Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Security in Building Automation Systems: a Study on Multi-party Key-agreement Protocols

Relatore:	Prof. Gerardo PELOSI
Correlatore:	Ing. Alessandro BARENGHI

Tesi di Laurea di:
Dario NAVONI Matr. 799119

Anno Accademico 2013–2014

Ringraziamenti

Il primo e più grande ringraziamento va alla mia Famiglia: ai miei genitori, Viviana e Gianni, a mio fratello Diego e a mia nonna Angela. Senza il loro costante supporto e amore incondizionato non sarei ciò che sono ora.

Il secondo ringraziamento va ai miei Amici, che mi sono sempre stati vicini durante questi 5 anni di Politecnico, anche nei momenti più difficili. La semplice inclusione dei loro nomi in una lista non rende giustizia a ciò che meriterebbero, spero comunque che apprezzino questo piccolo gesto. Ringrazio quindi i Giovani di Guascogna, ovvero Caste, Diego, Mazzo e Tama, per il loro supporto e per la loro stoica sopportazione del sottoscritto. Ringrazio anche il dott. Michele Asti, per il suo aiuto e i suoi consigli, preziosi und azzeccati. Meritano inoltre una menzione due ragazze speciali, che conosco da meno tempo ma sono comunque entrate nel mio cuore: grazie quindi alla Vanaz e alla Ghiso.

Vorrei infine ringraziare alcuni amici, nonché colleghi, conosciuti in questi anni di Politecnico, con i quali ho condiviso progetti e fatiche: Andrea, Francesco, Jacopo, Lorenzo, Luca e Stefano.

Abstract

The main objective of this thesis is to provide an evaluation of the most important multi-party key exchange algorithms, that are generalizations of the more famous two-party Diffie-Hellman key exchange, proposed in the literature. This analysis is carried out not as a simple theoretical review, but with a very precise practical target: the application of multi-party key exchange algorithms to secure Building Automation Systems (BAS). This target is reached through implementing and testing various multi-party key exchange algorithms on a realistic benchmark, that simulates the devices commonly deployed on BAS. The experimental validation has provided significant indications about which are the best solutions that should be employed in this specific context.

Sommario

Il principale obiettivo di questa tesi è di fornire una valutazione dei più importanti algoritmi di scambio di chiavi fra più partecipanti, cioè generalizzazioni del più famoso scambio di chiavi Diffie-Hellman, proposti nella letteratura. Questa analisi non è limitata a una semplice rivisitazione teorica, ma ha un preciso scopo pratico: l'applicazione dei suddetti algoritmi alla messa in sicurezza dei Building Automation System (BAS). Questo obiettivo viene raggiunto attraverso l'implementazione e la verifica di vari algoritmi di scambio di chiavi generalizzati su una realistica piattaforma di test, che simula i dispositivi normalmente utilizzati nei BAS. La valutazione sperimentale messa in atto ha fornito indicazioni significative riguardo quale sia la miglior soluzione adottabile in questo contesto specifico.

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

During the last years there has been an increasing attention around the so called Building Automation Systems (BAS), mainly due to the advantages provided by modern IT solutions. BAS aim at improving control and management of mechanical and electrical systems in buildings. The main goal of a BAS is to provide increased comfort while keeping an efficient use of all available resources. It is important to specify that building automation systems are not limited only to the field of domotronics, i.e. simple home automation networks, with a relatively small number of devices for the remote control of appliances or features, that are becoming popular in modern luxury houses. Indeed these systems scale up to large installations with thousands of devices, covering a vital role in the functioning of important infrastructures such as office buildings, hospitals, data centers, and hotels. They typically connect building actuators and sensors to data networks, in order to realise distributed systems able to achieve control and monitoring of different infrastructure services (e.g. lighting, security alarms, HVAC).

The general trend is that of making these systems more and more pervasive and involved in critical aspects, therefore the security features offered by the network technologies underlying these infrastructures are of considerable importance. In this context there are a lot of realistic scenarios that constitute a serious threat and that must be avoided. Let us think for example to the case of an adversary, either an insider or on outsider one, able to gain access priviliges to the nodes of the BAS and thus subverting their normal functioning. The consequences may be of different severity, depending on the type of devices connected to the network. Unfortunately it happens that the technology behind these systems have been developed and deployed without security in mind. As an example of this fact, it is worth to mention a work presented this year at the famous Black Hat Briefings conference [14]. In this work the author shows the vulnerabilities of a building automation protocol, the KNX protocol, focusing on a real world example: the automation network of an hotel. The main reason behind these vulnerabilities is the complete lack of encryption, a characteristic shared by all the most important building automation protocols, that allows any kind of attack: from simple passive eavesdropping of packets circulating on the network, to more sophisticated attacks such as injection of forged packets or denial of service.

The introduction of strong cryptographic primitives in these protocols would allow to guarantee important security properties and thus thwarting the kind of attacks mentioned previously; in order to achieve this objective a crucial point is the use of a shared ephemeral key, agreed among all the nodes of the building automation system. Since many different key exchange protocols have been proposed in the academic literature, the purpose of this work is to provide a comprehensive study of the most relevant ones, focusing on their applicability to the specific context of building automation systems. This thesis provides both a theoretical analysis of these algorithms and also an experimental evaluation of their efficiency when implemented on embedded platforms, like those usually found in building automation systems.
Chapter 1 provides an overview of the most important building automation protocols, focusing on their functionalities especially from the security point of view.
Chapter 2 discusses the general problem under analysis and introduces the key agreement protocols that have been taken under consideration in this work.
Chaper 3 provides an overview of the theory behind elliptic curves, the cryptographic technology that has been chosen to implement the key agreement protocols.
Chapter 4 presents the experimental setup and summarizes the results obtained by this study.
Finally in Chapter 5 there are the conclusions derived from this work and the possible future developments.

Chapter 1

State of the Art

This chapter will describe the most important building automation protocols that have been considered in this study, describing their characteristic, strengths, weaknesses and security features. Particular attention will be dedicated to the first protocol, the KNX protocol, since it has been chosen as the reference protocol for the implementation of the key exchange algorithms that have been analyzed. After that three other protocols will be presented: LonWorks, BACnet and Modbus.

1.1 KNX

The KNX protocol [2] is a standardized (EN 50090, ISO/IEC 14543) network protocol, defined in 2002, specifically designed to operate in the context of building automation. KNX is the successor to three previous standards: the European Home Systems Protocol (EHS), BatiBUS, and the European Installation Bus (EIB or Instabus), it is based on the OSI model and it is adopted by more than 300 manufacturers from more than 30 countries. The standard supports different physical communication media, both wired and wireless ones. The decision of employing this protocol as the reference for the implementation relies on the fact that it is an open standard, quite widespread on the market, and on its tight connectivity constraints: indeed the maximum bandwidth available on wired means with KNX is of 19.2 kbit/s. In the following two sections we provide an overview of the technical details of the protocol and of the security features offered by it.

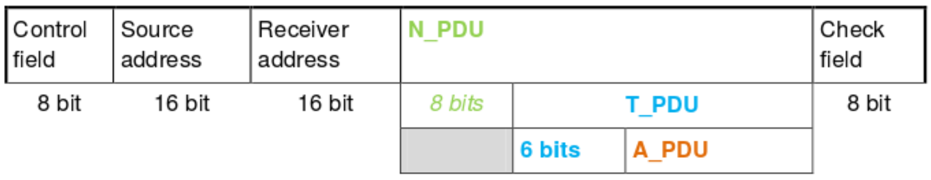


Figure 1.1: KNX datagram structure

1.1.1 KNX Datagram Overview

Figure 1.1 shows the complete structure of a standard KNX datagram; the different layers of the OSI model implemented in the protocol have been highlighted.

Layer 1 (Physical Layer) The principal aim of the services provided by layer 1 is to shield layer 2 from the physical means used for transmission. It is thus ensured that the upper layers of the network remain independent of the transmission physics used, so that the transmission media could be changed without having any effect upon the upper layers. As we have previously reported KNX supports multiple types of media both wired and wireless. In Chapter 4 we will describe how the physical layer was implemented in the tests.

Layer 2 (Data Link Layer) The data link layer comprises a header made of the control field, the sender and receiver addresses and a trailer field used to detect transmission errors. The control field contains information about the priority of the packet (KNX defines 4 different levels of priority) and if the packet is a repeated one or not. As it is possible to note from figure 1.1 the protocol features 16 bit long addresses, accomodating up to 65536 devices. An important characteristic of KNX is the support of both unicast and multicast communications, in fact, it is possible to specify if the receiver address has to be interpreted as a group address, i.e. an address identifying multiple devices, instead of a normal one referring to a single node. It is obviously possible to realise broadcast communication (since it is a particular case of multicast communication), simply setting to zero the receiver address. The trailer field is a bitwise parity code, computed bytewise on the entire message. CSMA/CA is used for bus access control. The KNX protocol specifies that telegrams should be immedeately acknowledged at the link layer, sending acknowledgment messages with a length of a single byte. Three different type of acknowledgement are defined: immediate acknowldge (IACK), immediate not acknowledge (INACK) and busy (BUSY).

Layer 3 (Network Layer) The header of the network layer is composed of a single byte coding different informations:

- The first bit indicates whether the receiver address needs to be interpreted as an individual address or a group address;
- The following three bits indicates the value of the routing counter, which will play a role in the routing of the telegram across the network;
- The following 4 bits indicate the length of the useful information in the telegram (actually the length of the transport protocol data unit).

Layer 4 (Transport Layer) The transport layer contains information about the type of communication. KNX distinguishes between two different types of communications: control data and data; the packets from each one of these communications can be optionally numbered, giving rise to four possible combinations, therefore the first two bits of the transport layer's header are devoted to the coding of the communication type. The next 4 bits indicate, only in the case of a numbered communication type, the sequence number.

1.1.2 KNX Security Features

As it is possible to note from the previous description, up to version 2.0, the standard KNX protocol doesn't offer any kind of explicit security feature. The datagram's payload is not encrypted, thus not providing the confidentiality of the communication. Furthermore there are no means to guarantee integrity (the check field provided by layer 2 can be easily bypassed by an attacker), authentication, or freshness of the messages. Although different solutions to secure this specific protocol have been proposed in the literature, e.g. [7], [12], none is actually in use. At the start of this year (2014) was presented, after four years from the previous release, the new version of the KNX standard: version 2.1. Unfortunately, even if KNX claims to be an open standard, the official KNX specifications are free for KNX Members, but they have to be ordered via the KNX Online Shop for the others.

1.2 LonWorks

LonWorks [11] is a network protocol developed by Echelon Corporation that has become a standard (ISO/IEC-14908). The LonWorks system comprises a specific communication protocol called LonTalk, a specific microcontroller, and a network management tool. LonWorks provides a simplified routed network with at most 255 subnets, each one able to address up to 127 nodes. Similarly to what we have seen for KNX, also this protocol supports multicast communications: indeed LonWorks defines 256 different group addresses. The medium access strategy employed in this protocol is CSMA/CD, allowing a maximum bandwidth of 1.25 Mbit/s.

This protocol is equipped with a security measure based on a single key shared among all devices. This key is used only to guarantee authentication of the sender (i.e. ensuring that the sender is a node of the network), through a challenge-reponse protocol. This protocol works as follows: the sender delivers a message setting the so called authentication bit. After having received this request, the receiver replies with a 64 bit random number (which is the challenge of the protocol). The sender receives this random number and calculates a 64 bit hash value over the content of the message and the random number itself using the shared secret key. This hash value (which is the response of the protocol) is sent back to the receiver that performs the same calculation comparing the result with the received value. The protocol can be trivially extended to the case of multicast communication (i.e. a sender authenticating to more nodes comtemporarily). It is clear that this security measure is insufficient, in fact there are some desirable security properties that cannot be granted, e.g. confidentiality: although authenticated the messages are sent in plaintext. Moreover this sort of membership check is rather rudimentary and suffers of a considerable number of flaws:

- The authentication is limited to the sender, the receiver is not verified;

- The challenge-response protocol can only be initiated by the sender, the receiver does not have the opportunity to demand secured requests;
- The length of the secret key is limited to 48 bits, clearly insufficient to prevent offline bruteforcing;
- There is a complete lack of a secure mechanism to distribute the shared key or to change it.

