the delegation-based key provisioning protocol: *Child* device is an edge IoT device who wants to join the IoT system and establish a *membership key* with the Guardian device; *Guardian* device is the group authority of the local IoT network, who contacts Parent device to verify Child device and obtains authorization; *Parent* device has a pre-existing trust relationship with Child device in terms of the *Parent key*, and provides verification/delegation to the Guardian device so that Guardian device and Child device can reach a resultant key agreement: membership key.

symmetric *membership key* shared only between the Child and the Guardian is the local long term membership credential established by our key provisioning protocol.

Thus in the context of Figure 5.8, the two parties to establish a trust relationship represented by a key are the Child and the Guardian. A third party Parent has an existing parent key provisioned to the Child, though the Child and the Parent may or may not have a direct communication channel at this point. The only available communication channels are the one between the Child and the Guardian and the one between the Guardian and the Parent through out a protocol session. With the delegation provided herein, key provisioning between the Child and the Guardian can occur, given that the Parent is a trusted third party to provide authentication and authorization of the Guardian to the Child. With this authorization decision, the Parent may delegate a certain subset of (or all) privileges to the Guardian, temporarily or permanently. Effectively, the Guardian now becomes a guardian (or a new parent) for the Child, represented by an established key, referred to as a membership key, between the Child and the Guardian. In the case of full privilege transfer, the established membership key is considered as the new parent key for the Child and the Guardian, which now becomes the new Parent for the Child.

The key provisioning process thus allows the Parent to delegate a subset or all of its privilege on the Child to the Guardian, as represented by the original permission key from the Parent to the Child.

5.6.3 Security Requirements

The delegation from the Parent to the Guardian with respect to the Child can occur to address a variety of concerns. First, IoT edge devices have highly heterogeneous capabilities. The protocol for establishing mutual trust and keys described in section 5.6.6 can support capable devices as well as highly resource-constrained devices. Second, establishment of the IoT device keys may be user-friendly, which conventionally requires the protocol to either leverage a trusted public key infrastructure (PKI) that is transparent to the user, or to remove dependency on trusted servers but minimize dependency on manual interaction by users. Traditional PKI or Kerberos services [121] can be challenging to establish at a global scale, and may require substantial IT professional expertise as well as computational resources for cryptographic processing. On the other hand, solutions for key provisioning without a server are often insecure, such as pre-shared key approaches, lacking resilience by working directly with fixed manufacture/device keys, or involving cumbersome and critical manual interaction such as pairing. We try to address these issues in our delegation-based key provisioning protocol.

Particularly, when developing the key provisioning protocol, the following properties and requirements should be taken into consideration:

- **Scalability** : As many IoT systems are large scale implementations, it would be extremely difficult to establish secure credential for each device if human intervention is required. To enhance usability and chance of practical deployment, it is important and necessary to remove manual operations by the user on an out-of-band (OOB) channel, such as secure pairing. Instead, a trusted third party is introduced to facilitate automatic trust establishment as well as reduce human error.

- **Consistency**: All the three protocol participants should have a consist view of the protocol session. Namely, the three participants involved know who the peers are with respect to the specific session.

- **Dynamic mutual trust and delegation**: With the success of the protocol, each pair of the involved participants can reach a mutual trust at the end of the procedure. Also, the Guardian establishes the mutual trust dynamically with the Parent and obtains delegation before executing the privilege with respect to the Child.
- **Secrecy and forward secrecy**: The principle of the least privilege should be followed. That means any resultant secret key should only be shared between involved two parties. No third party should be able to distinguish the resultant key from a random key. Also, the *forward secrecy* should be conserved so that the compromise of a long-term secret key will not affect the confidentiality of past communication sessions [97].
- **Efficiency and lightweight**: As we focus on those resource limited devices, efficiency and lightweight are among utmost priorities. When designing the protocol, the Child, which is typically, though not necessarily, a resource constrained device, should be protected by shifting work load to the Guardian and the Parent as much as possible.
- **Average administration requirements**: The protocol should require as little professional knowledge as possible. The configurations are automatic and user-friendly so that people do not need to be IT experts to operate the protocol.

5.6.4 Adversarial Model

A key exchange protocol without a well-defined adversarial model cannot be considered as secure and sophisticated. Hence, here we summarize the assumptions regarding to potential adversaries. We assume a active network adversary, who may take full control of all the communication channels (namely, the communication channel between the Guardian and the Child as well as the one between the Guardian and the Parent). This attacker can eavesdrop on messages, modify them at will, inject its own message, delete messages, delay message delivery, duplicate any message and/or replay it later. It may even be able to initiate new protocol sessions and interleave messages from different sessions [35, 83].

To facilitate the dynamic establishment of the mutual trust between the Guardian and the Parent, we assume a trusted certification authority, e.g., Public Key Infrastructure (PKI), is available so that the more capable parties (i.e., the Guardian and the Parent in our context) can use their long-term private information, such as private keys and certificates, to authenticate their identities with the assistance of the trusted certification authority. With respect to the Guardian and the Parent, if its long-term private information that is used to authenticate itself is leaked and is accessible to attackers, then we consider it to be corrupted.

As for the Child, we assume it is a resource constrained device with necessary hardware protections. That means the adversary is at most a weak software adversary but it is not a hardware adversary. The attacker can only access and/or overwrite regular applications or the operating system, but not the non-volatile secure storage, such as the Trusted Execution Environment (TEE) [58, 59]. Also, we assume the adversary cannot modify the hardware circuitry. With that being said, the "Parent" key (shown in Figure 5.8) representing the pre-existing trust relationship between the Parent and the Child should be protected properly, e.g., in a TEE or implemented using separate hardware circuitry. If the "Parent" key on the Child is leaked or modified, we consider the Child to be compromised.

5.6.5 Protocol Design Choices and Considerations

In order to develop a delegation-based key provisioning protocol that satisfies the requirements aforementioned in section 5.6.3 in the presence of attackers that were described in section 5.6.4, we first investigate several existing key exchange protocols and authentication protocols. Based upon the protocol analysis, we incorporate and customize Choo's three-party key distribution protocol (3PKD) protocol and SIGMA protocol to accommodate our needs. In the rest of this section, we will discuss the design choices and considerations.

Investigation of Three Party Key Exchange Protocols

There are several existing three party key exchange protocols. Particularly, we examine two representatives: Choo's 3PKD [39] protocol and Needham-Schroeder protocol [99]. One of the major differences between these two protocol is that Needham-Schroeder protocol is good for establishing one-time use shared key (e.g., session key), while Choo's 3PKD protocol can generate a long-lived shared key. In the Needham-Schroeder protocol, the server, which serve as the key distribution center (KDC), generates the secrete key and sends it to the one of the two peer parties directly. Then the party, who receives the key from the server, shares the key with the other peer party with proper challenge-response verification. With Needham-Schroeder protocol, all three protocol participants know the shared key which will be used between the two peer parties. However, the server does not necessarily need to know this shared key. This fact increases the risk of secret leakage and requires the generated key to be used and discarded as soon as possible because once the server is comprised, the shared key generated with the assistance of the server will be subject to leakage.

Choo's 3PKD, the sever provides a "session key" to the two peer parties directly at the same time, and thus this "session key" may be used as a seed, with which the two peer parties can compute a shared key by themselves without being known by the sever. This strategy follows the least privilege principle and is preferred for generating a long-term key. Even if the server is comprised later, the shared key derived from the seed will not be comprised.

Another problem with Needham-Schroeder protocol is its vulnerability regarding to a replay attack [46], which is fixed in the Kerberos protocol [121] by the adding a timestamp. Kerberos is another widely used three-party key distribution protocol based on Needham-Schroeder protocol. However, Kerberos is not suitable for our purposes due to the following reasons: (1) it suffers from the single point of failure; (2) the centralized KDC model is not flexible and scalable enough to support heterogeneous Parent devices at a global scale. Different edge IoT devices (Child devices) may have different Parent devices, which have pre-existing trust relationships and can provide verification and/or delegation regarding to the corresponding Child devices. Due to the vast type and quantity of the Child devices, the Parent devices are also heterogeneous and numerous. As a result, extending Kerberos to our context (e.g., a global-wised MobilityFirst-based IoT architecture) is not feasible. (3) Kerberos has a strict time requirement, which means the clocks of the involved devices must be synchronized within configured limits. However, accurate clock synchronization might be too expensive for many low end IoT devices. Hence, in our protocol design, we prefer to use nonces instead of timestamps.

On the other hand, Choo's 3PKD protocol has a solid security proof and no attack against it has been identified yet. Therefore, based upon the analysis above, we choose Choo's 3PKD protocol as the blueprint for the delegation-based key provisioning protocol. However, Choo's 3PKD protocol alone cannot satisfy our requirements and achieve our goal. There is an assumption in Choo's 3PKD protocol that the server is trusted by the two peer parties before the protocol execution. Unfortunately, this is not necessarily always true in our setting. If the Guardian and the Parent have already established a secure communication channel, then they may leverage such an advantage to facilitate the Parent to delegate the Guardian the privilege to establish a key with the Child. Alternatively, if the Guardian does not know the Parent previously and/or there is no secure communication channel between the Guardian and the Parent, these two parties need to build mutual trust on the fly during the on-going protocol session. This requires incorporating an authentication and delegation process between the Guardian and the Parent, which will be discussed in the following section.

SIGMA Protocol

The Diffie-Hellman (DH) key exchange protocol [47, 96] is one of the most fundamental two-party key exchange protocols. Thought itself is not an authenticated key agreement protocol, it is the basis for many popular authenticated protocols. One serious vulnerability of Diffie-Hellman key exchange is the man-in-the-middle-attack [24]. To address the weaknesses of the Diffie-Hellman protocol, a variety of variant two-party key exchange protocols are proposed, including STS protocol [48], Photuris protocol [11], ISO protocol [20] and SIGMA [83] protocol.

Unfortunately, many of those variants have flaws that were identified later on. To the best of our knowledge, SIGMA is still a secure two-party authenticated key exchange protocol that has no effective attack against it. SIGMA takes advantage of the public key infrastructure and adopts *“SIGn-and-MAc”* approach to authenticated Diffie-Hellman with digital signatures. It serves as the cryptographic basis for the Internet Key Exchange (IKE) standard (version 1 and version 2) [65, 78, 83]. In our key provisioning protocol, SIGMA is chosen as a building block and is overlaid with 3PKD protocol to enable the Guardian and the Parent to authenticate each other on demand. This choice also brings other benefits, such as forward secrecy.

