REVISITING SHAMIR’S NO-KEY PROTOCOL: A LIGHTWEIGHT KEY TRANSPORT PROTOCOL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ADNAN KILIÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2017

Approval of the thesis:

REVISITING SHAMIR'S NO-KEY PROTOCOL: A LIGHTWEIGHT KEY TRANSPORT PROTOCOL

submitted by **ADNAN KILIÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Ertan Onur
Supervisor, **Computer Engineering Department, METU**

Assist. Prof. Dr. Cansu Betin Onur
Co-supervisor, **Mathematics Dept., Atılım University**

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU

Assoc. Prof. Dr. Ertan Onur
Computer Engineering Department, METU

Prof. Dr. İbrahim Körpeoğlu
Computer Engineering Department, Bilkent University

Assoc. Prof. Dr. Murat Cenk
Cryptography Department, METU

Assist. Prof. Dr. Pelin Angın
Computer Engineering Department, METU

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ADNAN KILIÇ

Signature :

ABSTRACT

REVISITING SHAMIR'S NO-KEY PROTOCOL: A LIGHTWEIGHT KEY TRANSPORT PROTOCOL

Kılıç, Adnan
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Ertan Onur
Co-Supervisor : Assist. Prof. Dr. Cansu Betin Onur

September 2017, 42 pages

Key-transport protocols, subclasses of key-establishment protocols, are employed to convey secret keys from a principal to another to let them establish a security association. In this thesis, we propose a lightweight, practicable, energy-efficient, and secure key-transport protocol, convenient for wireless sensor networks (WSN), the Internet of things (IoT) and mobile networks. The proposed protocol is based on the Shamir's three-pass (no-key) protocol. Although Shamir's three-pass protocol does not require any pre-shared secret between principals, we show that it is impossible to employ the three-pass protocol over public commutative groups. We modify Diffie-Hellman key-agreement protocol to morph it into a key-transport protocol by applying a set of changes on the original protocol, and it becomes possible to compare both protocols in terms of memory usage and total time to complete a single key transportation. The experimental results point out that the proposed key transport protocol performs faster than the modified Diffie-Hellman protocol, and the total time to transport a single key by using the modified Diffie-Hellman protocol grows drastically with the increase in key size.

Keywords: Key transport protocols, Wireless Sensor Networks, Internet of Things

ÖZ

SHAMİR'İN ANAHTARSIZ PROTOKOLÜNÜN YENİDEN İNCELENMESİ: HAFİF-SIKLET ANAHTAR AKTARIM PROTOKOLÜ

Kılıç, Adnan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ertan Onur
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Cansu Betin Onur

Eylül 2017 , 42 sayfa

Anahtar kurulması protokollerinin alt sınıfı olan anahtar aktarma protokolleri, gizli anahtarları bir varlıktan diğerine iletip bir güvenlik ortaklığı kurmalarına izin vermek için kullanılır. Bu tezde, kablosuz duyarga ağları (WSN), nesnelerin interneti (IoT) ve mobil ağlar için uygun, hafif, pratik, enerji açısından verimli ve güvenli bir anahtar aktarım protokolü öneriyoruz. Önerilen protokol Shamir'in üç-geçişli (anahtarsız) protokolüne dayanıyor. Shamir'in üç-geçişli protokolü varlıklar arasında önceden paylaşılmış bir sır gerektirmese de, umumi değişmeli gruplar üzerinde üç geçişli protokolün kullanılmasının imkansız olduğunu gösteriyoruz. Diffie-Hellman anahtar değişimi protokolünü orijinal protokol üzerinde bir dizi değişiklik uygulayarak bir anahtar aktarım protokolüne dönüştürmek üzere değiştiriyoruz ve her iki protokolü de tek bir anahtar aktarımını tamamlamak için gereken bellek kullanımı ve toplam süre açısından karşılaştırmak mümkün hale geliyor. Deney sonuçları, önerilen anahtar aktarma protokolünün değiştirilmiş Diffie-Hellman protokolünden daha hızlı performans gösterdiğini ve değiştirilen Diffie-Hellman protokolünü kullanarak tek bir anahtarın taşınması için gereken toplam sürenin anahtar boyutundaki artışla birlikte önemli ölçüde arttığına işaret etmektedir.

Anahtar Kelimeler: Anahtar Aktarma Protokolleri, Kablosuz Duyarga Ağları, Nesnelerin İnterneti

To my mom

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Ertan Onur for his guidance, encourage and support. With his friendly attitude, I always feel motivated to continue working. Special thanks go to my co-advisor Cansu Betin Onur. Whenever we feel stuck in theory and did not foresee, she has helped us like an oracle with her expertise in group theory.

I am grateful for my colleagues and friends Hüsnü Yıldız, Özgür Kaya, Merve Aydınlılar, Ömer Ekmekçi, Abdullah Al-Shihabi, Ezgi Ekiz, Çağlar Seylan, Alperen Eroğlu and Selma Süloğlu. Without their friendship, support and encouragement, it would be impossible to complete this thesis.

I would like to express my thankfulness to my close friends Merve Asiler, Koray Uzun, Ali Alp Karabay, Çağrı Tepe, Ulaş Doğru, Eda Küçükkeskin, Eda Ceren Güngör, Merve Gürbüz, Sibel Soylu, Hüseyin Harun Uyan, Halim Gökhan Mert, Gülay Özdemir and Cem Alp Yapalak. If you know these great people, you would not have any reason to worry about anything. I also would like to thank Monika Imeri for her help and vision in creating the visual part of the work.

I would like to thank Fit/IoT-LAB. They provide an amazing opportunity for researchers to test their works.

I would like to express my sincere appreciations to my mom Hatice Kılıç, my sister Fatma Kılıç, my brother Cengiz Kılıç. They are always with me whenever I need their love and support.

Finally, I would like to thank my dad. It has been very difficult to go on being creative after his departure from my life, and now I can only hope that he would feel proud of me.

TABLE OF CONTENTS

CHAPTERS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

Nomenclature

$GF(p), \mathbb{F}_p$	A finite field of order (prime) p
$GL_n(A)$	The set of $n \times n$ matrices whose elements $\in A$
$G = \langle g \rangle$	A cyclic group generated by $g \in G$
$\varphi : G \to H$	A group homomorphism from G to H
$\{0,1\}^n$	The set of n-bit binary strings
$\{0,1\}^*$	The set of all finite binary strings
$\|A\|$	The number of elements in set A
$[a, b]$	The commutator of two elements a and b
Alice	First entity of a communication
Bob	Second entity of a communication
Eve	Eavesdropper
Mallory	Malicious active attacker
Text	Memory space for text segment
Data	Memory space for all initialized data
Bss	Memory space for all uninitialized data

CHAPTER 1

INTRODUCTION

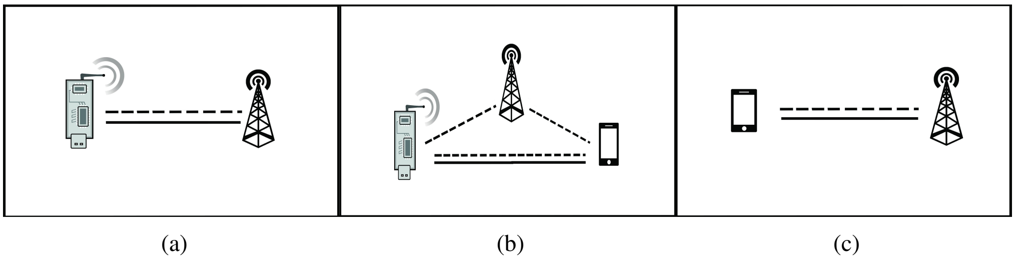
1.1 Scope

Given the recent developments in 5G technology, the Internet of Things (IoT) applications are expected to be integrated with mobile networks to form heterogenous future networks. Therefore, resource-constrained devices are expected to emerge in different platforms and to play an important role in daily life. Due to their restricted hardware capabilities, employing existent security protocols is not feasible. The trade-off between security and efficiency has led to the appearance of lightweight cryptographic protocols to replace symmetric cryptosystems, cryptographic hash functions and key establishment protocols. In this thesis, we concentrate on key transport protocols.

1.2 Problem Definition

According to Paar [31], *key establishment* can be classified into *key transport* and *key agreement* protocols. Key transport protocols are designed to transfer a secret key from an initiating principal that determines the key to another entity in a network, while all of the principals taking part in the protocol influence the key establishment process in key agreement protocols.

Consider the following scenarios in which lightweight security solutions will be critical with the integration of IoT devices given in Fig. 1.1 (1) an IoT device connected to mobile network over narrow-band Long-Term Evolution (LTE) or Long-Range (LoRa), to monitor the status of a critical system positioned outdoor, (2) an IoT device in a healthcare system, checking the vital signs of an elderly person, then reporting them to a smartphone using device-to-device (D2D) communication, (3) smart phones and mobile networks employing LTE or 5G technologies.



(a) (b) (c)

Figure 1.1: Scenarios (a) IoT device connected to mobile network, (b) Device-to-device (D2D) communication between IoT device and smartphone, (c) Smart phone connected to mobile network.

Possible problems with those scenarios are (1) The channel between IoT device and mobile network becomes vulnerable to attacks if the communication is in plaintext. (2) Public-key protocols, so is Diffie-Hellman key agreement protocol, are not suitable [25] for resource-constrained devices due to the computational burden compared to symmetric protocols and communication overhead as a result of frequent and large messages. (3) The vital signs of a person can be exploited with lack of key establishment between IoT device and smartphone. (4) The protocols that employ non-commutative cryptography does not specify the implementation details, including the choice of platform group. (5) Increasing number of IoT devices [2], given in Fig. 1.2, requires flexible, lightweight key-establishment protocols since the existing ones are not lightweight and IoT devices are hardware-constrained.

