Analysis of the PKMv2 Protocol in IEEE 802.16e-2005 Using Static Analysis

Ender Yuksel

Kongens Lyngby 2007
IMM-THESIS-2007-16

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-M.Sc: ISSN 0000000000, ISBN 00000000000

Summary

The IEEE 802.16e-2005 specification provides an air interface standard for metropolitan area wireless broadband service. IEEE 802.16 is the basis for Worldwide Interoperability for Microwave Access (WiMAX) certification which is the next evolution in wireless technology. The latest version of the standard, the IEEE 802.16e addresses mobility and also enhances the security sublayer of the IEEE 802.16 standard. Since wireless technology is broadcast and transmitted data can be intercepted, wireless users face more risks than wired users. The former IEEE 802.16 standards used the Privacy and Key Management (PKM) protocol which had many critical drawbacks. In IEEE 802.16e, a new version of this protocol called PKMv2 is released. PKMv2 has radical changes and in contrast with the previous version it seems to have an exaggerated mixture of security features like nonces, message authentication codes, key ids, certificates, etc.

The PKMv2 includes two main issues: an Authentication/Authorization protocol to establish a shared Authorization Key (AK), and a 3-Way Security Association (SA) Traffic Encryption Key (TEK) Handshake. The former is strengthened with de facto standards such as RSA and EAP, therefore the PKMv2 SA-TEK 3-Way Handshake (PKMv2 SA-TEK 3W HS), which is used for transferring TEKs to mobile stations (MS) after authentication will be the specific point of this thesis.

Static analysis is successfully used for automatically validating security properties of classical and modern cryptographic protocols. In this thesis we will show how the very same technique can be used to validate modern wireless network security protocols, in particular, we study the IEEE 802.16e PKMv2 SA-TEK 3W HS.

We derived a model of the protocol and described it using LySa, a process calculus in the pi/spi calculus family allowing communication protocols to be specified and annotated for validation of authentication properties. After that, we carried out a static analysis of our LySa model using the static analysis tool LySa-tool. Validating the base protocol, we studied our proposal on an optimized but still secure protocol. Having established systematic experiments on our models of modified versions of the protocol, we analyzed the robustness and security features. In conclusion we found improvements that increased the performance while being still secure.

Preface

This thesis was prepared at Informatics Mathematical Modelling, the Technical University of Denmark in fulfillment of the requirements for acquiring the M.Sc. degree in engineering.

The thesis deals with the static analysis of the IEEE 802.16e-2005 PKMv2 Protocol.

The thesis consists of a summary report and source codes for the experimented protocols.

Lyngby, February 2007

Ender Yuksel

Acknowledgements

I am very grateful for the advice and support from my supervisor, professor Hanne Riis Nielson, for her feedback and excellent guidance and for keeping me focused in my research.

I would like to thank Christoffer Rosenkilde Nielsen for his helps and support.

I would also like to thank professor Bulent Orencik for his support and guidance since 2003.

Last but not least, I want to thank my parents and my brother for their endless support and encouragement in my whole life.

Contents

CHAPTER 1

Introduction

Security issues in wireless networks became a growing concern with the spreading growth on wireless communication in recent years. Wireless networks face more and especially different security threats than wired networks. However, IEEE 802.16, the standard for wireless metropolitan area networks (WMAN), incorporated a pre-existing standard called Data Over Cable Service Interface Specifications (DOCSIS), which was designed for cable networks not wireless networks. Therefore, IEEE 802.16 security failed to protect the IEEE 802.16 link [1] and had significant changes in its Privacy and Key Management (PKM) protocol with the latest standard IEEE 802.16e-2005 [2].

The key distribution and management protocols which are used to establish secure communication between two principals, and authentication protocols which verify that the communicating principle is who it is supposed to be are one of the main issues that the applications of formal methods in the analysis of cryptographic protocols have been mainly concerned with. The tools that have been constructed based on the theoretical developments have successfully located subtle bugs in many cases, even in protocols that have been considered secure for

several years. One of the most famous success stories is the Lowe's attack [3, 4] on the Needham Schroeder public key protocol [5] using the process algebra Communicating Sequential Processes (CSP) and the Failures-Divergences Refinement (FDR) which is the model checker for CSP [6]. Also, Shmatikov and Stern [7] used Murphi, and Corin et al. [8] used symbolic traces and Pure-past Security - Linear Temporal Logic (PS-LTL) successfully.

In this thesis, a formal and automated method to verify the security protocol used in IEEE 802.16 is described and used. In particular, the PKMv2 SA-TEK 3-Way Handshake is studied using LySa process calculus and static analysis.

1.1 Authentication Protocols

An authentication protocol verifies the identity of principals by exchanging messages that have a specific form for authentication. These protocols usually have additional goals such as the distribution of session keys. Because of the illegitimate and/or malicious principals and active intruders, authentication requires complex protocols that are based on cryptography. The cryptographic protocols enable the principals to establish secure communications on insecure networks by using cryptographic functions and shared secrets for authentication and confidentiality.

Generally, a trusted server (i.e. Key Distribution Center) is used for authentication protocols. The principals communicate with the server to make sure that the corresponding principal is authenticated. In addition, in most of the protocols the principals agree on a session key for that specific session. Since there exists a trade-off between performance and security, the symmetric-key cryptography is used for all data traffic, and public-key cryptography is widely used for the authentication protocols themselves which aim to establish the session key. Roughly speaking, the symmetric-key cryptography is faster but less secure, whereas public-key cryptography is slower but more secure. Establishing the session key is done less frequently but it needs more security, whereas encrypting the

data traffic is done frequently but is not so critical as encrypting the session keys whose loss will affect all the encrypted data traffic.

In addition, session key is freshly created for each new connection and minimizes the amount of traffic that gets sent with confidential data like the users' secret keys or public keys, therefore reduces the amount of cipher text an intruder can obtain, and minimizes the damage in a case of intrusion. The loss of the session key is not as crucial as the loss of the secret key or any permanent key since the session key is renewed in each session [9].

Symmetric-key cryptography uses the same key for encryption and decryption. The notation between a simple encrypted communication between two principals is basically shown as:

$$A \rightarrow B : \{M\}_K$$

where principal *A* sends the message *M* to principal *B* by encrypting it with the key *K*. Certainly, *B* must possess the key *K* in order to be able to decrypt $\{M\}_K$ and read the message *M*.

Public-key cryptography or in other words asymmetric encryption is carried out using a private/public key pair e.g. $\boldsymbol{K^-/K^+}$**.** The private key is kept secret whereas the public key is common knowledge. The messages that are encrypted using the private key can only be decrypted by using the public key, so all the principals possessing the public key can decrypt them. Likewise, the messages that are encrypted using the public key can be decrypted by using the private key, so only the principal possessing the private key can decrypt them.

Description of asymmetric encryption is done in the following notation:

$$A \rightarrow B : \{|M|\}_K$$

This is the description of the scenario where principal A encrypts the message *M* using his private key K^- and sends it to the principal B. To be able to decrypt the message, the principal *B* must possess the corresponding public key K^+.

A protocol is formalized as a list of correct message transfers. For instance, the following notation in Table 1.1 describes a variant of the Wide Mouthed Frog protocol (WMF) [10]:

Table 1.1: The Wide Mouthed Frog Protocol

1.	$A \rightarrow S:$	$A, B, \{K\}_{K_A}$
2.	$S \rightarrow B:$	$A, \{K\}_{K_B}$
3.	$A \rightarrow B:$	$\{m_1, ..., m_k\}_K$

In this variant of the WMF protocol two principals *A* and *B* have shared master keys K_A and K_B with a trusted server S, and the protocol aims to establish a shared session-key *K* between two principals.

In step 1, the principal *A* initiates the protocol by sending the message $\boldsymbol{A,B,\{K\}_{K_A}}$ to the server *S*. *S* recognizes that *A* wants to arrange a secure communication with *B,* and since it possesses the shared master key K_A, it can decrypt the encrypted part of the message and also recognize that *A* wants to use *K* as a session key between *B.*

In step 2, *S* sends the message $\mathbf{A,\{K\}_{K_B}}$ to *B* using the shared master key K_B. Having the key K_B, *B* is able to decrypt the message and retrieve the session key *K.*

In step 3, since the symmetric session-key *K* is established between the two principals, *A* is now able to send a secret message $\mathbf{\{m_1,...,m_k\}_K}$ to *B* using the session key *K.*

There are also other ways of authentication without server, such as authentication based on shared-secret, and even more, authentication without neither server nor shared secret. The protocol that we study in this thesis is an example of authentication based on shared-secret whereas the Diffie-Hellman key exchange protocol [11] is a common example of authentication without using shared-secret.

1.1.1 Attacker Modelling and Scenarios

Only the correct message transfers of the protocol are described in the formalization of the protocol in the previous section, therefore it is important to be aware of the possibility of an attacker present on the network. A common way to model the ability of attackers to send and receive messages and to perform encryptions as well as decryptions on a public accessed network is to use the classical approach of Dolev and Yao [12], the notion of a “hardest attacker". This model allows the attacker to perform the following operations:

- The attacker is able to intercept any message.
- The attacker can decrypt an encrypted message if and only if he knows the key. The attacker can encrypt messages using keys in his possession. The attacker cannot guess a key.
- The attacker can construct new messages.
- The attacker can send constructed or intercepted messages on the network.

Some basic scenarios are listed below:

Deletion

The attacker can delete a message before it reaches to the receiver. This kind of attacks would halt or restart the protocol since usually timers are used in implementations.

Insertion

The attacker can send a message that is totally created by himself. This could be an initiation message, a request or a response.

Eavesdropping

Eavesdropping is possible when the attacker can intercept and read messages of the protocol. As shown in Figure 1.1, eavesdropper does not send any messages to principals nor take any messages from them, so this

is a passive attack. Encryption is used against eavesdropping since most attacks include eavesdropping to gain basic knowledge.

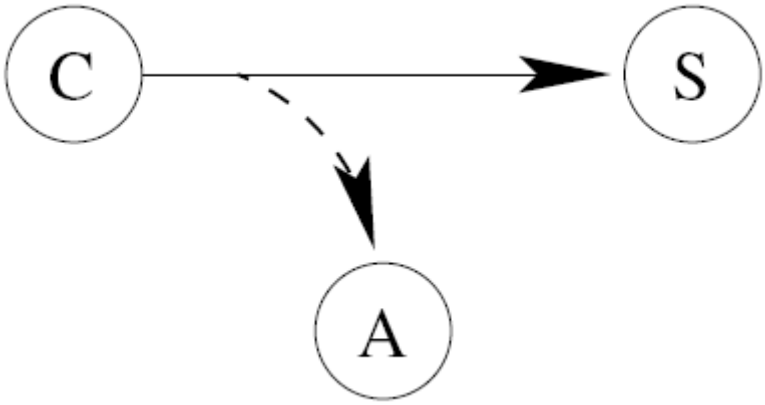

Figure 1.1: Eavesdropping

Replay Attack

After eavesdropping, the attacker could send the message that he gained to any principal in a new run of that protocol as shown in Figure 1.2. This type of attacks can be avoided by verifying freshness of the messages. Using nonces, timestamps or sequence numbers avoids replay attack.

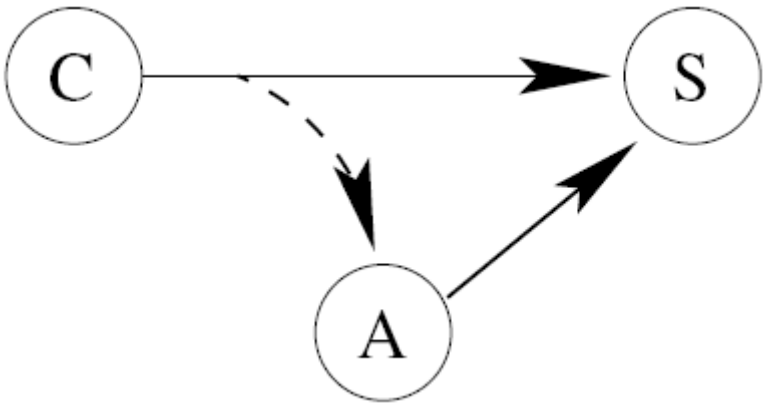

Figure 1.1: Replay Attack

Modification

This type of attacks needs interception to gain the message. Interception is different from eavesdropping since the recipient cannot receive the original message. The attacker modifies the original message and sends it to the recipient as in Figure 1.3. Encryption is not a complete solution to this problem because the message can be replaced with another (a previous one) message using the same key. To avoid modification, i.e. hashing can be used with digital signature. If a message is sent with its hash signed with the private key of the sender, then an attacker will have to posses that private key to modify the message with a valid hash value.

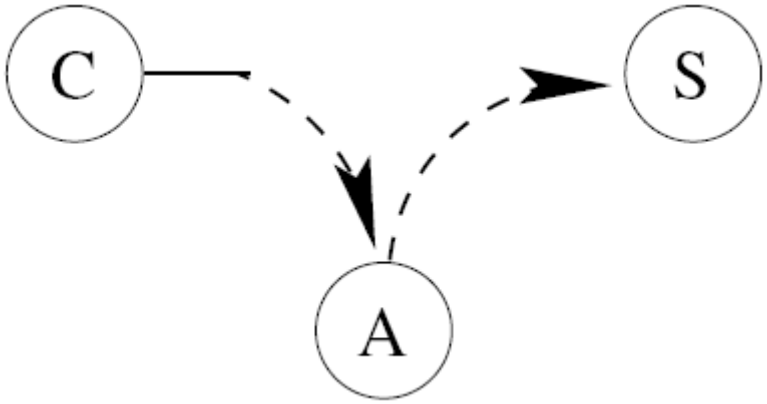

Figure 1.3: Modification

Man-In-the-Middle

In this type of attacks, the attacker works in a bidirectional manner. Namely, he uses eavesdropping and modification attacks to both of the principals in the protocol. As shown in Figure 1.4, the attacker is like the recipient and the sender of both sides. The solution of this attack is bilateral authentication which allows communicating principals to verify that received message comes from the genuine sender.

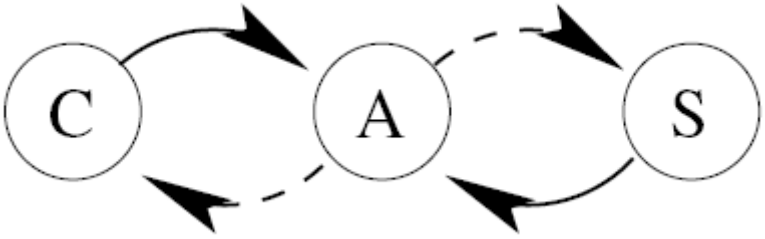

Figure 1.4: Man-in-the-middle Attack

Since an attacker is present the attacker can intercept and replay any messages in the global scenario, it is not possible to determine neither the sender nor the receiver of a message by looking at it. If the WMF protocol from the previous section is deployed on a network where an attacker is present, the following run of the protocol could occur in Table 1.2:

Table 1.2: Attack Scenario for WMF

1.	A	$\rightarrow$	$M(S)$	:	$A, B, \{K\}_{K_A}$
1.′	$M(A)$	$\rightarrow$	S	:	$A, M, \{K\}_{K_A}$
2.	S	$\rightarrow$	$M(B)$	:	$A, \{K\}_{K_M}$
3.	A	$\rightarrow$	$M(B)$	:	$\{m_1, ..., m_k\}_K$

In this message transfer, *M(S)* denotes the malicious attacker acting as *S*. The first message sent from *A* to the server is intercepted by the attacker. But the attacker cannot decrypt the session key since he does not possess the key K_A. Then the attacker changes the intercepted message by replacing *B* with his own identifier *M* and sends this message to the real server *S*. Receiving this message the server *S* believes that *A* wants to engage a secure communication with *M* which is in fact the attacker. Therefore, S encrypts the session key *K* with the master key K_M , which is shared between the attacker and the server, and sends it to the attacker. Inasmuch as the attacker got the session key, he is able to intercept and read messages sent from *A* to *B* encrypted under the session key *K*. *A* believes that the messages are to be secret between him and *B*, but in fact they are readable to the attacker.

1.2 Protocol Validation

In protocol validations, choosing the properties to be validated is an important issue. For instance, a protocol validated to be tolerant to denial of service (DoS) attacks could very well be flawed with respect to replay attacks. The most common properties to consider when validating cryptographic protocols are:

Authenticity Communication over a protocol that offers authenticity means that principals are communicating with the exact principals they believe to be communicating with. To be authenticated means to ensure that principals are actually who they say they are.

Authentication properties have been discussed in many different levels of abstraction. The authentication property studied in [13] describes authentication at the level of the individual messages used in communication. The idea is to be sure that the messages always have the intended destination and origin, no matter how an attacker interferes with communication.

Confidentiality A protocol that ensures confidentiality prevents the disclosure of transmitted data to unauthorized parties, such that only the intended receiver is able to read the confidential data. This is mostly established using cryptography.

Integrity Messages cannot be changed by any malicious user when data integrity is offered. Modification, insertion, deletion, or replay of transmitted data is detected. Hashing is a well known solution for integrity.

In addition, there are some other properties like non-repudiation [14].

Various approaches have been used in protocol validations. Formalizing protocols in some simplified programming language, process calculus or logic description and using automatic tools to verify the properties for the simplified description of the protocol is the tendency of the most recent research. The three main approaches in automatic verification are:

- **Theorem proving** The correctness of systems is determined by properties in a mathematical theory with deductive methods. Then these properties are proved using automatic tools such as theorem provers and proof checkers. As a real life example, this method is used in [15] to verify the SET protocol and in [16] for the automatic train operating system METEOR of the (first) driverless metro-line in Paris.

- **State exploration** A protocol is modelled as a finite-state system and then the verification is evaluated by exploring each state in the protocol and reporting if the protocol enters a state that violates the properties to be validated. A number of model checkers and state exploration methods have been applied to the security protocols as well. Murphi is a well-known example of this group[6,7].

- **Static analysis** An indispensable technique for language-based security which has successfully detected errors in protocols [18,13]. Control flow analysis is used to do an over-approximation of the possible variable bindings and message transfers. Constructing reference monitor semantics it is possible to know whether the properties to be validated are violated or not.

Theorem proving can deal with infinite state spaces and can verify the validity of properties for arbitrary parameter values and is a convenient method for protocols such as classical key distribution, where the reasoning about the formalization of the protocol into a logic description is relatively simple, and the assumption made prior to a run of the protocol. The main disadvantages of theorem proving are the slowness of the verification process, and the error-prone and labor-intensive character of application. Furthermore, the mathematical logic requires a rather high degree of user expertise. Although some successful applications of theorem proving, like the thorough verification of smartcard software have been reported, the drawbacks have restricted their use mainly to the academic world [17] .