5.6.6 Protocol Specifications

As illustrated in Figure 5.8, there are three participants in the protocol: Child, Guardian and Parent. The Child and the Parent share a symmetric key kek representing their pre-established trust relationship, but they can not communicate directly. The Guardian has direct communication channels with both the Child and the Parent. The output of the delegation-based key provisioning protocol is a symmetric key sk shared only between Child and Guardian.

Figure 5.9 presents the format and the flow of nine messages in the “full fledge” delegation-based key provisioning protocol. Table 5.1 lists the notations used in Figure 5.9. For simplicity, the Child, the Guardian and the Parent are denoted by C, G and P respectively in the following discussion. Below we present the delegation-based key provisioning protocol specifications:

1. $G \rightarrow C : \{N_1, ID_G\}$

   G sends Message 1 to C to initiate a protocol session. The goal is to establish a trust relationship and a corresponding symmetric key to represent such trust. To construct Message 1, G first generates a random nonce N_1 and sends it together with its own identifier ID_G to C in plaintext.

2. $C \rightarrow G : \{N_1, ID_G, N_2, ID_C, ID_P\}$

   Currently, C has no knowledge regrading G, but sends back Message 2 to direct G to P for further authentication and delegation. Message 2 contains identity information for all three parties, and initial random nonces from C and G for a protocol session.

3. $G \rightarrow P : \{N_1, ID_G, N_2, ID_C, ID_P\}$

   Upon receiving Message 2 from C, G first verifies N_1 and ID_G are from Message 1 as well as no duplicate identifiers, i.e., $ID_C \neq ID_G$ and $ID_P \neq ID_G$ and $ID_C \neq ID_P$. If all the verifications succeed, G then forwards Message 2 as Message 3 to C’s parent party P;

4. $P \rightarrow G : \{N_1, ID_G, N_2, ID_C, ID_P, N_3, g^x\}$

   Upon receiving Message 3 from G, P should validate the following:

   (a) All the identity information are correct.

   (b) G can be authenticated.

   (c) C is indeed its child entity.

   (d) Requested privilege on C can be authorized to G.

(e) Authorization decision could be based on G's identity and an additional data structure for the authorization. Understand that in different embodiments, various information may be included in the authorization data structure. Such data structure may be sent and verified by P and G in protocol exchange with Messages 4 - 6.

Then P generates its own nonce N_3, Diffie-Hellman private component x and public value g^x to construct Message 4, which corresponds to the first message of SIGMA protocol. Message 4 includes all three nonces so as to enhance three-party context. This works as a challenge to G to reduce the chance of a denial of service (DoS) attack on P.

5. $G \rightarrow P : \{N_1, N_2, N_3, g^y, sig_G(g^x || g^y), [ID_C, cert_G]_{ak}\}$

   Now G and P engage in a mutual authentication from Message 4 to Message 6. G first verifies that N_1, ID_G, ID_P, N_2 and ID_C are from Message 3 as well as g^x is an element of the Diffie-Hellman group. Then G sends Message 5, which corresponds to the second message of the SIGMA protocol, with the enhanced protocol context (e.g., matching identifiers and all three nonces) and its own Diffie-Hellman public value g^y. G signs Diffie-Hellman public values with its public key certificate $cert_G$, and then computes a Message Authentication Code (MAC) on G's own identity, further bound with C's identity using the derived key from both parties Diffie-Hellman values. Note that the key ak used to compute the MAC is derived by Equation 5.1 and Equation 5.2.

$$dk = prf(hash(N_1 || N_2 || N_3), g^{xy}) \tag{5.1}$$

$$ak = prf(dk, 1) \tag{5.2}$$

6. $P \rightarrow G : \{\alpha, \beta, sig_P(g^y || g^x), [cert_P]_{ak}\}$

   P parses and validates the following in Message 5:

   (a) Verify N_1, N_2, N_3, $cert_G$ and ID_C ($ID_C \neq ID_G$) are from Message 4.

   (b) Verify $cert_G$ and G is authorized to receive a key for C.

   (c) User the public key contained in $cert_G$ to verify the signature $sig_G(g^x || g^y)$.

   (d) Verify g^y is an element of the Diffie-Hellman group.

(e) Use ak derived as in Equation 5.1 and Equation 5.2 to verify the MAC tag on $[ID_C, cert_G]_{ak}$.

Then P sends back Message 6, which corresponds to the third message of the SIGMA protocol, with matching protocol context, signing Diffie-Hellman public values in reverse order (to protect against reflection attacks) with its own public key certificate $cert_G$, and computes the MAC on P's own identity, using the same derived key ak. Note that Message 6 introduces additional data structures as part of the 3PKD protocol, namely two key provisioning tokens, upon making delegation decision to G:

(a) P as the key provisioning server, generates a random key material k, which will be used later by G and C to compute shared secret key sk, to be delivered to both G and C.

(b) P generates token α for G using Equation 5.4, which includes authenticated data for this protocol instance (all identifiers and nonces) and encrypts the key material k using ek, which is derived by Equation 5.3.

(c) P generates an associated token β for C using Equation 5.5, which includes the same authenticated data for this protocol instance (all identifiers and nonces) and encrypts the key material k using kek, which is the existing parent key that P has established with C, prior to this protocol execution.

(d) As the key material k is securely encapsulated into α and β, P sends both tokens to G in Message 6, in addition to SIGMA authentication data structures.

$$ek = prf(dk, 2) \tag{5.3}$$

$$\alpha = [N_1||ID_G||N_2||ID_P||ID_C||N_3||\{k\}]_{ek} \tag{5.4}$$

$$\beta = [N_1||ID_G||N_2||ID_P||ID_C||N_3||\{k\}]_{kek} \tag{5.5}$$

7. $G \rightarrow C : \{\beta\}$

Upon receiving Message 6, G first validates as the following:

(a) Verify N_1, ID_G, N_2, ID_P, ID_C and N_3 are from Message 5.

(b) Parse $[cert_P]_{ak}$ as $cert_P$ and tag'.

(c) User ak to verify tag'.

(d) Verify $cert_P$.

(e) Verify signature $sig_P(g^y||g^x)$.

If all the validations succeed, at this point, G and P have completed mutual authentication and established a fresh secret symmetric key ek derived from the Diffie-Hellman handshake and the current session information as in Equation 5.3. Then G uses ek to decrypt token α and extract the secret k, which implies G receives an authorization from P. Then G directly extracts β from Message 6, and forwards it to C as Message 7 to further confirm with C whether they can reach the agreement that P has granted authorization to this delegation bounding to the same session.

8. $C \rightarrow G : \{[sid, ID_C]_{ck}\}$

   C validates the token β with matching session information (namely, N_1, ID_G, N_2, ID_P and ID_C are from Message 2), then recovers the same key material k with kek. To construct Message 8, C computes session ID sid as in Equation 5.6 and uses key ck derived by Equation 5.7 to authenticate the message. As an option, G and C could further engage in an additional exchange locally to establish a pair secret ps (ps could be $null$). This pair secret may be used as a contribution to the session information and further derivation, so that P has no knowledge of the established key between G and C;

$$sid = hash(N_1||ID_G||N_2||ID_P||ID_C||N_3||ps) \tag{5.6}$$

$$ck = prf(k, 1||sid) \tag{5.7}$$

9. $G \rightarrow C : \{[sid, ID_G]_{ck}\}$

   G parses Message 8 as sid, ID_C and MAC tag. G first verifies ID_C is from Message 2 and then verifies sid derived by Equation 5.6. G computes ck with Equation 5.7 and uses it to verify the tag. At last, G constructs Message 9 and sends it to C as an acknowledgement. Locally, G computes key sk with Equation 5.8 as the final outcome of the protocol.

$$sk = prf(k, 2||sid) \tag{5.8}$$

10. C: validate Message 9 and compute sk

    C parses Message 9 as sid, ID_G and MAC tag. C verifies ID_G is from Message 1, validates sid and then use ck to verify tag. C also computes the shared secret key sk with Equation 5.8.

    Message 8 and Message 9 are created by C and G, respectively and validated by G and C, respectively, to complete the 3PKD protocol to confirm the matching session information, and consistency in the provided key material from P. Therefore, all three parties reach an agreement that the delegation has been properly granted and confirmed by C, P and G, and derive sk as the final outcome of the protocol.

At this point, the protocol concludes with the successful establishment of a secret key between G and C to represent their trust relationship delegated by P. Stated another way, at completion of the protocol, all three parties confirm they reached a three-party agreement on the delegation. This is typically referred to as a matching conversation. For the agreement on the delegation, the result of the protocol is that:

(a) C agrees that P and G are corresponding parties in the delegation as P and G, as well as P and G know that C agrees;

(b) P agrees that G can be authorized to receive the delegation on C, and both C and G know that P agrees;

(c) G agrees that P has successfully authorized the delegation to C, and C is indeed P's child, and C and P know that G agrees.