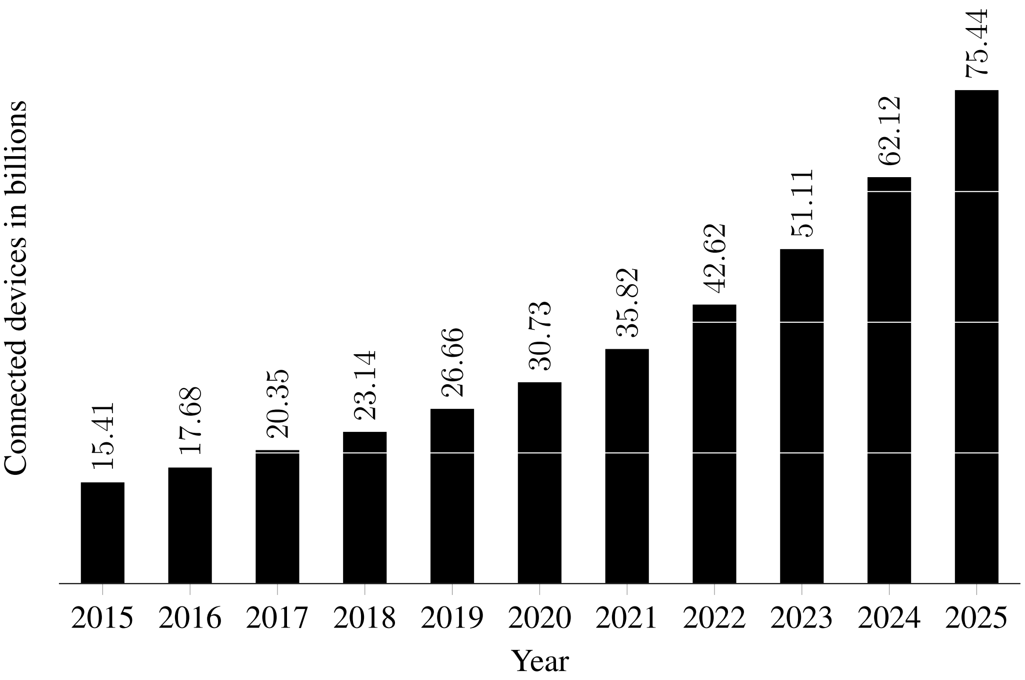


Figure 1.2: The expected number of connected devices installed worldwide from 2015 to 2025. (in billions)

1.3 Contributions

In this thesis, we introduce a lightweight, practical, energy-efficient, secure key-transport protocol suitable for Wireless Sensor Networks (WSN), IoT and mobile networks. The main contributions of the thesis can be listed as follows:

- We introduce the generalized one-time pad model as a symmetric cipher and use this generalized model to implement Shamir's no-key (three-pass) protocol for key transportation. If it were possible to employ public commutative groups to implement the three-pass protocol as we present in Section 3.2, we could use it in post-quantum cryptography for transporting keys providing information-theoretic security without relying on any computationally-difficult problem. As one of the contributions of this work, we prove in Section 3.5 that it is impossible to communicate without sharing secrets using three-pass protocol over public commutative groups. All in all, one has to rely on computational security instead of information-theoretic security approaches to implement the three-pass protocol.

- We propose the lightweight key-transport protocol (LKTP) that is applicable in above-mentioned scenarios relying on computational security for bootstrapping a prime number which will specify the group of the protocol, then information-theoretic security can be provided.

1.4 Outline

In Chapter 2, we will give information about group theory, cryptosystems, non-commutative cryptography and computationally-difficult problems. The related key establishment protocols using group theory related concepts and their overall assessment in comparison to LKTP will conclude the chapter. In Chapter 3, generalized one-time pad and Shamir's no-key protocol using commutative groups by mentioning requirements of public group will be explained. An easy implementation of the three-pass protocol over Klein four-group V will reveal the importance of the requirements. At the end of the chapter, we will show that it is impossible to employ three-pass protocol over public commutative groups. In Chapter 4, the general structure of the proposed lightweight key-transport protocol will be explained. An easy implementation of the protocol and security proof showing the conformance of the protocol will help us understand the idea behind the proposed protocol. We will conclude the chapter with the implementation methodolody, results and discussion of two key-transport protocols.

CHAPTER 2

RELATED WORK

In this chapter, the background information and related works will be explained to understand the structure of the proposed key-transport protocol. In Section 2.1, the definition of *group* and necessary concepts related to abstract algebra will be given. In Section 2.2, the definition of a simple cryptosystem and two types of cryptosystems will be explained to understand the differences between key-establishment protocols. This section also includes the concepts from non-commutative cryptography. In Section 2.3, the underlying computationally-difficult problems will be considered. In Section 2.4, key distribution techniques in WSN will be given. In Section 2.5, related works from non-commutative cryptography and the modified Diffie-Hellman key-transport protocol will be explained in details.

2.1 Group Theory

A *group* [5] [15] is a nonempty set G with binary operation $\cdot$ on G that satisfies the following properties:

G1. The set is closed under the operation : $x \cdot y \in G$ for every $x, y \in G$.

G2. The operation $\cdot$ is associative: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ for every $x, y, z \in G$.

G3. G contains a unique identity element e such that $e \cdot x = x \cdot e = x$ for every $x \in G$.

G4. Every element $x \in G$ has a unique inverse element $y \in G$ such that $x \cdot y = y \cdot x = e$.

Notes:

1. G is commutative (Abelian) if $x \cdot y \ = \ y \cdot x$ for every $x, y \in G$.
2. $|G|$ denotes the number of elements in G.

2.1.1 Kernel of Group Homomorphism

Let $\varphi : G \to H$ be a group homomorphism. The set $Ker(\varphi) = \{x \in G : \varphi(x) = 1_H\}$ is called the **kernel** of φ.

2.1.2 Stabilizer of a Point

Let G be a group acting on a set S and let x be an element of S. The set of elements in G, which fix the value of x is called the **stabilizer** of x in G and is denoted by G_x.

2.1.3 Transitive Action

A group G acting on a set S is said to be **transitive** if for any pair of elements $a, b \in S$, there exists $x \in G$ such that $a \circ x = b$.

2.1.4 Theorem 1.4A-ii in [14]

Suppose that $(G, *)$ is a group acting on a set S and that $r \in G$ and $a, x \in S$. Then, the stabilizer G_a is a subgroup of G and $G_x = r^{-1} * G_a * r$ whenever $x = a \circ r$.

2.2 Cryptography

2.2.1 Cryptosystems

A cryptosystem defined over $(\boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{C}})$ is a set of algorithms $\Pi = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ acting on a plaintext message m where

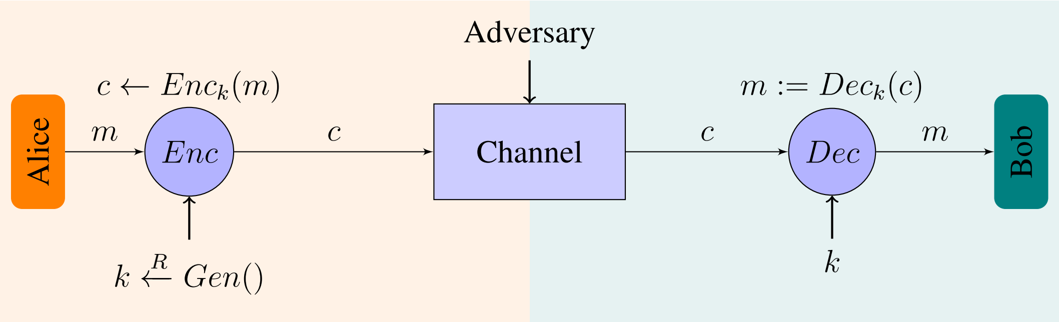


Figure 2.1: A simple model of symmetric encryption scheme.

- $\mathcal{K}$ denotes the set of keys k,
- $\mathcal{M}$ denotes the set of plaintexts (messages) m,
- $\mathcal{C}$ denotes the set of ciphertexts c,
- **Gen** is a key-generation algorithm which outputs k,
- **Enc**: $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ is an encryption algorithm $c \leftarrow Enc_k(m)$,
- **Dec**: $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$ is a decryption algorithm $m := Dec_k(c)$,

Correctness Property: For all messages $m \in \mathcal{M}$ and key $k \in \mathcal{K}$ which is generated by Gen,

$$Dec_k(Enc_k(m)) = m.$$

Symmetric-key and public-key (asymmetric-key) cryptography are types of cryptosystems. In symmetric-key cryptography, all principals share a private key and communicate with each other by encrypting messages. Whenever Alice wants to send a secret message to Bob or Bob tries to decrypt an encrypted message, the same private key will be the input for encryption or decryption, respectively as shown in Fig.2.1.

In public-key cryptography however, each principal has 2 different keys, called public and private keys. When Alice wants to send a secret message to Bob, she should encrypt the plaintext by using the public key of Bob. To decrypt the ciphertext, Bob should use his private key this time as shown in Fig.2.2. The essence of public-key

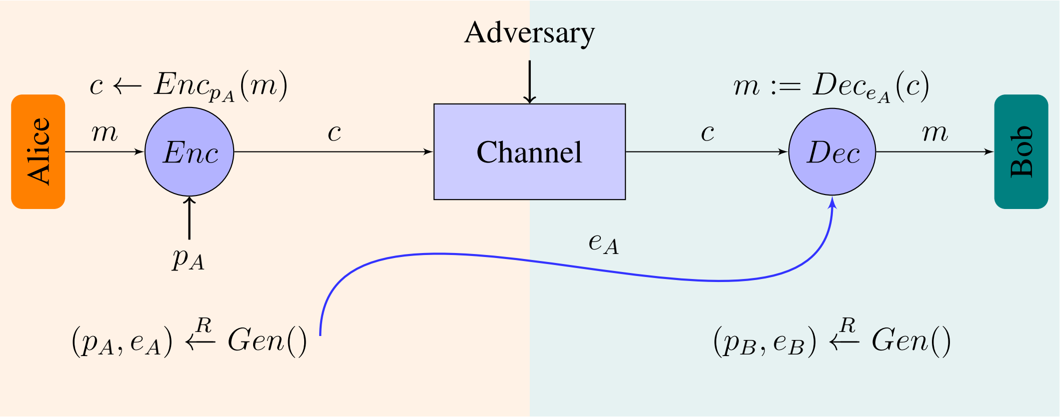

Adversary
$c \leftarrow Enc_{p_A}(m)$
$m := Dec_{e_A}(c)$
Alice
m
Enc
c
Channel
c
Dec
m
Bob
p_A
e_A
$(p_A, e_A) \overset{R}{\leftarrow} Gen()$
$(p_B, e_B) \overset{R}{\leftarrow} Gen()$

Figure 2.2: A simple model of public-key encryption scheme.

cryptography is that the difficulty of a specific mathematical problem makes the encryption be a one-way function, that is, to decrypt a ciphertext you need another key, namely private key, which is very difficult to derive with the existence of the public key, and vice versa. Since all entities should have a public key used in encryption, there is no need to distribute a secret key between entities. However, the public key cryptography is slower compared to symmetric key cryptography and requires more processing power to both encrypt and decrypt a message. [47] [43]

2.2.2 Non-commutative Cryptography

2.2.2.1 Conjugate [28] [9]

Let G be a non-commutative group. The conjugate of g by x for the elements $x, g \in G$ is denoted by $g^x = x^{-1}gx$. The conjugate operation is assumed to replace exponentiation in cryptographic contexts.