Model checking and static analysis methods are similar in the sense of the usage of the reachability analysis. Confidentiality is interpreted by ensuring the secret data does not reach the attacker. Authentication is reachability in the sense that information should end up at the intended

user from the intended provider of that information. These two methods have different advantages and disadvantages. Model checking approach returns a trace of the protocol that leads to the reported error, after investigating all possible traces trough the protocol. As the length of the protocol to be analyzed increases the number of different traces through a protocol raises significantly, and if an attacker is present, the number of states is infinite which makes it hard to use the method on full scale protocols. Murphi [6] is used as a prework of this thesis and this drawback is clearly seen, though it is out of the scope of this thesis. In static analysis, it is possible to create an over-approximation of the components, without investigating all possible traces, this makes it feasible to create automatic tools for validations with the presence of an attacker. If an error is reported by the static analysis, the trace leading to the error is however not part of the result.

1.3 Strategy and Concepts

In this thesis, the security properties of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol are analyzed. This task is done in several steps:

1. Derivation of a model of the base protocol and the modified versions of the protocol, and descriptions of the protocols in the LySa process calculus.

2. Static analysis of the LySa process' which reveals potential breaches in the protocols.

3. Analysis of the result of the static analysis..

LySa process calculus [18] is the framework that the analysis of the protocols are carried out. LySa is a process calculus in the pi[19]/spi[21] calculus family and used in validation of the authentication properties of communication protocols, specifically the destination/origin authentication properties.

Static analysis [20] is the basis of our analysis. This means that we construct an approximation of the behavior of the protocol. In doing so we focus on:

1. the communications that may take place over the network
2. the potential bindings of the variables occurring in the protocol
3. the potential violations of the destination/origin annotations of the protocol

The Dolev-Yao attacker [12] is used in the analysis therefore any message sent on the network may be intercepted by the attacker, any encryption with a key known to the attacker may be decrypted by him and furthermore the attacker may make use of all the information available to him to construct new messages, even more, new encryptions and to send messages on the network. The notion of a *perfect encryption library* is used in order to be able to model encryption. Simply, an encrypted message can only be decrypted if the correct key is used.

The protocol that is chosen to be analyzed in thesis is the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake. Executed after the initial Authentication Stage or on Handover, the basic purpose of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake is the distribution of keying parameters, such as the Traffic Encryption Keys (TEK) which are encrypted using Key Encryption Keys (KEK), related to all Security Associations (SA) active between a Mobile Station (MS) and the Base Station (BS).

1.4 Structure of this Report

Chapter 2 introduces the concepts and usage of the security protocols in the IEEE 802.16 standard that we analyze in this thesis. The intention in the thesis and the aimed contribution is also stated in this chapter.

Chapter 3 presents the LySa calculus which is be used in the static analysis and the modelling of the protocol in LySa calculus.

Chapter 4 presents static analysis, the technique used in the analysis of IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol. Includes an example analysis for a simple protocol.

Chapter 5 includes the experiments and the analysis of the protocol. We modelled the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake (described in chapter 2) using the LySa calculus (described in chapter 3) and analyzed using static analysis (described in chapter 4).

Chapter 6 summarizes our work and concludes on the security aspect of the analyzed protocol.

CHAPTER 2

IEEE 802.16 Security

The IEEE 802.16 Working Group on Broadband Wireless Access Standards develops the IEEE 802.16 WirelessMAN® Standard for Wireless Metropolitan Area Networks. This standard specifies the air interface of fixed broadband wireless access (BWA) systems.

While the 802.16 family of standards is officially called WirelessMAN, it has been entitled Worldwide Interoperability for Microwave Access (WiMAX) by an industry group called the WiMAX Forum, whose mission is to promote and certify compatibility and interoperability of broadband wireless products.

The first 802.16 standard, which was designed to provide a solution for the last mile problem for Wireless Metropolitan Area Networks (WMAN) with line-of-sight (LOS) working at 10-66GHz bands, was approved in 2001 and was followed by two amendments: 802.16a and 802.16c to address issues of radio spectrum and interoperability, respectively.

In 2003, a revision project called 802.16REVd commenced aiming to align the standard with aspects of the European Telecommunications Standards Institute (ETSI) HIPERMAN standard as well as lay down conformance and test specifications. This project concluded in 2004 with the release of IEEE standard 802.16-2004 which consolidates previous standards, also supports non-line-of-sight (NLOS) within 2-11GHz bands and mesh nodes [22]. In addition, the earlier 802.16 documents including the a/b/c amendments are now superseded.

An amendment and corrigendum to the standard that aims to provide mobility in BWA and presents new security protocols was concluded in 2005 and named as IEEE 802.16e-2005 [2].

Two types of principals communicate in IEEE 802.16 and since IEEE 802.16e-2005 comes up with mobility, the client principal which was called as the subscriber station (SS) in the previous versions is now called the mobile station (MS) and the other principal who acts as the server is still called the base station (BS).

With the entry of the MS to the network, using the ranging protocol, the communication starts. The purpose of the ranging protocol is to set up the physical communication parameters and to assign a basic connection identifier to the requesting MS. Later, the ranging protocol is periodically executed to recommunicate the physical communication parameters [23].

After that, the registration protocol is performed in order to allow the mobile station into the network. BS and MS' security capabilities are negotiated during the registration protocol. The stations may agree on authentication and key management protocols. Authentication options are: unilateral authentication, mutual authentication and no authentication. The mutual authentication was missing in the previous versions and it was one of the problems that were mentioned in the related papers such as [1] but now it is included in IEEE 802.16e-2005. Key management protocols are focused on this thesis and are described in details in the following sections.

The key management protocols are periodically executed to update the Traffic Encryption Keys (TEK) which can be thought as the session keys. After the establishment of the TEKs, user data protocols start. Traffic encryption keys are used as sequential pairs and have overlapping lifetimes to avoid service interruptions.

The authentication and key management protocols are specified in the security sublayer of IEEE 802.16 standard. The security sublayer is meant to provide subscribers with privacy and authentication and operators with strong protection from theft of service. The security sublayer consists of two component protocols, an encapsulation protocol for securing packet data across the network and a key management protocol providing the secure distribution of keying data from the base station to the mobile station. In the following sections we will focus on the key management protocol. The Privacy and Key Management (PKM) protocol of IEEE 802.16 and the second version of this protocol, which is announced within the IEEE 802.16e-2005 and aims to fix the bugs in the former protocol, are described in the following sections.

2.1 Overview of PKMv1 (IEEE 802.16-2004)

The first version of the Privacy and Key Management Protocol consists of two specific components, which are designed for IEEE 802.16 and defined in Security Sublayer. The first protocol is the PKM Authorization Protocol which is established by the subscriber station (SS) and responded by the base station (BS). As we mentioned before, until the PKMv2 announced the standard was not mobile and therefore we use the notation SS instead of MS (mobile station). At the end of a successful run of this protocol, an Authorization Key (AK) is created by BS and transmitted to SS. After that, each party generates a Key Encryption Key (KEK) using their AK. KEKs are used in encrypting and distributing Traffic Encryption Keys (TEK), TEKs can be taken as session keys, while AK/KEK are long term keys. Then comes the second part: the Privacy and Key Management protocol which lets SS to gather TEKs from BS, note that TEKs are encrypted by KEKs. The flow of the protocols in PKMv1 can be seen in Figure 2.1.

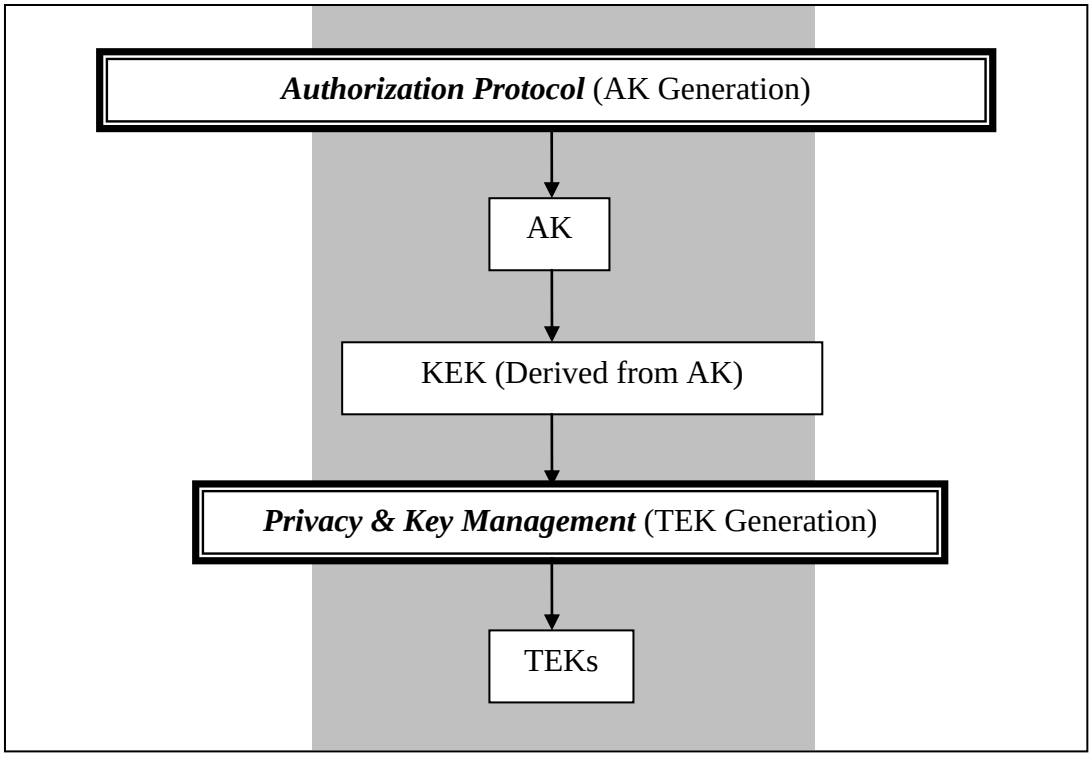

Figure 2.1: The PKMv1 Protocols

2.1.1 PKM Authorization

The PKM authorization protocol aims to distribute an authorization key (AK) to an authorized SS. The authorization protocol is a three-message exchange between an SS and a BS; but it is not in a one after the other manner since the first two messages are sent by the SS. When successful BS responds with the third message, which is actually the transmission of the AK from BS to SS. The messages can be seen in Table 2.1.

Table 2.1: The PKM Authorization Protocol

Message 1: ***Authentication Information Message***
SS → BS: Certificate(Manufacturer(SS))
Message 2: ***Authorization Request***
SS → BS: Certificate(SS) | Capabilities | SAID
Message 3: ***Authorization Reply***
BS → SS: RSA-Encrypt(PubKey(SS), AK) | Lifetime | SeqNo | SAIDList

SS uses Message 1, formally named as the Authentication Information Message, to push its X.509 certificate which identifies its manufacturer to BS. BS uses this certificate to decide whether SS is a trusted device. BS may use this message in order to allow access only to devices from recognized manufacturers, according to its security policy.

SS sends Message 2, named as the Authorization Request immediately after Message 1. Message 2 consists of SS's X.509 certificate with the SS public key, its security capabilities which are actually the authentication and encryption algorithms that SS support, and the security association identity (SAID) which is the id of the secure link between SS and BS. Using the certificate, BS determines whether to authorize SS; and the public key of SS which is also in the certificate lets BS construct Message 3.

If successful, namely SS is authorized after BS verifies its certificate, BS responds with Message 3, the Authorization Reply. This message includes the AK, encrypted using the RSA public-key encryption protocol using the public-key of SS which was obtained in the previous message, the lifetime of the AK as a 32-bit unsigned number in unit of seconds, the sequence number for AK as a 4-bit value and the list of SA descriptors each including an SAID and the SA cipher suit. The successful run of the protocol instantiates an authorization SA between the two stations. The design assumes that only BS and SS possess the

AK, which means that the key is confidential and never revealed to any other party.

2.1.2 Privacy and Key Management

The privacy and key management protocol aims to establish a data SA between BS and SS. The first message of the protocol is optional and used for forcing rekeying, therefore the protocol is a two or three message exchange between SS and BS. When successful, the BS sends TEKs to the SS in the last message of the protocol. The messages in the protocol can be seen in the Table 2.2.

Table 2.2: The Privacy and Key Management Protocol

[Message 1: BS → SS: SeqNo | SAID | HMAC(1)]
Message 2: ***Key Request***
SS → BS: SeqNo | SAID | HMAC(2)
Message 3: ***Key Reply***
BS → SS: SeqNo | SAID | OldTEK | NewTEK | HMAC(3)

If BS wants to rekey a data SA or create a new SA, it starts the protocol with the first message which contains the sequence number of the AK used for the exchange, the id of the data SA being created or rekeyed and HMAC-SHA1 digest of these two fields. Computation of the Hashed Message Authentication Code HMAC(1) requires a HMAC key which is derived from the AK, therefore it allows SS to detect forgeries.

The second message, named as the Key Request, is where SS requests the SA parameters. If the protocol was started by BS, SS takes SAID from message 1 with valid HMAC(1). Otherwise, SS takes SAID from the authorization protocol SAIDList. Then HMAC is computed with the sequence number of AK and the SAID.

The third message, the Key Reply, is sent if the HMAC and the SAID in message 2 is valid. As mentioned in the beginning of this chapter, TEKs have overlapping lifetimes to avoid service interruptions. The OldTEK value has the active SA parameters whereas the NewTEK value has the SA parameters to be used on the expiry of the current TEK. OldTEK includes the initialization vector, remaining lifetime and sequence number for the specified data SA for the previous generation TEK, and similarly NewTEK includes the same parameters for the next TEK. The TEKs are encrypted with 3-DES using the Key Encryption Key (KEK) which is derived from the AK. This message also has HMAC to avoid forgeries.

2.2 Overview of PKMv2 (IEEE 802.16e-2005)

The second version of the Privacy and Key Management (PKM) protocol of IEEE 802.16 is described in IEEE 802.16e-2005 and aims to fix the bugs in the former version.

The AK derivation is now established by the well known standards RSA and EAP. In PKMv2, RSA and EAP can be used in different ways which are defined in the standard [2], such as RSA, RSA+EAP, EAP and EAPinEAP. Therefore the AK derivation is now much more specific and with the contribution of two principals much more secure. In addition BS now has a certificate, and can authenticate itself to the MS by mutual authentication which was missing in PKMv1. Nonces are used against replay attacks. The process can be seen in Figure 2.2.

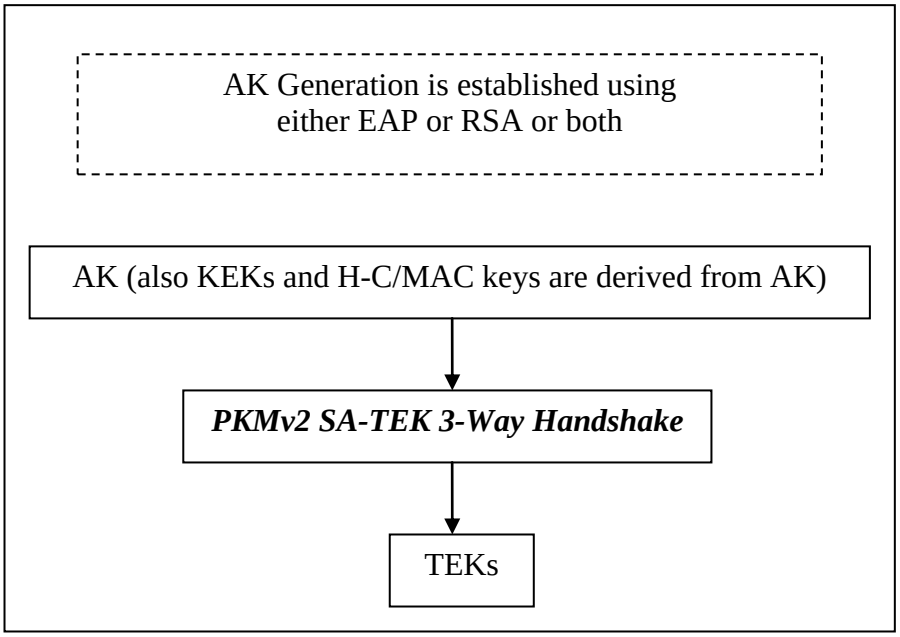

Figure 2.2: The PKMv2 Protocols

The important part of PKMv2 is the SA-TEK 3-Way Handshake. It is based on the second part of the former protocol, but now it has more security features. The original specification has three messages with H-MACs and in total twenty-one fields. The main fields are described in Table 2.3.

Table 2.3: The PKMv2 Protocols

Attribute	***Content***
MS_Random	Number received from MS
BS_Random	Number included in SA-TEK-Challenge or SA-Challenge
KeySeqNo	AK Sequence Number

AKID	Id of the AK that was used for protecting this message
SA-TEK-Update	TEKs encrypted by KEKs
Frame No	The frame number that old PMKs and associated AKs should be discarded
SA_Descriptors	Only for initial entry
SecNegParam.	Confirms messages security capabilities
HMAC/CMAC	Message Authentication Codes (Hashed/Cryptographic)

The PKMv2 SA-TEK 3-Way handshake sequence proceeds as shown in Figure 2.3.

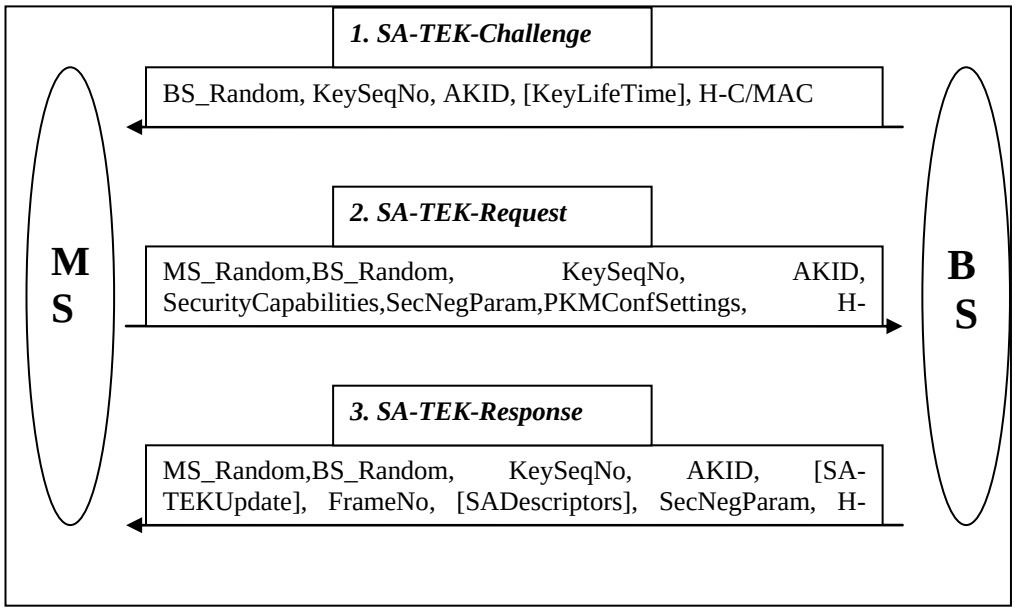

Figure 2.3: The PKMv2 SA-TEK 3-Way handshake

The first message, named as PKMv2 SA-TEK-Challenge, includes a random number generated by BS and similar to the previous version protected by HMAC/CMAC tuple.

The second message is the PKMv2 SA-TEK-Request and includes the random number generated by MS, the random number of BS received in the first message, and the similar fields as in the previous version of the protocol, just as described in the previous section.