Symbol	**Notation**
rng	Random Number Generator
$\|\|$	Concatenation
N_i	i^{th} nonce generated in the protocol session
ID_X	Identifier of the protocol participant X
g	Generator of Diffie-Hellman group
x	Parent's Diffie-Hellman private component
y	Guardian's Diffie-Hellman private component
$prf(k, s)$	Pseudo random function with k as the key and s as the seed
$hash(M)$	One way hash function on material M
$sig_X(M)$	Digital signature on material M using the private key of X
$[M]_k$	Message Authentication Code of M using key k
$[A\|\|\{B\}]_k$	Authenticated encryption: B is encrypted and $A\|\|B$ is authenticated using key k
$cert_X$	Certificate of the protocol participant X
α	Token generated by Parent and granted to Guardian
β	Token generated by Parent and granted to Child
kek	Parent key shared between Parent and Child
k	Random key generated by Parent for Guardian and Child
dk	Intermediate component derived from Diffie-Hellman
ak	Symmetric key derived from dk and used for computing MAC
ek	Symmetric key derived from dk and used for protecting token α
sid	Session identifier
ck	Symmetric key derived from sid and k, and shared between Guardian and Child
sk	Membership key shared between Guardian and Child
ps	Local secret shared only between Guardian and Child (could be $null$)

Table 5.1: Protocol Notations

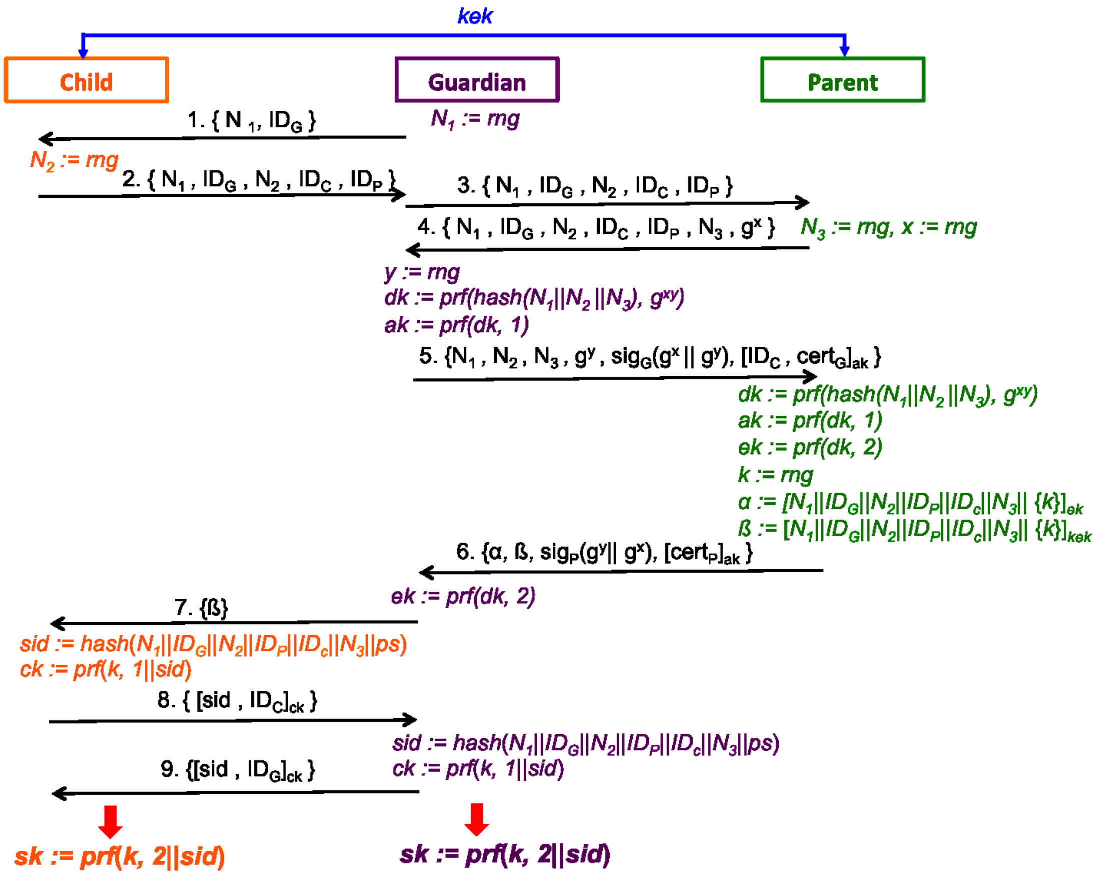

kek
Child
Guardian
Parent
1. { N_1, ID_G }
N_1 := rng
N_2 := rng
2. { N_1 , ID_G , N_2 , ID_C , ID_P }
3. { N_1 , ID_G , N_2 , ID_C , ID_P }
4. { N_1 , ID_G , N_2 , ID_C , ID_P , N_3 , g^x }
N_3 := rng, x := rng
y := rng
dk := prf(hash(N_1||N_2||N_3), g^{xy})
ak := prf(dk, 1)
5. {N_1 , N_2 , N_3 , g^y , sig_G(g^x || g^y), $[ID_C , cert_G]_{ak}$ }
dk := prf(hash(N_1||N_2||N_3), g^{xy})
ak := prf(dk, 1)
ek := prf(dk, 2)
k := rng
α := [N_1||ID_G||N_2||ID_P||ID_c||N_3|| $\{k\}]_{ek}$
ß := [N_1||ID_G||N_2||ID_P||ID_c||N_3|| $\{k\}]_{kek}$
6. {α, ß, sig_P(g^y|| g^x), $[cert_P]_{ak}$ }
ek := prf(dk, 2)
7. {ß}
sid := hash(N_1||ID_G||N_2||ID_P||ID_c||N_3||ps)
ck := prf(k, 1||sid)
8. { $[sid , ID_C]_{ck}$ }
sid := hash(N_1||ID_G||N_2||ID_P||ID_c||N_3||ps)
ck := prf(k, 1||sid)
9. {$[sid , ID_G]_{ck}$ }
sk := prf(k, 2||sid)
sk := prf(k, 2||sid)

Figure 5.9: Message flow of key provisioning protocol.

5.6.7 Discussions and Security Analysis

The delegation-based key provisioning protocol specified in section 5.6.6 involves three parties and contains nine messages. However, in the context of the MobilityFirst-based IoT architecture, there is another round of message exchange before the execution of this key provisioning protocol. When grouping a set of IoT devices to form an IoT network or when a device joining an existing IoT network, the IoT-NRS initiates the device discovery process by sending out registration beacons periodically. An IoT device responds to the beacon with its intention to join the network. After the beacon and the response message exchange, to register the new device and establish a long-term membership credential, the IoT-NRS initiates a new session of our key provisioning protocol by sending Message 1 of the protocol with its identifier and a fresh nonce. We do not explicitly include the first round of message exchange into the key provisioning protocol specification because: (1) the beacon and the response message exchange is sent over an open channel and is irrelevant to the security operations; and (2) the context of MobilityFirst IoT architecture does not limit the applicability of our key provisioning protocol. This protocol is general and can be applied to other scenarios, which will be discussed in section 5.6.9.

Both Choo's 3PKD protocol and SIGMA protocol are proved to be secure *individually*. However, combining them together and customizing them to be applicable for resource limited devices does not necessarily imply *secure*. One of our major contributions includes providing security guarantee for overlaying Choo's 3PKD protocol and SIGMA protocol as well as adopting lightweight measures to reduce overhead. Below, we will discuss our contributions in detail.

Protocol Security Analysis

The key provisioning protocol provides strong security on provisioned keys by leveraging two different protocols, namely a three-party key distribution protocol (3PKD) and a two-party SIGMA protocol, for parent-guardian authentication. These two protocols are overlaid with *enhanced protocol session context (all identifiers and nonces)* so that all the protocol participants have consistent and correct view of the protocol session. Upon the successful completion of the key provisioning protocol, each party in the protocol needs to confirm its commitment to the fact that they agree on, and contribute (e.g. contribute a nonce) to the shared secret(e.g., a shared key or a seed used to generate a shared key).

Introducing enhanced protocol session context into the protocol design excludes the possibility of an identity misbinding attack, where an attacker successfully convinces the peers that the key exchange succeeded, yet the key agreement is bound by each of the participants to a wrong "peer" [83]. One of the measures adopted in the key provisioning protocol to prevent identity-related attacks, e.g., a misbinding attack, is adding identifiers in the corresponding messages when necessary. We assume the identifiers of Child C, Guardian G and Parent P are ID_C, ID_G and ID_P respectively. In the context of the MobilityFirst-IoT, there are two types of identifiers : the globally-used GUID and the locally-used membership credential. Thus, ID_G and ID_P should be the GUIDs of G and P respectively. However, as C is typically a low-end IoT edge device, it does not necessarily have a GUID. And before the completion of the key provisioning protocol, as C has not successfully finished the device registration yet, C does not hold a local membership credential of G's network. In fact, ID_C should be a "local" identifier recognizable by P and used to facilitate the establishment of a long-term membership key shared with G. To summarize, ID_C is neither a GUID nor a local membership key. Instead, ID_C is only a local identifier that uniquely identifies C at P. For example, assuming P is a device manufacturer and C is a product device from P, then ID_C could be the model and serial number that is stored in P's database. In other application scenarios of the key provisioning protocol, ID_G and ID_P are not necessarily GUIDs, which is specific to the design of the MobilityFirst network. Alternatively, ID_G and ID_P could be any other forms of identifier that allow G and P to identify and further authenticate each other.

The first stage of our protocol contains the first three messages and corresponds to the first stage of Choo's 3PKD protocol, in which each of the two peer parties contributes a nonce to the new protocol session. Also, as the "server" P is unknown to G at this point, C explicitly specifies its Parent P by providing ID_P. This design provides flexibility, which many other existing server-based key distribution protocols, such as Kerberos, are lack of. In Message 2, the identity of the Parent is specified by the Child, and hence the Guardian knows who to contact for delegation from the Child. In practice, as a Child may have several pre-existing trust relationships that can be leveraged, the Child has the freedom to direct the Guardian to any of the Parent sharing an established trust. In case one Parent is unreachable by the Guardian or the mutual trust establishment between the Guardian and the Parent fails, the Guardian may inform the Child of the delegation failure. Then the Child can in turn provide another Parent as the new referee.

In the second stage (Message 4 - 6), which corresponds to the SIGMA protocol, a Parent-Guardian authentication is completed using Diffie-Hellman exchange. This authentication process, facilitated by a PKI, is essential to exclude the possible collusion of the Child and the Parent. As this authentication request is initiated by G in Message 3, P sends its Diffie-Hellman value, which can be pre-generated, as a challenge to G to reduce the chance of a denial of service (DoS) attack on P as the Diffie-Hellman algorithm requires a lot of computing resource and G has to pay the price first.

The last stage of our protocol (Message 7 - 9) corresponds to the second stage of the 3PKD protocol. In this stage, the key material k provided by P is delivered to C, and then G and C reach a key agreement based upon k. The protocol is concluded by a handshake between G and C so that both of them confirm that the other party has computed the correct shared secret with respect to the correct protocol session. In our embodiment, we introduced the session context in terms of *session* identifier sid, which is derived from the complete session information (all identifiers, nonces and optional local secret between G and C), and use it as a glue to combine SIGMA protocol between the first stage and the second stage of 3PKD protocol. With the approach of repeating the session information through the protocol, each of the parties has a fresh and consistent view of the current protocol session, which significantly increases the difficulty of launching attacks targeting the context of a protocol instance, such as replay attacks and misbinding attacks.