1.3 BACnet

BACnet [15] is a protocol stack devised in 1987 by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) that became an ISO standard in 2003 (ISO-16484-5). The BACnet specification is under continous development, extensions to the current BACnet standard are summarized in specific documents called BACnet Addenda. The main characteristic of this protocol is its flexibility: it allows to choose which network transport stack should be used for its commands. More precisely, the BACnet protocol defines a number of data link/physical layers, such as: ARCNET, Ethernet, BACnet/IP, Point-To-Point over RS-232, Master-Slave/Token-Passing over RS-485, and the LonTalk protocol discussed one section above.

From the point of view of security the situation has changed since the publication of the so called "Addendum g" (one of the aforementioned BACnet addenda). BACnet Addendum g replaces the old and vulnerable security concept of BACnet. The update from Addendum g makes it similar to what has been proposed for LonWorks, i.e., it is based on the establishment of common keys shared between all the nodes, while the security desiderata are guaranteed through the use of symmetric key ciphers. The required shared secret keys have to be distributed to the devices in advance, or they have to be retrieved from a key server during runtime. The latter solution, which does not employ any key exchange protocol, is not ideal for multiple reasons:

- First of all the use of a single key server introduces a single point of failure. It would be possible to use multiple key servers, at the cost of an increased complexity of the overall system, but the standard doesn't provide this possibility.

- If the distribution of the keys is handled by the key server, it happens that the distribution to the devices sets up in advance which of them are able to communicate with each other in a secure way and which are instead prevented in doing so. To allow more flexible settings it would be necessary to introduce multiple keys, but this option is not present in the specification.
- The use of asymmetric schemes, like the ones studied in this work, can avoid the need for a trusted, online key server.

1.4 Modbus

Modbus [10] is a serial communication protocol developed by Schneider Electric in 1979 in order to be used on the programmable logic controllers (PLCs) produced by the same company. Modbus is based on an underlying transport protocol, typically a TCP/IP stack (even if other options are possible). Communications are usually realised over Ethernet, thus allowing both unicast and multicast communications. The use of Ethernet implies that the access strategy to the shared medium is CSMA/CD and a high maximum transmission speed is available (up to 1 Gbit/s). This protocol is a sort of de facto standard in industrial applications, due to its simplicity and robustness and also to the facts that is openly published and free of royalties. Unfortunately, since it was developed in the late '70 and specifically tailored to industrial applications, it doesn't offer any kind of security features.

Chapter 2

Problem Description

This chapter is devoted to a general description of the problem of securing a Building Automation System and the associated Building Automation Network (BAN). In particular we will focus on the threat model that is reasonable to assume in this type of systems and on the security properties that we are interested in. After that, we will provide a description of the key agreement protocols that have been studied

2.1 Threat Model and Security Desiderata

To provide a sound analysis of the security of BAS it is important to describe the threat model taken under consideration, highlighting the security properties we are interested in. As we have already said the typical configuration of a BAS consists of nodes (sensors and actuators) communicating on a shared medium. In the following we specify the security desiderata needed in this particular context:

Confidentiality If the communication between the nodes happens in cleartext, as it is the case in the aforementioned protocols, it is reasonable to assume that an attacker may be able to eavesdrop it. The sensitivity of the informations circulating on the BAN is related to the attached devices, but it is not difficult to imagine cases where confidentiality is a desirable property, e.g, at commands directed to door control systems or video streaming coming from security cameras.

Authentication Considering that the common medium is usually easily accessible (via direct connection to the wires or generation of EM transmissions), it is reasonable to assume an attacker able to inject arbitrary messages on the BAN. This is another serious threat, basically generated by the lack of an authentication, that can lead to the unwanted scenario of an attacker able to send commands to the actuators of critical buildings.

Message integrity Even without injecting forged packets, an attacker may be able to interfere with the normal network functioning, simply tampering with legitimate packets. Therefore it is important to assess the integrity of the circulating messages.

Replay attacks The purpose of these kind of attacks is to force the system into doing something simply recording normal messages and repeating them later on. Considering the fact that messages containing actuator commands that cause physically evident actions (e.g. open a door, turning on/off ligths) can be fully inferred from an attacker through environmental observations, these attack strategies may be a valid option for a malicious adversary. Therefore the freshness of the messages must be granted by some means.

Forward secrecy A system is said to have forward secrecy, if the compromise of a long-term private key (at some point in the future) does not compromise the security of communications made using that key in the past. Since an attacker may be able to acquire a BAN node and to retrieve the long term key contained in it, thus compromising the confidentiality of previously recorded communications, the property of forward secrecy is relevant in order to mitigate this issue.

Availability Furthermore an attacker may try to compromise the availability of nodes realizing some sort of denial of service attack. In general it is important to detect when a node is no longer available, either to counteract a possible attack or to take appropriate maintenance actions, should the disappearance be caused by a node failure.

2.2 Securing BAS

After having described the threat model and stated the security desiderata, we must specify the security services and cryptographic technologies that are adequate in this context. Moreover it is important to consider that the solution must comply with another requirement: the tight computational and communication constraint imposed by the target devices commonly deployed in BANs.

A possible solution would be to apply the well-known Kerberos protocol [19], which guarantees mutual authentication, confidentiality and protection from replay attacks. This protocol is built on fast symmetric key cryptography, a characteristic that fits nicely the aforementioned computational constranints. There are multiple reasons for which, even if it is a viable option, the Kerberos protocol has been discarded in favor of other solutions:

- Kerberos cannot provide all the security desiderata stated above. For example, the availability of the nodes of the BAN or the forward secrecy property are not granted.
- One of the main disadvantages of Kerberos is that it has a single point of failure. It requires continuous availability of a central server; when this server is down the system is locked.
- Even if asymmetric key primitives are less efficient, with the introduction of Elliptic Curve Cryptography (ECC) it is possible to use asymmetric algorithms even on low-end embedded devices.

All these reasons point towards the adoption of an ad-hoc solution employing lightweight key agreement schemes to derive ephemeral cryptographic keys, to be used with fast symmetric-key primitives.

Antonini, Barenghi and Pelosi, in a recent work published in 2013 [1], have designed a complete protocol, based on a key agreement scheme, that is able to provide the aforementioned properties. The protocol, whose messages are depicted in Figure 2.1, is organized in three phases: bootstrap, regular functioning and key refreshment. During the bootstrap phase the value of the secret ephemeral cryptographic key is set by the installer to the same value for all nodes. Subsequently, the nodes are forced to perform a multi-party key agreement procedure to determine the first value of the shared

	Mgmt	Cmd	Ack	Command	Nonce	MAC
Beacon	*0*	*0*	*X*	*Beacon*	*nonce*	$\mathcal{T}$
BAN Command	*0*	*1*	*0*	*BAN command*	*nonce*	$\mathcal{T}$
BAN Notification	*0*	*1*	*1*	*Delivery Notification*	*nonce*	$\mathcal{T}$
Key Agreement	*1*	*0*	*X*	*Key Agreement Message*		$\mathcal{T}$
Add Node Command	*1*	*1*	*0*	*Add Node Command*	*nonce*	$\mathcal{T}$
Add Node Notification	*1*	*1*	*1*	*Delivery Notification*	*nonce*	$\mathcal{T}$

Figure 2.1: Description of the messages in the secure protocol proposed by Antonini, Barenghi and Pelosi

ephemeral symmetric key to be used. Once this key agreement is completed, we enter in the regular functioning mode. The confidentiality requirement is satisfied by encrypting the whole payload with a strong symmetric cipher, exploiting the shared key. The integrity of the messages is granted by appending at the end of each message a Message Authentication Code (MAC) (indicated as $\mathcal{T}$ in Figure 2.1), which prevents an attacker from both tampering with the contents of messages and injecting forged ones. To ensure the availability of the nodes the transmission time is logically divided into time slots, scheduled to the nodes according to the lexicographical order of their respective addresses. Each node willing to transmit will have to wait the start of the corresponding time slot, in case nothing has to be transmitted a beacon message will be sent; in this way any disruption of the transmission of a node can be promptly detected. Finally, the forward secrecy is ensured by the last phase of the protocol: the rekeying. Indeed, after a predetermined number of exchanged messages, the network's members will engage in a new key agreement procedure to renew the shared epeheral secret key.

The central point around which the entire protocol is built is the key agreement algorithm. It has to be both sound, i.e., it should not be broken by a motivated adversary, and efficient since the procedure will be repeated multiple times throughout the BAS lifecycle. In the following section we will provide a thorough analysis of the most important multi-party key agreement protocols described in the literature.

2.3 Key Agreement Protocols

This section will provide a theoretical insight of the group key agreement protocols that have been considered in this work. Since all the algorithms that will be covered are an attempt to generalize the Diffie-Hellman key exchange, it is worth to analyze this protocol. After that we will introduce the key exchange algorithms proposed by Burmester and Desmedt, the algorithms of Steiner et al. and the MKA algorithms devised by Antonini, Barenghi and Pelosi.

2.3.1 Diffie-Hellman Key Exchange

Algorithm 2.3.1: Diffie-Hellman Key Exchange

```
Globals: $(G, \cdot) = \langle g \rangle$, finite cyclic group with order $n = |g|$
Output: Shared secret: $k = g^{a_0 a_1}$
begin
    Private ephemeral key: $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0, 1\}$
    Public ephemeral key: $k_{pub,i} = g^{a_i}$
    Send $k_{pub,i}$ to the other peer
    Receive $k_{pub,j}$
    $k = {k_{pub,j}}^{a_i} = g^{a_0 a_1}$
    return $k$
```

This scheme was first published by Whitfield Diffie and Martin E. Hellman in 1976 [9]. The Diffie-Hellman key exchange protocol allows two parties that have no knowledge of each other to determine a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications (e.g. using a symmetric key cipher).

As shown in Algorithm 2.3.1 each peer randomly selects[1] a private ephemeral key $a_i \in \{2, \ldots, n-1\}$ and then computes and sends to the other one a public key $k_{pub,i} = g^{a_i}$. Each participant completes the key exchange exponentiat-

[1]When we say "randomly select" we always intend that the values are selected indepently and with an uniform distribution

ing the public key of the other peer with its own private key:

$$k = k_{pub,j}{}^{a_i} = g^{a_0 a_1}$$

The security of the Diffie-Hellman scheme against passive adversaries, i.e. that can only passively eavesdrop messages over the communication channel, is based on a supposedly computationally hard problem, the Computational Diffie-Hellman problem:

Definition 2.1 (CDHP)**.** Let $(G, \cdot)$ be a finite cyclic group with order $n = |G|$, and denote as $g \in G$ one of its generators: $(G, \cdot) = \langle g \rangle$. Given two positive integers $a_0,\ a_1 \in \mathbb{Z}_n \setminus \{0, 1\}$, the Computational Diffie Hellman Problem (CDHP) is defined as the one of computing g, a_0, a_1, given as input (G, g, g^{a_0}, g^{a_1}).

The Diffie-Hellman protocol is not effective against active adversaries, i.e. that can put themselves in the middle of the communication channel performing a so called Man In The Middle Attack. The reason for this is that the Diffie-Hellman protocol provides a non-authenticated key exchange, therefore authenticantion should be provided by some other mean. This characteristic is shared by all the other key exchange protocols that will be presented here. Another important property that is ensured by the Diffie-Hellman protocol is forward secrecy, that has been already cited in the previous section, we report here its formal definition:

Definition 2.2 (Forward secrecy)**.** A system is said to have forward secrecy, if the compromise of a long-term private key (at some point in the future) does not compromise the security of communications made using that key in the past.

The multi-party key agreement protocols that will be introduced in the following sections can be roughly divided into two cathegories:

- Those that are natural generalizations of the Diffie-Hellman key exchange, i.e, the algorithms proposed by Steiner et al. and the MKA of Antonini, Barenghi and Pelosi;
- Those that are not natural generalizations of the Diffie-Hellman key exchange, i.e. the algorithms proposed by Burmester and Desmedt.