2.2.2.2 Commutator [28] [9]

Let G be a group, and $a, b \in G$. The commutator of two elements, a and b, is the element of G, represented by $[a, b] = a^{-1}b^{-1}ab$.

2.2.2.3 Word [28]

Let $S = \{a_1, a_2, a_3, ...\}$ be a set. A finite string of elements which are selected from the set S, in which repetition of elements and inverse elements are allowed is called a *word*. For example, a_1a_2, $a_1a_3^{-1}a_2$, $a_1a_2^{-1}a_3a_1a_1^{-1}a_2a_3^{-1}$ are words in the set S. In other words, a word in S is a string of the form $a_1^{\epsilon_1}a_2^{\epsilon_2}...a_n^{\epsilon_n}$ where $a_i \in S$ and $\epsilon_i \in \{-1, +1\}$. The number n is called the length of the word.

2.3 Computationally-Difficult Problems

2.3.1 Cryptographic Hash Functions [16]

A cryptographic hash function $f : \{0,1\}^* \rightarrow \{0,1\}^n$ is a mapping, from a set of infinite input sequences to the set of $n-$bit hash values, which satisfies the following properties:

- The computation of a hash value for a given input should be deterministic and quick.
- A small change in input value should cause an important change in the hash value to satisfy the avalanche effect.
- f should be *pre-image resistant*: for a known hash value h, it should be infeasible to find a message m such that $f(m) = h$.
- f should be *second pre-image resistant*: for an input message m_1, it should be infeasible to find m_2, such that $f(m_1) = f(m_2)$ where $m_1 \neq m_2$.
- f should be *collision resistant*: it should be infeasible to find two different input messages m_1 and m_2 such that $f(m_1) = f(m_2)$.

Since the mapping f is from a set containing infinitely many input sequences, to a finite set of n-bit hash values, the collision is inevitable according to the Pigeonhole Principle [45]. To strengthen the security of a key-establishment protocol, a secure hash function should be considered. After Google announced the first successful SHA-1 collision, SHA-1 becomes insecure and is expected to be deprecated [38].

2.3.2 Integer Factorization Problem (IFP) [26]

The integer factorization problem is the decomposition of an integer n into prime numbers p_i such that $n = \prod p_i^{a_i}$.

In RSA [35], n is chosen as a semiprime, namely multiplication of two distinct, large prime numbers p and q. The security strength of RSA, hence the integer factorization problem, is based on the belief that there is no efficient, polynomial-time algorithm on a classical (non-quantum) computer [27] when the numbers are large. There are some studies showing that using elliptic curves [41] and estimating $\phi(n)$ [21] give advantage for finding the factorization of a semi-prime [21]; a polynomial algorithm for finding the factorization of n was proposed by Peter Shor [36], but it requires a quantum computer.

2.3.3 Discrete Logarithm Problem (DLP) [20]

Let G be a group, and $g \in G$. Assume that $\langle g \rangle = \{g^k : k \in \mathbb{Z}\}$ be a finite cyclic subgroup of G generated by g. Given an element $y \in \langle g \rangle$, find an element x such that $g^x = y$. The difficulty of the problem depends on the selection of a group, where the logarithm of b to the base g will be calculated.

There are several algorithms to solve the discrete logarithm problem. Baby-step, giant-step algorithm [12] has time complexity and space requirement $\mathcal{O}(\sqrt{|G|})$. To reduce the space requirement, Pollard-ρ [33] provides a method with time complexity $\mathcal{O}(\sqrt{|G|})$ with very small space requirement. The worst case of Pohlig Helman [32] is $\mathcal{O}(\sqrt{|G|})$, however if the order of the group $n = |G| = \prod p_i^{e_i}$, the complexity will become $\sum e_i(log\ n + \sqrt{p_i})$ [37].

2.3.4 Elliptic Curve Discrete Logarithm Problem (ECDLP) [17]

Let E be an elliptic curve over a finite field $\mathbb{F}_q$, and $Q \in E$ be a point of order n. Given $P \in \langle Q \rangle$, find an integer a, $0 \leq a \leq n-1$ such that $P = aQ$.

2.3.5 Conjugacy Search Problem (CSP) [28] [9]

Let G be a non-commutative group. For given two conjugate elements $g, h \in G$, find an element $x \in G$ such that the conjugate of g by x is equal to h. In other words, find an element $x \in G$ such that $x^{-1}gx = h$.

2.3.6 Decomposition Search Problem (DSP) [28] [9]

Let G be a non-commutative group. For given two elements $g, h \in G$, find two elements $a_1, a_2 \in A \subseteq G$ such that $h = a_1 g a_2$, in case the existence of at least one such pair is provided.

2.3.7 Word Search Problem (WSP) [28]

Let G be a non-commutative group and $g \in G$. Find a representation of g such that $g = a_{i_1} a_{i_2} ... a_{i_r}$ where $a_{i_j} \in \{a_1^{\mp 1}, a_2^{\mp 1}, ..., a_n^{\mp 1}\} \subseteq G$.

2.4 Key Distribution Techniques in WSN

Before explaining specific key establishment protocols, it is essential to classify key distribution techniques used in wireless sensor networks: [11]

- **Key Distribution over a Single Network-Wide Key** There is a single network-wide key and it is used to encrypt communication between legitimate nodes.

- **Key Distribution over a Pairwise-Shared Key** Instead of a single network-wide key, each pair has its unique key in a network.

- **Key Distribution by using Public-Key Cryptography** The strength (with computation burden) of public-key cryptography ensures a secure key distribution in a network.

- **Key Distribution by Bootstrapping a Trusted Base Node** The communication between sensor nodes are encrypted by the keys provided by a trusted base node.

2.5 Related Works

There are many key agreement protocols proposed in the literature that rely on computationally-difficult problems (or in general public-key systems). The hallmark is the Diffie-Hellman key exchange protocol that makes use of the difficulty of discrete logarithm problem over a finite group [13]. It is widely employed for exchanging secret keys on the Internet [46]. Additionally, Ko-Lee-Cheon-Han-Kang-Park [22], Anshel-Anshel-Goldfeld [4] and the Stickel key agreement [39] protocols rely on the computationally-difficult problems [28]. For details of the computationally-difficult problems, see Section 2.3).

2.5.1 Modified Diffie-Hellman Key Exchange Protocol

In their papers [13], W. Diffie and M. E. Hellman provided a clear introduction to public key cryptography and introduced their famous key agreement algorithm. To agree on a secret key, Alice and Bob first agree on a prime number p and a base element g that is a primitive root modulo p. Alice selects a private key a to compute $g^a \ (mod\ p)$ and sends it to Bob, where Bob selects a private key b to compute $g^b \ (mod\, p)$ and sends it to Alice. To complete the key exchange protocol, Alice calculates $(g^b)^a \ (mod\, p)$ easily by using her private key, and Bob calculates $(g^a)^b \ (mod\, p)$ with his private key. Now, Alice and Bob agrees on and establishes a secret key which is $k = (g^b)^a \ (mod\ p) \ = \ (g^a)^b \ (mod\ p) \ = \ g^{ab} \ (mod\ p)$. The essence of the security in Diffie-Hellman Key Exchange is without knowing the secret keys a or b, it is computationally difficult to calculate $g^{ab} \ (mod\, p)$ from g^b or g^a which is known as the discrete logarithm problem over a finite set $G = \mathbb{Z}_p^*$. (see Section 2.3.3)

Since we propose a key transport protocol in this thesis, we morphed Diffie-Hellman key exchange protocol to a key transport protocol by applying the following changes on the original protocol to make a fair comparison:

1. A prime number q is assumed to be bootstrapped in both devices.

2. Alice generates a prime number p randomly, and sends $n = p \times q$ to Bob.

3. Bob sets p and the finite cyclic group $G = \mathbb{Z}_p^*$, multiplicative group modulo p, after dividing n by q.

4. Alice selects an integer a, where Bob chooses an integer b such that a, b are in $\{1, ..., p-1\}$ and relatively prime to Euler's totient function [23] $\phi(p) = p - 1$ since p is prime. Then $\bar{a}, \bar{b}$, modulo classes of a and b, are elements of $\mathbb{Z}_{\phi(p)}^*$. Therefore there are integers c, d in $\{1, ..., p-1\}$ such that $\bar{c}$ and $\bar{d}$ are inverses of $\bar{a}$ and $\bar{b}$ in $\mathbb{Z}_{\phi(p)}^*$, respectively. Hence $ac = t_1(p-1)+1$ and $bd = t_2(p-1)+1$ for some integers t_1, t_2. To show the correctness of the protocol, let $g \in G = \mathbb{Z}_p^*$. Then the order of g divides the order of G which is $p - 1$. If we calculate, $g^{ac} = g^{t_1(p-1)+1} = g(g^{p-1})^{t_1} = g$. (This result can also be shown using number theoretic methods. Note that for any integer g, $g^{p-1} \equiv 1 \ (mod\, p)$ by Fermat's Little Theorem [24].)

5. When the key transportation starts, Alice selects a random key $k \in G$ and sends $m_1 = k^a$ to Bob.

6. Bob receives the encrypted message, encrypts it once more by calculating $m_2 = (m_1)^b$ and sends m_2 to Alice.

7. Alice calculates $m_3 = {m_2}^{a^{-1}}$ and sends it to Bob to start the last phase of the protocol.

8. Bob calculates ${m_3}^{b^{-1}} = k^{aba^{-1}b^{-1}} \equiv k \ (mod\, p)$ and recovers the original key k.

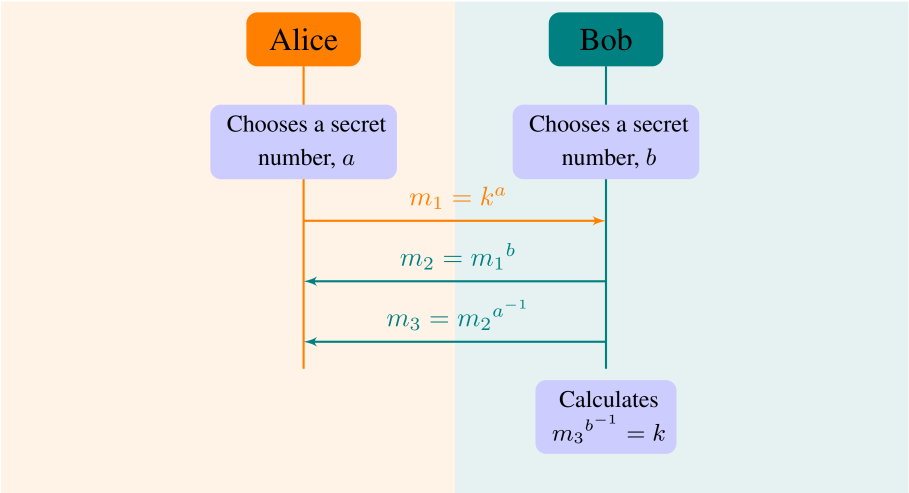

Alice
Bob
Chooses a secret number, a
Chooses a secret number, b
$m_1 = k^a$
$m_2 = {m_1}^b$
$m_3 = {m_2}^{a^{-1}}$
Calculates ${m_3}^{b^{-1}} = k$

Figure 2.3: Modified Diffie-Hellman Key Transport Protocol.