The key provisioning protocol may further realize stronger security on provisioned keys as compared with existing pairing protocols. In contrast to pairing protocols that do not have security proof (which cause them to be troublesome due to possible man-in-the-middle attacks and frequent human errors in the manual operations), the protocol as described herein may be performed with no manual intervention from the user for key provisioning. Compared with traditional key provisioning protocols that do not rely on server or key infrastructure, such as secure pairing, there is no manual operation by user on an out-of-band (OOB) channel. Instead, the device Parent party is leveraged as the trusted third party to facilitate trust establishment. Such design reduces human error, hence increasing usability. And it removes the requirement of close proximity for key provisioning, as required by most OOB channels involving manual operations by human, and hence increases the chance of practical deployment.

Lightweight Approach

This key provisioning protocol is carefully designed for IoT networks as a light key solution. As one of the most important features, the protocol design favours the resource-limited Child device C in all possible means. The strategy is to minimize the number of messages C needs to process and shift the work load to those more capable devices, namely G and P, as much as possible. Consequently, the key provisioning complexity is transferred to the edge device's Parent party and the Guardian device, making it possible for minimally complex IoT edge devices to operate with secure interactions.

Compared with public key certificates, our protocol implements only lightweight symmetric key operations and secure storage for at least a single symmetric key and protected key operations with the stored key. Specifically, all of the security operations on the Child are symmetric cryptographic computations, which makes the protocol feasible for low-end IoT edge devices.

Timestamps and *nonces* are two typical security measures to guarantee the freshness of a protocol instance and to prevent replay attacks. Timestamps requires strict time synchronization, and consequently increases the hardware cost as an oscillator with high accuracy is not cheap. On the other hand, nonces require a (pseudo) random generator and some form of database to retrieve if a nonce appeared before. As a comparison, nonces are relatively cheaper, and hence we chose to use nonces to protect against reply attack. Unlike a Kerberos system, the Child device is not required to have secure clock synchronization.

Furthermore, the protocol can be implemented with a minimal number of protocol messages handled by the child device. The protocol can thus minimize the computation and power costs to a Child device, such that lightweight devices like sensors, actuators, and microcontroller units (MCUs) can be provisioned with secret keys securely. In addition, the first four messages are sent as plaintext in open channels to reduce overhead. Modifying and/or replaying the messages in the first stage will result in protocol abortion later as each piece of session information will be verified in the following stage.

5.6.8 Proof of Concept

This section presents the prototype of the delegation-based key provisioning protocol, including the framework design, implementations choices and considerations as well as an example output of the demo version.

Modularized Prototype Framework

Figure 5.10 shows the modularized prototype framework design. Each party's protocol package includes three major components: *Instance Manager*, *Protocol Manager* and *Utility Functions*.

Each party should be able to work independently to perform tasks, such as receive messages, parse messages, perform designated computations, construct messages and send messages. In the protocol system model, there are two client/server based communication channels. For the channel between the Guardian and the Child, the Guardian serves as the “server”, while the Child is the “client”. On the other channel between the Parent and the Guardian, the Parent serves as the “sever” while the Guardian is the “client”.

Modularized design allows easy adjustment if the protocol design changes or the parameters change. The instance manager creates an instance of the corresponding party at the device, and then performs configurations, e.g., setting up a port number.

The protocol manager manages all of the sessions of the protocol instance on the device. The protocol manager consists of four modules: *Session Manager*, *Finite State Machine*, *Networking* and *Message*. As one party may open more than one protocol session related to different peer parties, the *Session Manager* is introduced to coordinate different sessions and handle related operations, including initiating a new protocol session, terminating an active protocol session, update and query local session database. For example, the Guardian as a local group authority may receive several registration requests from different edge IoT devices so that it needs to open multiple protocol sessions (one for each requesting IoT device), which requires a session manager to coordinate. This module is also necessary for even edge IoT devices as one edge device as a Child might join two or more different IoT networks at the same time. The *Networking* module is responsible for receiving and transmitting encoded messages through the network interface. The *Message* module parses and constructs messages as well as performs related cryptographic computations. The *Finite State Machine* (FSM) is the core module that handles protocol state transitions. We incorporate an event-driven FSM to take actions upon input events, including a message being constructed and sent successfully, successfully receiving a message, time out, result of parsing and verifying a message, etc.

Utility Functions component provides two categories of utility functions: *Message Encoding/Decoding*, which provide API for the *Message* module to parse and construct messages, and *Crypto Library Wrapper*, which wraps the underlying cryptography library and provides cryptographic computation APIs for the *Message* module.

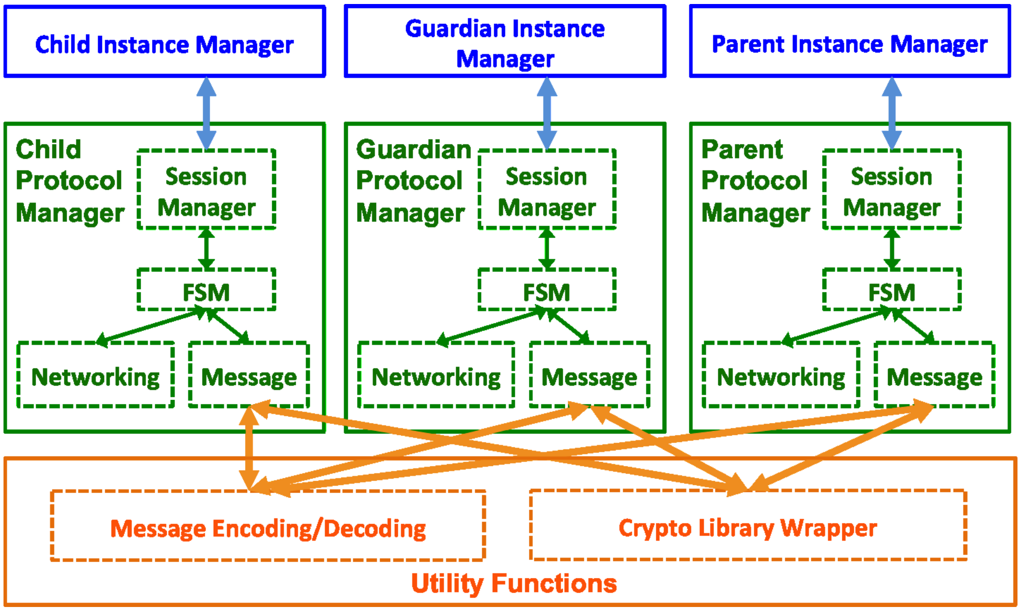


Figure 5.10: Modularized prototype framework design: the framework includes three major modules - instance manager, protocol manager and utility functions - for three protocol participants.

Implementation Choices and Considerations

As for the underlying crypto library, we mainly investigated two lightweight crypto libraries: wolfSSL (previously known as CyaSSL) [13] and mbed SSL (formerly known as PolarSSL) [5]. Both of them are lightweight embedded SSL libraries, providing cryptographic and SSL/TLS/DTLS capabilities for IoT devices with minimal coding overhead. We implemented two suites of crypto library wrappers, one for wolfSSL and one for mbed SSL. At the point of implementing the protocol prototype, to the best of our knowledge, mbed SSL is more stable and easier to use with respect to the cryptographic algorithms used in our protocol. In the final protocol prototype, we adopted mbed SSL as the underlying crypto library and its wrapper as crypto/networking APIs to facilitate the crypto and networking operations.

In the protocol specifications, authenticated encryption is used to protect two tokens carrying the secret key material k as wel as session information. The prototype adopts AES Galois/Counter Mode (GCM) [50] as the authenticated encryption algorithm, which is a block cipher with both encryption and authentication functionalities. With AES-GCM, the key material k is first encrypted and then authenticated together with the session information to generate an authentication tag. As a result, the token α and β consist of three parts: a plaintext of the session information, a ciphertext of the key material and an authentication tag. In our prototype, we set the size of the authentication tag to be 128 bits. Note that mbed SSL does not provide padding support, so if we use AES, we need to pad plaintext to multiples of 16 bytes to meet AES block size requirement. However, AES-GCM has a looser requirement: the input size of the plaintext and the additional authenticated data (AAD) only need to be in bytes (multiples of 8 bits). So we actually do not need to perform padding for AES-GCM as our message are all constructed in bytes.

Another trivial but important issue is the key management. As a device may have multiple keys on it for different purposes and/or shared with different parties, it is crucial to organize the key storage properly and securely so that it is easy to retrieve and restore the keys. In our implementation, for simplicity, the symmetric key is managed by the key identifier ($keyID$): a key is identified by $keyID$ and stored in the key file "keyID.der". To store a symmetric key, we first encoded the key material following Abstract Syntax Notation One (ASN.1), which is a standard and notation that describes rules for representing, encoding, transmitting, and decoding data through the Internet. [4, 77]. Particularly, we adopt the Distinguished Encoding Rules (DER) format defined in ASN.1 to encode all the symmetric keys appeared in the prototype. When retrieving a symmetric key, we read the key directly from the key file with the corresponding $keyID$ and decode the key. While for asymmetric keys, a key is identified and retrieved by the file name. In the future, we may improve the prototype by introducing a well-designed key store to manage all of the keys.

The identifiers are also kept in files. To differentiate various identifier files, which might be used in different scope and context, we encoded the identifier files in the same way as the key files. Generally, identifiers should be at least 32 bits long. If the identifier is generated randomly, it should be at least 128 bits long to be collision resistant.

The session manager module has a responsibility of monitoring and checking the session information. For example, in a new session, when a party receives the fresh session information, namely relevant identifiers and nonces, from an incoming message, it should first validate the freshness and the correctness of the session information. Specifically, it is important to verify whether the nonce has appeared before. Therefore, creating and maintaining a local database

Cryptography Operation	**Algorithm Choice**
hash	SHA-256
pseudo random function (prf)	HMAC-SHA256
DH Groups	2048-bit MODP Group specified in RFC3526 [82]
asymmetric crypto	RSA with key size 2048 bits and exponent 65537
Message Authentication Code (MAC)	HMAC-SHA256
Certificate standard	X.509

Table 5.2: Cryptograhpy Algorithm Choices

to store and retrieve all the past session information is necessary. Currently, we simply use a database file to store all the previous session data. The session manager looks up and updates the database file upon receiving the session data from a fresh protocol session.

Two of the three parties, i.e., the Guardian and the Parent, use certificates, while the Child, as a resource-limited device, does not use asymmetric cryptographic algorithms and certificates. The certificates we used follow the X.509 standard [68]. For simplicity, we assume the Guardian and the Parent share a root certificate authority (CA), otherwise we need to set up two root CAs and make them cross authenticate each other to build a trust anchor. To generate the certificates, we first made a certificate for a self-certified root CA, and then used this certificate to generate and sign two certificates for the Guardian and the Parent respectively.