Algorithm 2.3.2: Burmester-Desmedt Key Exchange (BD1)

```
Globals: $(G, \cdot) = \langle g \rangle$, finite cyclic group with order $n = |g|$
Input: $\mathcal{U} : \{U_0, \ldots, U_{t-1}\}$ ordered set of protocol partecipants
Output: Shared secret: $k = g^{a_0a_1+a_1a_2+\ldots+a_{t-1}a_0}$
begin
    $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0, 1\}$
    $z_i = g^{a_i}$
    Broadcast $z_i$
    Receive $z_{i-1}$ and $z_{i+1}$
    $X_i = \left(\frac{z_{i+1}}{z_{i-1}}\right)^{a_i}$
    Broadcast $X_i$
    for $U_j \in U | U_j \neq U_i$ do
        receive $X_j$
    $k = z_{i-1}^{ta_i} \cdot X_i^{t-1} \cdot X_{i+1}^{t-2} \cdot \ldots \cdot X_{i-2}$
    return $k$
```

A multi-party key exchange algorithm is considered as a natural generalization of the Diffie-Hellman key exchange if it satisfies the following definition:

Definition 2.3. A multi-party key exchange is defined as a natural generalization of the Diffie-Hellman key exchange if all the partecipants compute the key:

$$k = g^{\prod a_i}$$

where a_i is the private ephemeral key chosen by the generic i-th partecipant.

2.3.2 Burmester-Desmedt Key Exchange

Mike Burmester and Yvo G. Desmedt proposed two different systems to perform key exchange in a group of $t \geq 2$ partecipants. The first solution is described in [5], while the second more recent solution in [6]. These two schemes will be treated separately in the following subsections.

First Burmester-Desmedt Key Exchange (BD1)

An important prerequisite of this algorithm is that the partecipants should be ordered and each partecipant should be aware of this ordering. As specified by Algorithm 2.3.2, given a publicly known finite cyclic group $(G, \cdot) = \langle g \rangle$ with $n = |g|$ elements, each partecipant randomly selects a private ephemeral key $a_i \in \{2, \ldots, n-1\}$ and computes $z_i = g^{a_i}$. The first round of the algorithm is completed broadcasting z_i to all the other nodes. After that, in the second round of the algorithm, each partecipant, after having received the exponentiations of the preceding node z_{i-1} and of the following node z_{i+1}, computes and broadcasts $X_i = \left(\frac{z_{i+1}}{z_{i-1}}\right)^{a_i}$ [2]. After having received all the other values each partecipant will compute the shared key k as: $k = z_{i-1}^{ta_i} \cdot X_i^{t-1} \cdot X_{i+1}^{t-2} \cdot \ldots \cdot X_{i-2}$. Following this procedure all the nodes will compute the same key:

$$k = g^{a_0 a_1 + a_1 a_2 + \ldots + a_{t-1} a_0}$$

Indeed, set:

$$A_{i-1} = (z_{i-1})^{a_i} = g^{a_{i-1} a_i}$$

$$A_i = {z_{i-1}}^{a_i} \cdot X_i = g^{a_i a_{i+1}}$$

$$A_{i+1} = {z_{i-1}}^{a_i} \cdot X_i \cdot X_{i+1} = g^{a_{i+1} a_{i+2}}$$

$$\ldots$$

We have that the key computed by the i-th partecipant can be expressed as:

$$k_i = A_{i-1} \cdot A_i \cdot A_{i+1} \cdot \ldots \cdot A_{i-2}$$

From the above relation it follows that the key is a second order cyclic function of the a_i. It is interesting to note that for $t = 2$ we get $X_1 = X_2 = 1$ and $k = g^{2a_0 a_1}$, which is not the same key computed through the Diffie-Hellman key exchange, since this protocol, as stated previously, is not a natural generalization of it. In their work the authors prove that this scheme is secure against a passive adversary (i.e. it provides confidentiality). As noted discussing the Diffie-Hellman key exchange, this protocol, by itself, doesn't guarantee authentication.

From the point of view of efficiency we note two nice properties:

- The number of rounds of the protocol doesn't depend on the number of nodes involved (indeed it is constant and equal to 2);
- The exponentiations (which are the most expensive operations) are "cheap". Indeed each node will perform $t + 1$ exponentiations, but

[2]The indexes are taken modulo the number of partecipants

apart from three of them the exponent is at most $t-1$.

The efficiency of this scheme comes to a price, indeed, as we have stated before, the partecipants must be ordered and each partecipant must be aware of this ordering. Moreover there is another important practical aspect to consider: during the protocol the communications are performed using simultaneous broadcasts by all the nodes, however the ability to perform n simultaneous broadcasts is not a feature available in most network environments. Therefore a more realistic version of this algorithm would be obtained imposing that each broadcast is performed separately from the others, thus giving rise to $2t$ rounds instead of only 2.

Second Burmester-Desmedt key exchange (BD2)

In their second work Burmester and Desmedt try to enhance the previous one describing a new, more efficient protocol. The characteristic of this scheme is that it is based on a strong precondition: a spanning tree of the network to which the partecipating nodes are connected must be available. The main idea of this protocol is that a single node, corresponding to the root of the spanning tree, establishes the shared key that is then propagated to all the other nodes. The authors develop two different versions of the same algorithm, a sequential version in which the key is propagated through one level of the tree at a time and a multicast version where the key propagation is parallelized. In order to show how is it possible to propagate the key selected by the root node to all the other partecipants, let us consider the first algorithm proposed by the authors. First of all each node performs a key exchange with its adjacent nodes (e.g. through a Diffie-Hellman key exchange), let k_{ij} be the key exchanged between two generic nodes. Starting from the top of the tree (i.e. from the root, which is the node responsible for starting the propagation), each node sends to its adjacent nodes the encrypted key, computed in this way: $X_{ij} = K \cdot k_{ij}$. Trivially each adjacent node will retrieve the key simply calculating: $K = X_{ij} \cdot {k_{ij}}^{-1}$.

As we have already said, this algorithm requires the availability of a spanning tree of the underlying network. Since this is not a feature available in most building automation protocols, we consider a version of this algorithm that works without the spanning tree. Actually what we obtain dropping

Algorithm 2.3.3: Burmester-Desmedt Key Exchange (BD2)

```
Globals: $(G, \cdot) = \langle g \rangle$, finite cyclic group with order $n = |g|$
Input: $\mathcal{U} : \{U_0, U_1, \dots, U_{t-1}\}$ set of protocol partecipants
Output: Shared secret: $k$
begin
    $k \overset{random}{\longleftarrow} G$, $U_0$ selects the key
    $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0, 1\}$
    $z_i = g^{a_i}$
    $U_i$ sends $z_i$ to $U_0$, $U_0$ broadcasts $z_0$
    $U_0$ computes $k_{0i} = z_i^{a_0}$, $U_i$ computes $k_{0i} = z_0^{a_i}$
    for $i = 1$ to $t - 1$ do
        $X_i = k \cdot k_{0i}$
        Send $X_i$ to $U_i$
    $k = X_i \cdot {k_{0i}}^{-1}$
    return $k$
```

this hypothesis is a special case of the more general algorithm proposed by the authors. Indeed, without a spanning tree, the network can be considered as having a star topology, where the center of the star is the root node and all the other partecipants are adjacent to it. The resulting procedure is represented in Algorithm 2.3.3, similarly to what happens in the Diffie-Hellman key exchange each node computes a private-public keypair. All the child nodes send their public key to the root node U_0, while U_0 broadcasts its own public key. After this first phase the central node computes the shared key with each one of the adjacent nodes and uses it to encrypt the key that it has previously chosen. Each adjacent node, after having received the encrypted key, will be able to retrieve the original one applying the procedure discussed before.

The evident drawback of this protocol is that the root node has to perform $t-1$ more operations with respect to the other ones. This is a consequence of not constructing the spanning tree as supposed by the original algorithm, the usefulness of the spanning tree is exactly that of balancing the computational load between all the partecipants. Another more practical aspect to consider, is that all the nodes send their public keys to the central node at the same time. Similarly to what we have seen for the previous protocol, when we stated that it is difficult to admit the possibility of having simultaneous broadcasts by all the nodes, it is not possible in a practical implementation

```
Algorithm 2.3.4: Generalized Diffie-Hellman 2 (GDH2)
  Globals: $(G, \cdot) = \langle g \rangle$, finite cyclic group with order $n = |g|$
  Input: $\mathcal{U} : \{U_0, U_1, \ldots, U_{t-1}\}$ set of protocol partecipants
  Output: Shared secret: $k$
  begin
      1. Receive set of values from the preceding node
      2. Compute the new cardinal value, exponentiating the old one for $a_i$
      3. Construct new set of values
      4. Forward output values to the successive node
      5. Waiting for the final set of values from member $U_{t-1}$
      6. Compute shared key
```

to perform $n-1$ simultaneous unicast communications towards a single node, because most network architectures cannot receive more thant one packet at the same time. The solution to this issue is again to perform this transmission in distinct moments, thus increasing the number of rounds of the protocol.

2.3.3 Generalized Diffie-Hellman

This section is dedicated to the presentation of an entire family of protocols, called generalized Diffie-Hellman, introduced by the work of Steiner et al. [18]. As suggested by the name these protocols are natural extensions of the 2-party Diffie-Hellman to the n-party case. The definition of family of protocols is due to the fact that the authors propose a general protocol, from which it is possible to derive different schemes; three of these schemes are then explained in their work. As in all the other protocols, the parteciptans $U_0, U_1, \ldots, U_t$ agree on a finite cyclic group $(G, \cdot) = \langle g \rangle$ of order $n = |g|$ and choose a private key $a_i \in \{2, \ldots, n-1\}$. The main idea on which all these protocols are based is that given $g^{a_0 \cdots a_{i-1} a_{i+1} \cdots a_{t-1}}$ member U_i can easily compute the shared key $k = g^{\prod a_i}$ simply "adding", through exponentiation, its own private key. We have taken under consideration the second algorithm proposed by the authors, basing this choice on the fact that it is the most interesting from the point of view of efficiency.

In the first phase of the protocol the i-th node receives i values from the preceding one. The first value in this set is the so called cardinal value, i.e. a value containing $i-1$ exponents, while all the other values contain $i-2$ exponents. The recipient has to compose a new set of $i+1$ values, containing one cardinal value with i exponents and i values with $i-1$ exponents, and to forward it to the successive node. At the end U_{t-1} receives a set of t values, the first one, that is the cardinal value, has the form $g^{a_0 a_1 \cdots a_{t-2}}$. Therefore the last node is the first one able to compute the shared key, simply exponentiating the received cardinal value to its own exponent. After that, it will process all the remaining $t-1$ values, adding to each one of them its own exponent, and finally broadcasts this batch of values to the other nodes. The remaining group members will be able to compute the shared key, selecting the appropriate value and exponentiating it to their own exponent. As a running example, let's consider a 4 nodes BAN, $\mathcal{U} = \{U_0, U_1, U_2, U_3\}$:

1. Node 0 begins the key exchange by sending to node 1 its own public value: $\{g^{a_0}\}$
2. Node 1 sends to node 2: $\{g^{a_0 a_1}, g^{a_0}, g^{a_1}\}$
3. Node 2 sends to node 3: $\{g^{a_0 a_1 a_2}, g^{a_0 a_1}, g^{a_0 a_2}, g^{a_1 a_2}\}$
4. Node 3 will compute the shared key using the cardinal value: $g^{a_0 a_1 a_2}$ and then it computes and broadcasts $\{g^{a_0 a_1 a_3}, g^{a_0 a_2 a_3}, g^{a_1 a_2 a_3}\}$
5. Each node will compute the shared key from the appropriate intermediate value

An interesting characteristic of this protocol is the minimal number of messages exchanged between the nodes, indeed only t messages are sent. It is easy to see that at least t messages are required in any group key agreement protocol, i.e. each partecipant has to contribute to the key exchange with at least one message. Moreover in this protocols, unlike to what we have seen in the previous ones, there are neither simultaneous broadcasts nor simultaneous unicasts. A distinguishing characteristic of this protocol with respect to the previous ones, that can be considered a drawback, is that the message size is not constant but grows linearly; indeed each node adds a new value to the ones it has received.