2.5.2 Ko-Lee-Cheon-Han-Kang-Park Key Agreement Protocol

In this protocol, let G be a non-commutative group, and also let $g \in G$ be a public element. Let A and B be two commuting subgroups of G. In order words, $ab = ba$ for all $a \in A$ and $b \in B$. To agree on a shared secret key,

1. Alice chooses an element a randomly from A and sends g^a to Bob[1].
2. Bob chooses an element b randomly from B and sends g^b to Alice.
3. Alice calculates shared secret key $k_a = (g^b)^a$.
4. Bob calculates shared secret key $k_b = (g^a)^b$.

The secret key k becomes:

$$\begin{aligned} k &= k_a = g^{ab} &= (ab)^{-1}g(ab) \\ & &= (ba)^{-1}g(ba) = g^{ba} = k_b. \end{aligned}$$

This protocol [22] [28] [9] is based on the fact that the *conjugacy search problem* (see Section 2.3.5) is computationally-difficult, namely the problem of finding $y \in G$ such that $h = g^y = y^{-1}gy$ where g and h are known.

[1] g^a: The conjugate of g by a.

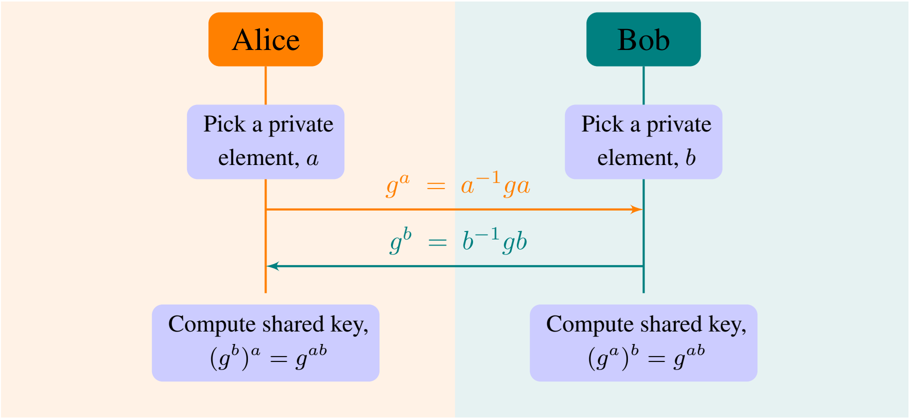

Alice
Bob
Pick a private element, a
Pick a private element, b
$g^a = a^{-1}ga$
$g^b = b^{-1}gb$
Compute shared key, $(g^b)^a = g^{ab}$
Compute shared key, $(g^a)^b = g^{ab}$

Figure 2.4: Ko-Lee-Cheon-Han-Kang-Park Key Agreement Protocol.

2.5.3 Anshel-Anshel-Goldfeld Key Agreement Protocol

Let G be a non-commutative group, and let $a_1, a_2, ..., a_k, b_1, b_2, ..., b_l \in G$ be public elements. To agree on a shared secret key:

1. Alice chooses a word (see Section 2.2.2.3) x from the set $\{a_1, a_2, ..., a_k\}$ and sends the set of conjugates $\{b_1^x, b_2^x, ..., b_l^x\}$ to Bob.

2. Bob chooses a word y from the set $\{b_1, b_2, ..., b_l\}$ and sends the set of conjugates $\{a_1^y, a_2^y, ..., a_k^y\}$ to Alice.

3. Alice calculates x^y whereas Bob calculates y^x. x is the private key of Alice, where y is the private key of Bob.

4. Therefore, the secret key k becomes $[x, y] \ = \ x^{-1}y^{-1}xy$.

Calculations will yield to

$$\begin{aligned} x^y &= (a_{i_1}a_{i_2}...a_{i_r})^y \\ &= (a_{i_1}^y a_{i_2}^y ... a_{i_r}^y) \end{aligned}$$

for given $x = a_{i_1}a_{i_2}...a_{i_r}$ where $a_{i_j} \in \{a_1^{\mp1}, a_2^{\mp1}, ..., a_k^{\mp1}\}$ and

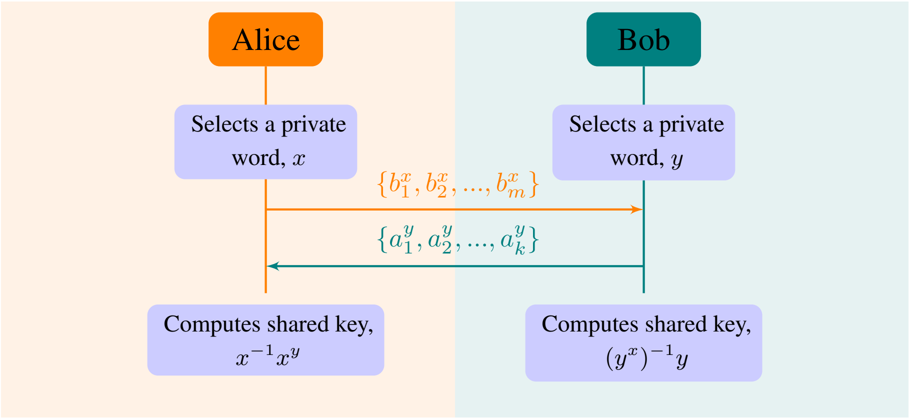

Alice
Bob
Selects a private word, x
Selects a private word, y
$\{b_1^x, b_2^x, ..., b_m^x\}$
$\{a_1^y, a_2^y, ..., a_k^y\}$
Computes shared key, $x^{-1}x^y$
Computes shared key, $(y^x)^{-1}y$

Figure 2.5: Anshel-Anshel-Goldfeld Key Agreement Protocol.

$$\begin{aligned} y^x &= (b_{j_1} b_{j_2} ... b_{j_r})^x \\ &= (b_{j_1}^x b_{j_2}^x ... b_{j_r}^x) \end{aligned}$$

for given $y = b_{j_1} b_{j_2} ... b_{j_r}$ where $b_{j_k} \in \{b_1^{\mp 1}, b_2^{\mp 1}, ..., b_l^{\mp 1}\}$.

To understand how the commutator (see Section 2.2.2.2) works as a shared secret key, we can write the commutator as;

$$\begin{aligned} [x, y] &= x^{-1} y^{-1} x y \\ &= x^{-1}(y^{-1} x y) \\ &= x^{-1} x^y \end{aligned}$$

which can be calculated by Alice's private value x. Similarly,

$$\begin{aligned} [x, y] &= x^{-1} y^{-1} x y \\ &= (x^{-1} y x)^{-1} y \\ &= (y^x)^{-1} y \end{aligned}$$

which can be calculated by Bob's private value y. Note that, since G is a non-commutative group, $[x, y]$ need not be the identity element for any $x, y \in G$.

This protocol [4] [28] [9] is based on the fact that the *Word Search Problem (WSP* (see Section 2.3.7) is computationally-difficult, namely the problem of finding a representation of an element $y \in G$ including elements from the subset of G.

2.5.4 The Stickel Key Agreement Protocol

Let $G = GL_n(\mathbb{F}_q)$ and $g \in G$. Take any two elements $a, b \in G$ of order n_a and n_b, respectively. In other words, $a^{n_a} = b^{n_b} = I$ and suppose that $ab \neq ba$. Assume that G, a and b are all publicly known. To agree on a shared secret key,

1. Alice chooses k, l such that $0 < k < n_a$ and $0 < l < n_b$, sends $u = a^k g b^l$ to Bob.
2. Bob chooses s, t such that $0 < s < n_a$ and $0 < t < n_b$, sends $v = a^s g b^t$ to Alice.
3. Alice calculates $k_a = a^k v b^l$.
4. Bob calculates $k_b = a^s u b^t$.

The secret key k becomes:

$$k = k_a = k_b = a^{k+s} g b^{l+t}.$$

This protocol [28] [9] [39] is based on the fact that the *decomposition search problem* (see Section 2.3.6) is computationally-difficult, namely the problem of finding $a_1, a_2 \in A$ such that $h = a_1 g a_2$ where g and h are known.

2.5.5 Overall Assessment

In addition to passive and active attacks, a protocol should be resistent against some other requirements: [42] [10]

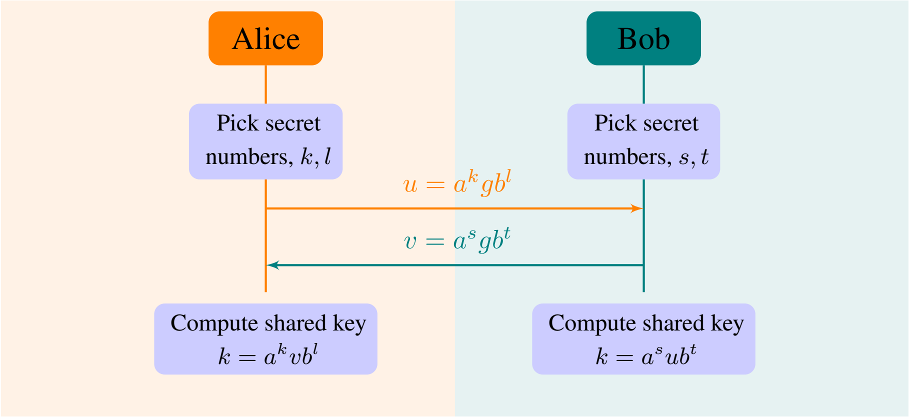

Alice
Bob
Pick secret numbers, k, l
Pick secret numbers, s, t
$u = a^k g b^l$
$v = a^s g b^t$
Compute shared key $k = a^k v b^l$
Compute shared key $k = a^s u b^t$

Figure 2.6: The Stickel Key Agreement Protocol.