Serializing and deserialzing is another trivial issue but worth mentioning in practice. As the network traffic uses big endian, but some end hosts use little endian while others use big endian, message serialization and deserialization are necessary to guarantee parsing messages correctly. Therefore, in the implementation, the message to be sent out through the network socket is encoded from a structured message into a big endian bit string for transmission, while at the receiver the received big endian bit string is decoded into a structured message for parsing.

Major implementation choices for the cryptography algorithms is listed in Table 5.2 and parameter settings are listed in Table 5.3.

Cryptographic Parameter	**Settings**
symmetric key size	32 bytes
key identifier size	32 bytes
identifier size	32 bytes
MAC tag size	32 bytes
AES-GCM tag size	16 bytes
SHA-256 digest size	32 bytes
RSA key size	2048 bits
Diffie-Hellman key size	2048 bits

Table 5.3: Cryptograhpic Parameter Settings

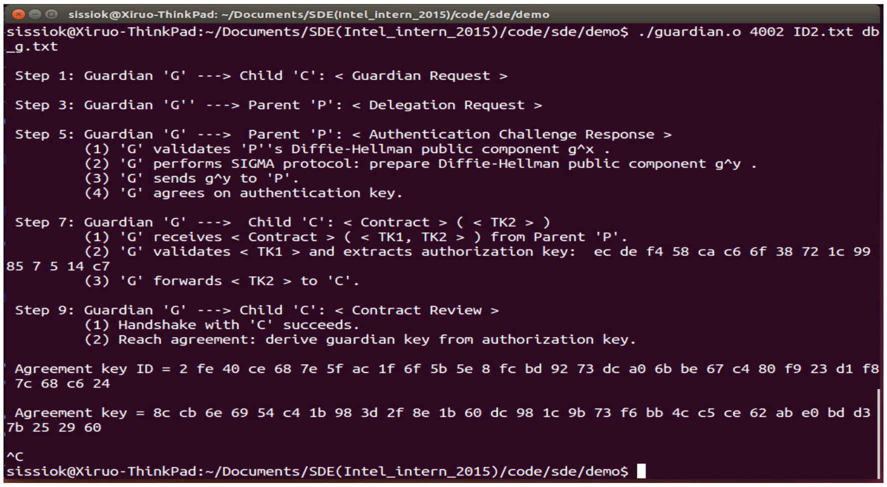
Demo of the Prototype

In this section, we present the demo output (shown in Figure 5.11) at Child, Guardian and Parent respectively. The demo was run on a single Linux machine so that we used different port number to differentiate the different communication channels. In practice, IP addresses and port numbers are usually used instead of purely port numbers.

At the beginning, the Parent P configured itself as a server and opened a listening socket waiting for the Guardian G. G played a dual role and thus configured itself as a server for the Child C and as a client of P. C was configured as a client of G. In the demo, we used the TCP socket provided by the mbed SSL library as the networking socket. In a practical IoT network deployment, this can be replaced by other networking sockets customized for IoT environments, e.g., DTLS [112]. Also, there is an additional Message 0 before the first message of the protocol to initiate the connection from the client (Child) to the server (Guardian).

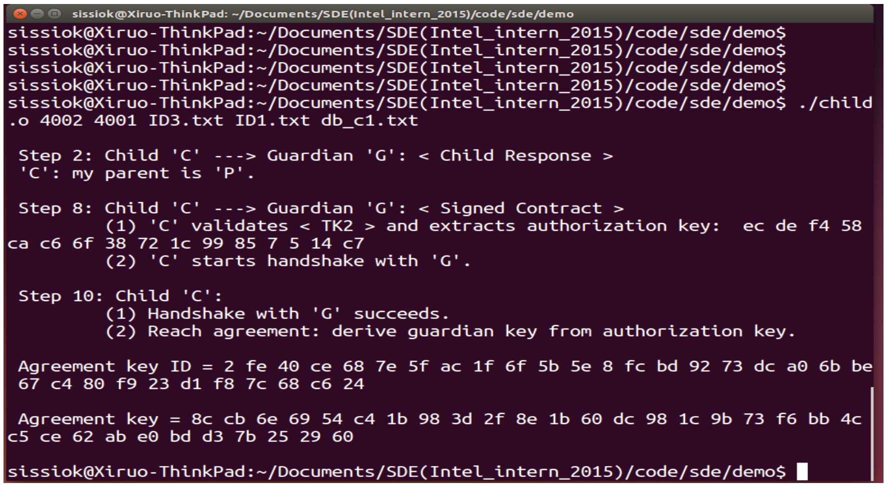
Below we briefly illustrates the message exchange flows shown in Figure 5.11.

- **Step 1**: G initiates the protocol and sends the first message $<$ *Guardian Request* $>$ to C.
- **Step 2**: C replies to G with $<$ *Child Response* $>$ indicating the identity of P.
- **Step 3**: G sends $<$ *Delegation Request* $>$ to P.


sissiok@Xiruo-ThinkPad: ~/Documents/SDE(Intel_intern_2015)/code/sde/demo

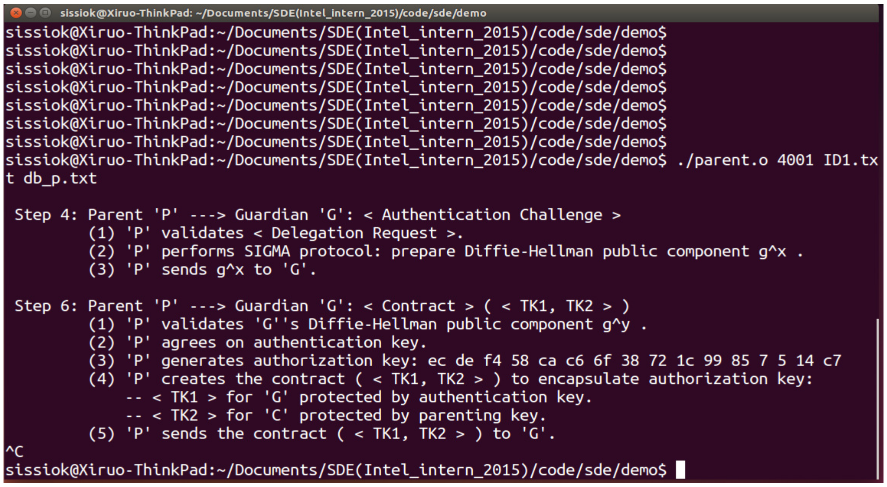
```
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$ ./guardian.o 4002 ID2.txt db
_g.txt

 Step 1: Guardian 'G' ---> Child 'C': < Guardian Request >

 Step 3: Guardian 'G'' ---> Parent 'P': < Delegation Request >

 Step 5: Guardian 'G' --->  Parent 'P': < Authentication Challenge Response >
         (1) 'G' validates 'P''s Diffie-Hellman public component g^x .
         (2) 'G' performs SIGMA protocol: prepare Diffie-Hellman public component g^y .
         (3) 'G' sends g^y to 'P'.
         (4) 'G' agrees on authentication key.

 Step 7: Guardian 'G' --->  Child 'C': < Contract > ( < TK2 > )
         (1) 'G' receives < Contract > ( < TK1, TK2 > ) from Parent 'P'.
         (2) 'G' validates < TK1 > and extracts authorization key:  ec de f4 58 ca c6 6f 38 72 1c 99
85 7 5 14 c7
         (3) 'G' forwards < TK2 > to 'C'.

 Step 9: Guardian 'G' ---> Child 'C': < Contract Review >
         (1) Handshake with 'C' succeeds.
         (2) Reach agreement: derive guardian key from authorization key.

 Agreement key ID = 2 fe 40 ce 68 7e 5f ac 1f 6f 5b 5e 8 fc bd 92 73 dc a0 6b be 67 c4 80 f9 23 d1 f8
 7c 68 c6 24

 Agreement key = 8c cb 6e 69 54 c4 1b 98 3d 2f 8e 1b 60 dc 98 1c 9b 73 f6 bb 4c c5 ce 62 ab e0 bd d3
7b 25 29 60

^C
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$ 
```

(a) Demo printout at the Guardian


sissiok@Xiruo-ThinkPad: ~/Documents/SDE(Intel_intern_2015)/code/sde/demo

```
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$ ./child
.o 4002 4001 ID3.txt ID1.txt db_c1.txt

 Step 2: Child 'C' ---> Guardian 'G': < Child Response >
 'C': my parent is 'P'.

 Step 8: Child 'C' ---> Guardian 'G': < Signed Contract >
         (1) 'C' validates < TK2 > and extracts authorization key:  ec de f4 58
ca c6 6f 38 72 1c 99 85 7 5 14 c7
         (2) 'C' starts handshake with 'G'.

 Step 10: Child 'C':
         (1) Handshake with 'G' succeeds.
         (2) Reach agreement: derive guardian key from authorization key.

 Agreement key ID = 2 fe 40 ce 68 7e 5f ac 1f 6f 5b 5e 8 fc bd 92 73 dc a0 6b be
 67 c4 80 f9 23 d1 f8 7c 68 c6 24

 Agreement key = 8c cb 6e 69 54 c4 1b 98 3d 2f 8e 1b 60 dc 98 1c 9b 73 f6 bb 4c
c5 ce 62 ab e0 bd d3 7b 25 29 60

sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$ 
```

(b) Demo printout at the Child


sissiok@Xiruo-ThinkPad: ~/Documents/SDE(Intel_intern_2015)/code/sde/demo

```
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$ ./parent.o 4001 ID1.tx
t db_p.txt

 Step 4: Parent 'P' ---> Guardian 'G': < Authentication Challenge >
         (1) 'P' validates < Delegation Request >.
         (2) 'P' performs SIGMA protocol: prepare Diffie-Hellman public component g^x .
         (3) 'P' sends g^x to 'G'.

 Step 6: Parent 'P' ---> Guardian 'G': < Contract > ( < TK1, TK2 > )
         (1) 'P' validates 'G''s Diffie-Hellman public component g^y .
         (2) 'P' agrees on authentication key.
         (3) 'P' generates authorization key: ec de f4 58 ca c6 6f 38 72 1c 99 85 7 5 14 c7
         (4) 'P' creates the contract ( < TK1, TK2 > ) to encapsulate authorization key:
             -- < TK1 > for 'G' protected by authentication key.
             -- < TK2 > for 'C' protected by parenting key.
         (5) 'P' sends the contract ( < TK1, TK2 > ) to 'G'.
^C
sissiok@Xiruo-ThinkPad:~/Documents/SDE(Intel_intern_2015)/code/sde/demo$ 
```

(c) Demo printout at the Parent

Figure 5.11: One demo example of the message flow and the session information printout at the Guardian, the Child and the Parent.