2.3.4 Antonini-Barenghi-Pelosi Multi-party Key Agreement (MKA)

Algorithm 2.3.5: Multi-party Key Agreement (MKA)

```
Globals: $(G, \cdot) = \langle g \rangle$, finite cyclic group with order $n = |g|$
Input: $\mathcal{U} : \{U_0, U_1, \dots, U_{t-1}\}$ set of protocol partecipants
Output: Shared secret: $k = g^{\prod_{i=0}^{t-1} a_i}$
begin
    $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0, 1\}$
    $k_{U_i} = g$
    for $r = 0$ to $t - 2$ do
        $k_{U_i,r} = (k_{U_i})^{a_i}$
        Broadcast $k_{U_i,r}$
        $tmp = \prod_{U_j \neq U_i} k_{U_j}$
        $k_{U_i} = (tmp \cdot (k_{U_i})^{-r})^{\frac{1}{r+1}}$
    $k = k_{U_i}^{a_i}$
    return $k$
```

The MKA algorithm [1] has been specifically designed for the case of BAN networks. This protocol is another example of true generalization of the Diffie-Hellman key exchange, indeed each partecipant will select a private ephemeral key a_i and, after the protocol execution, will compute the key $g^{\prod_{i=0}^{t-1} a_i}$. The first step of the protocol, as shown in Algorithm 2.3.4, consists in the selection by each partecipant of a private ephemeral key $a_i \in \{2, \dots, n-1\}$ and initialization of the local shared secret k_{U_i} to the group generator g. After that the protocol repeats the same steps for $t-1$ rounds. During each round, every partecipant broadcasts the result of the exponentiation ${k_{U_i}}^{a_i}$ and then waits for the $t-1$ values computed in the same way by the other nodes. All the received values are accumulated in a temporary variable. The last operation performed in each round updates the value of the local shared secret k_{U_i}, to both combine the secret exponents of the other parties and remove the dependence from the local secret a_i. At the end of the loop the value of the shared common secret for the $i-th$ partecipant will be equal to: $g^{a_0 a_1 \cdots a_{i-1} a_{i+1} \cdots a_{t-1}}$, therefore the shared key can be computed simply adding the i-th ephemeral key a_i, computing ${k_{U_i}}^{a_i}$. Theorem 2.3.1 formalizes the correctness of the algorithm.

Theorem 2.3.1. ***(Algorithm Correctness)*** *Given a finite cyclic group $(G, \cdot)$ with generator $g \in G$ and order $n = |g|$, each partecipant U_i in a set $U = \{U_0, U_1, \ldots, U_{t-1}\}$ runs Algorithm 2.3.4 to compute a shared ephemeral secret $k = g^{\prod_{i=0}^{t-1} a_i}$, where each $a_i \in \{2, \ldots, n-1\}$ denotes the ephemeral secret of the i-th partecipant.*

In order to tackle to demonstrate the security provided by this algorithm, the authors introduce a definition of Computational Multi-party Key Agreement Problem, which basically is a generalization of the Computational Diffie Hellman Problem defined in 2.1.

Definition 2.4 (CMKAP)**.** Let $(G, \cdot)$ be a finite cyclic group of generator $g \in G$ and order $n = |g|$. Given an integer $t \geq 2$ and a set of values $a_i \in \{2, \ldots, n-1\}$, $0 \leq i \leq t-1$, the Computational Multi-party Key Agreement is defined as the one of computing $k = g^{\prod a_i}$ given all the values $k_{U_i,r}$ by Algorithm 2.3.4 for all $0 \leq r \leq t-1$.

The security of the protocol is then verified by the following theorem, that states the equivalence between the CDHP and the CMKAP.

Theorem 2.3.2. ***Turing Equivalence of CDHP and CMKAP*** *Solving the CMKAP problem for $t \geq 2$ participants is polynomially equivalent to computing the solution of the Computational Diffie-Hellman problem on the same finite cyclic group.*

The demonstration of Theorem 2.3.2 and Theorem 2.3.1 is provided in Appendix A.

Table 2.1: Protocol Comparison. n indicates the number of protocol partecipants.

	BD1	**BD2**	**GDH2**	**MKA**
round	2	$2(n-1)$	n	$n-1$
tot. messages	$2n$	$3(n-1)$	n	$n(n-1)$
sent messages per U_i	2	1 $2n-1$ for U_0	1	$n-1$
rec messages per U_i	$n+1$	2 $n-1$ for U_0	2 1 for U_1 and U_n	$(n-1)^2$
exponentiations per U_i	$n+1$	2 $2(n-1)$ for U_0	$i+1$	$3n-2$
exponentiations	$n(n+1)$	$4n-2$	$\frac{(n+3)n}{2}-1$	$n(3n-2)$
symmetry	Y	N	N	Y

2.4 Protocol Comparison

In order to compare the key exchange algorithms, we now introduce a set of metrics which are relevant to the evaluation:

Rounds: The number of rounds is important in order to evaluate the time complexity of the protocol.

Total messages: The number of messages exchanged is a relevant characteristic, especially in network with a limited bandwith like KNX.

Messages sent and received The number of messages sent and received by a node is important in relation with the limitation of the underlying network technology.

Exponentiations: Exponentiations are by far the most expensive algebric operations, therefore it's important to minimize them.

Symmetry: Symmetry, i.e. all the nodes performing the same operations, is another desirable property that can simplify the deployment of a protocol.

Table 2.1 summarizes the scores of the discussed protocols with respect to the above metrics. Considering the number of rounds, BD1 comes out as the clear winnder from the point of view of protocol efficiency. Unfortunately, as we have previously pointed out, its efficiency is based on the assumption of being able to perform n simultaneous broadcasts. If we drop this hypothesis (which is reasonable in practical applications), the protocol doesn't compare with the others as favorably as before. The time complexity may be significantly influenced by the number of exponentiations, which is the heaviest arithmetic operation. As it is possible to see all the algorithms seem to have a similar number of exponentiations, one may argue that BD2 and GDH2 are better than the others, but in these algorithm the computational load is not fairly balanced, a characteristic that may result in loss of performance. Finally, considering the number of messages, GDH2 appears to be the best protocol (i.e. the one producing the lowest number of messages). It is important to remark that in this algorithm the dimension of a message is not constant, but grows during the execution of the key exchange, because the number of group elements exchanged increases linearly.

Chapter 3

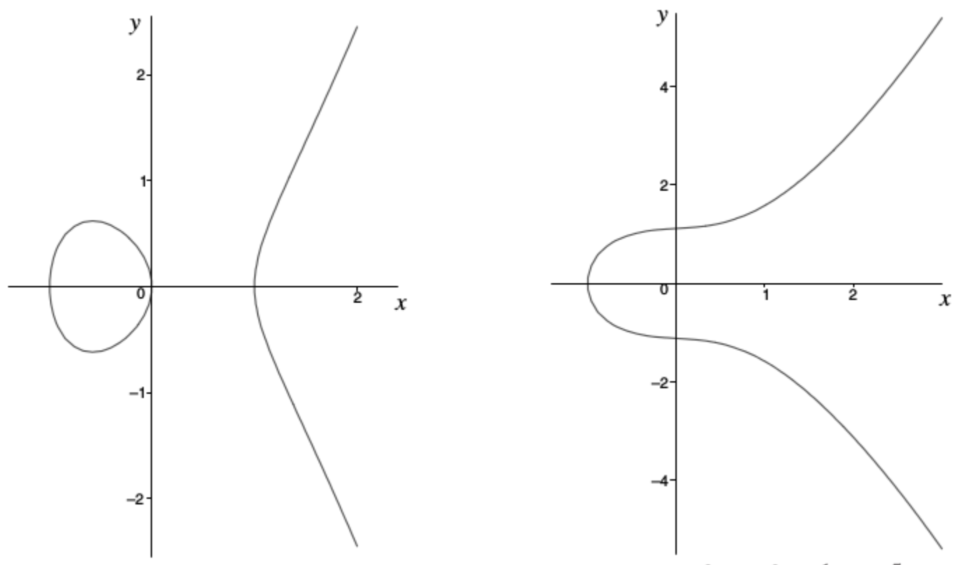
Adoption of ECC in Key Agreement Protocols

A critical aspect highlighted in the previous chapters is that BAS are characterized by very tight computational constraints, therefore the proposed security measures, including the key agreement schemes, should be implemented in an efficient way. Considering the state of the art of cryptographic technologies, Ellicptic Curve Cryptography (ECC) [8] is the best solution in terms of both security margin and efficiency. In the first part of this chapter we will analyse elliptic curve arithmetic, giving an introductory explanation of the underlying theory, while in the second part we will see how the key agreement protocols are defined using ECC.

3.1 Elliptic Curve Cryptography

ECC was introduced in 1985 by Victor Miller and Neal Koblitz, but it was only around the late 1990's that elliptic curve systems started being employed widely on a commercial scale, due to the introduction of standardized elliptic curve protocols. The security of elliptic curve cryptosystems is based on the discrete logarithm problem (ECDLP) in a finite cyclic group composed by the points of an elliptic curve. An important characteristic is that the ECDLP has fully exponential computational complexity, provided that a proper choice of the curve parameters and base fields are made, in turn means that, given a certain security margin ECC requires significantly smaller key size with respect to other public key cryptosystems, such as RSA or discrete logarithm defined over $\mathbb{F}_{p^m}$. The limited key size is beneficial for both the computational load and the storage space required by ECC algorithms, making this solution ideal for embedded systems.

3.1.1 Elliptic Curves



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Figure 3.1: Elliptic curves over $\mathbb{R}$

Definition 3.1. An elliptic curve $\mathbb{E}$ over a field $\mathbb{K}$ is defined by an equation:

$$\mathbb{E}(\mathbb{K}) = \{(x, y) \in \mathbb{K}^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{K}\}$$

where the curve has to be non-singular, i.e. every point of the curve should have a unique tangent line.

The definition is given for a generic field $\mathbb{K}$, indeed in figure 3.1 we have the example of two elliptic curves defined over the field of real numbers $\mathbb{R}$. However, for cryptographic purposes, only elliptic curves over finite fields $\mathbb{F}_{2^m}$ and $\mathbb{F}_{p^m}$ are considered. The above equation, also called Weierstrass equation, can be simplified using an admissible change of variables:

- if $\mathbb{K} = F_{p^m}$, the reduction is performed using the change of variables:

  $$(x, y) \rightarrow \left(\frac{x - 3{a_1}^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{{a_1}^3 + 4a_1a_2 - 12a_3}{24} \right)$$

  transforming the curve equation into:

  $$\mathbb{E}(\mathbb{F}_{p^m}) : y^2 = x^3 + ax + b \quad a, b \in \mathbb{F}_{p^m}, p > 3$$

  The non-singularity constraint can be imposed as:

  $$\Delta = 4a^3 + 27b^2 \neq 0$$

- if $\mathbb{K} = F_{2^m}$ and $a_1 \neq 0$, the reduction is performed using the change of variables:

  $$(x, y) \rightarrow \left({a_1}^2x + \frac{a_3}{a_1}, {a_1}^3y + \frac{{a_1}^2a_4 + {a_3}^2}{{a_1}^3} \right)$$

  transforming the curve equation into:

  $$\mathbb{E}(\mathbb{F}_{2^m}) : y^2 + xy = x^3 + ax^2 + b \quad a, b \in \mathbb{F}_{2^m}$$

  Such a curve is called non-supersingular and the non-singularity constraint can be imposed as:

  $$\Delta = b \neq 0$$

3.1.2 Group Law

In order to define a cryptosystem based on the ECDLP, the set of points of the elliptic curve should form a proper algebraic structure (i.e. a cyclic group). The simplest structure that satisfies the group properties is defined as $(G, +)$, where G is a set of curve points and the internal composition law, i.e. the + operation, is the so called chord and tangent rule. Together with this addition operation, the set of points $\mathbb{E}(\mathbb{K})$ forms an additive abelian group.