The BS checks the AKID, HMAC/CMAC (Hashed-MAC/Cryptographic-MAC) and the BS_Random of the message 2 and if any of these values are invalid, than ignores the message. Otherwise, it checks the security capabilities provided by the MS and if the properties does not match it reports this inconsistency to the higher layers.

If the second message is successfully validated by the BS then message 3 which is named as the PKMv2 SATEK-Response is sent to MS. This message has the SA-TEKUpdate unless for the handover and the security capabilities that BS wishes to specify for the session with the MS.

If the last message is successfully verified by MS using the HMAC/CMAC, the received TEKs and associated parameters will be installed by the MS. The security negotiation parameters of BS should also be verified by MS but the failure of this verification may not cause halt of the protocol since MS may continue by adopting the security negotiation parameters encoded in SA-TEK Response message.

2.3 Overview of Contribution

The PKMv1 was defined in IEEE 802.16-2004 and it had many problems and flaws in it which are mainly discussed in [1]. For example, the data SA (Security Association) was explicitly defined but the Authorization SA was not. The SS had an X.509 certificate, but BS did not. BS did not even authenticate itself to the SS. Even the IV (initial vector) in the encryption phase was predictable. Therefore, IEEE 802.16 PKMv1 did not provide any data authenticity. Besides, the rogue AP problem in

802.11 Wireless Local Area Networks [24] was still existing in the sense that there was no BS identity in Authorization SA and there could be rogue BS'. Furthermore, the TEK identifiers were only 2-bits in length, Data Encryption Standard-Cipher Block Chaining (DES-CBC) was not a convenient way of encryption, the AK derivation was only BS's job and SS did not have chance to participate it, and because security features were used against replay attacks.

PKMv2 left the first part of PKMv1, namely the AK derivation to the well known standards RSA and EAP. In fact, RSA already existed in the PKMv1 but in PKMv2 RSA and EAP can be used in different ways which are defined in the standard such as RSA, RSA+EAP, EAP and EAPinEAP. Therefore the AK derivation is now much more specific and with the contribution of two participation much more secure. In addition BS now has a certificate, and can authenticate itself to the MS by mutual authentication. Nonces are used against replay attacks.

PKMv1 had many missing parts in it, but PKMv2 is over-strengthened. This does not mean that PKMv2 can be considered as the ultimate secure protocol, but it definitely has degraded efficiency since it needs more sources and time for the security features it has. The aim of this thesis is to argue that, PKMv2 can still pursue its security with less features than it has. In other words, PKMv1 was a failure in wireless security, just like the WEP in IEEE 802.11 [31], so PKMv2 is now overloaded, but a light version of PKMv2 should serve as good as now it is.

Our approach is to see the limits of robustness in IEEE 802.16 PKMv2. The way we do it is removing the extras and the improvements in PKMv2 one by one, and in different combinations. We want to see when the robustness will be lost, what preserves the robustness and how is this accomplished. We also want to see if some improvements are unnecessary then what are they, and can we provide better efficiency with less strength? The result may lead us to a simplified by still strong and secure protocol.

The experiments could be established by constructing the PKMv2 SA-TEK 3-Way Handshake beginning from the simple PKMv1. In order to see where the problems arise and where the flaws start, the experiments are held in the reverse direction, namely from the full protocol to a simpler but still secure revised protocol.

2.4 Specifying IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake

Obviously, the description of IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake contains many details that won't be used in modelling. Not all the fields make the protocol secure, so there should be a simplification such as removing the fields which have no effect on security itself.

John Mitchell [25] simplified the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake to make a formal verification with Murphi as shown in Table 2.4. This work was used in a security review together with IETF EAP Work Group.

Table 2.4: Mitchell's Simplified version of the PKMv2 SA-TEK 3-Way HS

1.	BS → MS :	BSNonce, AKID, MIC[AK](BSNonce, AKID)
2.	MS→ BS :	BSNonce, MSNonce, AKID, MSSuite, MIC[AK](MSNonce, BSNonce, AKID, MSSuite)
3.	BS → MS :	SAUpdate, BSNonce, MSNonce, AKID, MIC[AK](SAUpdate, MSNonce, BSNonce, AKID)

Rewriting the simplified PKMv2 SA-TEK 3-Way Handshake in a more familiar protocol narration style is shown in Table 2.5.

Table 2.5: The Simplified PKMv2 SA-TEK 3-Way Handshake

1. BS → MS: N_{BS}, AKID, MIC{ N_{BS}, AKID}$_{AK}$
2. MS→ BS : N_{BS}, N_{MS}, AKID, MSSuite, MIC{ N_{BS}, N_{MS}, AKID, MSSuite}$_{AK}$
3. BS → MS: SAUpdate, N_{BS}, N_{MS}, AKID, MIC{ SAUpdate, NBS, NMS, AKID}$_{AK}$

Using the abbreviations below we can have a shorter narration:

BS = A, MS = B, $N_{BS} = N_A$, $N_{MS} = N_B$, MSSuite = S, SAUpdate = T, AK = K, AKID = Id

Mitchell uses MIC (Message Integration Code) instead of MAC (Message Authentication Code), we will use the notation MAC as used in the standard.

The simplified and abbreviated version of PKMv2 SA-TEK 3-Way Handshake is shown in Table 2.6.

Table 2.6: Simplified and Abbreviated Version of PKMv2 SA-TEK 3-Way HS

1. A → B: N_A, ID, MAC{ N_A, Id}$_K$
2. B → A: N_A, N_B, Id, S, MAC{ N_A, N_B, Id, S}$_K$
3. A → B: T, N_A, N_B, Id, MAC{ T, N_A, N_B, Id}$_K$

In order to use this handshake specification in LySa, some parameters should be reordered and for simplicity some of them need to be reabbreviated. This reordering does not affect the security features but it is needed for LySa which will be explained in the following chapter.

The reordered, simplified and reabbreviated PKMv2 SA-TEK 3-Way Handshake protocol narration is shown in Table 2.7, named as Pkmv2-simple.protocol.

Table 2.7: Pkmv2-simple.protocol

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$
3. A → B: na, nb, id, T, MAC{ na, nb, id, T}$_K$

2.5 Considerations in Modelling

In our specification of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake in section 2.4, we have six different fields. First of all, every message has a nonce which are freshly generated values: ***na***, ***nb***. Then comes the authorization key (AK) which is a shared-secret long term key: ***K***. This key also has an id which is used in every message: ***id***. The second message, which is actually called the *SA-TEK-Request*, includes a field that provides information about the security capabilities of the MS: ***S***. Last message, which is actually called the *SA-TEK-Response*, includes the session keys (TEKs) which are encrypted by special keys that are generated from AK (KEK: Key Encryption Key): ***T***. Finally, all the messages have message authentication codes which are generated from the whole message and using AK: ***MAC***.

In the modelling phase of these entities, three major studies are taken as guidelines. Nonces are modelled as they are modelled by Buchholtz's implementation of The Bauer, Berson, and Feiertag (BBF) protocol which aims at establishing a fresh shared key, between two principals using nonces [26].

The long term key and the id of it are modelled as they are modelled in the impressive study about static validation which is also the basis for this thesis [13]. Wide Mouthed Frog protocol [21] (WMF) which aims at establishing a secret (symmetric) session key between two principals who share master keys with a trusted server, has an implementation in this study which includes the long term key usage that can be used for *K* in our specification of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake and also the id of *K*.

The encrypted session key can be taken as an ordinary message field, like in Mitchell's work [25]. In addition, the security capabilities field is no doubt an ordinary field for us, therefore these two fields *S* and *T* will be modelled as they are in modelled in WMF.

The modelling of the message authentication codes is a difficult problem but in the SAML-TLS implementation in [27] comes a clever solution for the problem which is described in Chapter 3, Section 3.2.2.

The summary of the considerations in modelling and the basis implementations are given in the Table 2.8.

Table 2.8: Modelling Summary

Field	***Definition***	***Implementation***
Nonces	Freshly generated	BBF
K	Long term key	WMF
Id	Long term key id	WMF
T	Encrypted session key	WMF (as a message)
S	Security Capabilities	WMF (as a message)
MACs	Message Authentication Codes	SAML-TLS

CHAPTER 3

LySa

This chapter is about LySa [13], a process calculus based on the π-calculus [28] and incorporates cryptographic operations using ideas from the Spi-calculus [21]. Though, there are two main differences between LySa and spi/pi calculus. First difference is that, LySa does not have channels but one global ether. That is because in usual implementations like ethernet-based or wireless, anyone can eavesdrop or act as an active attacker and that's definitely not the channel based communication. The second difference is about the usage of pattern matching in the expression of the tests associated with input and decryption.

3.1 LySa Calculus

To analyze the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol we need to formalize it in LySa calculus. The distinguishing features of LySa can be summarized as: LySa has only one global

communication channel or network. Everyone in the network can see the messages between processes. The LySa calculus has primitives for symmetric and asymmetric cryptography. Besides, decryption is modelled using pattern matching.

3.1.1 Syntax

LySa consists of terms and processes; terms consist of names (keys, nonces, messages, etc.), variables, public/private keys and the compositions of them using symmetric/asymmetric encryptions. The syntax of terms E is shown in Table 3.1:

Table 3.1: LySa Terms

E ::=	*terms*	
	n	name ($n \in \mathcal{N}$)
	x	variable ($x \in \mathcal{X}$)
	k^+, k^-	public and private keys
	$\{ E_1,...,E_k \}^{\ell}_{E_0}$ [dest $\mathcal{L}$]	symmetric encryption (k ≥ 0)
	$\{\| E_1,...,E_k \|\}^{\ell}_{E_0}$ [dest $\mathcal{L}$]	asymmetric encryption (k ≥ 0)

In Table 3.1, $\mathcal{N}$ denotes the sets of names and $\mathcal{X}$ denotes the sets of variables. Tuples of terms $E_1,...,E_k$ are encrypted under a term E_0 representing a key in the cases of symmetric or asymmetric encryption. An assumption of perfect cryptography is adopted, meaning that the only inverse function of encryption is to use decryptions with the correct key.

The syntax of processes P which is mostly familiar to the polyadic Spi-calculus [21] is shown in Table 3.2:

Table 3.2: Processes

P ::=	*processes*	
	0	nil
	$\langle E_1,...,E_k \rangle.P$	output
	$(E_1,...,E_j\ ;\ x_{j+1},...,x_k).P$	input (with matching)
	$P_1 \mid P_2$	parallel composition
	$(\nu\ n)P$	restriction
	$!P$	replication
	decrypt E as $\{E_1,...,E_j\ ;\ x_{j+1},...,x_k\}^{\ell}_{E_0}$ [orig $\mathcal{L}$] in P	
		symmetric decryption (with matching)
	decrypt E as $\{\|E_1,...,E_j\ ;\ x_{j+1},...,x_k\|\}^{\ell}_{E_0}$ [orig $\mathcal{L}$] in P	
		asymmetric decryption ($k \geq 0$)

The input operation with pattern matching will only succeed if the prefix of the message matches the terms specified before semi-colon in the input operation. The input process $(E_1,...,E_j\ ;\ x_{j+1},...,x_k).P$ means that a k-tuple of values $(E_1^{'},...,E_k^{'})$ is taken as the input and if the first $1 \leq i \leq j$ values $E_i^{'}$ are pairwise matched to the values E_i, the remaining k-j values of the input will be binded to the variables $x_{j+1},...,x_k$. In other words, the values before the semi-colon are to matched to the beginning part of the input and if the matching is successful the remaining part of the input will be assigned to variables after the semi-colon. This pattern matching is also used in decryptions as shown in table 3.2. If no matching will be performed, then nothing is written before the semi-colon. Similarly, if no binding will be performed, then nothing is written after the semi-colon. For example,

$$\text{P = decrypt } \{y\}_K \text{ as } \{x;\}_K \text{ in P'}$$

means that the decryption in P succeeds only if $x = y$ whereas

$$\text{Q = decrypt } \{y\}_K \text{ as } \{;x\}_K \text{ in Q'}$$

means that the decryption in Q always succeeds, binding x to y.

LySa syntax also have annotations for origin and destination in order to describe the intentions of the protocols. Encryptions can be annotated

with fixed labels, called *crypto-points* defining its position in the process, and with *assertions* specifying the origin and destination of encrypted messages. Crypto-points ℓ are from some enumerable set C (disjoint from N and X) and added to state where the encryptions and decryptions occur. The LySa term for encryption:

$$\{ E_1,...,E_k \}^{\ell}_{E_0} \textbf{ [dest } \mathcal{L}\textbf{]}$$

means that the encryption is created at crypto-points ℓ and specifies the intended crypto-points $L \subseteq C$ for decryption of the encrypted value in the assertion **[dest $\mathcal{L}$]**

Similarly, in the LySa term for decryption:

$$\textbf{decrypt } E \textbf{ as } \{E_1,...,E_j ; x_{j+1},...,x_k\}^{\ell}_{E_0} \textbf{ [orig } \mathcal{L}\textbf{] in } P$$

[orig $\mathcal{L}$] specifies the crypto-points $L \subseteq C$ that E is allowed to have been encrypted.

For the terms with all annotations removed $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$ is used, and in particular:

$$\lfloor\!\lfloor \{ E_1,...,E_k \}^{\ell}_{E_0} \textbf{ [dest } \mathcal{L}\textbf{]} \rfloor\!\rfloor = \{ \lfloor\!\lfloor E_1 \rfloor\!\rfloor,..., \lfloor\!\lfloor E_k \rfloor\!\rfloor \}^{\ell}_{\lfloor E_0 \rfloor} \textbf{ [dest } \mathcal{L}\textbf{]}$$

In addition, for each name n there is a canonical representative $\lfloor n \rfloor$ and similarly, the function $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$ is extended homomorphically to terms: $\lfloor E \rfloor$ is the term where all names and variables are replaced by their canonical versions.

3.1.2 Semantics

This section gives a short description of the reduction semantics defined for LySa following the tradition of the π -calculus. We use the notation of $P[E/x]$ to describe that all occurrences of x in process P should be replaced by the term E, in other words the value of E is bound to variable x in P. In addition, names used in a LySa process are global, for instance if a name “X" occurs in two places in the process they have the same

meaning. Therefore, it is impossible to use local variables and each name should only be used in one meaning.

As described in the previous section, all occurrences of a bound name *n* is mapped to one canonical name $\lfloor n \rfloor$ and the same mapping applies for variables. The function applied to terms $\lfloor E \rfloor$ replaces all names and variables in the term with their canonical versions. We say that two processes are α -equivalent only if the mapping of names and variables correspond.

Structural congruence, ≡, is defined on processes to be the least congruence satisfying the following conditions:

- $P \equiv Q$ if P and Q are disciplined α-equivalent;
- $(\mathcal{P} / \equiv, |, 0)$ is a commutative monoid:
  - $P \mid Q \equiv Q \mid P$
  - $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$
  - $P \mid \mathbf{0} \equiv P$
- $(\nu\ n)\mathbf{0} \equiv \mathbf{0}$,

  $(\nu\ n)\ (\nu\ n')P \equiv (\nu\ n')\ (\nu\ n)P$, and

  $(\nu\ n)\ (P|Q) \equiv P|(\nu\ n)Q$ if $n \notin fn(P)$;
- $!P \equiv P \mid !P$

We consider two variants of *reduction relation* $\rightarrow_R$: the reference monitor semantics ($\rightarrow_{RM}$) takes advantage of annotations, whereas the standard semantics ($\rightarrow$) discards them. After the reduction semantics we will describe the reference monitor semantics in details.

The rules for the reduction semantics $\rightarrow_R$ are shown in table 3.3 and described below:

Communication

The rule Communication expresses that an output $\langle E_1,...,E_k\rangle.P$ is matched by an input $(E_1^{'},...,E_j^{'};x_{j+1},...,x_k).Q$ if the first j elements are pairwise the same, namely E_i with all annotations removed is compared with $E_i^{'}$ with all its annotations removed. If these comparisons are successful, rest of the terms each E_{j+1} ,..., E_k is bound to the variables $x_{j+1},...,x_k$.

Decryption / Asymmetric Decryption

The rule Decryption expresses matching the term $\{ E_1,...,E_k \}^{\ell}_{E_0}$ [dest $\mathcal{L}$], which is a result of an encryption, against the pattern in decrypt E as $\{E_1,...,E_j ; x_{j+1},...,x_k\}^{\ell}_{E_0}$ [orig $\mathcal{L}$] in P if the key used for decryption corresponds to the one used to create the encrypted term. This is accomplished by adding the condition $[\![E_0]\!] = [\![E_0']\!]$ in addition to the case for communication which required the first j components to be pairwise the same. This models perfect symmetric cryptography. If the matching is successful rest of the terms are binded to the variables as in the previous rule. In the case of asymmetric decryption, the decryption key should be the opposite of the encryption key, namely $\{E_0, E_0^{'}\}=\{m^{+},m^{-}\} \vee \{E_0, E_0^{'}\}=\{m^{-},m^{+}\}$ which is shortly expressed by $\{E_0, E_0^{'}\}=\{m^{\pm},m^{\mp}\}$

In the reference monitor semantics we ensure that the crypto-point of the encrypted value is acceptable at the decryption (i.e. $\ell \in \mathcal{L}'$) and the crypto-point of the decryption is acceptable for the encryption (i.e. $\ell' \in \mathcal{L}$). But in the standard semantics the condition $R(\ell,\mathcal{L}', \ell', \mathcal{L})$ is universally true and therefore can be ignored.

Parallel

The rule for parallel construction is standard; using the reduction semantics, two parallel processes $P \mid Q$ are reduced on either one of them.

Restriction / Asymmetric Restriction

The rule for restriction $(\nu\ n)P$ applies the reduction semantics on the restricted processes. In the case of asymmetric restriction, the same rule applies on asymmetric keys.

Congruence

The rule for congruence expresses that, if the reduction semantics are applied then the two congruent processes $P \equiv Q$, are reduced to two congruent processes $P' \equiv Q'$.