- *Known-key security (Ks)* The compromise of the past session keys should not reveal the future session keys.
- *Forward secrecy (Fs)* The compromise of the future session keys should not reveal the previously established session keys.
- *Key-compromise impersonation resilience (KiR)* After revealing the long-term private key of A, the attacker is able to impersonate the entity of A.
- *Unknown key-share resilience (UkR)* It should not be possible that A is tricked to establish a key with party C, when A wants to establish a key with B.
- *Key control (Kc)* The secret key should be established with the participation of A and B together.

In Table 2.1, we list several key establishment protocols and compare them with respect to underlying computationally-difficult problem that security of a protocol depends on, number of messages, specific security requirements (Kks, Fs, KciR, UksR, KC) presented and existence of a pre-shared key.

In the second column of the table, we analyse the underlying computationally-difficult problems on which the security of the protocols depend. Most of the protocols depend on computationally-difficult problems while Wen-Lin-Hwang's protocol employ the properties of hash functions, Otway-Rees and Needham-Schroeder rely on symmetric encryption. The original Shamir's no-key protocol relies on symmetric encryption but LKTP also benefits from integer factorization problem in the initialization phase.

Since resource-constrained devices consume energy while sending and receiving messages, a protocol should not need to exchange too many messages between entities. Although Diffie-Hellman based protocols accomplish key establishment with two messages only, LKTP requires three messages as a three-pass protocol. Needham-Schroeder consumes the highest energy while sending and receiving messages by exchanging five messages.

Specific security requirements (Kks, Fs, KciR, UksR, KC) of protocols are given starting from the fourth column of the table. It is important to note that protocols without authentication could not satisfy KciR and UksR. Moreover, key transport protocols are usually preferred for transporting an already-selected private key from Alice to Bob, they could not satisfy the key control requirement, either.

Non-commutative cryptographic key agreement protocols are not practically usable and choice of platform groups for each of the protocols are not well-defined. If the chosen groups are not appropriate, the underlying computationally-difficult problems can be easily solvable. [9] [28]

Table 2.1: Comparison of key establishment protocols, [42], [26], [6].

Protocol	Security Dependence	Number of Messages	KkS	Fs	KciR	UksR	KC	PSK	Reference
LKTP	Symmetric	3	YES	YES	NO	NO	-	YES	
Diffie-Hellman	DLP	2	YES	YES	NO	NO	YES	NO	[13]
Ko-Lee-Cheon-Han-Kang-Park	CSP	2	YES	YES	NO	NO	YES	NO	[22], [28]
Anshel-Anshel-Goldfeld	WSP	2	YES	YES	NO	NO	YES	NO	[4], [28]
The Stickel	DSP	2	YES	YES	NO	NO	YES	NO	[28], [39]
Tseng's Protocol	DLP	4	YES	YES	YES	YES	YES	NO	[40]
Popescu's Protocol	ECDLP	2	YES	YES	NO	YES	YES	NO	[34]
Wen-Lin-Hwang's Protocol	Hash Function	3	YES	NO	YES	YES	YES	YES	[44]
Harn-Hsin-Mehta's Protocol	IFP	3	YES	NO	YES	YES	YES	NO	[18]
Otway-Rees	Symmetric	4	YES	YES	NO	NO	-	YES	[30]
Beller-Yacobi Protocol	IFP / DLP	4	YES	YES	YES	YES	-	YES	[7], [8]
Needham-Schroeder	Symmetric	5	YES	YES	NO	NO	-	YES	[29]

CHAPTER 3

IMPOSSIBILITY OF THREE-PASS PROTOCOL OVER PUBLIC COMMUTATIVE GROUPS

In this chapter, we will show why we decided to hide the group G. In Section 3.1, we will generalize the one-time pad by using a group automorphism. In Section 3.2, we will explain the three-pass protocol over public commutative groups. To transport a key successfully, the requirements of the group G will be mentioned in Section 3.3. A simple example in Section 3.4 using Klein four-group V will help the reader understand how three-pass works. Finally, in Section 3.5, we will show that it is impossible to use three-pass protocol over public commutative groups since an attacker does not need to find the private element to calculate the private key.

Confidentiality in secure communications is defined as ensuring that an adversary gains no intelligence from a sent message. Alice and Bob would like to communicate with each other. However, they do not share any secrets. They only share the endpoints of a communication channel that is fast, albeit insecure. Anything put on the channel may be tapped by a passive eavesdropper, Eve. We assume there is no active attacker (e.g., Mallory) in the system when Alice and Bob talk to each other; i.e., active attacks such as the man in the middle attack is out of the scope of the thesis. Is it possible for Alice and Bob to exchange messages in a *confidential* manner without having pre-shared secrets (keys)? The answer to this question falls in the scope of key establishment protocols over insecure channels.

The three-pass protocol, also known as Shamir's no-key protocol (protocol 12.22 in [26]), is a key transport protocol developed by Adi Shamir in 1980 where Alice wants to transport a secret message m to Bob over an insecure channel, and they do not share any secret information. They have mutually agreed on a symmetric encryption scheme that is a pair of encryption and decryption algorithms (E, D) acting on a message space $\mathcal{M}$ (where $|\mathcal{M}| > 1$) and a key space $\mathcal{K}$ such that for all messages $m \in \mathcal{M}$ and keys $k \in \mathcal{K}$, $\Pr[D_k(E_k(m) = m] = 1$ where $E_k(m)$ is the notation for encryption of message m with key k. Both Alice and Bob determine their secret keys k_a and k_b respectively without sharing with each other. Alice encrypts m with her secret key k_a and sends $c_1 = E_{k_a}(m)$ to Bob. Bob does not have any idea about k_a, therefore, cannot possibly decrypt c_1. Bob sends back $c_2 = E_{k_b}(c_1)$ to Alice. If E and D commutes, Alice can produce $c_3 = D_{k_a}(c_2) = E_{k_b}(m)$ and send it to Bob. Finally, Bob can decrypt it and retrieve the secret message m.

Based on this brief overview, we will generalize the idea of one-time pad to commutative groups, and then we will mention employing the original three pass protocol by using commutative groups.

3.1 Generalized One-time Pad

Let $(G, *)$ be a commutative group acting on a set S. The encryption is defined as the action of G on S. If there exists a group homomorphism

$$\varphi : G \to Sym(S)$$
$$g \to \varphi_g$$

where $Sym(S)$ is the group of all permutations on the set S under composition, then we say that G acts on S. Here, the map φ is called the permutation representation of G on S. To simplify the notation, we prefer to use the right action notation $a \circ g$ to represent $\varphi_g(a)$. Note that a group action satisfies $a \circ (g * h) = (a \circ g) \circ h$ and $a \circ e = a$, where e is the identity element of G. This is the generalization of the one-time pad cipher to commutative groups with arbitrary orders.

3.2 Three Pass Protocol Using Commutative Groups

During the thesis, we assume that Alice is a sink node, who chooses a random key k and transport to Bob who is a client. Suppose that Alice and Bob agree on a public

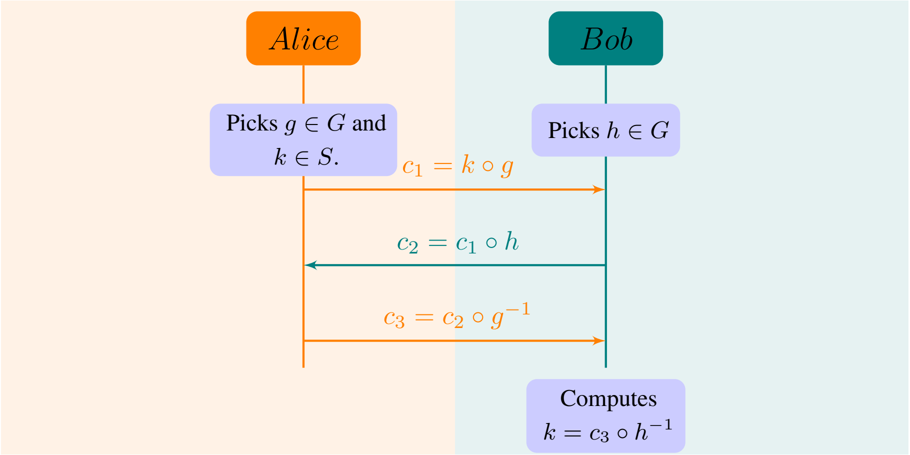


Figure 3.1: Three-pass protocol using public Abelian group actions.

commutative group G and a public homomorphism φ. Notice that, G and φ have to be **public** since the three-pass protocol assumes no pre-shared information. As shown in Fig.3.1, Alice selects the to-be-transported secret key $k \in S$ and a private group element $g \in G$ uniform randomly, then sends Bob $c_1 = k \circ g$. Notice here that this operation is the encryption of the secret message k using the group action g (i.e., $c_1 = E_g(k)$). Bob selects his private group element $h \in G$ uniform randomly and sends back to Alice $c_2 = c_1 \circ h$. After that, Alice computes $c_3 = c_2 \circ g^{-1}$. Following the properties of group actions $c_3 = ((k \circ g) \circ h) \circ g^{-1} = k \circ (g * h * g^{-1})$. Since we chose a commutative group, $c_3 = k \circ (g * g^{-1} * h) = k \circ (e * h) = k \circ h$. In this protocol, all the messages conveyed over the channel, $c_1, c_2, c_3 \in C = S$ are encrypted. At the end of these three exchange of messages, Bob is able to secretly compute the key k by $(k \circ h) \circ h^{-1} = k \circ e = k$.

3.3 Requirements of the Public Group G

G has to satisfy the following properties.

Property 1 *Commutativity: The group G must be commutative. If G is not commutative, then c_3 need not be equal to $k \circ h$.*

Property 2 *Uniform Randomness: Let* **C**, **S** *and* **G** *be the random variables denoting* *the values of ciphertexts (e.g., c_1, c_2 or $c_3 \in C$), secret keys to-be-transported (e.g., $k \in S$) and group elements (e.g., $g, h \in G$), respectively. To be able to provide information-theoretic security, uniform randomness must be satisfied, i.e.,*

$$Pr[\mathbf{C} = c \mid \mathbf{S} = k, \mathbf{G} = g] = Pr[\mathbf{C} = c \mid \mathbf{S} = k, \mathbf{G} = h]$$

and

$$Pr[\mathbf{C} = c \mid \mathbf{S} = k_0, \mathbf{G} = g] = Pr[\mathbf{C} = c \mid \mathbf{S} = k_1, \mathbf{G} = g]$$

providing computational indistinguishability for any probability distribution over the set S, the group G and ciphertext space $C = S$ where $Pr[\mathbf{C} = c] > 0$. Here, we claim that the probability of conveying any ciphertext over the channel in this protocol is equally likely. Therefore, eavesdropping does not give any statistical advantage to an attacker, i.e.,

$$Pr[\mathbf{S} = k \mid \mathbf{C} = c] = Pr[\mathbf{S} = k].$$

We select the to-be-transported secret key $k \in S$ and a private group element $g \in G$ shown in Fig. 3.1 uniform randomly to satisfy Property 2. Furthermore, for any pair $(a, b) \in S \times S$, there must be exactly $M \geqslant 1$ group elements sending[1] a to b. In particular, this action must be transitive as defined in Section 2.1.3.