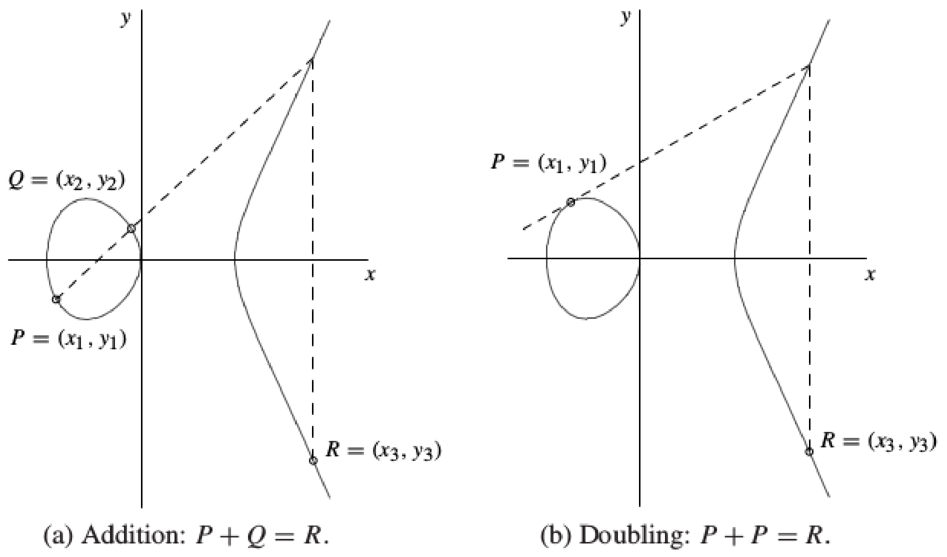


Figure 3.2: Geometric addition and doubling of elliptic curve points

Definition 3.2 (Chord and Tangent Rule)**.** Let $P, Q \in \mathbb{E}(\mathbb{K})$ be two points and $r = \overline{PQ}$ be the straight line passing through P and Q (or the tangent line if $P = Q$). Denote with R the third point intercepted by the straight line $r = \overline{PQ}$ (elliptic curves can be expressed as $y^2 = f(x)$, therefore being $f(x)$ a 3rd degree polynomial it will always intercept a straight line in at most 3 points). Denote with $r' = \overline{R\mathcal{O}}$ the vertical line passing through R (and $\mathcal{O}$, which is the only point on the line at infinity that satisfies the projective form of the Weierstrass equation). The sum of the two points P and Q, denoted as $S = P + Q$, is defined to be the third point intercepted by r' over $\mathbb{E}(\mathbb{K})$.

Figure 3.2 shows an example of the geometric interpretation of the composition law on the elliptic curve $\mathbb{E}(\mathbb{R}) : y^2 = x^3 - x$. The generic formulae to apply the composition law on curves defined over $\mathbb{F}_{p^m}$ and $\mathbb{F}_{2^m}$ are provided in Appendix B.

It is easy to show that the group properties are satisfied by the algebraic structure composed by the points of a generic elliptic curve $\mathbb{E}(\mathbb{K})$ and the chord and tangent rule:

1. Associativity. $\forall P, Q, R \in \mathbb{E}(\mathbb{K}), \quad P + (Q + R) = (P + Q) + R$
2. Identity. $\forall P \in \mathbb{E}(\mathbb{K}), \quad P + \mathcal{O} = \mathcal{O} + P = P$
3. Inverse. $\forall P \in \mathbb{E}(\mathbb{K}), \quad \exists! Q = (-P) \in \mathbb{E}(\mathbb{K}) : P + Q = \mathcal{O}$
4. Commutativity. $\forall P, Q \in \mathbb{E}(\mathbb{K}), \quad P + Q = Q + P$

3.1.3 Group Order and Structure

After having defined a proper composition law and verified that it satisfies the group properties, there are two important open problems: the computation of the number of points of the curve, i.e. the order of the group, and the conditions under which this group is cyclic, i.e. the structure of the group.

There is an interesting result about the order of an elliptic curve group, the Hasse theorem:

Theorem 3.1.1 (Hasse Theorem)**.** *The number of points of an elliptic curve $\mathbb{E}(\mathbb{F}_q)$ lies between:*

$$q + 1 - 2\sqrt{q} \leq \#\mathbb{E}(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$$

The exact number of points can be computed in polynomial time via either the Schoof's algorithm or the Schoof-Elkis-Atkin algorithm.

The structure of the group of points $\mathbb{E}(\mathbb{F}_q)$ is described by theorem 3.1.2. We use $\mathbb{Z}_n$ to denote a cyclic group of order n.

Theorem 3.1.2. *Let E be an elliptic curve defined over $\mathbb{F}_q$. Then $\mathbb{E}(\mathbb{F}_q)$ is isomorphic to $Z_{n_1} \oplus Z_{n_2}$ where n_1 and n_2 are uniquely determined positive integers such that n_2 divides both n_1 and $q - 1$*

A corollary of this theorem provides a sufficient condition for an elliptic curve set of points to be a cyclic group.

Corollary 3.1.3. *If $\#\mathbb{E}(\mathbb{F}_q)$ can be factorized in the product of distinct primes, then the group $G = (\mathbb{E}(\mathbb{F}_q), +)$ is cyclic*

3.1.4 Elliptic Curve Discrete Logarithm Problem

The hardness of the elliptic curve discrete logarithm problem is the cornerstone upon which the security of ECC is based.

Definition 3.3. Given an elliptic curve $\mathbb{E}$ defined over a field $\mathbb{F}_q$, a point $P \in \mathbb{E}(\mathbb{F}_q)$ of order n, and a point $Q \in \langle P \rangle$, the Elliptic Curve Discrete Logarithm Problem (ECDLP) requires to find an integer $k \in [0, n-1]$ such that

$$Q = [k]P = \underbrace{P + P + \ldots + P}_{k-times}$$

The integer k is called the discrete logarithm of Q.

Since the security of all EC cryptoschemes is based on this problem, it is important to properly choose the parameters of the elliptic curve in use, to avoid all the possible attacks. The most trivial way to solve the ECDLP is to perform a sort of brute force attack, computing all the multiples of the base point P, until Q is found. In the worst case n point multiplications will be required, while on average only $n/2$ operations should be performed; therefore to circumvent this attack it is sufficient to pick a sufficiently large n. In general $n \geq 2^{80}$ represents a good threshold to guarantee the unfeasability of this attack. However, the best attack strategies against the ECDLP are two other methods to compute discrete logarithms: the Pohlig-Hellman algorithm and Pollard's rho algorithm. This approach has a fully exponential time complexity of $O(\sqrt{p})$, where p is the largest prime divisor of the elliptic curve group of order n. To circumvent this attack it is necessary to select a value of n that is divisible by a sufficiently large prime number p. In practice, only curves with nearly prime cardinality are considered, in particular those curves where: $\#\mathbb{E}(\mathbb{F}_q) = cp, \quad p > 2^{160}$. The idea is to employ the subgroup of prime order p (the existence of which is guaranteed by the inverse of Lagrange's theorem for finite abelian groups), which is surely cyclic (because of the prime order) and robust against the aforementioned attacks (provided that p is sufficiently large).

Table 3.1: Comparable strenghts as reported by NIST SP 800-57

Bits of security [bit]	**Symmetric key algorithms**	**FFC [bit]**	**IFC [bit]**	**ECC [bit]**
80	2TDEA	$L = 1024$ $N = 160$	$k = 1024$	$f = 160 - 223$
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224 - 255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256 - 383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384 - 511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

3.1.5 Level of security

To conclude this overview of ECC we provide a brief analysis of the level of security guaranteed with respect to other cryptographic technologies, both symmetric and asymmetric. Two algorithms, having key sizes X and Y, provide the same level of security if the computational effort needed to break them or retrieve the keys is approximately the same. It is common to express the security margin of an algorithm for a given key size in terms of the amount of work it takes to try all keys for a symmetric algorithm that has no short cut attacks [1]. An algorithm that has a Y bit key, but whose strength is comparable to an X bit key of such a symmetric algorithm is said to provide a "security strength of X bits". Table 3.1 provides comparable security strenghts according to the National Institute of Standard and Technology (NIST) [17].

- Column 1 indicates the number of bits of security provided by the algorithm and key sizes of a particular row;

[1]i.e. the most efficient attack is to try all possible keys

- Column 2 indicates the symmetric key algorithm that has been considered as a reference for a certain security level;
- Column 3 indicates the minimum parameters for asymmetric algorithms based on Finite Field Cryptography (FFC) (e.g. DSA, DH). L is the size of the public key, while N is the size of the public key;
- Column 4 indicates the minimum parameters for asymmetric algorithms based on Integer Factorization Cryptography (IFC), where the most relevant example is surely the RSA algorithm. The value of k, that is commonly considered the key size, corresponds to the bit size of RSA field modulus;
- Column 5 indicates the range of f, the bit size of the order of $\langle g \rangle$, the subgroup generated by g to be used in the ECC based algorithms.

The data reported in the table reflect the great advantage provided by ECC: the reduced key size with respect to other asymmetric key algorithms necessary to achieve a certain security margin. This characteristic results in more efficient implementations, making ECC a worthy choice when dealing with low power embedded devices.

3.2 Key Agreement Protocols in ECC

This section is dedicated to a general revision of the key agreement protocols covered previously, considering them as realized using ECC. The main difference is a substantial change of notation: instead of using the classic multiplicative notation, we adapt the algorithms to the additive notation typically employed when dealing with elliptic curves.

3.2.1 Elliptic Curve Diffie-Hellman

The Elliptic Curve Diffie-Hellman (ECDH) is the implementation of the Diffie-Hellman key exchange using ECC. In this version of the algorithm the two parties have to agree on an elliptic curve, generate public-private key pairs performing the usual exchange to estalish a shared secret over an insecure channel.

Algorithm 3.2.1: Elliptic Curve Diffie-Hellman Key Exchange

```
Globals: $(G,+) = \langle P \rangle, \quad G \subseteq \mathbb{E}(\mathbb{F}_q), \quad n = |G|$
Output: Shared secret: $k = [a_0 \cdot a_1]P$
begin
    Private ephemeral key: $a_i \xleftarrow{random} \mathbb{Z}_n \setminus \{0,1\}$
    Public ephemeral key: $k_{pub,i} = [a_i]P$
    Send $k_{pub,i}$ to the other peer
    Receive $k_{pub,j}$
    $k = [a_i]k_{pub,j} = [a_0 \cdot a_1]P$
    return $k$
```

3.2.2 Elliptic Curve Burmester-Desmedt

Algorithm 3.2.2: Elliptic Curve BD1

```
Globals: $(G,+) = \langle P \rangle, \quad G \subseteq \mathbb{E}(\mathbb{F}_q), \quad n = |G|$
Input: $\mathcal{U} : \{U_0, \ldots, U_{t-1}\}$ ordered set of protocol partecipants
Output: Shared secret: $k = [a_0 \cdot a_1 + a_1 \cdot a_2 + \ldots + a_{t-1} \cdot a_0]P$
begin
    $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0,1\}$
    $z_i = [a_i]P$
    Broadcast $z_i$
    Receive $z_{i-1}$ and $z_{i+1}$
    $X_i = [a_i](z_{i+1} - z_{i-1})$
    Broadcast $X_i$
    for $U_j \in U | U_j \neq U_i$ do
        receive $X_j$
    $k = [t \cdot a_i]z_{i-1} + [t-1]X_i + [t-2]X_{i+1} + \ldots + X_{i-2}$
    return $k$
```

Algorithm 3.2.2 presents the first Burmester-Desmedt key exchange updated to additive notation. Note how multiplications between group elements become additions. As expected the division operation between z_{i+1} and z_{i-1} becomes a subtraction, i.e. an addition of the inverse of the point, computed summing z_{i+1} to the inverse of z_{i-1}.