Table 3.3: Operational semantics

(Communication)

$$\frac{\wedge_{i=1}^{j} \llbracket E_i \rrbracket = \llbracket E_i' \rrbracket}{\langle E_1, \cdots, E_k \rangle.\, P \ \mid\ (E_1', \cdots, E_j';\, x_{j+1}, \cdots, x_k).\, Q \rightarrow_{\mathcal{R}} P \ \mid\ Q[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]}$$

(Decryption)

$$\frac{\wedge_{i=0}^{j} \llbracket E_i \rrbracket = \llbracket E_i' \rrbracket \quad \wedge \quad \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{\begin{array}{c}\mathsf{decrypt}\ \{E_1, \cdots, E_k\}_{E_0}^{\ell}[\mathsf{dest}\ \mathcal{L}]\ \mathsf{as}\ \{E_1', \cdots, E_j';\, x_{j+1}, \cdots, x_k\}_{E_0'}^{\ell'}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P \\ \rightarrow_{\mathcal{R}} P[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]\end{array}}$$

(Asymmetric-Decryption)

$$\frac{\wedge_{i=1}^{j} \llbracket E_i \rrbracket = \llbracket E_i' \rrbracket \quad \wedge \quad \{E_0, E_0'\} = \{m^{\pm}, m^{\mp}\} \quad \wedge \quad \mathcal{R}(\ell, \mathcal{L}', \ell', \mathcal{L})}{\begin{array}{c}\mathsf{decrypt}\ \{\!|E_1, \cdots, E_k|\!\}_{E_0}^{\ell}[\mathsf{dest}\ \mathcal{L}]\ \mathsf{as}\ \{\!|E_1', \cdots, E_j';\, x_{j+1}, \cdots, x_k|\!\}_{E_0'}^{\ell'}\ [\mathsf{orig}\ \mathcal{L}']\ \mathsf{in}\ P \\ \rightarrow_{\mathcal{R}} P[E_{j+1}/x_{j+1}, \cdots, E_k/x_k]\end{array}}$$

(Parallel)

$$\frac{P \rightarrow_{\mathcal{R}} P'}{P \mid Q \rightarrow_{\mathcal{R}} P' \mid Q}$$

(Restriction)

$$\frac{P \rightarrow_{\mathcal{R}} P'}{(\nu\, n)\, P \rightarrow_{\mathcal{R}} (\nu\, n)\, P'}$$

(Asymmetric-Restriction)

$$\frac{P \rightarrow_{\mathcal{R}} P'}{(\nu_{\pm}\, m)P \rightarrow_{\mathcal{R}} (\nu_{\pm}\, m)P'}$$

(Congruence)

$$\frac{P \equiv Q \ \wedge\ Q \rightarrow_{\mathcal{R}} Q' \ \wedge\ Q' \equiv P'}{P \rightarrow_{\mathcal{R}} P'}$$

Reference Monitor Semantics

In reference monitor semantics, the reduction rules of the semantics are the same but instead of defining the *R* relation, the reference monitor takes this relation as input: RM(ℓ,$\mathcal{L}$', ℓ', $\mathcal{L}$) = ($\ell \in \mathcal{L}' \wedge \ell' \in \mathcal{L}$) which means that the encryption made at ℓ must be in the set $\mathcal{L}$' of expected origins of the data, as well as the actual place where decryption takes place ℓ' must be in the set of expected destinations $\mathcal{L}$. In other words, decryptions may only occur at crypto-points specified in the corresponding encryption and vice-versa, otherwise the execution is halted.

3.2 Modelling Protocols in LySa

The translation from ordinary protocol narration into a LYSA process is done in two stages:

1. The ordinary protocol narration is refined into an extended protocol narration.

2. The extended protocol narration is translated into LySa.

A discussion on the need for extending the ordinary protocol narration can be found in [29].

3.2.1 Extended Protocol Narrations

As shown in Section 1.1, the protocol narrations only list the messages to be exchanged. Here is the first message of the WMF protocol which was also used in Section 1.1:

$$A \rightarrow S : A,B,\{K\}_{K_A}$$

This line means that, the message containing *A, B* and *K* encrypted with the key K_A is sent from *A* to *S*.

To formalize the protocols to be analyzed, we have to use an extended version of this notation. The extended protocol narration distinguishes between inputs and corresponding outputs and also makes clear which checks must be performed [13]. The protocol narrations only list the messages to be exchanged, therefore the actions to be performed upon receiving the messages are left unspecified.

The first step of unfolding the protocol narrations to the extended protocol narrations is to distinguish between outputs and the corresponding inputs. This is done also for encryptions and corresponding decryptions. In addition, the check on the received values and the freshness of the keys should be explicitly stated. In addition, the source and destination addresses may be added to the messages. Using this extension the first step of the WMF protocol would now be split into three parts:

$$
\begin{array}{lll}
1. & A \rightarrow \quad : A, S, A, B, \{K\}_{K_A} & \\
1.' & \quad \rightarrow S : x_A, x_S, x'_A, x_B, x_{Mess}, & [\texttt{check}\ x_S = S, x_A = x'_A] \\
1.'' & \quad \rightarrow S : \texttt{decrypt}\ x_{Mess}\ \texttt{as}\ \{x_{Key}\}_{K_A} &
\end{array}
$$

First line consists of the message sent from *A* with the source and destination addresses added as a prefix. Second line has the variables that are bound to the received messages fields. In the end of the second line exists the checks in the brackets. First check is to make sure that the message is really sent for *S* and the second check is to make sure that the sender of the message is the one in the first part (in extended narration third part) of the message (the one that wishes to communicate). Third line shows the decryption of the encrypted value using the key K_A and as a result x_{Key} is bound to *K*.

The last step is about the security goals namely the authenticity properties to be verified. The protocol narration is refined by specifying the origin and destination of encrypted messages. This will help us to be sure of the confidential data is sent and received by the principals intended by the protocols. So the final result for the extended protocol narration of the WMF protocol is given below:

$$
\begin{array}{lll}
1. & A \rightarrow & : A, S, A, B, \{K\}_{K_A}\,[\texttt{dest}\ \{S\}] \\
1.' & \rightarrow S & : x_A, x_S, x'_A, x_B, x_{Mess} \quad [\texttt{check}\ x_S = S, x_A = x'_A] \\
1.'' & & S : \texttt{decrypt}\ x_{Mess}\ \texttt{as}\ \{x_{Key}\}_{K_A}\,[\texttt{orig}\ \{A\}]
\end{array}
$$

The annotations in brackets including the tags **dest** and **orig** means that, the encrypted message sent in line 1 should only be decrypted at the principal S and the decrypted (part of the) message in line 1" should have been encrypted at principal A.

$$
\begin{array}{lll}
1. & A \rightarrow S : & A, B, \{K\}_{K_A} \\
2. & S \rightarrow B : & A, \{K\}_{K_B} \\
3. & A \rightarrow B : & \{m_1, ..., m_k\}_K
\end{array}
$$

The protocol narration of the WMF protocol is given above and the extended narration of this protocol is given in Table 3.4 below:

Table 3.4: The Extended protocol narration of the WMF protocol

$$
\begin{array}{lll}
1. & A \rightarrow & : A, S, A, B, \{K\}_{K_A}\,[\texttt{dest}\ \{S\}] \\
1.' & \rightarrow S & : x_A, x_S, x'_A, x_B, x_{Mess} \quad [\texttt{check}\ x_S = S, x_A = x'_A] \\
1.'' & & S : \texttt{decrypt}\ x_{Mess}\ \texttt{as}\ \{x_{Key}\}_{K_A}\,[\texttt{orig}\ \{A\}] \\
2. & S \rightarrow & : S, x_B, x_A, \{x_{Key}\}_{K_B}\,[\texttt{dest}\ \{B\}] \\
2.' & \rightarrow B & : y_S, y_B, y_A, y_{Mess} \quad [\texttt{check}\ y_B = B] \\
2.'' & & B : \texttt{decrypt}\ y_{Mess}\ \texttt{as}\ \{y_{Key}\}_{K_B}\,[\texttt{orig}\ \{S\}] \\
3. & A \rightarrow & : A, B, \{m_1, ..., m_k\}_K \\
3.' & \rightarrow B & : y'_A, y'_B, y_{Messages} \quad [\texttt{check}\ y'_B = B, y'_A = y_A] \\
3.'' & & B : \texttt{decrypt}\ y_{Messages}\ \texttt{as}\ \{y_{m_1}, ..., y_{m_k}\}_{y_{Key}}
\end{array}
$$

First three lines of this extended narration was explained and the remaining part is similarly derived from the narration. The important point is that, the variables that are set in line 1' are sent in line 2, in convenience with the original protocol. Also in lines 3' and 3" some previous variables are used again, for example in checks.

The LySa model of the WMF protocol is given in Table 3.5. The number in the left margin refer to the message number of the extended protocol narration for WMF (Table 3.4). Step 0 in the LySa process is used to setup the long term keys between the server *S* and the principals *A* and *B*. The scope of the restriction of K_A and K_B include the definition of both *A* and *B* and the Server *S*. Similarly, the first line of Step1 shows that the newly created key has to be restricted, and the first line of Step3 shows that the message containing confidential data must also be restricted. The remaining parts are similar to the extended protocol narration but in LySa syntax.

Table 3.5: LySa model of the WMF protocol.

$$
\begin{array}{ll}
0. & (\nu\, K_A)\,(\nu\, K_B)\,(\\
1. & (\nu\, K) \\
 & \langle A, S, A, B, \{K\}_{K_A}[\mathsf{at}\, A_1\ \mathsf{dest}\, S1\,]\rangle. \\
3. & (\nu\, Secret) \\
 & \langle A, B, \{Secret\}_K\rangle[\mathsf{at}\, A_2\ \mathsf{dest}\, B\,] \\
 & \mid \\
1.' & (A, S, A;\ xB, xMess). \\
1.'' & \mathsf{decrypt}\ xMess\ \mathsf{as}\ \{;\ xKey\}_{K_A}[\mathsf{at}\, S1\ \mathsf{orig}\, A_1\,]\ \mathsf{in} \\
2. & \langle S, xB, A, \{xKey\}_{K_B}\rangle[\mathsf{at}\, S2\ \mathsf{dest}\, B1\,].0 \\
 & \mid \\
2.' & (S, B;\ yA, yMess). \\
2.'' & \mathsf{decrypt}\ yMess\ \mathsf{as}\ \{;\ yKey\}_{K_B}[\mathsf{at}\, B1\ \mathsf{orig}\, S1\,]\ \mathsf{in} \\
3.' & (yA, B;\ yMessages). \\
3.'' & \mathsf{decrypt}\ yMessages\ \mathsf{as}\ \{;\ ySecret\}_{yKey}[\mathsf{at}\, B2\ \mathsf{orig}\, A_2\,]\ \mathsf{in}\ 0 \\
) &
\end{array}
$$

3.2.2 Modelling of Message Authentication Codes

The message authentication codes which include the hash functions are the important parts of the model which need special modelling

considerations. The method used in [27] is suitable for our analysis so we employed the usage as follows.

Hash functions

Since the hash functions are one way functions, they can be modelled with public-key encryption where we have different keys for encryption and decryption. If we can manage the paradigm that the encrypted value can never be decrypted we can use this as a hash function model. This can be done by modelling the hash function using a public name for the encryption key and with no corresponding key for decryption.

Message Authentication Codes

In PKMv2 a keyed MAC is used to verify the integrity of messages. [27] also uses keyed MACs and they modelled it using a shared secret key and a cryptographic hash function. The message is hashed along with the key and then encrypted with the MAC key. Therefore, the message is encrypted by asymmetric encryption first. After that symmetric encryption is applied.

3.2.3 LySa Model of IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake

We are now ready to model protocols in LySa; in particular we will model the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol.

We had simplified the protocol, making use of the work of John Mitchell [25], made the necessary changes that are necessary for LySa and obtained the following protocol narration in section 2.4:

A → B: id, na, MAC{ id, na}$_K$

B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$

A → B: na, nb, id, T, MAC{ na, nb, id, T}$_K$

The extended protocol narration for IEEE 802.6 SA-TEK 3W HS is listed in Table 3.6 where we use the LYSA terms and syntax for writing the cryptographic operations.

Table 3.6: PKMv2 Extended Protocol Narration

0. [new K][new id]
1. A → : id, na, $\{\{|$ id, na $|\}_{Hash}\}_K$ [dest B] [new na]
1'. → B: yid, yna, ymac [check yid = id]
1''. B: decrypt ymac as $\{yh\}_K$ [orig A] [check yh = $\{|$ yid, yna $|\}_{Hash}$]

2. B→ : yna, id, nb, S, $\{\{|$ yna, id, nb, S $|\}_{Hash}\}_K$ [dest A] [new B] [new S]
2'. → A: xna, x_{id}, xnb, xS, xmac [check xna = na, xid = id]
2''. A: decrypt xmac as $\{xh\}_K$ [orig B] [xh = $\{|$ na, id, xnb, xS $|\}_{Hash}$]

3. A → : na, nb, id, T, $\{\{|$ na, nb, id, T $|\}_{Hash}\}_K$ [dest B] [new T]
3'. → B: yna, ynb, yid, yT, ymac [check yna = na, ynb = nb, yid = id]
3''. B: decrypt ymac as $\{yh\}_K$ [orig A] [check yh = $\{|$ na, nb, id, yT $|\}_{Hash}$]

The extended narration can be translated into LYSA by dividing the narration into two processes, one for each principal. The LYSA specification of the protocol is given in table. Notice that the checks in the extended narration are represented by the pattern matchings on input and decryption.

In the LYSA specification we add annotations to all cryptographic operations as described before in this chapter. The LySa model of the PKMv2 SA-TEK 3-Way Handshake is given in Table 3.7.

Table 3.7: PKMv2 LySa Model

```
(υ K ) (υ  id) (
        ! (υ  na) <id, na, {{| id, na |}Hash}K [ at a1 dest {b1}]>.
        (na, id; xnb, xS, xmac).
        decrypt xmac as { {| na, id, xnb, xS |}Hash;} K [ at a2 orig {b2}] in
        (υ  T) <na, nb, id, T, {{| na, nb, id, T |}Hash}K [ at a3 dest {b3}]>.0
        |
        !(id; yna, ymac)
        decrypt ymac as {{| id, yna |}Hash;}K [ at b1 orig {a1}] in
        (υ  nb) (υ  S) < yna, id, nb, S, { {| yna, id, nb, S |}Hash}K [ at b2 dest {a2}] >
        (na, nb, id; yT, ymac).
        decrypt ymac as { {| na, nb, id, yT |}Hash;}K [ at b3 orig {a3}] in 0
)
```

CHAPTER 4

Static Analysis

Static Analysis is a formal method which enables the security analysis of LySa processes. The analysis is based on tracking messages communicated on the network along with the possible values of the variables in the protocol and recording the potential violations of the destination/origin annotations.

A LySa process describes a set of possible operations, the analysis uses an over-approximation of this set, therefore the analysis could investigate a trace which is impossible at all. But this is needed to do a safe approximation because under-approximation could miss some traces. The over-approximation of a LySa-process is shown in Figure 4.1.

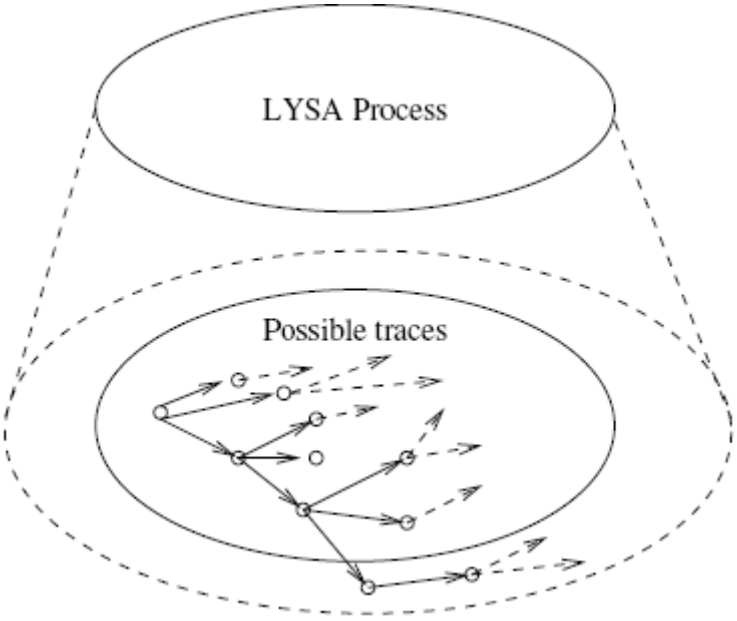

Figure 4.1: The over-approximation of a LySa-process

The approximation of a term *E* is represented by a pair (ρ, ϑ) and called *estimate* for *E*. Similarly, the approximation of a process *P* is represented by a triple (ρ, κ, ψ) and called *estimate* for *P*.

4.1 Terms

The estimate for terms satisfies the judgements defined by the axioms and rules of Table 4.1. The analysis of terms uses a global *abstract environment* in order to keep track of the potential values of variables so that the analysis will determine a superset of the possible canonical values that each tem *E* may evaluate to.

$$\rho : \lfloor \mathcal{X} \rfloor \rightarrow \wp(\mathcal{V})$$

The abstract environment ρ maps the canonical variables to the set of canonical values that may be bound to. In the formula, $\mathcal{V}$ is written for the set of canonical terms with no free variables. The analysis of terms uses the abstract environment to make a judgement of the form:

$$\rho \models E : \vartheta$$

This shows that $\vartheta \subseteq \mathcal{V}$ is a safe approximation of the set of values that E may evaluate to in the abstract environment.

Table 4.1: Analysis of terms, $\rho \models E : \vartheta$.

$$\frac{\lfloor n \rfloor \in \vartheta}{\rho \models n : \vartheta} \qquad \frac{\rho(\lfloor x \rfloor) \subseteq \vartheta}{\rho \models x : \vartheta} \qquad \frac{\lfloor m^{+} \rfloor \in \vartheta}{\rho \models m^{+} : \vartheta} \qquad \frac{\lfloor m^{-} \rfloor \in \vartheta}{\rho \models m^{-} : \vartheta}$$

$$\frac{\begin{array}{c}\wedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \\ \forall V_0, V_1, \cdots, V_k : \wedge_{i=0}^{k} V_i \in \vartheta_i \;\Rightarrow\; \{V_1, \cdots, V_k\}_{V_0}^{\ell}[\mathsf{dest}\,\mathcal{L}] \in \vartheta\end{array}}{\rho \models \{E_1, \cdots, E_k\}_{E_0}^{\ell}[\mathsf{dest}\,\mathcal{L}] : \vartheta}$$

$$\frac{\begin{array}{c}\wedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \\ \forall V_0, V_1, \cdots, V_k : \wedge_{i=0}^{k} V_i \in \vartheta_i \;\Rightarrow\; \{\!|V_1, \cdots, V_k|\!\}_{V_0}^{\ell}[\mathsf{dest}\,\mathcal{L}] \in \vartheta\end{array}}{\rho \models \{\!|E_1, \cdots, E_k|\!\}_{E_0}^{\ell}[\mathsf{dest}\,\mathcal{L}] : \vartheta}$$

The rules in Table 4.1. defines that ϑ contains all the canonical values associated with the components of a term. The first, third and fourth rules in the first line are for names, private and public keys, respectively. These rules say that the canonical names must be in ϑ. The second rule in the first line of the table is the rule for variables and it expresses that the set of canonical value the canonical variable maps to from the environment must be a subset of ϑ: $\rho(\lfloor x \rfloor) \subseteq \vartheta$.

The second line is the rule for k-ary symmetric encryption and the third line is the rule for k-ary asymmetric encryption. These rules express that each term is analyzed and all combinations of values from this analysis

must be in ϑ belonging to the analysis of the overall encryption term $\{V_1,...,V_k\}^{\ell}_{V_0}$ [dest $\mathcal{L}$] $\in \vartheta$. $V \in \vartheta$ notation tests if V is in the set ϑ.