3.4 An Easy Example Implementation

An easy example is to use the natural action of Klein four-group V. Klein four-group is a well-known commutative permutation group on $S = \{1, 2, 3, 4\}$ whose non-identity elements has order two. That is, every element is its own inverse. It is a subgroup of $Sym(S)$. For the sake of simplicity and clarity, we present V using the two-row notation where images are given in the second row of $\tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ \tau(1) & \tau(2) & \tau(3) & \tau(4) \end{pmatrix}$.

The Klein four-group is $V = \{\tau_0, \tau_1, \tau_2, \tau_3\}$ where $\tau_0 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$, $\tau_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$, $\tau_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$, and $\tau_3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$.

[1] When we say g sends a to b, we mean $b = a \circ g$.

The implementation of the three-pass protocol using V is as follows. Alice and Bob will employ V. Suppose Alice selects $k = 3$ as the secret key and her private group element $g = \tau_1$ uniform randomly. She sends Bob $c_1 = 3 \circ \tau_1 = 4$. Suppose Bob selects his private group element $h = \tau_2$ uniform randomly as well. Then, he sends back $c_2 = 4 \circ \tau_2 = 2$. Alice computes $c_3 = 2 \circ \tau_1^{-1} = 2 \circ \tau_1 = 1$ and sends it to Bob. Finally, Bob recovers the secret key by $k = 1 \circ \tau_2^{-1} = 1 \circ \tau_2 = 3$, which is the secret key determined by Alice following some probability distribution.

V satisfies both Property 1 and Property 2. Any message c_1, c_2 or c_3 put on the insecure channel is encrypted. Therefore, the reader may initially think it is not possible for Eve to recover the key k. However, this implementation is prone to brute-force attacks. Therefore, an additional requirement is imposed on the order of G:

Property 3 *Large Order: Trivially, the order of the group G and the cardinality of S must be very large to make the brute force attack almost infeasible. A brute-force attack can be (theoretically) used against any cryptosystem to find out the secrets. In this protocol, the private key is an element of the selected group G. Hence, the probability that an attacker successfully finds the correct element of the group G by a random trial is equal to* $1/|G|$.

We present in the sequel that a much deeper security flaw resides in such implementations of the three-pass protocol even if we can find a G that satisfies all these three properties.

3.5 Impossibility of Three-pass Protocol over Public Commutative Groups

When Eve observes $c_1 = 4$ and $c_2 = 2$ in the above example, she immediately finds out Bob's private group element $h = \tau_2$, since there is only one element τ in V sending a to b for any pair of elements $(a, b) \in S \times S$. It is computationally easy to determine τ in V when a and b are given as in the three-pass protocol. In fact, this security flaw is valid for any publicly known transitive commutative group G even when the order of G is very large and even when there exist many distinct group elements sending a to b. Determining one group element sending a to b is enough to break the system and to determine the secret group elements $g, h \in G$ and the secret key k to-be-transported shown in Fig. 3.1.

One may get this result as follows. Let G be a commutative group acting transitively on S via a homomorphism $\varphi : G \to Sym(S)$. Take any $a, b \in S$. Suppose that there exist two distinct group elements g_1, g_2 sending a to b. Then $g_1 * g_2^{-1}$ is in the point stabilizer of a (see Section 2.1.2). To put it plainly, let $a \circ g_1 = b = a \circ g_2$. Then $a \circ (g_1 * g_2^{-1}) = (a \circ g_1) \circ g_2^{-1} = b \circ g_2^{-1} = a$. Therefore, $g_1 * g_2^{-1}$ is an element of G_a. Say $g_1 = \alpha * g_2$ for some $\alpha \in G_a$. Now consider the kernel (see Section 2.1.1) of the permutation representation φ, which is $Ker\varphi = \bigcap_{x \in S} G_x$. Take any element $x \in S$. Since, G is transitive on S, there exists an element $r \in G$ such that $x = a \circ r$. Then by Theorem 1.4A-ii in [14] presented in Section 2.1.4, $G_x = r^{-1} * G_a * r$. As G is commutative, we get $G_x = r^{-1} * G_a * r = r^{-1} * r * G_a = G_a$ for all x and so $Ker\varphi = G_a$. Hence $\varphi(g_1) = \varphi(\alpha * g_2) = \varphi(g_2)$. For the convenience of the reader, let us verify the last equality element-wise; $x \circ g_1 = (a \circ r) \circ (\alpha * g_2) = a \circ (r * \alpha * g_2) = a \circ (\alpha * r * g_2) = (a \circ r) \circ g_2 = x \circ g_2$ since $a \circ \alpha = a$. That is, for any pair a, b in S, there may be two distinct group elements sending a to b but in fact the corresponding permutation is unique.

Let's relax the assumptions and consider that the above action of the group G on the set S is not transitive. In this case, we cannot satisfy Property 2 since Pr$[\mathbf{C} = c] = 0$ for some ciphertext c. Even if we do not get the equality $\varphi(g_1) = \varphi(g_2)$ for the elements satisfying $a \circ g_1 = b = a \circ g_2$ the security flaw still exists.

Note that the equality $g_1 = \alpha * g_2$ for some $\alpha \in G_a$ holds whenever $a \circ g_1 = b = a \circ g_2$. As a consequence of this equality, it will be sufficient for the attacker to find any g that satisfies $a \circ g = b$ instead of trying to find the selected private group element. In the three pass-protocol, Alice selects a secret key $k \in S$ and a private group element $g \in G$, sends Bob $c_1 = k \circ g$. Then Bob selects his private group element h and sends back to Alice $c_2 = c_1 \circ h$. Suppose Eve finds a group element h' satisfying $c_1 \circ h' = c_2$. Then $h' = \alpha * h$ for some $\alpha \in G_{c_1}$. When Alice sends $c_3 = (c_1 \circ h) \circ g^{-1}$, Eve is able to recover the secret key by computing $c_3 \circ (h')^{-1}$. Indeed, $c_3 \circ (h')^{-1} = ((c_1 \circ h) \circ g^{-1}) \circ (h')^{-1} = c_1 \circ (h * g^{-1} * (h')^{-1}) = c_1 \circ (h * g^{-1} * (\alpha * h)^{-1}) = c_1 \circ (h * g^{-1} * h^{-1} * \alpha^{-1})$. Note that $\alpha^{-1} \in G_{c_1}$ as $\alpha \in G_{c_1}$. Since the group is commutative, one has $c_3 \circ (h')^{-1} = c_1 \circ (\alpha^{-1} * g^{-1} * h * h^{-1}) = (c_1 \circ \alpha^{-1}) \circ g^{-1} = c_1 \circ g^{-1} = k$. Therefore, the three-pass protocol is insecure when public commutative groups are employed.

Since it is impossible to use three-pass protocol over public commutative groups, we decide to hide the group and propose a technique where the preshared secret is the group G and have the information-theoretic security throughout the sequel of the key transportation process.

CHAPTER 4

LIGHTWEIGHT KEY TRANSPORT PROTOCOL

In this chapter, we will explain the structure and details of the proposed key-transport protocol. In Section 4.1, phases of the protocol are described in details. A feasible implementation of the underlying idea over multiplicative groups will be mentioned in Section 4.2. In Section 4.3, we will analyse the protocol to see whether it provides several security requirements and conforms to the requirements. We will conclude the chapter by explaining the hardware capabilities of the platforms on which we implemented the protocol, results related to memory size, total time and total number of calculations required to transport a single key in Section 4.4.

4.1 LKTP

In this section, we will explain the details of LKTP, and then give a feasible implementation over finite multiplicative groups. To understand the implementation details of the protocol, the notation is presented in Table 4.1.

Table 4.1: List of symbols in LKTP.

p	Installed prime number on each node
q	Master key for the Key-Transportation Phase to hide a group.
n	Initial key message from Alice to Bob, $n = p \times q$
a	Alice's element.
c	Inverse of Alice's element.
b	Bob's element.
d	Inverse of Bob's element.

Suppose Alice and Bob want to share a secret key to establish a security association. Before the key-transportation, both entities should agree on the secret commutative group in the initialization phase.

4.1.1 Initialization Phase

We assume both Alice and Bob are pre-initialized with a prime number p which needs to be stored in a tamper-resistant hardware since an operation will be defined over a finite commutative group, $(G, *)$, which will be hidden by using the value of p. Afterwards;

- When Bob learns Alice's address, it sends a unicast request ESTABLISH_GROUP to Alice to agree on $(G, *)$.
- When Alice receives the ESTABLISH_GROUP request, it sets q and sends it to Bob after employing one of the computationally-difficult problems given in Section. 2.3, by using p.
- Bob receives the encrypted message from Alice, and easily sets the value of q by being a legitimate node that can solve the computationally-difficult problem by using the necessary information it has. For example, if Bob receives a number $n = pq$ where p and q are prime numbers and the related computationally-difficult problem is Integer Factorization, Bob can easily calculate the hidden prime q after dividing n by p since p is known by Bob.

After Alice and Bob agree on the hidden group G, they select integers a and b from G, and calculate their inverses c and d in G, respectively. This completes the initialization phase.

4.1.2 Key-transportation Phase

This phase is the three-pass protocol shown in Fig. 3.1.

Necessary steps to transport a single key k from Alice to Bob can be found in Fig. 4.1.