Algorithm 3.2.3: Elliptic Curve BD2

```
Globals: $(G,+) = \langle P \rangle, \quad G \subseteq \mathbb{E}(\mathbb{F}_q), \quad n = |G|$
Input: $\mathcal{U} : \{U_0, U_1, \ldots, U_{t-1}\}$ set of protocol partecipants
Output: Shared secret: $k$
begin
    $k \overset{random}{\longleftarrow} G$, $U_0$ selects the key
    $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0,1\}$
    $z_i = [a_i]P$
    $U_i$ sends $z_i$ to $U_0$, $U_0$ broadcasts $z_0$
    $U_0$ computes $k_{0i} = [a_0]z_i$, $U_i$ computes $k_{0i} = [a_i]z_0$
    for $i = 1$ to $t-1$ do
        $X_i = k + k_{0i}$
        Send $X_i$ to $U_i$
    $k = X_i + (-k_{0i})$
    return $k$
```

Algorithm 3.2.3 shows instead the adaptation of the second Burmester-

Desmedt protocol to elliptic curves.

3.2.3 Elliptic Curve GDH and MKA

Algorithm 3.2.4: Elliptic Curve Multiparty Key Agreement (MKA)

```
Globals: $(G, +) = \langle P \rangle, \quad G \subseteq \mathbb{E}(\mathbb{F}_q), \quad n = |G|$
Input: $\mathcal{U} : \{U_0, U_1, \ldots, U_{t-1}\}$ set of protocol partecipants
Output: Shared secret: $k = \left[\prod_{i=0}^{t-1} a_i\right] P$
begin
    $a_i \overset{random}{\longleftarrow} \mathbb{Z}_n \setminus \{0, 1\}$
    $k_{U_i} = P$
    for $r = 0$ to $t - 2$ do
        $k_{U_i,r} = [a_i]k_{U_i}$
        Broadcast $k_{U_i,r}$
        $tmp = \sum_{U_j \neq U_i} k_{U_j}$
        $k_{U_i} = \left[\frac{1}{r+1}\right](tmp + [-r]k_{U_i})$
    $k = [a_i]k_{U_i}$
    return $k$
```

The adaptation of the Generalized Diffie-Hellman algorithm to elliptic curves is really effortless: each partecipant, instead of computing a sequence of exponentiations on the received group elements, performs scalar point multiplication (i.e. repeated addition). The new version of the Multiparty Key Agreement Protocols is shown in Algorithm 3.2.4. It is interesting to note that the Diffie-Hellman key, that is computed by both these protocols, is now expressed as:

$$k = \left[\prod_{i=0}^{t-1} a_i\right] P$$

Chapter 4

Experimental Results

This chapter contains the experimental results and the implementation details of this work and it is logically divided into two parts. In the first part we present our experimental setup, discussing both the platform chosen for our evaluation and the implementation details, including for example the cryptographic library that has been used. Successively, in the second part, we show and discuss the results of this work.

4.1 Experimental Setup and Implementation

This section is dedicated to a general description of how the protocols have been implemented and tested. First of all we provide the technical details of the hardware platform that have been used. We move on discussing how the algorithms have been implemented, the cryptographic library used and the employed elliptic curve standards. Finally we briefly talk about how the metrics of interest have been measured.

4.1.1 Hardware Platform

The platform selected for the implementation of the key agreement algorithms is an evaluation board produced by STMicroelectronics, precisely the STM32F407VGT6 Discovery microcontroller. The board mounts an ARM Cortex-M4 32 bit processor and is equipped with 1 MB of flash memory and 192 KB of RAM. The power supply is provided through USB bus and the memory is written using the onboard ST-LINK/V2 interface, an in-

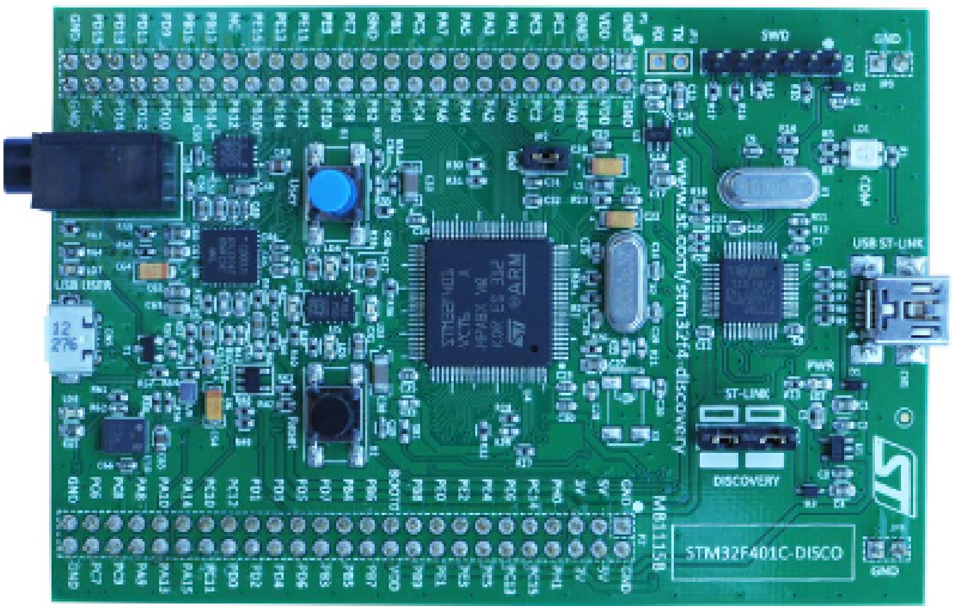


Figure 4.1: STM32F4 Discovery

circuit debugger and programmer provided directly by STMicroelectronics for different families of microcontrollers. The board features a wide range of peripherals such as: general purpose timers, multiple UART [1] transceivers, motion sensors, audio sensors, cryptographic core and general purpose IO pins. The Cortex-M4 belongs to the Cortex-M family, a group of 32 bit RISC processors specifically designed by ARM for embedded applications, supporting both Thumb and Thumb 2 instruction sets. The core's working frequency can be set up to 168 MHz. The choice of this particular platform is based mainly on two reasons. First of all in order to be able to perform a realistic simulation of the algorithms under exam, it is important to test them on a platform similar to those usually deployed on the field. According to this requirement we have chosen the STM32F4 Discovery board which is suited to simulate the computational constraints found in BAS. Another non negligible aspect to consider is the assessment of our work: from this perspective it is desirable to adopt a workbench that is widely employed and accepted by the research community. The STM32F4 evaluation board satisfies this requirement, allowing interested readers to reproduce the same results presented here.

[1] Universal Asynchronous Receiver Transmitter

Table 4.1: Comparison between the performances of the PolarSSL implementation of NIST curves and Brainpool curves, considering the number of handshake per second performed with ECDH `https://polarssl.org/kb/cryptography/elliptic-curve-performance-nist-vs-brainpool`

Elliptic Curve	**Number of Op. [handshake/sec]**
P-521	15
P-384	20
P-256	41
P-224	63
P-192	85
BP-512	2
BP-384	4
BP-256	8

4.1.2 Cryptographic Library

The algorithms have been implemented using the PolarSSL library, version 1.3.7 [3]. This library, which is entirely written in C, provides all the tools for an SSL/TLS implementation as a series of loosely coupled modules. Unlike other implementation of SSL/TLS, such as OpenSSL, PolarSSL is designed to fit on small embedded devices, which makes it ideal for our application. It is also highly modular: each component can be used separately from the rest of the framework, thus limiting the code size. The most important module provided by PolarSSL is the `ecp` module, which basically offers all the functions to implement ECC. PolarSSL supports only curves defined over prime fields (i.e. defined over $\mathbb{F}_p$) from different standards, in particular:

- Five NIST standard curves: P-192, P-224, P-256, P-384 and P-521 [16]
- Three Brainpool curves: BP-256, BP-384, BP-512 [13]
- The special curve 25519 [4]
- Three Koblitz curves: KP-192, KP-224, KP-256

All the test have been carried out using the standard NIST curve P-256, which offers a good balance between efficiency and security margin. Indeed

Table 4.2: Memory size of the TLS ECPoint record for PolarSSL's EC. The special curve 25519 is not reported here because it is not a standard one.

Elliptic Curve	**Point Dimension [byte]**
P-192	50
P-224	58
P-256	66
P-384	98
P-521	132.25
BP-256	66
BP-384	98
BP-512	130
KP-192	50
KP-224	58
KP-296	76

Brainpool curves, although accepted as a standard, are significantly slower, because the prime moduli of the underlying field are chosen randomly to guarantee extra security, while NIST curves employ particular prime moduli to accelerate computations. The result of this design choice is visible in Table 4.1, provided directly by PolarSSL's authors, that shows the number of handshake per second of ECDH performed by the different curves. The special curve 25519, which was designed to provide the same security guarantees of Brainpool curves with an higher level of efficiency, has been discarded because at the moment is not part of any standard (this is why it is called "special"). The points of these curves are the objects effectively exchanged during key agreement algorithms, therefore, in order to to facilitate their transmission, the library offers the possibility to export these points in a standard data structure, that is the TLS ECPoint record (defined in RFC 4492). Table 4.2 shows the memory sizes of the exported points for the various elliptic curves supported by PolarSSL. Since we have decided to employ a standard NIST curve, the interested reader will find the details about these curves in Appendix B.

4.1.3 Experimental Setup

We have already remarked that KNX, which was described in Chapter 1, has been chosen as the reference protocol for our analysis, indeed its tight bandwidth is well suited to verify the feasability of the key agreement schemes. The various layers of the protocol have been implemented with an ad-hoc library, that provides the functions to perform packets encapsulation and decapsulation. The data are physically transmitted exploiting the UART interfaces provided by the microcontroller. These peripherals perform serial transmission of data, allowing to configure data format and transmission speed. During our experiments the transmission speed has been set to 19200 bit/s, that is the maximum transmission speed offered by the KNX protocol.

The main interest of our experiments is to measure the efficiency of the key agreement protocols, that is their running time, in order to evaluate the feasability of their implementation in BAS. Unfortunately it is not possible to measure the running time of our tests through conventional methods, because the evaluation board is a bare metal environment, i.e. without operating system. Therefore the measurements have been carried out using a digital oscilloscope to monitor specific output pins of the microcontroller. The idea is to associate to each portion of code of interest a certain output pin. Before executing the target code the corresponding pin is set up and at the end it is set down, using the oscilloscope to measure the elapsed time interval.

4.2 Performances

Table 4.3 reports the measurements concerning the running time of the algorithms. For each key agreement scheme we have implemented a test that performs a key exchange between two different microcontrollers, measuring the execution time on one of the two boards as explained before. The only exception is the Generalized Diffie-Hellman, where the formulation of the protocol prevents its realization with only two parties, forcing us to implement the key exchange with three parties, using one board to contemporarily simulate two nodes. During these experiments we have measured multiple things: the total running time of the algorithms, the running time

Table 4.3: Performance measures for all the key agreement protocols. The times reported in the table are expressed in seconds.

Freq. [MHz]	Metric	BD1	BD2 1	BD2 2	MKA	GDH2 1-2	GDH2 3
16	Total time	39.82	61.5	82	78.93	211	159
	Group Op.s	39.82	61.5	39.03	78.64	131.64	79
64	Total time	13.96	12.2	19.15	15.6	41.33	31.74
	Group Op.s	13.84	12.043	13.715	15.514	25.643	15.38
168	Total time	5.32	4.716	7.344	5.963	17.26	12.76
	Group Op.s	5.288	4.588	5.233	5.906	11.127	5.86

of each scalar point multiplication (i.e. the most time consuming arithmetic operation), time necessary to perform packet encapsulation and transmission, and packet decapsulation. We have performed each measurement with multiple clock frequencies, in order to verify if the algorithms can be scaled down to less powerful devices. The two columns corresponding to the second Burmester-Desmedt key exchange and the Generalized Diffie Hellman have been split because these protocols are asymmetric, therefore different nodes perform substantially different operations, thus requiring separate monitoring. One may note that we have only reported the total running time and the time spent performing group operations (that is the sum of all scalar point multiplications), the reason is that the encapsulation and transmission time and the delay for packets decapsulation were invariant with respect to all tested clock frequencies. This is reasonable since the transmission speed was kept constant at 19200 bit/s during all the tests and the delays due to packet encapsulation or decapsulation are negligible. In particular the time spent to encapsulate and transmit a packet of data (we recall that a standard KNX frame is 23 bytes long with 15 bytes of payload) is around 15 ms, thus completely dominated by the time spent computing arithmetic operations, which are the real bottleneck of the algorithms.