4.2 Processes

In the analysis of processes we focus on which values can flow on the network. The abstract network environment that includes all the message sequences that may flow on the network is shown as:

$$\kappa \subseteq \wp(\mathcal{V}^*)$$

The estimate for processes satisfies the judgements defined by the axioms and rules of Table 4.2. The judgements for processes takes the form:

$$(\rho, \kappa) \models_{\mathsf{RM}} P : \psi$$

Here the symbol ψ represents the set of error messages of the form (ℓ,ℓ') which indicates that something encrypted at ℓ was unexpectedly decrypted at ℓ'. In the end of the section 3.1 we defined the reference monitor and here the analysis uses the reference monitor. If the reference monitor aborts, the annotation leading to the abortion should be placed in the error component ψ and the execution should continue. Namely, ψ contains an over-approximation of the potential origin/destination annotations.

Table 4.2: Analysis of processes, $(\rho,\ \kappa)\ \models_{\mathrm{RM}} P:\ \psi$.

$$(\rho,\kappa)\models_{\mathsf{RM}} 0:\psi \qquad \frac{(\rho,\kappa)\models_{\mathsf{RM}} P:\psi}{(\rho,\kappa)\models_{\mathsf{RM}} (\nu\, n)\,P:\psi} \qquad \frac{(\rho,\kappa)\models_{\mathsf{RM}} P:\psi}{(\rho,\kappa)\models_{\mathsf{RM}} (\nu_{\pm}\, m)P:\psi}$$

$$\frac{(\rho,\kappa)\models_{\mathsf{RM}} P_1:\psi\ \wedge\ (\rho,\kappa)\models_{\mathsf{RM}} P_2:\psi}{(\rho,\kappa)\models_{\mathsf{RM}} P_1|P_2:\psi} \qquad \frac{(\rho,\kappa)\models_{\mathsf{RM}} P:\psi}{(\rho,\kappa)\models_{\mathsf{RM}} !P:\psi}$$

(Output)

$$\frac{\begin{array}{c}\wedge_{i=1}^{k}\ \rho\models E_i:\vartheta_i\ \wedge\\ \forall V_1,\cdots,V_k:\ \wedge_{i=1}^{k} V_i\in\vartheta_i\ \Rightarrow\ \langle V_1,\cdots,V_k\rangle\in\kappa\ \wedge\\ (\rho,\kappa)\models_{\mathsf{RM}} P:\psi\end{array}}{(\rho,\kappa)\models_{\mathsf{RM}} \langle E_1,\cdots,E_k\rangle.\,P:\psi}$$

(Input)

$$\frac{\begin{array}{c}\wedge_{i=1}^{j}\ \rho\models E_i:\vartheta_i\ \wedge\\ \forall\langle V_1,\cdots,V_k\rangle\in\kappa:\ \wedge_{i=1}^{j} V_i\ \mathsf{E}\ \vartheta_i\Rightarrow\ \ \wedge_{i=j+1}^{k} V_i\in\rho(\lfloor x_i\rfloor)\ \wedge\\ (\rho,\kappa)\models_{\mathsf{RM}} P:\psi\end{array}}{(\rho,\kappa)\models_{\mathsf{RM}} (E_1,\cdots,E_j;\,x_{j+1},\cdots,x_k).\,P:\psi}$$

(Decryption)

$$\frac{\begin{array}{c}\rho\models E:\vartheta\ \ \wedge\ \ \wedge_{i=0}^{j}\,\rho\models E_i:\vartheta_i\ \wedge\\ \forall\ \{V_1,\cdots,V_k\}_{V_0}^{\ell}[\mathsf{dest}\,\mathcal{L}\,]\in\vartheta:\ \wedge_{i=0}^{j} V_i\ \mathsf{E}\ \vartheta_i\ \ \Rightarrow\wedge_{i=j+1}^{k} V_i\in\rho(\lfloor x_i\rfloor)\ \wedge\\ (\neg\mathsf{RM}(\ell,\mathcal{L}',\ell',\mathcal{L})\Rightarrow(\ell,\ell')\in\psi)\ \wedge\\ (\rho,\kappa)\models_{\mathsf{RM}} P:\psi\end{array}}{(\rho,\kappa)\models_{\mathsf{RM}} \mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1,\cdots,E_j;\,x_{j+1},\cdots,x_k\}_{E_0}^{\ell'}\ [\mathsf{orig}\,\mathcal{L}'\,]\ \mathsf{in}\ P:\psi}$$

(Asymmetric decryption)

$$\frac{\begin{array}{c}\rho\models E:\vartheta\ \ \wedge\ \ \wedge_{i=0}^{j}\,\rho\models E_i:\vartheta_i\ \wedge\\ \forall(m^{\pm},m^{\mp}):\forall\ \{\!|V_1,\cdots,V_k|\!\}_{V_0}^{\ell}[\mathsf{dest}\,\mathcal{L}\,]\in\vartheta:\forall V_0'\ \mathsf{E}\ \vartheta_0:\{V_0,V_0'\}=\{m^{\pm},m^{\mp}\}\wedge\\ \wedge_{i=1}^{j} V_i\ \mathsf{E}\ \vartheta_i\ \ \Rightarrow\ \ \wedge_{i=j+1}^{k} V_i\in\rho(\lfloor x_i\rfloor)\ \wedge\\ (\neg\mathsf{RM}(\ell,\mathcal{L}',\ell',\mathcal{L})\Rightarrow(\ell,\ell')\in\psi)\ \wedge\\ (\rho,\kappa)\models_{\mathsf{RM}} P:\psi\end{array}}{(\rho,\kappa)\models_{\mathsf{RM}} \mathsf{decrypt}\ E\ \mathsf{as}\ \{\!|E_1,\cdots,E_j;\,x_{j+1},\cdots,x_k|\!\}_{E_0}^{\ell'}\ [\mathsf{orig}\,\mathcal{L}'\,]\ \mathsf{in}\ P:\psi}$$

Table 4.2. defines the axioms and the rules for the analysis of the processes and ϑ gives the set of values that the terms can evaluate to.

The first line of the Table 4.2 includes the rules for inactive processes and restriction (the last one is for public/private keys). The second line includes the rules for parallel composition and replication. The rule for the inactive processes does not restrict the analysis result while the rules for parallel composition, restriction and replication ensure that the analysis also holds for the immediate subprocesses.

The rule **k-ary output** which is for sending message on the network finds the sets ϑ_i for each term E_i and requires that all *k*-tuples of values $\langle V_1,...,V_k\rangle$ taken from $\vartheta_1 \times \ldots \times \vartheta_k$ can flow on the network and also requires that (ρ, κ, ψ) are also valid analysis estimates of process *P*.

The rule **input** checks whether the first j terms of $E_1,...,E_k$ have acceptable estimates ϑ_i and whether the first j values of any message $\langle V_1,...,V_j,V_{j+1},...,V_k\rangle$ in κ are pointwise included in ϑ_i. When the check is successful, the remaining values $V_{j+1},...,V_k$ are included in the estimates for the corresponding variables $x_{j+1},...,x_k$.

The rules for **decryption** have a similar pattern matching with the previous rule. All the terms are evaluated to their respectable estimates ϑ_i and the first j values of the evaluation of the encrypted term $\{ V_1,...,V_k \}^{\ell}_{V_0}$ [dest $\mathcal{L}$] $\in \vartheta$ are checked whether they are pointwise included in ϑ_i. The rule for **symmetric decryption** ensures that only the correct key can be used to decrypt encrypted values. The rule for a**symmetric decryption** ensures that the key used for decryption must be the opposite of the one used for encryption. Similar to the input rule, if the matching succeeds for the first j values and in addition the keys for decryption matches the ones used for decryption, then the remaining values $V_{j+1},...,V_k$ are added to the acceptable estimates for the corresponding variables $x_{j+1},...,x_k$. If the encrypted term *E* is decrypted at an unexpected place ($\ell' \in \mathcal{L}$) or the decrypted values are encrypted at an unexpected place ($\ell \in \mathcal{L}'$), then the error component must contain the annotations where the error occurred $(\ell, \ell') \in \psi$.

If $(\rho, \kappa) \models_{RM} P : \psi$, then (ρ, κ, ψ) is a valid estimate for all the states passed through in an execution of P. Also, when $\psi = \emptyset$ in an estimate of the form $(\rho, \kappa) \models_{RM} P : \psi$ then the reference monitor cannot abort the execution of P. These are all proved in [18].

4.3 Modelling The Attacker

In practice, the protocols – especially the ones in wireless networks – are executed in medium with malicious attackers. As mentioned in subsection 1.1.1 LySa processes will be analyzed in parallel with Dolev-Yao attacker[12] which can perform operations like sending/receiving messages and encryption/decryption same as a legitimate principal. The analysis result of a process P analyzed in parallel with the attacker contain the least solution that satisfy the rules from the previous Section for the estimate (ρ, κ, ψ) and the variable bindings for the attacker in addition to the variable bindings for the process P.

We have new canonical name and variables for the attacker: all the canonical names of the attacker are mapped to $n_\bullet$ and all the canonical variables of the attacker are mapped to $z_\bullet$. We also have $\ell_\bullet$ which is a crypto-point in the attacker, and we have the set C which is the set of crypto-points in the original process P in parallel with the attacker. Finally, there exists a public/private key-pair belonging to the attacker $\{m_\bullet^+, m_\bullet^-\}$.The formal definition of the Dolev-Yao attacker is given in Table 4.3.

A process P is of type $(\mathcal{N}_f, \mathcal{A}_\mathcal{K}, \mathcal{A}_{Enc}^+)$ if (1) it is a closed process (it has no free variables, namely no variables that are never bound to a name), (2) its free names are in $\mathcal{N}_f$, (3) all the arguments used for sending and receiving are in $\mathcal{A}_\mathcal{K}$ and (4) all the arguments used for encryption and decryption are in $\mathcal{A}_{Enc}^+$.

Table 4.3: Dolev-Yao condition..

$$(1)\ \wedge_{k\in\mathcal{A}_\kappa}\ \forall\langle V_1,\cdots,V_k\rangle\in\kappa:\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)$$

$$(2)\ \wedge_{k\in\mathcal{A}^+_{\mathsf{Enc}}}\ \forall\{V_1,\cdots,V_k\}^{\ell}_{V_0}[\mathsf{dest}\,\mathcal{L}\,]\in\rho(z_\bullet):$$
$$V_0\ \mathsf{E}\ \rho(z_\bullet)\Rightarrow(\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\wedge(\neg\mathsf{RM}(\ell,\mathcal{C},\ell_\bullet,\mathcal{L})\Rightarrow(\ell,\ell_\bullet)\in\psi))$$

$$(3)\ \wedge_{k\in\mathcal{A}^+_{\mathsf{Enc}}}\ \forall V_0,\cdots,V_k:\wedge_{i=0}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow\{V_1,\cdots,V_k\}^{\ell_\bullet}_{V_0}[\mathsf{dest}\,\mathcal{C}\,]\in\rho(z_\bullet)$$

$$(4)\ \wedge_{k\in\mathcal{A}_\kappa}\ \forall V_1,\cdots,V_k:\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow\langle V_1,\cdots,V_k\rangle\in\kappa$$

$$(5)\ \{n_\bullet\}\cup\lfloor\mathcal{N}_\mathsf{f}\rfloor\subseteq\rho(z_\bullet)$$

$$(6)\ \forall(m^+,m^-):\wedge_{k\in\mathcal{A}^+_{\mathsf{Enc}}}\ \forall\{\!|V_1,\cdots,V_k|\!\}^{\ell}_{V_0}[\mathsf{dest}\,\mathcal{L}\,]\in\rho(z_\bullet):$$
$$\forall V_0'\ \mathsf{E}\ \rho(z_\bullet):\{V_0,V_0'\}=\{m^\pm,m^\mp\}\Rightarrow(\wedge_{i=1}^{k}\ V_i\in\rho(z_\bullet)\wedge$$
$$(\neg\mathsf{RM}(\ell,\mathcal{C},\ell_\bullet,\mathcal{L})\Rightarrow(\ell,\ell_\bullet)\in\psi))$$

$$(7)\ \wedge_{k\in\mathcal{A}^+_{\mathsf{Enc}}}\ \forall V_0,V_1,\cdots,V_k:\wedge_{i=0}^{k}\ V_i\in\rho(z_\bullet)\Rightarrow\{\!|V_1,\cdots,V_k|\!\}^{\ell_\bullet}_{V_0}[\mathsf{dest}\,\mathcal{C}\,]\in\rho(z_\bullet)$$

$$(8)\ \{m^+_\bullet,m^-_\bullet\}\subseteq\rho(z_\bullet)$$

The descriptions of the conditions given in Table 4.3 are below:

1. The attacker can improve his knowledge by eavesdropping all messages sent on the network.

2. The attacker can improve his knowledge by decrypting messages with the keys he already knows. Unless the intended recipient of the message was attacker, an error ($\ell, \ell_\bullet$) should be added to the error component ψ which means that something encrypted at ℓ was actually decrypted by the attacker at $\ell_\bullet$.

3. The attacker can construct new encryptions using the keys he already knows. If this message is received and decrypted by a principal, then an error ($\ell_\bullet, \ell$) should be added to the error component ψ which means that something encrypted at the attacker was decrypted by the attacker by a process *P* at ℓ

4. The attacker can send messages on the network using his knowledge and thus forge new communications.

5. The attacker initially has the knowledge of the canonical name $n_\bullet$ and all free names of the process *P*.

6. In addition to condition 2, if the attacker possesses the corresponding decryption key used for encryption, he can decrypt a term encrypted with asymmetric encryption.

7. In addition to condition 3, the attacker can create an encrypted term using asymmetric encryption.

8. The attacker has his own private/public key pair.

This conditions enable the attacker to establish the attack scenarios that were defined in subsection 1.1.1. The soundness of Dolev-Yao condition is proved in [13]

4.4 Analysis

The flow of the analysis starts with a LySa code which contains the LySa model of the protocol. The LySa-tool parses the LySa code and transforms it into the Alternation-free Least Fixed Point (ALFP) logic equations which are definitely outside of the focus of this thesis. These equations are solved by the Succinct Solver which is a tool for solving constraints specified in ALFP. The Succinct Solver computes the minimum solution satisfying the input equations and returns a result. This is transformed by the LySa-tool to a readable version of the estimate (ρ, κ, ψ). The LySa-tool makes use of the Succinct Solver and the Standard ML of New Jersey. The overall process is shown in Figure 4.2.

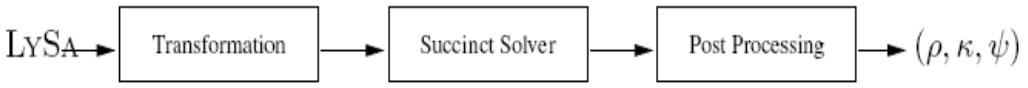

Figure 4.2: Analysis process.

4.4.1 Analysis of the WMF Protocol

We have defined the protocol narration of the WMF protocol in Table 1.1 of the first chapter and the extended narration of this protocol in Table 3.4 of the third chapter. Here we give the LySa model for the WMF protocol in Table 4.4 below.

Table 4.4: LySa model of the WMF.

$$
\begin{array}{ll}
0. & (\nu\, K_A)\,(\nu\, K_B)\,(\\
1. & (\nu\, K) \\
 & \langle A, S, A, B, \{K\}_{K_A}[\mathsf{at}\, A_1\ \mathsf{dest}\, S1\,]\rangle. \\
3. & (\nu\, Secret) \\
 & \langle A, B, \{Secret\}_K\rangle[\mathsf{at}\, A_2\ \mathsf{dest}\, B\,] \\
 & | \\
1.' & (A, S, A;\ xB, xMess). \\
1.'' & \mathsf{decrypt}\ xMess\ \mathsf{as}\ \{;\ xKey\}_{K_A}[\mathsf{at}\, S1\ \mathsf{orig}\, A_1\,]\ \mathsf{in} \\
2. & \langle S, xB, A, \{xKey\}_{K_B}\rangle[\mathsf{at}\, S2\ \mathsf{dest}\, B1\,].0 \\
 & | \\
2.' & (S, B;\ yA, yMess). \\
2.'' & \mathsf{decrypt}\ yMess\ \mathsf{as}\ \{;\ yKey\}_{K_B}[\mathsf{at}\, B1\ \mathsf{orig}\, S1\,]\ \mathsf{in} \\
3.' & (yA, B;\ yMessages). \\
3.'' & \mathsf{decrypt}\ yMessages\ \mathsf{as}\ \{;\ ySecret\}_{yKey}[\mathsf{at}\, B2\ \mathsf{ori} \\
) &
\end{array}
$$

This model is coded in LySa and after the LySa tool processes the analysis gives an estimate where $\psi = \emptyset$. The variable environment is given in Table 4.5.

Table 4.5: The variable environment for the WMF analysis.

$\rho(xB)$	$=$	$V_\bullet, B$
$\rho(xKey)$	$=$	K
$\rho(xMess)$	$=$	$V_\bullet, \{K\}_{K_A}$
$\rho(yA)$	$=$	$V_\bullet, A$
$\rho(yKey)$	$=$	K
$\rho(yMess)$	$=$	$V_\bullet, \{K\}_{K_A}$
$\rho(yMessages)$	$=$	$V_\bullet, \{Secret\}_K$
$\rho(ySecret)$	$=$	$Secret$
K	$\notin$	$rho(z_\bullet)$
$Secret$	$\notin$	$rho(z_\bullet)$

In Table 4.5, $V_\bullet$ denotes any value that the attacker has the knowledge of. The error component $\psi = \emptyset$ ensures that no encryptions/decryptions occur at unexpected places. So, the last row of the table means that the attacker does not possess the knowledge of the Secret. But the attacker can send out a message of any length matching any term receiving values, and the names the process matches on are all free names so the attacker has the knowledge to create messages on the right form. Therefore, All variables bound to values received directly on the network, which means that they are not decrypted from values received on the network, can all be bound to anything inside the attacker $V_\bullet$. However, the attacker does not posses enough information to create encrypted messages to be decrypted by legitimate principals which would to an error in ψ of the form $(\ell_\bullet, \ell)$, where ℓ is any point in the LySa-process. In the same manner, the attacker does not posses enough information to decrypt any of the encrypted terms from the message on the network by legitimate principals which would lead to an error in ψ of the form $(\ell, \ell_\bullet)$.

The analysis shows that the messages to be kept secret are not leaked to the attacker and the messages are originated and received by intended principals. But this does not verify the authenticity and confidentiality of the WMF protocol. Because, the LySa model in Table 4.4 describes a

simple scenario with one initiator A, one server S and one responder B which limits the attacker to act as a passive attacker. Therefore, we need a more flexible scenario, where a number of initiators I_i, and a number of responders I_j exist. The difference between the simple and the flexible scenarios are shown in Figure 4.3.