- **Step 4**: P performs SIGMA protocol and sends $<$ *Authentication Challenge* $>$ to G.
- **Step 5**: G replies P's challenge with $<$ *Authentication Challenge Response* $>$.
- **Step 6**: P grants two tokens to G in $<$ *Contract* (*TK1, TK2*) $>$. P close the socket.
- **Step 7**: G forwards $<$ *Contract* (*TK2*) $>$ to C.
- **Step 8**: C starts the final handshake to confirm the key agreement in $<$ *Signed Contract* $>$ with G.
- **Step 9**: G finishes the handshake with $<$ *Contract Review* $>$ to confirm the key agreement with C.
- **Step 10**: C confirms the success of the handshake and key agreement.

At the end, we can see the key ID, which is derived from the session data, and the key material are consistent at both the Child and the Guardian.

5.6.9 Use Cases

The delegation-based key provisioning protocol is not only a lightweight membership enrolment protocol specified for the MobilityFirst-based IoT architecture, but can also serve as the cryptographic core of general-purpose key provisioning, where established keys may represent various trust relationships with different scope, semantics and criticality/impact.

Delegation of key provisioning in accordance with our protocol assumptions can be applied to a wide variety of use cases. As an example, the delegation techniques can also be used for a special function delegation. For instance, a symmetric attestation scheme may involve the Parent as an established Verifier to delegate the capability for the Guardian (a new Verifier) to challenge and verify the Prover (Child) using a symmetric attestation key. Embodiments may further support provisioning an additional Verifier (Guardian) for the same Prover. In such cases, a fresh and unique key may be established between the Prover and the new Verifier, with the original Verifier's authorization and delegation. A device provisioning function that establishes a new root of trust key for a device could also use the above protocol.

Another example can be applied to an ownership transfer protocol, which allows the original owner (Parent) to transfer all its privileges on the target device (Child) to the new owner (Guardian). In this case, a new ownership key is established on the Child, and replaces the old ownership key as the root of trust.

To summarize, with respect to key management, different keys have different usage purposes and thus key semantics are essential to clarifying any ambiguity. Our delegation-based key provisioning protocol can work with various key semantics and provide a lightweight approach to establish key material for IoT systems.

5.7 Conclusion

In this chapter, we first investigated existing IoT solutions and analyzed their disadvantages. We focused in depth on security aspects of the various IoT architectures. Starting with general security analysis, we identified security and privacy concerns unique to IoT systems. Then we proposed a unified IoT architecture design based on the MobilityFirst network. With support from the underlying MobilityFirst infrastructure, we present several schemes that work at different layers in our IoT architecture to provide security and privacy so as to address security concerns and increase confidence about the assurable operation of the Internet of Things. Furthermore, to complete the naming framework of MobilityFirst-IoT and facilitate device discovery process, we propose a delegation-based key provisioning protocol as the membership enrolment protocol within the context of local IoT networks. Our efforts extend the Internet to include the real world and therefore form a huge *new inclusive network*. This is a promising direction and leads to many opportunities for further research.

Chapter 6

Conclusion

6.1 Summary of Thesis Contributions

This thesis began by exploring the security challenges associated with maintaining a two-tier name resolution service. One theme that has emerged from MobilityFirst and other future Internet designs is the notion that separating names from network addresses is advantageous. However, such designs still require having components of the network design that resolve names into network addresses, and thus there critical challenges associated with securing such name resolution services remain. Using the global name resolution service (GNRS) of MobilityFirst as a motivating example of such an emerging name resolution service, we thoroughly explored the protocol-level security challenges facing the administration of the GNRS. Specifically, we identified several specific security threats, and presented secure protocols that support the updating of name-to-address associations, and the querying of name resolutions. In particular, our solution also examined the unique challenges associated with IP holes that arise due to BGP churn.

With a secure name resolution service in place, it is possible to then use the name resolution service to provide improved security and privacy to the broader network. We next examined the possibility of integrating access control enforcement into emerging name resolution services by using the GNRS as a motivating example. Specifically, the name resolution service can be modified to enforce access control policies that regulate how users or network services fetch the GNRS mapping between names/identifiers and network addresses. For example, it is possible to describe access control lists so that illegitimate or unauthorized users cannot lookup specific users. Further, such access control policies can be specified by the end-users themselves, allowing intentional data receipt, whereby the end-host may receive data only if the transmission is consistent with its receipt policy. Additionally, we also explored how spatio-temporal access control policies can be specified, thereby supporting secure mobility-oriented services in the future Internet.

We next explored security aspects associated with two emerging network paradigms: mobile ad hoc networks (MANETs) and the Internet of Things (IoT). For MANETs, we explored the use of overlay tunneling as a policy tool that can support the security and reliability of network communications. In particular, for networks like MANETs, the introduction of specific security mechanisms into the actual network protocols is often problematic as ensuring the security properties of the network protocols (such as routing protocols) is often at odds with supporting the basic operations of the protocol and network itself. Consequently, it is often desirable to have a light-weight approach to introducing security, where one employs conventional and well-understood network protocols, and attempt to achieve the assurability of the network through higher-layer, overlay mechanisms. Specifically, in this thesis, we explored the use of overlay tunneling in an OLSR-based MANET as a means to mitigate network failures and attacks directed at the network without involving the redesign of the underlying OLSR protocol. Lastly, we explored the security challenges facing the Internet of Things. We presented a unified IoT architecture based on MobilityFirst that addresses several aspects related to security and privacy of IoT services. We introduced a new layer in the architecture, which we refer to as an IoT middleware, that connects heterogeneous hardware in local IoT systems to the global MobilityFirst network. One of the core components of the IoT middleware is the IoT-NRS (IoT name resolution service), which is a device registration service that provides naming and key management. Further, we developed a delegation-based key provisioning protocol that establishes a long-term membership key between an IoT device and the IoT-NRS, with the assistance of a trusted third party. The protocol is specifically designed to be trustworthy while also being lightweight. Moreover, this protocol can serve general purposes, such as ownership transfer, establish attestation key, etc.

6.2 Future Work

The Internet is extremely large and involves a wide variety of participants and components, from end users to ISPs, from upper layer applications to the underlying physical layer hardware, from various software virtualizations to complex network infrastructure, from the main hub of the Internet to being interconnected with various emerging network paradigms (such as MANETs, vehicle networks, smart grid and IoT). The size and complexity of the future Internet makes the task of securing it extremely difficult. Our efforts in this thesis serve as a starting point and there is still a long way to go for building a sophisticated and secure future Internet.

6.2.1 Integrating Security Measures into the GNRS Prototype

Currently, following the work in [131], which presents a performance evaluation of a GNRS simulation, the GNRS has been implemented by the MobilityFirst team. A next step towards securing MobilityFirst would be to integrate the security solutions proposed in Chapter 2 into the GNRS prototype for a complete performance evaluation. A network infrastructure without proper security protection is untrustworthy, and thus cannot convince end users to adopt such a system. Actually implementing the security measures into the existing GNRS is an essential foundation for prototyping the complete MobilityFirst architecture, and for further evaluating other network components and services. As there is always a trade off between the security and efficiency, besides the GNRS security analysis discussed in this thesis, it is worthwhile to measure and evaluate the overhead that is actually introduced to the GNRS protocol suite by the security measures in the real implementation.

6.2.2 Name Certificate & Resolution Service

Another incomplete work in MobilityFirst is the Name Certificate & Resolution Service (NCRS), which is the other core service in the name-based service layer besides the GNRS. As previously discussed in Section 2.3, the NCRS serves as both a certificate authority, which manages user certificates and provides trust anchor, and a search engine, which allows users lookup GUIDs with human readable names as well as lookup human readable key words with GUIDs. Therefore, the NCRS is indeed the starting point to initiate a communication as the involved operations begin at the NCRS: an end host needs to first register itself at the NCRS, obtain the GUID of the other end host, use the GUID to query the GNRS for the associated network address, with which the communication between two end hosts can be established.

However, as for the working progress, only the functionalities of the NCRS have been defined to date. The service protocol specifications are still under development. Below we outline the challenges that are realized and the specific functionalities the NCRS needs to support:

(a) As a certificate registry:

- The certificate should encapsulate both a list of well-defined human readable key words and the cryptographic identifier.
- A fully fledged NCRS should support three types of operations: *certificate registration*, *certificate renewal* and *certificate revocation*.
- It is desirable to allow both self-certifying and authority assignment approaches for the certificate registration. The protocol should be carefully designed to exclude the opportunities for attacks and exploits.
- Certificate revocation is a challenge due to scalability and there being many participants, including the GNRS and end users. The certificate revocation may impact the NCRS and the GNRS, both of which are logically centralized but physically distributed services, and hence may require a secure synchronization mechanism to coordinate among multiple service participants.

(b) As a search engine:

- It is challenging to define the standards that regulate human readable key words due to the complexity of human languages as well as the subjectivity of human beings.
- The NCRS provides two-directional name translation: from human readable names to the GUID, as well as from the GUID to human readable names. How to improve the accuracy of the searching result for querying a GUID with a *partial* set of human readable key words input is a challenge and requires further examination.

Since the NCRS serves as one of the security foundations for the MobilityFirst network, its design would have a huge impact on the rest of the network and may lead to alteration of the design for other network components. Therefore, it is important to have a sophisticated NCRS in place before moving forward.

6.2.3 Securing MobilityFirst Routing Protocols

In addition, routing is another critical component for a future Internet infrastructure, and hence requires security protection. The intra-domain and inter-domain routing protocols in MobilityFirst are the GSTAR [100] and the EIR (not published yet) respectively. Besides the routing protocols' core functionality of finding "best" routing paths, it is also important to enhance the security and robustness of the routing protocols. Below is a list of desirable characteristics for the inter-domain and the intra-domain routing protocols:

- High availability, even for networks with malicious parties.
- Strong isolation from untrusted parties.
- Enable route control to achieve policies, such as trust.
- Simplicity, efficiency, flexibility and scalability.