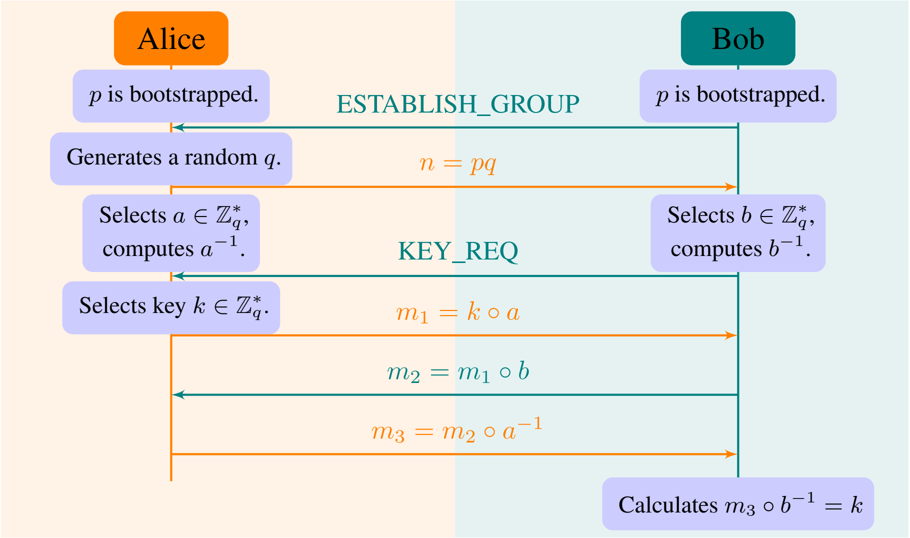

Alice
Bob
p is bootstrapped.
ESTABLISH_GROUP
p is bootstrapped.
Generates a random q.
$n = pq$
Selects $a \in \mathbb{Z}_q^*$, computes a^{-1}.
KEY_REQ
Selects $b \in \mathbb{Z}_q^*$, computes b^{-1}.
Selects key $k \in \mathbb{Z}_q^*$.
$m_1 = k \circ a$
$m_2 = m_1 \circ b$
$m_3 = m_2 \circ a^{-1}$
Calculates $m_3 \circ b^{-1} = k$

Figure 4.1: Flow Diagram of LKTP.

4.2 A Feasible Implementation of Key Transport Protocol

An easy example is to use multiplication as the operation over finite cyclic group $G = \mathbb{Z}_q^*$, multiplicative group modulo q. To transport a key k,

1. A prime number p is assumed to be bootstrapped in both devices.

2. Alice generates a random prime number q, and send $n = p \times q$ to Bob.

3. Bob sets q and G after dividing n by p.

4. Bob sends KEY_REQ packet to Alice.

5. Alice selects an integer a, where Bob selects an integer b such that a, b are in $\{1, ..., q\}$. Then $\bar{a}, \bar{b}$, modulo classes of a and b, are elements of $\mathbb{Z}_q^*$. Therefore there are integers c, d in $\{1, ..., p\}$ such that $\bar{c}$ and $\bar{d}$ are inverses of $\bar{a}$ and $\bar{b}$ in $\mathbb{Z}_q^*$, respectively. Hence $ac = t_1 p + 1$ and $bd = t_2 p + 1$ for some integers t_1, t_2.

6. When the key transportation starts, Alice chooses a random key $k \in G$ and sends $m_1 = k \times a$ to Bob.

7. Bob receives the encrypted message, encrypts it once more by calculating $m_2 = m_1 \times b$ and sends m_2 to Alice.

8. Alice calculates $m_3 = m_2 \times c$ and sends it to Bob to start the last phase of the protocol.

9. Bob calculates

$$\begin{aligned} m_3 \times d &= k \times (a \times b \times c \times d) \\ &= k \times (a \times c) \times (b \times d) \\ &= k \times (t_1 p + 1) \times (t_2 p + 1) \\ &= k \times (t_1 t_2 p^2 + t_1 p + t_2 p + 1) \\ &\equiv k \ (mod\, p) \end{aligned}$$

   and recovers the original key k.

Now, consider the following simple, working example given shown in Fig.4.2:

1. A prime number $p = 11$ is bootstrapped in both nodes.

2. Alice selects $q = 13$ and calculates $n = pq = 11 \times 13 = 143$. The finite cyclic group is $G = \mathbb{Z}_{13}^*$, multiplicative group modulo 13.

3. Alice selects $a = 5$ and determines $c \in G$ such that $ac \equiv 1 \ (mod\ 13)$. The correct value is $c = 8$.

4. Bob selects $b = 7$ and determines $d \in G$ such that $bd \equiv 1 \ (mod\ 13)$. The correct value is $d = 2$.

5. Alice selects $k = 4$, calculates $4 \times 5 \equiv 7 \ (mod\ 13)$ and sends 7 to Bob.

6. Bob calculates $7 \times 7 \equiv 10 \ (mod\ 13)$ and sends 10 to Alice.

7. Alice calculates $10 \times 8 \equiv 2 \ (mod\ 13)$ and sends 2 to Bob.

8. Finally, Bob calculates $2 \times 2 \equiv 4 \ (mod\ 13)$ which is equal to the value of k.

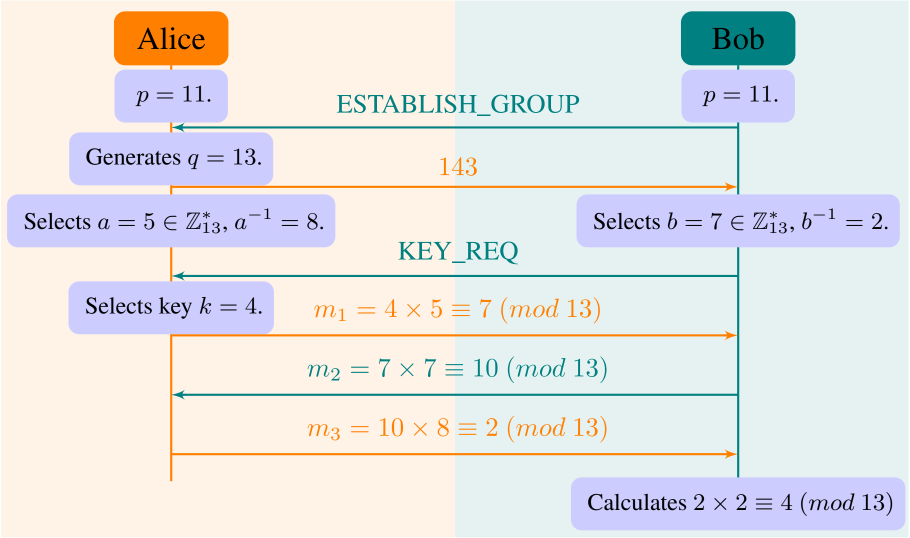

Alice
Bob
$p = 11.$
$p = 11.$
ESTABLISH_GROUP
Generates $q = 13.$
143
Selects $a = 5 \in \mathbb{Z}_{13}^*$, $a^{-1} = 8.$
Selects $b = 7 \in \mathbb{Z}_{13}^*$, $b^{-1} = 2.$
KEY_REQ
Selects key $k = 4.$
$m_1 = 4 \times 5 \equiv 7 \ (mod\ 13)$
$m_2 = 7 \times 7 \equiv 10 \ (mod\ 13)$
$m_3 = 10 \times 8 \equiv 2 \ (mod\ 13)$
Calculates $2 \times 2 \equiv 4 \ (mod\ 13)$

Figure 4.2: A simple example working over a multiplicative group $\mathbb{Z}_{13}^*$.

4.3 Security Analysis

The proposed key transport protocol satisfies known-key security and forward secrecy because the session keys are generated randomly from a uniform distribution. The protocol fails to meet key-compromise impersonation resilience and unknown key-share resilience security features since the authentication mechanism is not considered in this thesis. Due to the nature of key transport protocols in which transportation of a key from Alice to Bob is achieved, key control feature does not fit into content of key transport protocols.

The multiplication of two prime numbers can be computed efficiently when Alice calculates $k_i = p \times q$, however, factoring the product of two prime numbers is a computationally-difficult problem which is known as *Integer Factorization Problem* (see Section. 2.3.2). However, the factorization of k_i can be done efficiently by only a legitimate node.

The proposed protocol satisfy the three properties over a multiplicative group $G = (\mathbb{Z}_q^*, \times)$,

- The multiplication is commutative over G.

- If we consider Cayley table for G, each row includes every element of G exactly once. To show that, let a, x and y be elements of G such that $x \neq y$. If $a \times x = a \times y$, multiplying both sides with a^{-1} from left-hand side, then we will have $x = y$ which contradicts the assumption.

- LKTP can work on any commutative group with arbitrary order. To satisfy the large order property, prime number p can be chosen as 2048, 3072-bit long.

4.4 Implementation, Results and Discussion

In this section, we will give implementation details of LKTP and modified Diffie Hellman key transport protocol, and then mention about the hardware properties of the test environments, choice of the programming language and the library.

4.4.1 Methodology

The proposed key transportation protocol LKTP and the modified Diffie-Hellman Key Transport protocol are implemented in IoT-Lab [1] test-bed and in a more powerful desktop environment.

IoT-Lab provides an infrastructure of varying number of wireless sensor nodes to submit new experiments and collect results. Their test-beds are located at six different places and they give access to 2728 wireless sensor nodes of three different types. The selected node type for experiment is M3, and the experiment mostly runs on Lyon and Grenoble location since the first provides a small and quite infrastructure of 18 M3 nodes whereas the latter provides a large scale infrastructure consisting of 384 M3 nodes.

M3 nodes provide:

- ARM Cortex M3, 32-bits, 72 Mhz MCU, 64kB RAM

- Atmospheric pressure and temperature sensor

- 802.15.4 PHY standard, 2.4 Ghz radio communication
- 128 Mbits external memory
- 3,7V LiPo (650 mAh) battery

Desktop environment provides:

- Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz, 8GB RAM

For the programming language choice, C is the available option for programming a protocol on M3 nodes. On the other hand, since the arithmetic library MPIR [3] provides an easy interface for big integers, C++ becomes the better choice to lessen the code length of the implementations.

4.4.2 Results and Discussion

After the implementations of the protocols run correctly on both platforms, they have been investigated in terms of memory requirements and total time to complete a single key transport from Alice to Bob.

Since neither protocol has any extra storage requirement, a small difference between the memory usage is expected. In Table 4.2, the memory usage of both protocols are compared for IoT-Lab and desktop implementations.