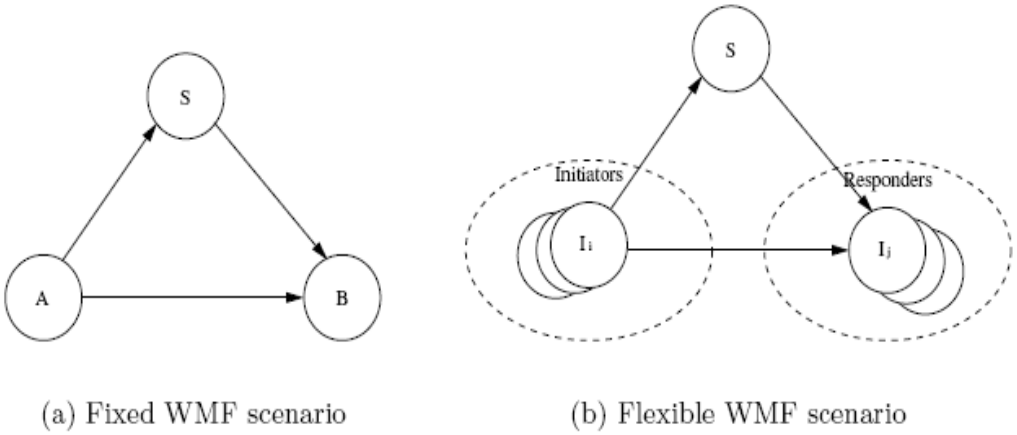

Figure 4.3: Different WMF scenarios.

In the flexible scenario, all initiators, responders and the attacker share keys with the server S. Therefore, the attacker is able to act as either an initiator or as a responder in a protocol run. The LySa model of the flexible WMF scenarios is described in Table 4.6.

Table 4.6: The Flexible WMF scenario.

$(\nu^{n}_{i=1\ldots n}\ KA_i)$
$(\nu^{n}_{j=1\ldots n}\ KB_j)$
$|^{n}_{i=1\ldots n}\ |^{n}_{j=1\ldots n\ j\neq i}$
$!(\nu\ K_{ij})$
$\langle I_i, S, I_i, I_j, \{K_{ij}\}_{KA_i}[\mathsf{at}\ A1_{ij}\ \mathsf{dest}\ S1_{ij}\ ]\rangle.$
$(\nu\ Secret_{ij})$
$\langle I_i, I_j, \{Secret_{ij}\}_{K_{ij}}[\mathsf{at}\ A2_{ij}\ \mathsf{dest}\ B2_{ij}\ ]\rangle.$
0
|
$|^{n}_{i=0\ldots n}\ |^{n}_{j=0\ldots n\ j\neq i}$
$!(I_i, S, I_i;\ xB_{ij}, xMess_{ij}).$
$\mathsf{decrypt}\ xMess_{ij}\ \mathsf{as}\ \{;\ xKey_{ij}\}_{KA_i}[\mathsf{at}\ S1_{ij}\ \mathsf{orig}\ A1_{ij}\ ]\ \mathsf{in}$
$\langle S, xB_{ij}, I_i, \{xKey_{ij}\}_{KB_j}[\mathsf{at}\ S2_{ij}\ \mathsf{dest}\ B1_{ij}\ ]\rangle.$
0

|
$|^{n}_{j=1\ldots n}\ |^{n}_{i=0\ldots n\ i\neq j}$
$!(S, I_j;\ yA_{ij}, yMess_{ij}).$
$\mathsf{decrypt}\ yMess_{ij}\ \mathsf{as}\ \{;\ yKey_{ij}\}_{KB_j}[\mathsf{at}\ B1_{ij}\ \mathsf{orig}\ S2_{ij}\ ]\ \mathsf{in}$
$(yA_{ij}, I_j;\ yMessages_{ij}).$
$\mathsf{decrypt}\ yMessages_{ij}\ \mathsf{as}\ \{;\ ySecret_{ij}\}_{yKey_{ij}}[\mathsf{at}\ B2_{ij}\ \mathsf{orig}\ A2_{ij}\ ]\ \mathsf{in}$
0

In the first rows of the model, it can be seen that initially shared keys KA_i and KB_j are restricted for the valid principals $1 \leq i\ ,\ j \leq n$. The keys KA_0 and KB_0 belonging to the attacker are not restricted, therefore KA_0 and KB_0 are threaded as free names in the analysis.

The indexing of the principals is important. The initiating principal I_i is indexed from $i = 1$ and $j = 1$ since we only describe the legitimate part of the system. The server S is indexed from $i = 0$ and $j = 0$ so that the attacker can act as either a initiator or a responder in the protocol. The responder I_j is indexed from $i = 0$ and $j = 1$ allowing the responder to actually receive messages from the attacker. The indices i and j cannot be equal, because that would lead principals to authenticate themselves.

The result of the flexible WMF model is shown in Table 4.7. This result bears a lot of differences compared to the result of the simple model shown in Table 4.5. The error component ψ in the result is not empty, this means that some encryptions and decryptions have occurred at unexpected places. In the first line of Table 4.7, there are three types of pairs in the error components: the first type includes $\ell_\bullet$ as the decryption point for example ($A2_{i,j}, \ell_\bullet$) and means that information encrypted at $A2$ in any principal initiating the protocol can be decrypted by the attacker. When we look at the model in Table 4.6 we see that $A2$ is the place where the $Secret_{i,j}$ is encrypted and therefore the pair ($A2_{i,j}, \ell_\bullet$) in the error component implies that all secrets are known by the attacker. This is also clearly seen in the last row of the analysis result in Table 4.7. Second type of pairs includes $\ell_\bullet$ as the encryption point for example ($\ell_\bullet$, $B2_{i,j}$) and means that the attacker can send any information in his knowledge $V_\bullet$ to any responder I_j and make him believe that the information originated from the initiator I_i. This causes the variables that store such information contain values from the attacker as shown in the second and third rows of Table 4.7. Third type of pairs does not include $\ell_\bullet$ as neither the encryption nor the decryption point for example ($A1_{1,2}$, $S1_{2,1}$) and means that messages encrypted in one run of the protocol can be decrypted in another run of the protocol. This can cause the situation that a secret meant for principal I_1 could end up at principal I_j. Such a problem can be seen in the second and third lines of the analysis again.

Table 4.7: The result of the flexible WMF analysis.

ψ	$=$	$(A2_{i,j}, l_\bullet), (l_\bullet, B2_{i,j}), (A1_{1,2}, S1_{2,1}), \ldots, (A2_{1,2}, B2_{2,1}), \ldots$
$\rho(ySecret_{1,2})$	$=$	$V_\bullet, Secret_{1,2}, Secret_{2,1}, \ldots$
$\rho(ySecret_{2,1})$	$=$	$V_\bullet, Secret_{2,1}, Secret_{1,2}, \ldots$
	$\vdots$	
$K_{i,j}$	$\in$	$rho(z_\bullet)$
$Secret_{i,j}$	$\in$	$rho(z_\bullet)$

Since the analysis is an over-approximation, the error component does not necessarily imply that there exist an error. In other words, the value in the error-component ψ could come from a trace that is not actually possible. So, we have to find actual traces leading to the errors in the

analysis. The trace in Table 4.8 leads to the error ($A2_{1,2}$, $\ell_\bullet$). In the line 1, the message intended from the initiating principal I_1 to the server S which includes the session key $K_{1,2}$ in the aim of establishing communication with principal I_2 is eavesdropped by the attacker denoted by M. In the line 1', the attacker modifies the second field of the message as if the principal I_1 wants to engage communication with I_0 which is in fact the attacker himself. The server gets this message and sends the message 2 (includes the session key $K_{1,2}$ but encrypted with the shared key of the attacker) to the intended responder of message 1 which is the attacker. So the attacker gets the session key, and now he (or she) is able to decrypt any messages sent from I_1 to I_2. This situation was indicated by ($A2_{1,2}$, $\ell_\bullet$)

Table 4.8: Trace of the error ($A2_{1,2}$, $\ell_\bullet$).

$$
\begin{array}{llcll}
1. & I_1 & \rightarrow & M(S) & : & I_1, I_2, \{K_{1,2}\}_{KA_1} \\
1.' & M(I_1) & \rightarrow & S & : & I_1, I_0, \{K_{1,2}\}_{KA_1} \\
2. & S & \rightarrow & I_0 & : & I_1, \{K_{1,2}\}_{KB_0} \\
3. & I_1 & \rightarrow & M(I_2) & : & \{Secret\}_{K_{1,2}}
\end{array}
$$

Using this session key, the attacker can cause different type of errors in the analysis, as shown in Table 4.9. In the first line, the attacker uses the session key he (or she) got in the first message, which is in fact a replay attack. This persuades the server that I_1 has created a fresh key for communication with I_2. In the second line, the server sends this session key to I_2 who believes that it is shared with I_1, but actually it is the attacker. After that the attacker is now able to impersonate I_1, by sending false secrets to I_2 as described in the line 3. This leads to the error ($\ell_\bullet$, $B2_{1,2}$) since the secret that was believed to be encrypted by I_1 and decrypted at B2 was in fact encrypted at the attacker.

Table 4.9: Trace of the error ($\ell_\bullet$, $B2_{1,2}$).

$$
\begin{array}{llcll}
1. & M(I_1) & \rightarrow & M(S) & : & I_1, I_2, \{K_{1,2}\}_{KA_1} \\
2. & S & \rightarrow & I_2 & : & I_1, \{K_{1,2}\}_{KB_2} \\
3. & M(I_1) & \rightarrow & I_2 & : & \{FalseSecret\}_{K_{1,2}}
\end{array}
$$

The details of the analysis of the WMF protocol is discussed in [18, 13].

CHAPTER 5

Analysis of IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake

As we described in the section 2.3, our approach is based on checking the limits of robustness in IEEE 802.16 PKMv2 by removing enhancements in PKMv2 one by one, and in different combinations. Thus, we can see if some improvements are unnecessary and the result may lead us to a simplified by still strong and secure protocol. Our experiments are accomplished using the LySa-tool which runs with our LySa code.

5.1 Experiments

We based our model on John Mitchell's simplified version (that was used in his security review together with IETF EAP Work Group) [25] of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake in section 2.4. After that we developed our LySa model in section 3.2.3.

We start with our base protocol model and try to simplify the model by removing components and analyzing with attacker to find flaws.

The Experiment Logic

We made the experiments systematically, and the road map of the experiment can be seen in Figure 5.1. First, we start with the base protocol and show that it has no flaws. After that, we have three major paths: Removing the Nonces, Removing the Ids and Removing both Nonces and Ids. The shaded nodes in the figure shows the experiments with violations.

In the first path, we start by removing the outermost nonces, namely the nonces in the last message. We remove one nonce at a time, and both nonces also. Therefore, we have three experiments about the nonces in the last message. There is a fourth experiment in this path which includes removing another nonce in the second message, in addition to the ones in the last message.

In the second path, we remove the key ids. We start with the key id in the last message. Then we remove another key id which is actually in the second message.

In the last path, we join the successful experiments, in other words the modification of the base model where no flaws could be found. There are two successful experiments in the first path and one in the second, therefore we have two experiments in the last path.

The results of the experiments are discussed in the section 5.3 Analysis Results.

Figure 5.1: Experiment Road Map.

5.1.1 The PKMv2 SA-TEK 3-Way Handshake

In our base model of the protocol we have three messages each of them consisting of at least identities, nonces and message authentication codes as shown in Table 5.1.

Table 5.1: The base protocol narration.

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$
3. A → B: na, nb, id, T, MAC{ na, nb, id, T}$_K$

The LySa model for the protocol is shown in Table 5.2. First part of the model is the initiator, who is actually the BS in IEEE 802.16. Then comes the responder, who is the MS. The last part shows the attackers knowledge. This means that the analysis include an attacker as described in section 4.3.

Table 5.2: PKMv2 LySa Model

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}_{-} \text{ s.t. } \lfloor \mathbf{N}_{-} \rfloor = \{0\} \text{ in} \\
|_{i \in X}\, |_{j \in Z}\, !(\nu\ na_{ij})\, \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij} \text{ dest } \{b1_{ij}\}]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{\!|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij} \text{ orig } \{b2_{ij}\}] \text{ in} \\
(\nu\, T_{ij})\, \langle na_{ij}, nb_{ij}, id_{ij}, T_{ij}, \{\!|na_{ij}, nb_{ij}, id_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij} \text{ dest } \{b3_{ij}\}]\rangle. \\
0 \\
| \\
|_{j \in X}\, |_{i \in Z}\, !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij} \text{ orig } \{a1_{ij}\}] \text{ in} \\
(\nu\ nb_{ij})\, (\nu\ S_{ij})\, \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{\!|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij} \text{ dest } \{a2_{ij}\}]\rangle. \\
(na_{ij}, nb_{ij}, id_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|na_{ij}, nb_{ij}, id_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij} \text{ orig } \{a3_{ij}\}] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}_{-} \text{ s.t. } \lfloor \mathbf{N}_{-} \rfloor = \{0\}}\, |_{j \in \mathbf{N}_{-} \text{ s.t. } \lfloor \mathbf{N}_{-} \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

The result of the analysis is: ***no violations possible***. This means that the protocol is secure and the attacker couldn't violate the authentication properties. This result is important because it ensures us that the base protocol, the PKMv2 SA-TEK 3-Way Handshake, is secure and this result is similar to Mitchell's work [25] with model checking using Murphi. Now, we can make our modifications convenient with our experiment logic.

5.1.2 Removing the Nonces

In the first part of the experiments we will be dealing with the nonces in the second and the last message. The nonce na is the same nonce that was used in message 1 and message 2, therefore seems to be redundant. But we have to show it with the static analysis. The nonce nb seems to be convenient but we deal with it too.

5.1.2.1 Removing nb in the Last Message

We removed the nonce of B, nb, and now the protocol is as shown in Table 5.3. No doubt that this modification affects the MAC of the message three. In fact, this modification makes the na in message two meaningless, but we have to try and see the result.

Table 5.3: PKMv2 without nb in message 3

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$
3. A → B: na, id, T, MAC{ na, id, T}$_K$

The LySa model after the modification is shown in Table 5.4. The changes are in the initiators last output and the responders second input.

Table 5.4: LySa model of PKMv2 without nb in message 3

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\} \text{ in} \\
|_{i \in X}\, |_{j \in Z}\, !(\nu\, na_{ij})\, \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij} \text{ dest } \{b1_{ij}\}\,]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{\!|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij} \text{ orig } \{b2_{ij}\}\,] \text{ in} \\
(\nu\, T_{ij})\, \langle na_{ij}, id_{ij}, T_{ij}, \{\!|na_{ij}, id_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij} \text{ dest } \{b3_{ij}\}\,]\rangle. \\
0 \\
| \\
|_{j \in X}\, |_{i \in Z}\, !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij} \text{ orig } \{a1_{ij}\}\,] \text{ in} \\
(\nu\, nb_{ij})\, (\nu\, S_{ij})\, \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{\!|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij} \text{ dest } \{a2_{ij}\}\,]\rangle. \\
(na_{ij}, id_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|na_{ij}, id_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij} \text{ orig } \{a3_{ij}\}\,] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\, |_{j \in \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash \rangle. \\
0
\end{array}
$$

The result of the analysis is: ***no violations possible***. This means that the protocol is still secure and the attacker still couldn't violate the authentication properties even though we didn't use the nonce of principal B in the last message. This is an interesting result because now the na in message two seems to be meaningless because there is no response for it. MAC's seem to save the protocol to verify the security properties. In addition, this is also an important result because it supports our assertion But we have to try the other combinations to conclude about the analysis.

5.1.2.2 Removing na in the Last Message

In this experiment, we removed the nonce of principal A, na, and now the protocol is as shown in Table 5.5. This nonce was sent to B in message-1, and was responded by B in message-2. Removing it from message-3 shouldn't affect the result, but we have to show it with LySa results.

Table 5.5: PKMv2 without na in message 3

1. A → B: id, na, MAC{ id, na}$_K$ 2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$ 3. A → B: nb, id, T, MAC{ nb, id, T}$_K$

The LySa model after the modification is shown in Table 5.6. Similar to the previous experiment, the changes are in the initiators last output and the responders second input.

Table 5.6: LySa model of PKMv2 without na in message 3

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}_- \text{ s.t. } \lfloor \mathbf{N}_- \rfloor = \{0\} \text{ in} \\
|_{i \in X}\, |_{j \in Z}\, !(\nu\ na_{ij})\, \langle id_{ij}, na_{ij}, \{|id_{ij}, na_{ij}|\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij} \text{ dest } \{b1_{ij}\}]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij} \text{ orig } \{b2_{ij}\}] \text{ in} \\
(\nu\, T_{ij})\, \langle nb_{ij}, id_{ij}, T_{ij}, \{|nb_{ij}, id_{ij}, T_{ij}|\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij} \text{ dest } \{b3_{ij}\}]\rangle. \\
0 \\
| \\
|_{j \in X}\, |_{i \in Z}\, !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{|id_{ij}, yna_{ij}|\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij} \text{ orig } \{a1_{ij}\}] \text{ in} \\
(\nu\ nb_{ij})\, (\nu\ S_{ij})\, \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij} \text{ dest } \{a2_{ij}\}]\rangle. \\
(nb_{ij}, id_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{|nb_{ij}, id_{ij}, yT_{ij}|\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij} \text{ orig } \{a3_{ij}\}] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}_- \text{ s.t. } \lfloor \mathbf{N}_- \rfloor = \{0\}}\, |_{j \in \mathbf{N}_- \text{ s.t. } \lfloor \mathbf{N}_- \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

The result of the analysis is: ***no violations possible***. This means that the protocol is still secure and the attacker still couldn't violate the authentication properties even though we didn't use the nonce of principal A in the last message. Actually, this result supports our assertion and this is an optimized alternative to the protocol.

5.1.2.3 Removing All the Nonces in the Last Message

In the previous two experiments, we removed na and nb one by one and we couldn't find any violations. In this experiment, we removed both na and nb in the base protocol as shown in Table 5.7. Again we did the necessary changes in the MAC.

Table 5.7: PKMv2 without na and nb in message 3

1. A → B: id, na, MAC{ id, na}$_K$ 2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$ 3. A → B: id, T, MAC{ id, T}$_K$

The LySa model after the modification is shown in Table 5.8. Similar to the previous experiment, the changes are in the initiators last output and the responders second input.