Specifically, the GSTAR unifies the techniques from MANETs and DTN (Delay Tolerant Network) protocols, and leverages the concept of *in-network storage* to cope with bandwidth fluctuations and improve the routing decision, which enables *delay-tolerant* modes and allows the packets to be stored at the router and sent out later without being simply dropped. This design classifies the routing path quality based on the short term expected transmission time (SETT) and the long term expected transmission time (LETT), and hence is able to compute the forwarding table to better reflect "real time" network capabilities. While transmitting a chunk of data, if the SETT is larger than a threshold calculated from the LETT, the router will store the chunk; otherwise, it will forward the chunk. Here, the router's available storage is an important factor for the routing table computation as well as the forwarding decision. As storage hardware has become cheap, large volumes of storage at routers is now affordable, and therefore provides more choices to the routers and end host when facing a bottleneck.

However, every coin has two sides. This design has also raised concerns involving storage-based attacks. For example, a compromised router may declare it has "infinite" storage by falsely claiming "Disk Space Remaining" field in Flooded link state advertisement (F-LSA) to be infinitely large and therefore redirecting many traffic flows through it. An adversarial router may also fill buffers of other routers on parallel paths with garbage data to manipulate the routing table computation and to redirect the traffics to itself. How to validate the actual storage on routers and prevent malicious storage manipulation is critical not only for the GSTAR intra-domain routing protocol but also for other protocols that take advantage of in-network storage.

In summary, this thesis has provide an initial exploration into various security aspects facing many future Internet designs, and there are many exciting directions for follow-on investigation that should be explored in order to arrive at a holistic and trustworthy design for a future Internet.

References

[1] eXpressive Internet Architecture. http://www.cs.cmu.edu/∼./xia/.

[2] Firechat. firechat.opengarden.com.

[3] Global Sensor Networks. https://github.com/LSIR/gsn/wiki.

[4] Information Technology ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)), author=International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), month=November, year=2008.

[5] mbed TLS, note = https://tls.mbed.org/, author=ARM.

[6] MelsecNet. https://eu3a.mitsubishielectric.com/fa/en/.

[7] MobilityFirst Future Internet Architecture. http://mobilityfirst.winlab.rutgers.edu/.

[8] Named Data Networking. http://named-data.net/.

[9] Nebula Future Internet Architecture Project. http://nebula-fia.org.

[10] NS-3. https://www.nsnam.org/.

[11] Photuris: Session-Key M]anagement Protocol, author=Karn, Phil and Simpson, William, year=1999.

[12] Smart Distributed System (SDS). http://holjeron.com/products/sds-products/.

[13] wolfSSL, note = https://www.wolfssl.com/wolfSSL/Home.html.

[14] DF-1 protocol and command set reference manual. *IETF standards, CoRE working group*, October 1996.

[15] NSA Suite B Cryptography. IETF Internet Standard, 2005.

[16] Six Technologies with Potential Impacts on US Interests out to 2025. Technical report, Washington: the National Intelligence Council, 2008.

[17] eXtensible Access Control Markup language (XACML) version 3.0, 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

[18] Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020. Gartner Inc., December 2013.

[19] Low Throughput Networks (LTN): Functional Architecture. European Telecommunications Standards Institute(ETSI) GS LTN 002 v1.1.1, September 2014.

[20] I. I. 9798-3. Entity Authentication Mechanisms – Part 3: Entity Authentication using Asymmetric Techniques. 1993.

[21] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317. ACM, 2001.

[22] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for Data Processing in Large-scale Interconnected Sensor Networks. In *Mobile Data Management, 2007 International Conference on*, pages 198–205. IEEE, 2007.

[23] C. Adams and S. Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Professional, 2nd edition, November 2002.

[24] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17. ACM, 2015.

[25] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. *Resilient Overlay Networks*, volume 35. ACM, 2001.

[26] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). In *ACM SIGCOMM*, August 2008.

[27] J. Ariyakhajorn, P. Wannawilai, and C. Sathitwiriyawong. A Comparative Study of Random Waypoint and Gauss-markov Mobility Models in the Performance Evaluation of MANET. pages 894–899, 2006.

[28] K. Ashton. That ‘Internet of Things’ Thing. *RFID Journal*, 22:97–114, 2009.

[29] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A Survey. *Computer networks*, 54(15):2787–2805, 2010.

[30] F. Azzedin and A. Ridha. Feedback Behavior and its Role in Trust Assessment for Peer-to-Peer Systems. *Telecommunication Systems*, 44(3-4):253–266, 2010.

[31] R. Balakrishnan and S. Kambhampati. SourceRank: Relevance and Trust Assessment for Deep Web Sources based on Inter-Source Agreement. In *Proceedings of the 20th international conference on World wide web*, pages 227–236. ACM, 2011.

[32] M. Bansal, R. Rajput, and G. Gupta. Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. *The Internet Society*, 1999.

[33] C. J. Bennett, S. W. Edge, and A. J. Hinchley. Issues in the Interconnection of Datagram Networks, July 1977.

[34] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. ROFL: Routing on Flat Labels. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 363–374, 2006.

[35] R. Canetti and H. Krawczyk. Security Analysis of IKEs Signature-based Key-Exchange Protocol. In *Advances in CryptologyCRYPTO 2002*, pages 143–161. Springer, 2002.

[36] A. Chander, J. C. Mitchell, and D. Dean. A State-Transition Model of Trust Management and Access Control. In *CSFW*, 2001.

[37] R. Chandramouli and S. Rose. Secure Domain Name System (DNS) Deployment Guide. NIST Special Publication 800-81r1, May 2006.

[38] S. Chen, Y. Zhang, and W. Trappe. Inverting Sensor Networks and Actuating the Environment for Spatio-temporal Access Control. In *Proc. 4th ACM SASN*, 2006.

[39] K.-K. R. Choo, C. Boyd, Y. Hitchcock, and G. Maitland. On Session Identifiers in Provably Secure Protocols. In *Security in Communication Networks*, pages 351–366. Springer, 2004.

[40] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2014-2019. *Cisco white paper*, May 2015.

[41] T. Clausen and P. Jacquet. IETF RFC 3626: Optimized link state routing protocol olsr. *The Internet Society http://www. ietf. org/rfc/rfc3626. txt*, 2003.

[42] F. T. Commission et al. Internet of Things Privacy and Security in a Connected World. In *workshop held on November*, volume 19, page 2013, 2013.

[43] S. Corson and J. Macker. Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, RFC 2501. 1999.

[44] S. R. Das, R. Castaneda, J. Yan, and R. Sengupta. Comparative Performance Evaluation of Routing Protocols for Mobile Ad Hoc Networks. In *Proceedings of the 7th International Conference on Computer Communications and Networks*, pages 153–161. IEEE, 1998.

[45] H. Deng, W. Li, and D. P. Agrawal. Routing Security in Wireless Ad Hoc Networks. *Communications Magazine, IEEE*, 40(10):70–75, 2002.

[46] D. E. Denning and G. M. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533–536, 1981.

[47] W. Diffie and M. E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[48] W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992.

[49] R. Droms and T. Lemon. *The DHCP Handbook*. Sams Publishing, 2nd edition, November 2002.

[50] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. In *NIST Special Publication 800-38D*. National Institute of Standards and Technology, 2007.

[51] A. Elmokashfi, A. Kvalbein, and C. Dovrolis. BGP Churn Evolution: a Perspective from the Core. In *Proceedings IEEE INFOCOM*, pages 1–9, 2010.

[52] H. Eriksson. MBone: the Multicast Backbone. *Communications of the ACM*, 8:54–60, 1994.

[53] D. Farinacci, P. Traina, S. Hanks, and T. Li. Generic Routing Encapsulation (GRE). 1994.

[54] R. Fink and R. Hinden. 6bone (IPv6 Testing Address Allocation) Phaseout (RFC 3701). 2004.

[55] L. Garber. Melissa Virus Creates a New Type of Threat. *Computer*, (6):16–19, 1999.

[56] S. Ghazizadeh, O. Ilghami, E. Sirin, and F. Yaman. Security-aware Adaptive Dynamic Source Routing Protocol. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, pages 751–760, 2002.

[57] D. Giusto, A. Lera, G. Morabito, and L. Atzori. *The Internet of Things*. Springer, 2010.

[58] GlobalPlatform. TEE Systems Architecture v1.0, December 2011.

[59] GlobalPlatform. The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, Februrary 2011.

[60] GlobalSecurity.org. Solar Sunrise. May 2009.

[61] C. Gui and P. Mohapatra. Efficient Overlay Multicast for Mobile Ad Hoc Networks. In *Wireless Communications and Networking (WCNC)*, volume 2, pages 1118–1123. IEEE, 2003.

[62] C. Gui and P. Mohapatra. Overlay Multicast for MANETs using Dynamic Virtual Mesh. *Wireless Networks*, 13(1):77–91, 2007.

[63] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, et al. XIA: Efficient Support for Evolvable Internetworking. *Proc. 9th Usenix NSDI*, 2012.

[64] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[65] D. Harkins, D. Carrel, et al. The Internet Key Exchange (IKE). Technical report, RFC 2409, november, 1998.

[66] Y. K. Hassan, M. H. A. El-Aziz, and A. S. A. El-Radi. Performance Evaluation of Mobility Speed over MANET Routing Protocols. *Network Security*, 11(3):128–138, 2010.

[67] M. Hilbert and P. López. The Worlds Technological Capacity to Store, Communicate, and Compute Information. *science*, 332(6025):60–65, 2011.

[68] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation list (CRL) Profile, April 2002.

[69] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad hoc networks*, 1(1):175–192, 2003.

[70] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: a Defense against Wormhole Attacks in Wireless Networks. In *INFOCOM 2003*, volume 3, pages 1976–1986. IEEE.

[71] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole Attacks in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, 2006.

[72] Y.-a. Huang and W. Lee. A Cooperative Intrusion Detection System for Ad Hoc Networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 135–147. ACM, 2003.

[73] Y.-a. Huang and W. Lee. Attack Analysis and Detection for Ad Hoc Routing Protocols. In *Recent advances in intrusion detection*, pages 125–145. Springer, 2004.

[74] J. S. Hunter. The Exponentially Weighted Moving Average. *Journal of Quality Technology*, 18(4):203–210, 1986.

[75] S. Jiang. An Enhanced Prediction-based Link Availability Estimation for MANETs. *Communications, IEEE Transactions on*, 52(2):183–186, 2004.

[76] D. Johnson, Y. Hu, D. Maltz, et al. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. Technical report, RFC 4728, February, 2007.