Table 4.2: Size of memory sections. (in bytes)

	IoT-Lab		Desktop	
	DH	LKTP	DH	LKTP
Text	112796	108064	54367	54067
Data	4472	4464	2448	2416
Bss	54872	54872	608	608
Total	**172140**	**167400**	**57423**	**57091**

Modified Diffie-Hellman protocol to transport a key makes use of mpz_powm function (in MPIR library) to calculate the exponentiation g^e of big integers g and e, em-

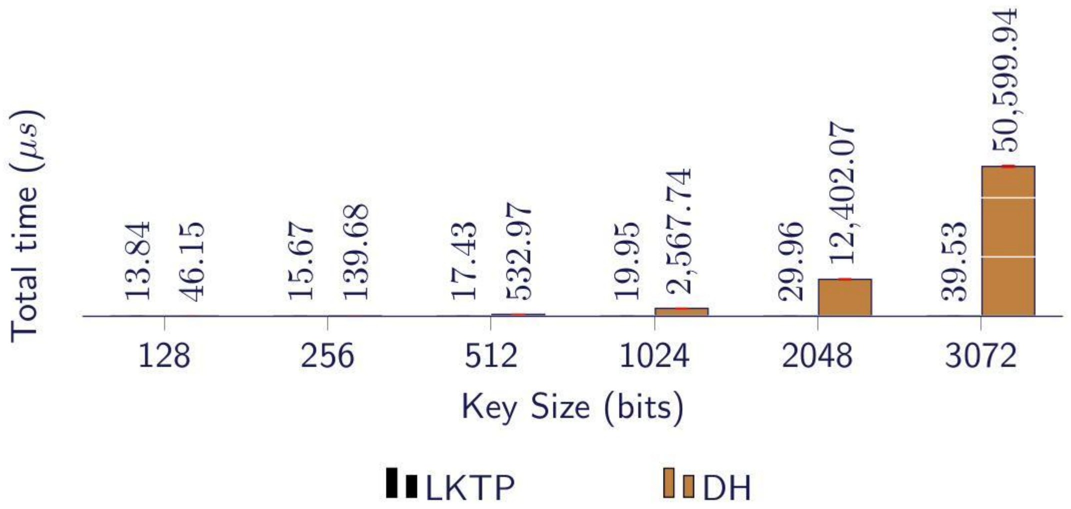


Figure 4.3: Key size versus total time to transport a single key in desktop.

ploys *right-to-left binary exponentiation* proposed by Knuth, Bach and Shallit [19]. For a random e, chosen from a finite set $G = \mathbb{Z}_n^*$, the total number of squarings and multiplication will be $\lfloor lgn \rfloor$, $\frac{1}{2}(\lfloor lgn \rfloor + 1)$, respectively as shown in Table 4.3, where n is the number of elements in G. However, in the key transportation phase of LKTP, a single multiplication is computed by each entity in every step. The other important parameter that we considered while comparing two protocols is the total time to transport a single key from Alice to Bob. To see the difference between two platforms, we considered different key sizes as inputs and corresponding run time of both protocols as shown in Fig.4.3. Since the number of multiplication and squarings increase, the total run time of the modified Diffie-Hellman key transport protocol increases drastically while LKTP grows almost linearly.

Table 4.3: Number of computations to calculate k^a in DH and $k \times a$ in LKTP.

	Multiplication	**Squaring**
DH	$\frac{1}{2}(\lfloor lgn \rfloor + 1)$	$\lfloor lgn \rfloor$
LKTP	1	-

CHAPTER 5

CONCLUSION

5.1 Conclusion

In this thesis, we proposed the lightweight key transport protocol LKTP, which is based on Shamir's no-key protocol, after evaluating the computational burden of current public-key protocols and the necessity for a lightweight key establishment protocol in Internet of Things and wireless sensor networks. By showing the impossibility of the three-pass protocol over public commutative groups, we decided to hide the group by employing a computationally-difficult problem in the initialization phase. To validate the protocol and do comparison, first we morphed the original Diffie-Hellman key exchange protocol into a key-transport protocol and then implemented both key transport protocols on IoT-lab and a powerful desktop environment. The major findings of the experiments are that LKTP performs a single key transport faster than the modified Diffie-Hellman protocol, and the memory requirements of both protocols are almost the same. The required time to transport a single key grows almost linearly for LKTP while Diffie-Hellman takes too long to complete the transport.

5.2 Future Work

The proposed key-transport protocol does not provide any authentication mechanism and therefore is vulnerable against key-compromise impersonation and unknown key-share resilience. An authentication mechanism by using the private elements of both entities can be added. Moreover, possible attack models can be defined to understand the threats. Although the computational burden of the proposed protocol is low, power consumption while encryption and sending messages can be considered to validate the findings.

REFERENCES

[1] FIT/IoT-LAB: Very large scale open testbed. http://www.iot-lab.info. Accessed: 2017-07-11.

[2] Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). http://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. Accessed: 2017-09-05.

[3] MPIR: Multiple Precision Integers and Rationals. http://mpir.org. Accessed: 2017-07-11.

[4] I. Anshel, M. Anshel, and D. Goldfeld. An algebraic method for public-key cryptography. *Mathematical Research Letters*, 6:287–292, 1999.

[5] M. Artin. *Algebra*. Pearson, 2nd edition, 2011.

[6] A. O. Baalghusun, O. F. Abusalem, Z. A. Al Abbas, and J. Kar. Authenticated key agreement protocols: A comparative study. *Journal of Information Security*, 6(1):51, 2015.

[7] M. J. Beller, L.-F. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. *IEEE Journal on selected areas in communications*, 11(6):821–829, 1993.

[8] M. J. Beller and Y. Yacobi. Fully-fledged two-way public-key authentication and key agreement for low-cost terminals. *Electronics Letters*, 29(11):999–1001, 1993.

[9] S. R. Blackburn, C. Cid, and C. Mullan. Group theory in cryptography. *Proceedings of Group St Andrews 2009 in Bath*, pages 133–149, 2011.

[10] C. Boyd and A. Mathuria. Key agreement protocols. In *Protocols for Authentication and Key Establishment*, pages 137–199. Springer, 2003.

[11] H. Chan, A. Perrig, and D. Song. Key distribution techniques for sensor networks. In *Wireless sensor networks*, pages 277–303. Springer, 2004.

[12] J. S. Coron, D. Lefranc, and G. Poupard. A new baby-step giant-step algorithm and some applications to cryptanalysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 47–60. Springer, 2005.

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

[14] J. D. Dixon and B. Mortimer. *Permutation Groups, Graduate Texts in Mathematics*, volume 163. Springer-Verlag, New York, 1996.

[15] D. S. Dummit and R. M. Foote. *Abstract Algebra*. Wiley Hoboken, 2004.

[16] P. Gauravaram and L. R. Knudsen. Cryptographic hash functions. In *Handbook of Information and Communication Security*, pages 59–79. Springer, 2010.

[17] D. Hankerson and A. Menezes. Elliptic curve discrete logarithm problem. In *Encyclopedia of Cryptography and Security*, pages 186–189. Springer, 2005.

[18] L. Harn, M. Mehta, and W.-J. Hsin. Integrating Diffie-Hellman Key Exchange into the Digital Signature Algorithm (DSA). *IEEE communications letters*, 8(3):198–200, 2004.

[19] A. Jakubski and R. Perliński. Review of general exponentiation algorithms. *Scientific Research of the Institute of Mathematics and Computer Science*, 10(2):87–98, 2011.

[20] S. M. Kevin. The discrete logarithm problem. *Cryptology and computational number theory*, 42:49, 1990.

[21] K. Kloster. Factoring a semiprime n by estimating $\varphi(n)$. *Honor's thesis, Fordham University*, 2010.

[22] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J.-s. Kang, and C. Park. *New Public-Key Cryptosystem Using Braid Groups*, pages 166–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[23] D. Lehmer. On euler's totient function. *Bulletin of the American Mathematical Society*, 38(10):745–751, 1932.

[24] M. Liskov. Fermat's little theorem. In *Encyclopedia of Cryptography and Security*, pages 456–456. Springer, 2011.

[25] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos. Lightweight cryptography for embedded systems - a comparative analysis. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 333–349. Springer, 2014.

[26] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1996.

[27] J. Müller. Computer Algebra: Primality Testing and Integer Factorisation Friedrich-Schiller-Universität Jena, WS. 2014.

[28] A. G. Myasnikov, V. Shpilrain, A. Ushakov, and N. Mosina. *Non-commutative Cryptography and Complexity of Group-theoretic Problems*, volume 177. American Mathematical Society Providence, RI, USA, 2011.

[29] 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.

[30] D. Otway and O. Rees. Efficient and timely mutual authentication. *ACM SIGOPS Operating Systems Review*, 21(1):8–10, 1987.

[31] C. Paar and J. Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Science & Business Media, 2009.

[32] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $gf(p)$ and its cryptographic significance (corresp.). *IEEE Transactions on information Theory*, 24(1):106–110, 1978.

[33] J. M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[34] C. Popescu. A secure authenticated key agreement protocol. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, volume 2, pages 783–786. IEEE, 2004.

[35] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[36] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[37] M. L. Sommerseth and H. Hoeiland. Pohlig-hellman applied in elliptic curve cryptography. `https://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Sommerseth+Hoeiland.pdf`, 2015.

[38] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The First Collision for Full SHA-1. *IACR Cryptology ePrint Archive*, 2017:190, 2017.

[39] E. Stickel. A new method for exchanging secret keys. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*, volume 2, pages 426–430. IEEE, 2005.

[40] Y.-M. Tseng. Efficient authenticated key agreement protocols resistant to a denial-of-service attack. *International Journal of Network Management*, 15(3):193–202, 2005.

[41] B. S. Verkhovsky. Integer factorization of semi-primes based on analysis of a sequence of modular elliptic equations. *International Journal of Communications, Network and System Sciences*, 4(10):609, 2011.

[42] B. Vesterås. Analysis of key agreement protocols. Master's thesis, Department of Computer Science and Media Technology Gjøvik University College, 2006.

[43] H. Wang, B. Sheng, C. C. Tan, and Q. Li. Comparing symmetric-key and public-key based security schemes in sensor networks: A case study of user access control. In *Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on*, pages 11–18. IEEE, 2008.

[44] H.-A. Wen, C.-L. Lin, and T. Hwang. Provably secure authenticated key exchange protocols for low power computing clients. *Computers & Security*, 25(2):106–113, 2006.

[45] D. B. West et al. *Introduction to Graph Theory*, volume 2. Prentice Hall Upper Saddle River, 2001.

[46] L. Xie, Y. Zhang, Z. Zheng, and X. Zhang. Trip: A tussle-resistant internet pricing mechanism. *IEEE Communications Letters*, PP(99):1–1, 2016.

[47] X. Zhang, H. M. Heys, and C. Li. Energy efficiency of symmetric key cryptographic algorithms in wireless sensor networks. In *Communications (QBSC), 2010 25th Biennial Symposium on*, pages 168–172. IEEE, 2010.