Table 5.8: LySa model of PKMv2 without na and nb in message 3

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i\in X, j\in X}\ K_{ij}) \\
(\nu_{i\in X, j\in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}_- \text{ s.t. } \lfloor \mathbf{N}_- \rfloor = \{0\} \text{ in} \\
|_{i\in X}\,|_{j\in Z}\,!(\nu\, na_{ij})\,\langle id_{ij}, na_{ij}, \{\{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}}[\text{at } a1_{ij} \text{ dest } \{b1_{ij}\}\,]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{\{\!|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}}[\text{at } a2_{ij} \text{ orig } \{b2_{ij}\}\,] \text{ in} \\
(\nu\, T_{ij})\,\langle id_{ij}, T_{ij}, \{\{\!|id_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}}[\text{at } a3_{ij} \text{ dest } \{b3_{ij}\}\,]\rangle. \\
0 \\
| \\
|_{j\in X}\,|_{i\in Z}\,!(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}}[\text{at } b1_{ij} \text{ orig } \{a1_{ij}\}\,] \text{ in} \\
(\nu\, nb_{ij})\,(\nu\, S_{ij})\,\langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{\{\!|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}}[\text{at } b2_{ij} \text{ dest } \{a2_{ij}\}\,]\rangle. \\
(id_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\{\!|id_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}}[\text{at } b3_{ij} \text{ orig } \{a3_{ij}\}\,] \text{ in} \\
0 \\
| \\
|_{i\in X\cup \mathbf{N}_- \text{ s.t. } \lfloor \mathbf{N}_- \rfloor = \{0\}}\,|_{j\in \mathbf{N}_- \text{ s.t. } \lfloor \mathbf{N}_- \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

This time we find violation of authentication properties. The result is given as:

$\psi = (a1_{1\,1}, b3_{1\,1}), (a3_{1\,1}, b1_{1\,1}), (a1_{2\,1}, b3_{2\,1}), (a3_{2\,1}, b1_{2\,1}), (a1_{1\,2}, b3_{1\,2}), (a3_{1\,2}, b1_{1\,2}), (a1_{2\,2}, b3_{2\,2}), (a3_{2\,2}, b1_{2\,2})$

Sample trace for $(a1_{1\,1}, b3_{1\,1})$ can be shown as:

1. $A_1 \rightarrow B_1$: id_{11}, na_{11}, MAC{ id_{11}, na_{11}}$_{K11}$
1'. $A_1 \rightarrow M(B_1)$: id_{11}, na_{11}, MAC{ id_{11}, na_{11}}$_{K11}$
2. $B_1 \rightarrow A_1$: na_{11}, id_{11}, nb_{11}, S_{11}, MAC{ na_{11}, id_{11}, nb_{11}, S_{11}}$_{K11}$
3. $M(A_1) \rightarrow B_1$: id_{11}, T_0, MAC{ id_{11}, na_{11}}$_{K11}$

The results show that some encrypted values are decrypted in wrong places and some decrypted values were actually encrypted in the wrong places. The crypto-points are all from legitimate principals so there can be a replay attack. A possible trace of this error can be summarized as: the attacker eavesdropped the first message and he used the encrypted value in the first message, which is actually the MAC of the message, that he couldn't decrypt in a reply attack. In the third message, he replayed the MAC's, namely he used the MAC of message one in messag-3. This is a flaw so we found a level that the protocol lost its robustness property.

This results show that in the implementation, the length of the fields are important. If somehow the lengths of the na value and the T value are the same, then there exists the security flaw.

5.1.2.4 Removing All Nonces in the Last Message and nb in the Second Message

In this experiment, we removed all the nonces of principal A from both second and third messages while removing the nonce of principal B from the third message. In fact, we wanted to go one step further, and tried to see if the point we stopped (which is actually the previous experiment, removing all the nonces in message 3) is the right point to stop. Thus we obtained the modified version of the protocol shown in Table 5.9. We did the necessary changes in the MAC fields of both message-2 and message-3.

Table 5.9: PKMv2 without na in message 3 and no nbs

1. A → B: id, na, MAC{ id, na}$_K$ 2. B → A: na, id, S, MAC{ na, id, S}$_K$ 3. A → B: id, T, MAC{ id, T}$_K$

The LySa model after the modification is shown in Table 5.10. Compared to the base model, the initiator has changes in its only input and last output, whereas the responder has changes in its only output and second input.

Table 5.10: LySa model of PKMv2 without na in message 3 and no nbs

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\} \text{ in} \\
|_{i \in X}\ |_{j \in Z}\ !(\nu\ na_{ij})\ \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij}\ \text{dest}\ \{b1_{ij}\}\ ]\rangle. \\
(na_{ij}, id_{ij};\ xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij}\ \text{as}\ \{\!|na_{ij}, id_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij}\ \text{orig}\ \{b2_{ij}\}\ ]\ \text{in} \\
(\nu\ T_{ij})\ \langle id_{ij}, T_{ij}, \{\!|id_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij}\ \text{dest}\ \{b3_{ij}\}\ ]\rangle. \\
0 \\
| \\
|_{j \in X}\ |_{i \in Z}\ !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij}\ \text{as}\ \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij}\ \text{orig}\ \{a1_{ij}\}\ ]\ \text{in} \\
(\nu\ S_{ij})\ \langle yna_{ij}, id_{ij}, S_{ij}, \{\!|yna_{ij}, id_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij}\ \text{dest}\ \{a2_{ij}\}\ ]\rangle. \\
(id_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij}\ \text{as}\ \{\!|id_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij}\ \text{orig}\ \{a3_{ij}\}\ ]\ \text{in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ |_{j \in \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

Again we have found violation of authentication properties. The result is given as:

ψ = ($a1_{1\,1}$, $b3_{1\,1}$), ($a3_{1\,1}$, $b1_{1\,1}$), ($a1_{2\,1}$, $b3_{2\,1}$), ($a3_{2\,1}$, $b1_{2\,1}$), ($a1_{1\,2}$, $b3_{1\,2}$), ($a3_{1\,2}$, $b1_{1\,2}$), ($a1_{2\,2}$, $b3_{2\,2}$), ($a3_{2\,2}$, $b1_{2\,2}$)

This result is same as the one in the previous experiment and the explanation including the sample trace is in use for this one too.

5.1.3 Removing the Key Ids

In this part of the experiments we will be dealing with the ids in the messages. The important point is that, all the id fields in the base protocol are the same and sent in plaintext.

5.1.3.1 Removing the key id in the Last Message

We removed the id from the last message and modified the MAC as needed. The protocol is now as shown in Table 5.11.

Table 5.11: PKMv2 without id in message 3

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$
3. A → B: na, nb, T, MAC{ na, nb, T}$_K$

The LySa model after the modification is shown in Table 5.12. The changes are in the initiators last output and the responders last input.

Table 5.12: LySa model of PKMv2 without id in message 3

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\} \text{ in} \\
|_{i \in X}\, |_{j \in Z}\, !(\nu\ na_{ij})\, \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij} \text{ dest } \{b1_{ij}\}\,]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{\!|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij} \text{ orig } \{b2_{ij}\}\,] \text{ in} \\
(\nu\, T_{ij})\, \langle na_{ij}, nb_{ij}, T_{ij}, \{\!|na_{ij}, nb_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij} \text{ dest } \{b3_{ij}\}\,]\rangle. \\
0 \\
| \\
|_{j \in X}\, |_{i \in Z}\, !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij} \text{ orig } \{a1_{ij}\}\,] \text{ in} \\
(\nu\ nb_{ij})\, (\nu\ S_{ij})\, \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{\!|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij} \text{ dest } \{a2_{ij}\}\,]\rangle. \\
(na_{ij}, nb_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|na_{ij}, nb_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij} \text{ orig } \{a3_{ij}\}\,] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\, |_{j \in \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

The result of the analysis is: ***no violations possible***. This means that the protocol is still secure and the attacker still couldn't violate the authentication properties even though we didn't use the key id in the last message.

5.1.3.2 Removing the key ids in the Second and the Third Message

Seeing that removing one id did not make any effect, this time we removed the key ids in the last two messages and modified the MACs as needed. The protocol is now as shown in Table 5.13.

Table 5.13: PKMv2 without ids in message 2 and 3

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, nb, S, MAC{ na, nb, S}$_K$
3. A → B: na, nb, T, MAC{ na, nb, T}$_K$

The LySa model after the modification is shown in Table 5.14. The changes are in the initiators last output and the responders last input.

Table 5.14: LySa model of PKMv2 without ids in message 2 and 3

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\} \text{ in} \\
|_{i \in X}\ |_{j \in Z}\ !(\nu\ na_{ij})\ \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij}\ \text{dest } \{b1_{ij}\}\ ]\rangle. \\
(na_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{\!|na_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij}\ \text{orig } \{b2_{ij}\}\ ] \text{ in} \\
(\nu\ T_{ij})\ \langle na_{ij}, nb_{ij}, T_{ij}, \{\!|na_{ij}, nb_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij}\ \text{dest } \{b3_{ij}\}\ ]\rangle. \\
0 \\
| \\
|_{j \in X}\ |_{i \in Z}\ !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij}\ \text{orig } \{a1_{ij}\}\ ] \text{ in} \\
(\nu\ nb_{ij})\ (\nu\ S_{ij})\ \langle yna_{ij}, nb_{ij}, S_{ij}, \{\!|yna_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij}\ \text{dest } \{a2_{ij}\}\ ]\rangle. \\
(na_{ij}, nb_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|na_{ij}, nb_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij}\ \text{orig } \{a3_{ij}\}\ ] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ |_{j \in \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

Now we have found violation of authentication properties. The result is given as:

ψ = ($b2_{1\,1}$, $b3_{1\,1}$), ($a3_{1\,1}$, $a2_{1\,1}$), ($b2_{2\,1}$, $b3_{2\,1}$), ($a3_{2\,1}$, $a2_{2\,1}$), ($b2_{1\,2}$, $b3_{1\,2}$), ($a3_{1\,2}$, $a2_{1\,2}$), ($b2_{2\,2}$, $b3_{2\,2}$), ($a3_{2\,2}$, $a2_{2\,2}$), ($a3_{1\,0}$, $a2_{1\,0}$), ($a3_{2\,0}$, $a2_{2\,0}$), ($b2_{0\,2}$, $b3_{0\,2}$), ($b2_{0\,1}$, $b3_{0\,1}$)

We found traces for specific types of violation. Sample trace for ($b2_{1\,1}$, $b3_{1\,1}$) can be shown as:

1. A_1 → B_1: id_{11}, na_{11}, MAC{ id_{11}, na_{11}}$_{K11}$
2. B_1 → A_1: na_{11}, nb_{11}, S_{11}, MAC{ na_{11}, nb_{11}, S_{11}}$_{K11}$

2'. B_1 → M(A_1): na_{11}, nb_{11}, S_{11}, MAC{ na_{11}, nb_{11}, S_{11}}$_{K11}$

3. M(A_1) → B_1: na_{11}, nb_{11}, T_0, MAC{ na_{11}, nb_{11}, S_{11}}$_{K11}$

Sample trace for ($b2_{0\,1}$, $b3_{0\,1}$) can be shown as:

1. $M(A_0) \rightarrow B_1$: id_{01}, na_{01}, MAC{ id_{01}, na_{01}}$_{K01}$
2. $B_1 \rightarrow M(A_0)$: na_{01}, nb_{01}, S_{01}, MAC{ na_{01}, nb_{01}, S_{01}}$_{K01}$
3. $M(A_0) \rightarrow B_1$: na_{01}, nb_{01}, T_{01}, MAC{ na_{01}, nb_{01}, S_{01}}$_{K01}$

The difference between the possible non-confidential values in this experiment are:

nb_{01}, nb_{02}, nb_{10}, nb_{20}, na_{01}, na_{02}

This result shows that we cannot remove both ids in the protocol.

5.1.4 Removing Nonces and the Key Ids

In this part of the experiments we will be dealing with both the key ids and the nonces in the messages. We will only use the successful results in section 5.1.2 and 5.1.3. Therefore, this part can be seen as a synthesis of the two previous parts.

5.1.4.1 Removing the key id and nb in the Last Message

We removed the id and nb from the last message and modified the MAC as needed. As we showed before, removing those fields one by one did no changes, so this time we remove them together. The protocol is now as shown in Table 5.15.

Table 5.15: PKMv2 without id and nb in message 3

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$
3. A → B: na, T, MAC{ na, T}$_K$

The LySa model after the modification is shown in Table 5.16. The changes are in the initiators last output and the responders last input.

Table 5.16: LySa model of PKMv2 without id and nb in message 3

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\} \text{ in} \\
|_{i \in X}\ |_{j \in Z}\ !(\nu\ na_{ij})\ \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij}\ \text{dest } \{b1_{ij}\}\ ]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{\!|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij}\ \text{orig } \{b2_{ij}\}\ ] \text{ in} \\
(\nu\ T_{ij})\ \langle na_{ij}, T_{ij}, \{\!|na_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij}\ \text{dest } \{b3_{ij}\}\ ]\rangle. \\
0 \\
| \\
|_{j \in X}\ |_{i \in Z}\ !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij}\ \text{orig } \{a1_{ij}\}\ ] \text{ in} \\
(\nu\ nb_{ij})\ (\nu\ S_{ij})\ \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{\!|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij}\ \text{dest } \{a2_{ij}\}\ ]\rangle. \\
(na_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{\!|na_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij}\ \text{orig } \{a3_{ij}\}\ ] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ |_{j \in \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

The result of the analysis is: ***no violations possible***. This means that the protocol is still secure and the attacker still couldn't violate the authentication properties even though we didn't use the key id and nb in the last message. Definitely, this is a better result and better optimization. But now nb in the second message is useless, therefore this result is not practical.

5.1.4.2 Removing the key id and na in the Last Message

We removed the id and na from the last message and modified the MAC as needed. The protocol is now as shown in Table 5.17.

Table 5.17: PKMv2 without id and na in message 3

1. A → B: id, na, MAC{ id, na}$_K$
2. B → A: na, id, nb, S, MAC{ na, id, nb, S}$_K$
3. A → B: nb, T, MAC{ nb, T}$_K$

The LySa model after the modification is shown in Table 5.18. The changes are in the initiators last output and the responders last input.

Table 5.18: LySa model of PKMv2 without id and na in message 3

$$
\begin{array}{l}
\quad \mathsf{let}\ X \subseteq \mathbf{N}\ \mathsf{s.t.}\ \lfloor \mathbf{N} \rfloor = \{1,2\}\ \mathsf{in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\mathsf{let}\ Z \subseteq X \cup \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}\ \mathsf{in} \\
|_{i \in X}\, |_{j \in Z}\, !(\nu\ na_{ij})\, \langle id_{ij}, na_{ij}, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\mathsf{at}\ a1_{ij}\ \mathsf{dest}\ \{b1_{ij}\}\,] \rangle . \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\mathsf{decrypt}\ xmac_{ij}\ \mathsf{as}\ \{\!|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\mathsf{at}\ a2_{ij}\ \mathsf{orig}\ \{b2_{ij}\}\,]\ \mathsf{in} \\
(\nu\, T_{ij})\, \langle nb_{ij}, T_{ij}, \{\!|nb_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\mathsf{at}\ a3_{ij}\ \mathsf{dest}\ \{b3_{ij}\}\,] \rangle . \\
0 \\
| \\
|_{j \in X}\, |_{i \in Z}\, !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\mathsf{decrypt}\ ymac_{ij}\ \mathsf{as}\ \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\mathsf{at}\ b1_{ij}\ \mathsf{orig}\ \{a1_{ij}\}\,]\ \mathsf{in} \\
(\nu\ nb_{ij})\, (\nu\ S_{ij})\, \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{\!|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\mathsf{at}\ b2_{ij}\ \mathsf{dest}\ \{a2_{ij}\}\,] \rangle . \\
(nb_{ij};\ yT_{ij}, ymac_{ij}). \\
\mathsf{decrypt}\ ymac_{ij}\ \mathsf{as}\ \{\!|nb_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\mathsf{at}\ b3_{ij}\ \mathsf{orig}\ \{a3_{ij}\}\,]\ \mathsf{in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}}\, |_{j \in \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash \rangle . \\
0
\end{array}
$$

The result of the analysis is: ***no violations possible***. This means that the protocol is still secure and the attacker still couldn't violate the authentication properties even though we didn't use the key id in the last message.

Finally, this point is the best point of optimization since it is still secure and also practical. Namely, this version makes use of both nonces of A and B (actually BS and SS), and also key ids. Now we have seen the limits of the protocol and removed the redundant fields.

5.2 Fixing the Violations

As seen in the experiments, in some modified versions of the protocol, violations are seen and the problems occur in the encryption/decryption parts of the protocol, which are in fact the message authentication codes in our model. In this part, we go one step further and fix the errors and present secure versions of the protocols that had violations.

We change the implementation of message authentication codes. We hash the messages along with the key and a sequence number. The sequence numbers ensure messages within a single session cannot be confused with one another. We model thin LySa by using a sequence of public values Seq1, Seq2, . . . and each message will be encrypted along with one of these numbers using the current session key. For example the i'th message transfer from principal A to principal B will be:

$$
\begin{array}{lll}
1. & A \rightarrow & : A, B, \{Seq_i, Mess\}_{K_{Session}} \\
1.' & \rightarrow B : x_A, x_B, x_{EncryptedMess} & [\texttt{check } x_B = B] \\
1.'' & B : \texttt{decrypt } x_{EncryptedMess} \texttt{ as } \{x_{Seq}, x_{Mess}\}_{K_{Session}} & [\texttt{check } x_{Seq} = Seq_i]
\end{array}
$$

5.2.1 Fix for Removing All the Nonces in the Last Message

Adding sequence numbers into the message authentication codes fix the violations in the version without nonces in message three. The LySa model of this version is given below. The LySa results are explained in section 5.2.4.

Table 5.19: LySa model of the fixed version of the experiment 5.1.2.3.

$$
\begin{array}{l}
\quad \mathsf{let}\ X \subseteq \mathbf{N}\ \mathsf{s.t.}\ \lfloor \mathbf{N} \rfloor = \{1,2\}\ \mathsf{in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\mathsf{let}\ Z \subseteq X \cup \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}\ \mathsf{in} \\
|_{i \in X} |_{j \in Z} !(\nu\ na_{ij})\ \langle id_{ij}, na_{ij}, \{Seq1_i, \{|id_{ij}, na_{ij}|\}_{Hash}\}_{K_{ij}} [\mathsf{at}\ a1_{ij}\ \mathsf{dest}\ \{b1_{ij}\}\ ]\rangle. \\
(na_{ij}, id_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\mathsf{decrypt}\ xmac_{ij}\ \mathsf{as}\ \{Seq2_i, \{|na_{ij}, id_{ij}, xnb_{ij}, xS_{ij}|\}_{Hash};\ \}_{K_{ij}} [\mathsf{at}\ a2_{ij}\ \mathsf{orig}\ \{b2_{ij}\}\ ]\ \mathsf{in} \\
(\nu\ T_{ij})\ \langle id_{ij}, T_{ij}, \{Seq3_i, \{|id_{ij}, T_{ij}|\}_{Hash}\}_{K_{ij}} [\mathsf{at}\ a3_{ij}\ \mathsf{dest}\ \{b3_{ij}\}\ ]\rangle. \\
0 \\
| \\
|_{j \in X} |_{i \in Z} !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\mathsf{decrypt}\ ymac_{ij}\ \mathsf{as}\ \{Seq1_j, \{|id_{ij}, yna_{ij}|\}_{Hash};\ \}_{K_{ij}} [\mathsf{at}\ b1_{ij}\ \mathsf{orig}\ \{a1_{ij}\}\ ]\ \mathsf{in} \\
(\nu\ nb_{ij})\ (\nu\ S_{ij})\ \langle yna_{ij}, id_{ij}, nb_{ij}, S_{ij}, \{Seq2_j, \{|yna_{ij}, id_{ij}, nb_{ij}, S_{ij}|\}_{Hash}\}_{K_{ij}} [\mathsf{at}\ b2_{ij}\ \mathsf{dest}\ \{a2_{ij}\}\ ]\rangle. \\
(id_{ij};\ yT_{ij}, ymac_{ij}). \\
\mathsf{decrypt}\ ymac_{ij}\ \mathsf{as}\ \{Seq3_j, \{|id_{ij}, yT_{ij}|\}_{Hash};\ \}_{K_{ij}} [\mathsf{at}\ b3_{ij}\ \mathsf{orig}\ \{a3_{ij}\}\ ]\ \mathsf{in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}} |_{j \in \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

5.2.2 Fix for Removing All Nonces in the Last Message and nb in the Second Message

Adding sequence numbers into the message authentication codes fix the violations in the version without nonces in message three and nb in message two. The LySa model of this version is given below. The LySa results are explained in section 5.2.4.