[77] B. S. Kaliski Jr. A Layman's Guide to a Subset of ASN.1, BER, and DER. 1993.

[78] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. 2005.

[79] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 61–72. ACM, 2002.

[80] M. Khabbazian, H. Mercier, and V. K. Bhargava. Severity Analysis and Countermeasure for the Wormhole Attack in Wireless Ad Hoc Networks. *IEEE Transactions on Wireless Communications*, 8(2):736–745, 2009.

[81] K.-I. Kim and S.-H. Kim. A Novel Overlay Multicast Protocol in Mobile Ad Hoc Networks: Design and Evaluation. *IEEE Transactions on Vehicular Technology*, 54(6):2094–2101, 2005.

[82] T. Kivinen. More Modular Exponential (MODP) Diffie-Hellman Groups for Internet Key Exchange (IKE). 2003.

[83] H. Krawczyk. SIGMA: The SIGn-and-MAc Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In *Advances in Cryptology-CRYPTO 2003*, pages 400–425. Springer, 2003.

[84] S. Kurosawa, H. Nakayama, N. Kato, A. Jamalipour, and Y. Nemoto. Detecting Blackhole Attack on AODV-based Mobile Ad Hoc Networks by Dynamic Learning Method. *Network Security*, 5(3):338–346, 2007.

[85] I. T. Laboratory. FIPS PUB 186-3 DSS. Federal Information Processing Standards Publication, 2009.

[86] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 1981.

[87] J. Li, Y. Zhang, K. Nagaraja, and D. Raychaudhuri. Supporting Efficient Machine-to-machine Communications in the Future Mobile Internet. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 181–185. IEEE, 2012.

[88] Q. Li, Y.-C. Hu, M. Zhao, A. Perrig, J. Walker, and W. Trappe. SEAR: A Secure Efficient Ad Hoc On Demand Routing Protocol for Wireless Networks. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 201–204. ACM, 2008.

[89] S. Li, Y. Zhang, D. Raychaudhuri, and R. Ravindran. A Comparative Study of MobilityFirst and NDN based ICN-IoT Architectures. In *Proceedings of the 10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, pages 158–163. IEEE, 2014.

[90] S. Li, Y. Zhang, D. Raychaudhuri, R. Ravindran, Q. Zheng, L. Dong, and G. Wang. IoT Middleware Architecture over Information-Centric Network. In *2015 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, 2015.

[91] E. Limer. How Heartbleed Works: The Code Behind the Internet's Security Nightmare. April 2014.

[92] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson. Tor instead of IP. In *Proc. 10th ACM HotNets*, 2011.

[93] X. Liu, W. Trappe, and J. Lindqvist. A Policy-driven Approach to Access Control in Future Internet Name Resolution Services. In *Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture*, pages 7–12. ACM, 2014.

[94] X. Liu, W. Trappe, and Y. Zhang. Secure Name Resolution for Identifier-to-Locator Mappings in the Global Internet. In *Proc. 22nd ICCCN*, 2013.

[95] J. Lundberg. Routing Security in Ad Hoc Networks. *Helsinki University of Technology, http://citeseer.nj.nec.com/400961.html*, 2000.

[96] R. C. Merkle. Secure Communications over Insecure Channels. *Communications of the ACM*, 21(4):294–299, 1978.

[97] Microprocessor, M. Committee, et al. IEEE Standard Specifications for Public-Key Cryptography. *IEEE Computer Society*, 2000.

[98] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. IETF Internet Standard, RFC 4423, May 2006.

[99] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[100] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. GSTAR: Generalized Storage-aware Routing for Mobilityfirst in the Future Mobile Internet. In *Proceedings of the sixth international workshop on MobiArch*, pages 19–24. ACM, 2011.

[101] B. C. Neuman and T. Ts'o. Kerberos: an Authentication Service for Computer Networks. *Communications Magazine, IEEE*, 1994.

[102] M. Newman. *BACnet: the Global Standard for Building Automation and Control Networks*. Momentum Press, 2013.

[103] I. Onat and A. Miri. An Intrusion Detection System for Wireless Sensor Networks. In *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'2005)*, volume 3, pages 253–259, 2005.

[104] H. Orman. The Morris Worm: A Fifteen-year Perspective. *IEEE Security & Privacy*, (5):35–43, 2003.

[105] J. Pan, R. Jain, S. Paul, and C. So-In. MILSA: A New Evolutionary Architecture for Scalability, Mobility, and Multihoming in the Future Internet. *Selected Areas in Communications, IEEE Journal on*, 28(8):1344 –1362, october 2010.

[106] J. Pan, S. Paul, and R. Jain. A Survey of the Research on Future Internet Architectures. *Communications Magazine, IEEE*, 49(7):26–36, 2011.

[107] C. Perkins. IP Mobility Support for IPv4, revised. IETF Internet Standard, RFC 5944, 2010.

[108] C. Perkins, E. Belding-Royer, and S. Das. Ad Hoc on Demand Distance Vector (AODV) Routing (RFC 3561). *IETF MANET Working Group*, 2003.

[109] I. Ray and M. Toahchoodee. A Spatio-temporal Role-Based Access Control Model. Springer Berlin Heidelberg, 2007.

[110] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani. MobilityFirst: a Robust and Trustworthy Mobility-centric Architecture for the Future Internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2012.

[111] RealClearPolitics. Moonlight Maze, 1998-1999. February 2013.

[112] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. January 2012.

[113] J. Saltzer. On the Naming and Binding of Network Destinations. IETF Internet Standard, RFC 1498, August 1993.

[114] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 78–87, 2002.

[115] C. Scott, P. Wolfe, and M. Erwin. *Virtual Private Networks*. O'Reilly Media, Inc., 1999.

[116] Z. Shelby. RFC6690: Constrained RESTful Environments (CoRE) Link Format. *IETF standards, CoRE working group*, 2012.

[117] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. RFC7252: The Constrained Application Protocol (CoAP). *IETF standards, CoRE working group*, 2014.

[118] S. Shenker. Fundamental Design Issues for the Future Internet. *Selected Areas in Communications, IEEE Journal on*, 13(7):1176–1188, 1995.

[119] A. C. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *Proc. 6th MobiCom*. ACM, 2000.

[120] W. Stallings. *Cryptography and Network Security: Principles and Practices*. Pearson Prentice Hall, 4th edition, 2005.

[121] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *USENIX Winter*, pages 191–202, 1988.

[122] R. Stone et al. Centertrack: An IP Overlay Network for Tracking DoS Floods. In *USENIX Security Symposium*, volume 21, page 114, 2000.

[123] K. Su, F. Bronzino, K. Ramakrishnan, and D. Raychaudhuri. MFTP: A Clean-Slate Transport Protocol for the Information Centric MobilityFirst Network. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 127–136. ACM, 2015.

[124] Telecommunication Development Bureau and International Telecommunication Union (ITU). ICT Facts and Figures 2005, 2010, 2014. May 2015.

[125] M. Thottan and C. Ji. Anomaly Detection in IP Networks. *Signal Processing, IEEE Transactions on*, 51(8):2191–2204, 2003.

[126] J. Touch. Dynamic Internet Overlay Deployment and Management Using the X-Bone. *Computer Networks*, 36(2):117–135, 2001.

[127] W. Trappe and L. C. Washington. *Introduction to Cryptography with Coding Theory*. Pearson, 2nd edition, July 2005.

[128] J. R. Vacca. *Public Key Infrastructure: Building Trusted Applications and Web Services*. Auerbach Publications, 1st edition, May 2004.

[129] S. Vanstone and C. Corporation. ECC Holds Key to Next-Gen Cryptography, 2004.

[130] J. Viega, M. Messier, and P. Chandra. *Network Security with OpenSSL*. O'Reilly Media, 1st edition, June 2002.

[131] T. Vu, A. Baid, Y. Zhang, T. Nguyen, J. Fukuyama, R. Martin, and D. Raychaudhuri. DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference*, June 2012.

[132] R. Wakikawa, K. Okada, R. Koodli, and A. Nilsson. Design of Vehicle Network: Mobile Gateway for MANET and NEMO Converged Communication. In *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, pages 81–82, 2005.

[133] T. Y. Woo and S. S. Lam. Authentication for Distributed Systems. *Computer*, 1992.

[134] B. Wu, J. Chen, J. Wu, and M. Cardei. A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks. In *Wireless Network Security*, pages 103–135. Springer, 2007.

[135] C. Xiao. Novel Malware Xcodeghost Modifies Xcode, Infects Apple iOS Apps and Hits App Store. Technical report, Palo Alto Netorks, 2015. http://researchcenter. paloaltonetworks. com/2015/09/novel-malware-xcodeghost-modifies-xcode-infects-appleios-apps-and-hits-app-store.

[136] S. Yi, P. Naldurg, and R. Kravets. Security-aware Ad Hoc Routing for Wireless Networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 299–302, 2001.

[137] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. 2006.

[138] F. Zhang, K. Nagaraja, Y. Zhang, and D. Raychaudhuri. Content Delivery in the MobilityFirst future Internet Architecture. In *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pages 1–5. IEEE, 2012.

Appendix A

Acknowledgment of Previous Publications

Chapter 2 revises previous publication [94]:

X. Liu, W. Trappe, and Y. Zhang. Secure Name Resolution for Identifier-to-Locator Mappings in the Global Internet. In Proceedings of International Conference on Computer Communications and Networks (ICCCN), 2013.

Chapter 3 revises a previous publication [93]:

X. Liu, W. Trappe, and J. Lindqvist. A Policy-driven Approach to Access Control in Future Internet Name Resolution Services. In Proceedings of ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch), 2014.

Chapter 4 revises a pending paper:

X. Liu, and W. Trappe. Overlay Tunneling as a Policy Tool for Defending Mobile Ad Hoc Networks.

Chapter 5 revises a pending patent:

M. Zhao, J. Walker, X. Liu, S. Schulz, and J. Zhang. System, Apparatus and Method for Key Provisioning Delegation, 2016.

Appendix B

Refereed Publications as a Ph.D. Candidate

Y. Liu, **X. Liu**, W. Trappe, and R. Roy: Distance-aware Overlay Routing with AODV in Large Scale Ad Hoc Networks. In Proceedings of Vehicular Technology Conference (VTC), 2014.

Appendix C

Funding Acknowledgement

Graduate assistantship support from the National Science Foundation grants CNS-134529 and CNS-1040735 is gratefully acknowledged.