For the sake of completeness we have also compared the key exchange algorithms with a symmetric block cipher, specifically AES-128. Our microcontroller, like many others available on the market, features a hardware implementation of AES, which is able to compute the encryption of a single block (128 bit) in 14 cycles of the internal bus. As it is reasonable to expect the key exchange algorithms are orders of magnitudes slower than a block cipher, indeed the transmission of a KNX packet encrypted using AES is performed in 16 ms. Considering that the encapsulation and transmission time of a packet is around 15 ms, the delay due AES encryption is substantially negligible.

Observing the reported data it is possible to make some interesting considerations. The first algorithm proposed by Burmester and Desmedt is the most efficient, especially when the clock frequency is very low. Nonetheless both the second algorithm proposed by the same authors and the MKA provide similar performances at higher clock frequencies, while the Generalized Diffie-Hellman appears to be the less efficient protocol, even considering that the key exchange is performed on three nodes instead of only two. It is possible to note that, when the clock frequency is reduced from 168 MHz to 64 MHz, the running times scale correctly with a factor of 2.625. Instead, when the clock frequency is scaled down to 16 MHz, the measured values scale with a factor of 5, while one would expect a factor of 4 (since the processor speed has been divided by 4). The motivation lies in the fact that the access to the flash memory is regulated by the number of wait states, which should be set according to CPU clock frequency. The higher the clock frequency is, the higher the number of wait states should be, otherwise the program will not be able to cope with the requests in time. In our tests the number of wait states has been fixed to 5, a value that is reasonable if the clock frequency is set at 150 MHz or more. At lower clock frequencies, this has a more significant impact on the execution times, thus causing the observed skew in timings. As it can be seen, the slowdown is the same for all the algorithms, thus effectively pointing to the aforementioned architectural issue to be the cause of our problems.

In conclusion our performance analysis suggests that the algorithms proposed by Burmester-Desmedt and the multi-party Key Agreement of Antonini-Barenghi-Pelosi are more efficient than the solution proposed by Steiner, with the BD1 appearing as the clear winner. In our opinion also the MKA algorithm should be considered as a valid option, because, contrarily to what happens with BD1, it doesn't require any preconditions. We recall that, as explained in detail in Chapter 2, the BD1 protocol requires the parteci-

Table 4.4: Code sizes of the tests. The table reports the sizes of the various sections of the ELF executable of each test, in particular: code segment (.text), initialized data segment (.data), uninitialized data segment (.bss)

Test	**.text [byte]**	**.data [byte]**	**.bss [byte]**	**Total [byte]**
BD1	44388	36	33064	77488
BD2 node	44284	36	33064	77384
BD2 root	44372	36	33064	77472
GDH2 1-2	44644	36	33064	77744
GDH2 3	44340	36	33064	77440
MKA	44516	36	33064	77616

pating nodes to be numbered. The BD2 algorithm instead should not be considered as an ideal solution for multiple reasons. First of all, although its performance measures are not bad at all, the unbalanced distribution of the computational load may become a huge problem with an higher number of nodes. Secondly the instrinsic asymmetry of the algorithm may introduce other issues in phase of deployment. Finally, the solution proposed by Steiner, as we have already said, is not particularly appealing. Indeed it is an asymmetric protocol and it is not efficient. A good characteristic of this algorithm is the low number of messages (and so of packets) exchanged by the nodes, but it is of little interest considering that the real bottleneck of the system are the arithmetic operations. In general we argue that, employing one of these algorithm, it is possible to implement a transparent multi-party key exchange procedure. This critical operation can be performed in background, when the network is less loaded (e.g. during night time) or at fixed intervals, depending on the desired degree of security.

4.2.1 Code size

Code size is another critical metric in embedded environments, because today's CPUs should fit in increasingly smaller packages in order to reduce chip area and consequently the overall cost. We have measured code size considering the ELF executable file of each test. Table 4.4 provides the resulting sizes, detailing the dimension of the different sections of the executable. The tests have been compiled to the Thumb instruction set, using a cross compilation toolchain provided and maintained directly by ARM. It is possible to see that the code sizes of the algorithms are very similar one to the other, therefore it is not possible to rank them with respect to this metric. Furthermore the memory occupation is sufficiently low, indeed we recall that the microcontroller used as benchmark platform features only 192 KB of RAM and 1 MB of flash memory. Therefore the memory occupancy is below 40%, and thus it is adequate to implement these algorithms in the restrained environments of low power embedded devices.

Chapter 5

Conclusion

This work has proposed a comprehensive study of the most important multiparty key exchange algorithms in the context of BAS. We have proposed a security analysis that highlighted how the most widespread BAS protocols are vulnerable to different kinds of attacks, due to the lack of security mechanisms in their design. This study should be considered complementary to the work published by Antonini, Barenghi and Pelosi [1], in which the authors propose a secure protocol for BAS. The central idea of this protocol is to employ a multiparty key exchange algorithm, that is a generalization of the famous Diffie-Hellman key exchange, to establish a shared secret key that is subsequently used to implement certain security desiderata. To this end the authors propose a multiparty key exchange algorithm of their own invention. The main limitation of their work is that they don't consider pre-existing solutions and they don't provide an experimental evaluation of the proposed algorithm. In our study we have tried to complete their work, providing a comparison of the most important key exchange algorithms proposed in the literature and an experimental evaluation of their efficiency. The implementation of the tested algorithms have been adapted to the particular environment of BAS, characterized by low power embedded devices with significant constraints in terms of memory and computational capacity.

The main limitation of this thesis, in our opinion, resides in the experimental part. Indeed this could be enhanced, testing the algorithms on different platforms and on a more realistic setting. A possible future development could be that of testing the key exchange procedures on a real network, implementing also the protocol proposed in [1], thus verifying the feasability of the complete solution.

Appendix A

Theorem Demonstrations

A.1 Proof of Theorem 2.3.1

Let $\mathcal{A} = \{a_0, a_1, \ldots, a_{t-1}\}$ be the set of private keys chosen by the $t \geq 2$ protocol partecipants in the set $\mathcal{U}$. Denote as $\mathcal{A}_{z,r+1}$, with $0 \leq r \leq t-2$, $z \in \left\{1, \ldots, \binom{t}{r+1}\right\}$, the generic element of the power-set $2^{\mathcal{A}}$ composed by r elements of $\mathcal{A}$ and denote as $\mu_{z,r+1} = \left(\prod_{a \in \mathcal{A}_{z,r+1}} a\right)$ the monomial obtained by the product of the values included in $\mathcal{A}_{z,r+1}$. A monomial that includes a_i as factor is denoted as $\mu_{z,r+1}{}^{(i)}$, while one that does not contain a_i is denoted as $\mu_{z,r+1}{}^{(\backslash i)}$. The whole algorithm can be described as a set of recursive equations involving the value k_{U_i} and the number of iterations $r \in \{0, \ldots, t-2\}$:

$$\begin{cases} k_{U_i,-1} = g & r = -1 \\ k_{U_i,0} = g^{\sum_z \mu_{z,1}}, \text{ for all } z = 1, \ldots, t \text{ s.t. } \mu_{z,1} \neq \mu_{z,1}{}^{(i)} & r = 0 \\ k_{U_i,r} = \left(g^{r \sum_z \mu_{z,r+1}{}^{(i)} + (r+1) \sum_z \mu_{z,r+1}{}^{(\backslash i)}} \cdot (k_{U_i,r-1})^{-a_i r}\right)^{\frac{1}{r+1}} & r > 0 \end{cases}$$

It can be easily verified that the last instruction in the body of Algorithm 2.3.1 computes the same value of the above relation (for $r \geq 0$), that is:

$$k_{U_i,r} = g^{\sum_z \mu_{z,r+1}{}^{(i)}} \text{ for all } z = 1, \ldots, \binom{t}{r+1} \text{ s.t. } \mu_{z,r+1} \neq \mu_{z,r+1}{}^{(i)}$$

Therefore at the end of the last iteration the exponent of the generator g comes down to a single monomial $\mu_{z,t-1} = (a_0 \cdot \ldots \cdot a_{i-1} \cdot a_{i+1} \cdot \ldots \cdot a_{t-1})$. The algorithm returns the desired shared key raising this monomial to the power of the missing ephemeral key a_i.

$\square$

A.2 Proof of Theorem 2.3.2

First of all we will show that CDHP $\leq$ CMKAP. Let $\mathcal{O}_{CMKAP}(G, g, \mathcal{L}, t)$ be an oracle able to solve the CMKAP on $(G, \cdot) = \langle g \rangle$, with t parties, taking $\mathcal{L}$ as the input list of intermediate values $k_{U_i,r}$. It is possible to solve the CDHP invoking the oracle in the following way: $\mathcal{O}_{CMKAP}(G, g, (g^{a_0}, g^{a_1}), 2)$. The oracle will compute $g^{a_0 a_1}$ because g^{a_0} and g^{a_1} correspond to $k_{U_0,0}$ and $k_{U_1,0}$ computed by the MKA with two partecipants. This proof by itself would be sufficient to guarantee the security of MKA against a passive adversary, since the hypothetical existence of a method to solve efficiently the CMKAP would imply the possibility to solve efficiently the CDHP, resulting in a violation of the intractability of CDHP. Nonetheless the proof can be extended proving the complete equivalence of the the two problems, that is proving CMKAP $\leq$ CDHP. Similarly to what we have done before let's assume to have an oracle able to solve the CDHP: $\mathcal{O}_{CDHP}(G, g, g^x, g^y)$. Consider the values $k_{U_i,0}$ (i.e. the values broadcasted by the partecipants in the first round of the algorithm), they are all of the type g^{a_i}. Now let's call k_{01} the value containing the ephemeral key corresponding to nodes 0 and 1, that is $k_{01} = g^{a_0 a_1}$. This value can be computed as: $k_{01} = \mathcal{O}_{CDHP}(G, g, k_{U_0,0}, k_{U_1,0})$. We can iterate the procedure to compute $k_{012} = g^{a_0 a_1 a_2}$, by doing: $k_{012} = \mathcal{O}_{CDHP}(G, g, k_{01}, k_{U_2,0})$. It is clear that iterating this procedure $t-1$ times we recover the complete key $g^{\prod_{j=0}^{t-1} a_i}$, thus solving the CMKAP.

$\square$

Appendix B

Appendix on ECC

B.1 Composition Law and Point Representation

Table B.1: Composition law for elliptic curves defined over a generic finite field with characteristic greater than 3: $\mathbb{E}(\mathbb{F}_{p^m}) : y^2 = x^3 + ax + b, \quad p > 3$

$P_1 + P_2$	$x_3 = \frac{y_1-y_2}{x_1-x_2}^2 - x_1 - x_2$ $y_3 = \frac{y_1-y_2}{x_1-x_2}(x_1 - x_3) - y_1$
$[2]P_1$	$x_3 = \left(\frac{3{x_1}^2+a}{2y_1}\right)^2 - 2x_1$ $y_3 = \left(\frac{3{x_1}^2+a}{2y_1}\right)(x_1 - x_3) - y_1$
$-P_1$	$(x_1, -y_1)$

Considering elliptic curves defined over $\mathbb{F}_{p^m}, \quad p > 3$ Table B.1 reports the explicit formulas to apply the composition law (chord & tangent rule). In particular, given the points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, we have:

- Point addition: $P_1 + P_2 = (x_3, y_3)$
- Point doubling: $[2]P_1 = (x_3, y_3)$
- Point inversion

On the other hand Table B.2 reports the same formulas for elliptic curves defined over binary fields $\mathbb{F}_{2^m}$.

Table B.2: Composition law for elliptic curves defined over a generic binary field (non-supersingular curve): $\mathbb{E}(\mathbb{F}_{2^m}) : y^2 + xy = x^3 + ax^2 + b$

$P_1 + P_2$	$x_3 = \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a$ $y_3 = \frac{y_1+y_2}{x_1+x_2}(x_1 + x_3) + x_3 + y_1$
$[2]P_1$	$x_3 = {x_1}^2 + \frac{b}{{x_1}^2}$ $y_3 = {x_1}^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3$
$-P_1$	$(x_1, x_1 + y_1)$

As one may note for both types of curves the formulas of point addtion and doubling require a field inversion and several field multiplications. For implementation purposes this is a serious problem, because division in finite fileds is an expensive operation, that cannot be implemented as efficiently as multiplications. This issue is solved considering elliptic curves in projective coordinates.

Definition B.1. Let $\mathbb{K}$ be a field. Let c and d be positive intergers. One can define an equivalence relation $\sim$ on the set $\mathbb{K}^3 \setminus \{(0,0,0)\}$ of nonzero triples over $\mathbb{K}$ by

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \text{ if } X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2, \quad \lambda \in \mathbb{K} \setminus \{0\}$$

The equivalence class containing (X, Y, Z) is denoted as:

$$(X : Y : Z) = \{\lambda^c X, \lambda^d Y, \lambda Z\}, \quad \lambda \in \mathbb{K} \setminus \{0\}$$

and (X, Y, Z) is called the representative point of the class. The set of all the projective points is called projective plane and is indicated as $\mathbb{P}(\mathbb{K})^2$. In general every point of a class can serve as its representative, if $Z \neq 0$ then there exists a unique point in its equivalence class $(x, y, 1)$ where $x = X/Z^c$ and $y = Y/Z^d$. This allows to state that there is a one-to-one correspondence between the set of projective points:

$$\mathbb{P}(\mathbb{K})^* = \{(X : Y : Z), X, Y, Z \in \mathbb{K}, Z \neq 0\}$$

and the set of affine points:

$$\mathbb{A}(\mathbb{K}) = \{(x, y), x, y \in \mathbb{K}\}$$

The projective plane is identified with the affine one plus the adjoint of the points that have $Z = 0$, which are called points at infinity. A generic point at inifinity $(X, Y, 0)$ can't be put in correspondence with any affine point, instead it is possible to establish a one-to-one correspondence with the direction of the straight lines having slope $m = Y/X$. In practise implementations several projective coordinate systems are employed, depending on the characteristics of the specific application. The most important coordinate systems are derived from the previous definion:

Standard projective coordinates obtained setting $c = 1$ and $d = 1$. In this system each affine point (x, y) is represented by three coordinates X, Y, Z using the following relation: $x = \frac{X}{Z}, y = \frac{Y}{Z}$.

Jacobian Coordinates which are used for NIST standard curves. In this case $c = 2$ and $d = 3$, therefore a point (x, y) is represented as (X, Y, Z) where $x = \frac{X}{Z^2}, y = \frac{Y}{Z^3}$.

López-Dahab system characterized by $c = 1$ and $d = 2$, therefore the relation is $x = \frac{X}{Z}$ and $\frac{Y}{Z^2}$.

Modified Jacobian system the same relations are used but four coordinates are stored and used for calculations (X, Y, Z, aZ^4).

Chudnovsky Jacobian system where the Jacobian point $(X : Y : Z)$ is represented as $(X : Y : Z : Z^2 : Z^3)$.

Table B.3: Bit length of the underlying fields for NIST recommended curves.

Bit Length of n [bit]	**Bit lenght of p [bit]**
160-223	192
224-255	224
256-383	256
384-511	384
≥ 512	521

B.2 Standard NIST Curves supported by PolarSSL

The official document of NIST [17], as we have seen in Chapter 3, defines 5 levels of security. Each level of security is characterized by a different key length, defined as the bit length of the order of the generator of the group. For each key length two kinds of fields are provided:

- A prime field $\mathbb{F}_p$ which contains p elements, which are the integers modulo p.
- A binary field $\mathbb{F}_{2^m}$ which contains 2^m elements. The elements of this kind of field are the polynomials defined over $\mathbb{F}_2$ of degree lower than m.

The PolarSSL library, up to version 1.3.7, supports only elliptic curves defined over prime fields, discarding the other proposed by the standard. For this type of curves defined over $\mathbb{F}_p$, the security strength is dependent not only on the number of points n, but also on the length of the binary expansion of p, that is the prime modulus of the underlying field. Table B.3 provides the bit lengths of the various underlying fields of the curves provided in this appendix. The first column lists the ranges for the bit length of n. The second column reports the value of p used for the curves over prime field.

NIST curves are actually pseudo random curves, that is curves whose coefficients are generated from the output of a seeded cryptographic hash function. If the domain parameter seed value is given along with the coefficients, it is possible to perform the validation of domain parameters. This is done in order to guarantee that the curve has not been selected appropriately, with some kind of ad-hoc weakness. The generic NIST prime field curve has the following form (in affine coordinates):

$$\mathbb{E}(\mathbb{F}_p) : y^2 \equiv x^3 - 3x + b \quad (mod\ p)^1$$

The arithmetic of this class of curves is implemented expoliting two different projective coordinates.

- Standard projective coordinates. The projective equation of the curve is expressed as
  $$Y^2Z = X^3 + aXZ^2 + bZ^3$$
  The point at infinity, indicated here as $\mathcal{O}$, corresponds to $(0 : 1 : 0)$. The inverse of a point $X : Y : Z$ is computed as $X : -Y : Z$.

- Jacobian projective coordinates. The projective equation in this case becomes
  $$Y^2 = X^3 + aXZ^4 + bZ^6$$
  The point at infinity is $(1 : 1 : 0)$, while the inverse of a point $(X : Y : Z)$ is again $(X : -Y : Z)$

For each curve the standard specifies all the domain parameters:

- The prime modulus p, which is chosen appropriately as a generalized Mersenne prime in order to speed up the computation of field multiplications.
- The order n, which is always a big prime number.
- The 160-bit input seed to the SHA-1 based algorithm (i.e., the domain parameter seed).
- The output c of the SHA-1 based algorithm.
- The coefficient b (satisfying $b^2c \equiv -27 \quad (mod\ p)$).
- The generator.

[1]Parameter a is fixed to -3 to speed up computations

The five prime field curves proposed are identified by the bit lenght of the prime modulus: P-192, P-224, P-256, P-384 and P-521. The arithmetic operations of point addition and point doubling are implemented in the PolarSSL library respectively with Algorithm B.2.2 and Algorithm B.2.1, exploiting both affine and Jacobian coordinates. As it is possible to note both algorithms don't make use of field inversions.

Algorithm B.2.1: Point doubling ($y^2 = x^3 - 3x + b$, affine-Jacobian coordinates)

```
**Input**: $P = (X_1 : Y_1 : Z_1)$ in Jacobian coordinates
**Output**: $2P = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates
**if** $P = \mathcal{O}$ **then**
    **return** $\mathcal{O}$
$T_1 = Z_1^2$
$T_2 = X_1 - T_1$
$T_1 = X_1 + T_1$
$T_2 = T_2 \cdot T_1$
$T_2 = 3T_2$
$Y_3 = 2Y_1$
$Z_3 = Y_3 \cdot Z_1$
$Y_3 = Y_3^2$
$T_3 = Y_3 \cdot X_1$
$Y_3 = Y_3^2$
$Y_3 = Y_3/2$
$X_3 = T_2^2$
$T_1 = 2T_3$
$X_3 = X_3 - T_1$
$T_1 = T_3 - X_3$
$T_1 = T_1 \cdot T_2$
$Y_3 = T_1 - Y_3$
**return** $(X_3 : Y_3 : Z_3)$
```

Algorithm B.2.2: Point addition ($y^2 = x^3 - 3x + b$, affine-Jacobian coordinates)

```
Input: $P = (X_1 : Y_1 : Z_1)$ in Jacobian coordinates, $Q = (x_2, y_2)$ in
       affine coordinates
Output: $P + Q = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates
if $Q = \mathcal{O}$ then
    return $(X_1 : Y_1 : Z_1)$
if $P = \mathcal{O}$ then
    return $(x_2 : y_2 : 1)$
$T_1 = Z_1^2$
$T_2 = T_1 \cdot Z_1$
$T_1 = T_1 \cdot x_2$
$T_2 = T_2 \cdot y_2$
$T_1 = T_1 - X_1$
$T_2 = T_2 - Y_1$
if $T_1 = 0$ then
    if $T_2 = 0$ then
        $(X_3 : Y_3 : Z_3) = 2(x_2 : y_2 : 1)$
        return $(X_3 : Y_3 : Z_3)$
    else
        return $\mathcal{O}$
$Z_3 = Z_1 \cdot T_1$
$T_3 = T_1^2$
$T_4 = T_3 \cdot T_1$
$T_3 = T_3 \cdot X_1$
$T_1 = 2T_3$
$X_3 = T_2^2$
$X_3 = X_3 - T_1$
$X_3 = X_3 - T_4$
$T_3 = T_3 - X_3$
$T_3 = T_3 \cdot T_2$
$T_4 = T_4 \cdot Y_1$
$Y_3 = T_3 - T_4$
return $(X_3 : Y_3 : Z_3)$
```

Bibliography

[1] Alessio Antonini, Alessandro Barenghi, and Gerardo Pelosi. Security Analysis of Building Automation Networks - Threat Model and Viable Mitigation Techniques. In *Secure IT Systems - 18th Nordic Conference, NordSec 2013, Ilulissat, Greenland, October 18-21, 2013, Proceedings*, pages 199–214, 2013.

[2] KNX Association. KNX Association: Network Communications Protocol for Intelligent Buildings. ISO/IEC 14543. `http://www.knx.org/`.

[3] Paul Bakker. PolarSSL Library. `https://polarssl.org`.

[4] Daniel J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, pages 207–228, 2006.

[5] Mike Burmester and Yvo Desmedt. A Secure and Efficient Conference Key Distribution System (Extended Abstract). In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 275–286, 1994.

[6] Mike Burmester and Yvo Desmedt. Efficient and Secure Conference-Key Distribution. In *Security Protocols, International Workshop, Cambridge, United Kingdom, April 10-12, 1996, Proceedings*, pages 119–129, 1996.

[7] Salvatore Cavalieri, Giovanni Cutuli, and Michele Malgeri. A study on security mechanisms in KNX-based home/building automation networks. In *Proceedings of 15th IEEE International Conference on Emerg-*

ing Technologies and Factory Automation, ETFA 2010, September 13-16, 2010, Bilbao, Spain, pages 1–4, 2010.

[8] Scott Vanstone Darrel Hankerson, Alfred Menezes. *Guide To Elliptic Curve Cryptography.* Springer, 2003.

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

[10] Schneider Electric. Modbus: Open protocol. `http://http://www.modbus.org`.

[11] International Organization for Standardization. ISO/IEC 14908-[1-4]:2012. `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=60203`.

[12] Wolfgang Granzer, Georg Neugschwandtner, and Wolfgang Kastner. EIBsec: A Security Extension to KNX/EIB. In *Konnex Scientific Conference*, November 2006.

[13] J. Merkle. Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS). Technical report, Internet Engineering Task Force (IETF), 2013.

[14] Jesus Molina. Learn How to Control Every Room at a Luxury Hotel Remotely: The Dangers of Insecure Home Automation Deployment. `https://www.blackhat.com/us-14/briefings.html#learn-how-to-control-every-room-at-a-luxury-hotel-remotely-the-dangers-of-insecure-home-automation-deployment`.

[15] M. Newman. Bacnet-a tutorial overview. `http://www.bacnet.org/`.

[16] NIST. Digital Signature Standard. Technical report, NIST, 2013.

[17] Elaine Barker William Barker William Burr William Polk and Miles Smid. Recommendations for Key Management - Part 1: General. Technical report, NIST, 2012.

[18] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *CCS '96,*

Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 14-16, 1996., pages 31–37, 1996.

[19] K. Raeburn T. Yu, S. Hartman. RFC4120 - The Kerberos Network Authentication Service (V5). Technical report, MIT, 2005.