Table 5.20: LySa model of the fixed version of the experiment 5.1.2.4.

$$
\begin{array}{l}
\quad \mathsf{let}\ X \subseteq \mathbf{N}\ \mathsf{s.t.}\ \lfloor \mathbf{N} \rfloor = \{1,2\}\ \mathsf{in} \\
(\nu_{i\in X, j\in X}\ K_{ij}) \\
(\nu_{i\in X, j\in X}\ id_{ij}) \\
\mathsf{let}\ Z \subseteq X \cup \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}\ \mathsf{in} \\
|_{i\in X}\,|_{j\in Z}\,!(\nu\ na_{ij})\,\langle id_{ij}, na_{ij}, \{Seq1_i, \{|id_{ij}, na_{ij}|\}_{Hash}\}_{K_{ij}}[\mathsf{at}\ a1_{ij}\ \mathsf{dest}\ \{b1_{ij}\}\,]\rangle. \\
(na_{ij}, id_{ij};\ xS_{ij}, xmac_{ij}). \\
\mathsf{decrypt}\ xmac_{ij}\ \mathsf{as}\ \{Seq2_i, \{|na_{ij}, id_{ij}, xS_{ij}|\}_{Hash};\ \}_{K_{ij}}[\mathsf{at}\ a2_{ij}\ \mathsf{orig}\ \{b2_{ij}\}\,]\,\mathsf{in} \\
(\nu\, T_{ij})\,\langle id_{ij}, T_{ij}, \{Seq3_i, \{|id_{ij}, T_{ij}|\}_{Hash}\}_{K_{ij}}[\mathsf{at}\ a3_{ij}\ \mathsf{dest}\ \{b3_{ij}\}\,]\rangle. \\
0 \\
| \\
|_{j\in X}\,|_{i\in Z}\,!(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\mathsf{decrypt}\ ymac_{ij}\ \mathsf{as}\ \{Seq1_j, \{|id_{ij}, yna_{ij}|\}_{Hash};\ \}_{K_{ij}}[\mathsf{at}\ b1_{ij}\ \mathsf{orig}\ \{a1_{ij}\}\,]\,\mathsf{in} \\
(\nu\, S_{ij})\,\langle yna_{ij}, id_{ij}, S_{ij}, \{Seq2_j, \{|yna_{ij}, id_{ij}, S_{ij}|\}_{Hash}\}_{K_{ij}}[\mathsf{at}\ b2_{ij}\ \mathsf{dest}\ \{a2_{ij}\}\,]\rangle. \\
(id_{ij};\ yT_{ij}, ymac_{ij}). \\
\mathsf{decrypt}\ ymac_{ij}\ \mathsf{as}\ \{Seq3_j, \{|id_{ij}, yT_{ij}|\}_{Hash};\ \}_{K_{ij}}[\mathsf{at}\ b3_{ij}\ \mathsf{orig}\ \{a3_{ij}\}\,]\,\mathsf{in} \\
0 \\
| \\
|_{i\in X\cup \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}}\,|_{j\in \mathbf{N}\_\ \mathsf{s.t.}\ \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

5.2.3 Fix for Removing the key ids in the Second and the Third Message

Adding sequence numbers into the message authentication codes fix the violations in the version without key ids in message two and three. The LySa model of this version is given below. The LySa results are explained in section 5.2.4.

Table 5.21: LySa model of the fixed version of the experiment 5.1.3.2.

$$
\begin{array}{l}
\quad \text{let } X \subseteq \mathbf{N} \text{ s.t. } \lfloor \mathbf{N} \rfloor = \{1,2\} \text{ in} \\
(\nu_{i \in X, j \in X}\ K_{ij}) \\
(\nu_{i \in X, j \in X}\ id_{ij}) \\
\text{let } Z \subseteq X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\} \text{ in} \\
|_{i \in X}\, |_{j \in Z}\, !(\nu\ na_{ij})\, \langle id_{ij}, na_{ij}, \{Seq1_i, \{\!|id_{ij}, na_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a1_{ij} \text{ dest } \{b1_{ij}\}\,]\rangle. \\
(na_{ij};\ xnb_{ij}, xS_{ij}, xmac_{ij}). \\
\text{decrypt } xmac_{ij} \text{ as } \{Seq2_i, \{\!|na_{ij}, xnb_{ij}, xS_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } a2_{ij} \text{ orig } \{b2_{ij}\}\,] \text{ in} \\
(\nu\, T_{ij})\, \langle na_{ij}, nb_{ij}, T_{ij}, \{Seq3_i, \{\!|na_{ij}, nb_{ij}, T_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } a3_{ij} \text{ dest } \{b3_{ij}\}\,]\rangle. \\
0 \\
| \\
|_{j \in X}\, |_{i \in Z}\, !(id_{ij};\ yna_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{Seq1_j, \{\!|id_{ij}, yna_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b1_{ij} \text{ orig } \{a1_{ij}\}\,] \text{ in} \\
(\nu\, nb_{ij})\, (\nu\, S_{ij})\, \langle yna_{ij}, nb_{ij}, S_{ij}, \{Seq2_j, \{\!|yna_{ij}, nb_{ij}, S_{ij}|\!\}_{Hash}\}_{K_{ij}} [\text{at } b2_{ij} \text{ dest } \{a2_{ij}\}\,]\rangle. \\
(na_{ij}, nb_{ij};\ yT_{ij}, ymac_{ij}). \\
\text{decrypt } ymac_{ij} \text{ as } \{Seq3_j, \{\!|na_{ij}, nb_{ij}, yT_{ij}|\!\}_{Hash};\ \}_{K_{ij}} [\text{at } b3_{ij} \text{ orig } \{a3_{ij}\}\,] \text{ in} \\
0 \\
| \\
|_{i \in X \cup \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\, |_{j \in \mathbf{N}\_ \text{ s.t. } \lfloor \mathbf{N}\_ \rfloor = \{0\}}\ \langle K_{ij}, K_{ji}, id_{ij}, id_{ji}, Hash\rangle. \\
0
\end{array}
$$

5.2.4 LySa Results for the Fixes

The LySa results of the fixed models for the modifications that caused violations are all the same: ***no violations possible***. This means that this versions of the protocol are **again** secure. Since the problems occurred from the message authentication codes, changing the implementations of them fixed the violations. As a simple note, this was not included in our proposal at the beginning. Therefore, the fixes are just for showing how to fix a violation about message authentication codes.

5.3 Analysis Results

As seen in the experiments, we established the analysis in four steps. First of all, we analyzed the base model. We had successful results for the base model. Then we removed the nonces starting from the ones in the last message. The results for nonces showed us that removing neither the nonce of the principal A nor the principal B does not change the

secure standing of the protocol. But removing them both, causes problems especially replay attacks. After that we checked the nonces in the second message but removing them also caused problem so we stopped. We didn't try some combinations such as removing nb in message since it was the first usage of it. As a third step we removed the key ids which were always the same in three messages and sent in plaintext. Removing the last id still preserved the robustness, but doing more with the ids caused problems. Finally we got the secure paths from the nonce experiments and the id experiments and merged them to get a combined path which is optimized but secure. Thus we found out that removing the id and one of the nonces in the last message does not cause any flow. Whereas, removing both nonces in the last message or removing a nonce from the second message with a missed id makes the protocol fail.

As a result we may have a simplified but still strong and secure protocol if we make the optimizations that we found successful in our analysis. In addition, reducing the number of fields will also have better performance results since the bandwidth usage is also important in wireless networks.

Another result of this analysis is that the lengths of the fields are also very important since the error components in the static analysis show that the same MACs can be created when the implementations take some field lengths the same.

As we mentioned in the static analysis chapter, the errors in this analysis do not always show that the protocol has flaws, whereas the successful runs of the analysis are always successful. The reason of this behavior is the over-approximation of the analysis. Therefore, the experiments where we got the results with no violations show that the studied protocols are secure. But the experiments with violations in the result needed some traces to show the flaws. We found traces for the flaws to show that that versions of the protocol really had flaws.

In conclusion, the results of the static analysis support our assertion that the PKMv2 can be improved by optimization without any loss of security.

CHAPTER 6

Conclusion

Security is important in all types of data communications but it is an essential and also a tough subject in wireless networks. The IEEE 802.16 standard which is certified as WiMAX is the strongest competitor of 3G and still rapidly growing. In this thesis, the latest version of the standard, the IEEE 802.16e-2005 is considered. This version of standard's most important feature is mobility but it also has significant improvements in security. The reason for such security improvements in this version was the big failure in the previous security protocol PKMv1. Similar to the drawbacks in IEEE 802.11b which was fixed in IEEE 802.11i, the drawbacks of IEEE 802.16-2004 is now fixed by IEEE 802.16e-2005.

In this thesis, the studies are divided into four groups. First of all, the security sublayers of the current and former IEEE 802.16 protocols are studied. These studies led us to the PKM protocol which has two main issues: an Authentication/Authorization scheme to establish a shared authorization key, and a second scheme to distribute the traffic encryption keys. The latest version of the protocol, PKMv2, leaves the first issue to de facto standards which are proved to be secure such as RSA and EAP, therefore fixes the ambiguities in the first version. The

second issue was very weak in the PKMv1 and is highly strengthened by the PKMv2 named as PKMv2 SA-TEK 3-Way Handshake. PKMv1 had many missing security features, whereas PKMv2 is over-strengthened which does not mean that PKMv2 is the ultimate secure protocol. The truth is that, it has definitely degraded efficiency since it needs more sources and time for the security features it has. The assertion of this thesis is that, PKMv2 can still pursue its security with less features than it has. The redundancy in the PKMv2 is being questioned.

The second part of the work is about LySa process calculus. LySa allows communication protocols to be specified and annotated allowing for validation of authentication properties. To make the static analysis of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol we had to formalize it in LySa calculus which is based on pi calculus. The next part of the work is the static analysis method. Static analysis is successfully used for automatically validating security properties of classical protocols. Using these three parts of work, we were able to derive a model of the protocol and describe it using LySa and carry out a static analysis of the LySa process using the static analysis tool LySa-tool.

Last part of our work was the analysis to see the limits of robustness in IEEE 802.16 PKMv2. The way we do that was removing the extras and the improvements in PKMv2 one by one, and in different combinations. We wanted to see when the robustness would be lost and also if there were some unnecessary improvements. Since this was an over-strengthened protocol we could try to provide better efficiency with less strength. So that the result may lead us to a simplified by still strong and secure protocol.

We established many experiments and took the important ones here. Our analysis results shows that some fields are unnecessary and does not affect security at all. Special combinations of those fields are also redundant and shown by our experiments. This results support our ideas about optimizing the protocol. Thus, according to our static analysis results based on LySa process calculus, we can say that this protocol is secure enough itself and will still be secure even though some components are removed. The limits of the robustness is measured and given in the analysis results. In addition, the possible flaws when this limits are exceeded are mentioned.

The soundness of the analysis based on Lysa is proved in previous studies, especially in [13] and [26]. The method of the analysis is described in details in [20].

As a future work, the former parts of the PKMv2 can be modeled and analyzed so that the results can be joined with the results of this thesis and a security analysis framework can be developed.

APPENDIX A

Lysa Codes

A.1 The PKMv2 SA-TEK 3-Way Handshake

```
/*********************************************************/
/*                                                       */
/*      LySa Codes for thesis          1.lysa            */
/*      <01-02-2007 Ender Yuksel>                         */
/*                                                       */
/*      5.1.1 The PKMv2 SA-TEK 3-Way Handshake           */
/*                                                       */
/*********************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}
```

```
      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <na_{i, j}, nb_{i,
j}, id_{i, j}, T_{i, j}, {{|na_{i, j}, nb_{i, j}, id_{i,
j}, T_{i, j}|}:Hash}:K_{i, j} [at a3_{i, j} dest {b3_{i,
j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (na_{i,j}, nb_{i,j}, id_{i,j};
yT_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|na_{i,j},
nb_{i,j}, id_{i,j}, yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j}
orig {a3_{i,j}}] in 0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.2 Removing nb in the Last Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for thesis           2.1.lysa        */
/*      <01-02-2007 Ender Yuksel>                       */
/*                                                     */
/*      5.1.2.1 Removing nb in the Last Message        */
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <na_{i, j}, id_{i,
j}, T_{i, j}, {{|na_{i, j}, id_{i, j}, T_{i,
j}|}:Hash}:K_{i, j} [at a3_{i, j} dest {b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
```

```
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (na_{i,j}, id_{i,j}; yT_{i,j},
ymac_{i,j}).
                       decrypt ymac_{i,j} as {{|na_{i,j},
id_{i,j}, yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig
{a3_{i,j}}] in 0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.3 Removing na in the Last Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for thesis          2.2.lysa        */
/*      <01-02-2007 Ender Yuksel>                      */
/*                                                     */
/*      5.1.2.2 Removing na in the Last Message       */
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <nb_{i, j}, id_{i,
j}, T_{i, j}, {{|nb_{i, j}, id_{i, j}, T_{i,
j}|}:Hash}:K_{i, j} [at a3_{i, j} dest {b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
```

```
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (nb_{i,j}, id_{i,j}; yT_{i,j},
ymac_{i,j}).
                       decrypt ymac_{i,j} as {{|nb_{i,j},
id_{i,j}, yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig
{a3_{i,j}}] in 0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.4 Removing All the Nonces in the Last Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for thesis          2.3.lysa        */
/*      <01-02-2007 Ender Yuksel>                      */
/*                                                     */
/*      5.1.2.3 Removing All the Nonces in the Last    */
/*              Message                                */
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <id_{i, j}, T_{i, j},
{{|id_{i, j}, T_{i, j}|}:Hash}:K_{i, j} [at a3_{i, j} dest
{b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
```

```
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (id_{i,j}; yT_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig {a3_{i,j}}] in
0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.5 Removing All nonces in the Last Message and nb in the Second Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for thesis            3.3.lysa      */
/*      <01-02-2007 Ender Yuksel>                       */
/*                                                     */
/*      5.1.2.4 Removing All Nonces in the Last Message */
/*               and nb in the Second Message
*/
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xS_{i, j},
xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at a2_{i, j} orig
{b2_{i, j}}] in
                        (new T_{i, j}) <id_{i, j}, T_{i, j},
{{|id_{i, j}, T_{i, j}|}:Hash}:K_{i, j} [at a3_{i, j} dest
{b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
```

```
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
                        (new S_{i,j})<yna_{i,j}, id_{i,j},
S_{i,j}, {{|yna_{i,j}, id_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (id_{i,j}; yT_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig {a3_{i,j}}] in
0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.6 Removing the key id in the Last Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for thesis            4.1a.lysa     */
/*      <01-02-2007 Ender Yuksel>                       */
/*                                                     */
/*      5.1.3.1 Removing the key id in the Last Message */
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <na_{i, j}, nb_{i,
j}, T_{i, j}, {{|na_{i, j}, nb_{i, j}, T_{i,
j}|}:Hash}:K_{i, j} [at a3_{i, j} dest {b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
```

```
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (na_{i,j}, nb_{i,j}; yT_{i,j},
ymac_{i,j}).
                       decrypt ymac_{i,j} as {{|na_{i,j},
nb_{i,j}, yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig
{a3_{i,j}}] in 0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.7 Removing the key ids in the Second and the Third Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for                  4.1b.lysa      */
/*      <01-02-2007 Ender Yuksel>                      */
/*                                                     */
/*      5.1.3.2 Removing the key ids in the Second     */
/*              and the Third Message
*/
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}; xnb_{i, j}, xS_{i, j},
xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at a2_{i, j} orig
{b2_{i, j}}] in
                        (new T_{i, j}) <na_{i, j}, nb_{i,
j}, T_{i, j}, {{|na_{i, j}, nb_{i, j}, T_{i,
j}|}:Hash}:K_{i, j} [at a3_{i, j} dest {b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
```

```
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, nb_{i,j}, S_{i,j}, {{|yna_{i,j},
nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j} [at b2_{i,j} dest
{a2_{i,j}}]>.
                        (na_{i,j}, nb_{i,j}; yT_{i,j},
ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|na_{i,j},
nb_{i,j}, yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig
{a3_{i,j}}] in 0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.8 Removing the key id and nb in the Last Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for                  4.1c.lysa      */
/*      <01-02-2007 Ender Yuksel>                       */
/*                                                     */
/*      5.1.4.1 Removing the key id and nb in the      */
/*              Last Message                           */
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <na_{i, j}, T_{i, j},
{{|na_{i, j}, T_{i, j}|}:Hash}:K_{i, j} [at a3_{i, j} dest
{b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
```

```
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (na_{i,j}; yT_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|na_{i,j},
yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig {a3_{i,j}}] in
0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

A.9 Removing the key id and na in the Last Message

```
/*******************************************************/
/*                                                     */
/*      LySa Codes for                  4.1d.lysa      */
/*      <01-02-2007 Ender Yuksel>                       */
/*                                                     */
/*      5.1.4.2 Removing the key id and na             */
/*              in the Last Message                    */
/*                                                     */
/*******************************************************/


let X subset NATURAL2 in

(new_{i in X, j in X} K_{i, j})
(new_{i in X, j in X} id_{i, j})

let Z subset X union ZERO in(

/* Principal A_{i} */
(|_{i in X}

      /* Initiating a session with principal B_{j} */
      (|_{j in Z} !(new na_{i,j})
                  <id_{i, j}, na_{i,j}, {{|id_{i, j},
na_{i, j}|}:Hash}:K_{i, j} [at a1_{i, j} dest {b1_{i,
j}}]>.
                        (na_{i, j}, id_{i, j}; xnb_{i, j},
xS_{i, j}, xmac_{i, j}).
                        decrypt xmac_{i, j} as {{|na_{i, j},
id_{i, j}, xnb_{i, j}, xS_{i, j}|}:Hash;}:K_{i, j} [at
a2_{i, j} orig {b2_{i, j}}] in
                        (new T_{i, j}) <nb_{i, j}, T_{i, j},
{{|nb_{i, j}, T_{i, j}|}:Hash}:K_{i, j} [at a3_{i, j} dest
{b3_{i, j}}]>.0 )
)
|
/* Principal B */
(|_{j in X}

     /* Responding to a session from principal A_{i} */
     (|_{i in Z} ! (id_{i,j}; yna_{i,j}, ymac_{i,j}).
```

```
                        decrypt ymac_{i,j} as {{|id_{i,j},
yna_{i,j}|}:Hash;}:K_{i,j} [at b1_{i,j} orig {a1_{i,j}}] in
                        (new nb_{i,j})(new
S_{i,j})<yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j},
{{|yna_{i,j}, id_{i,j}, nb_{i,j}, S_{i,j}|}:Hash}:K_{i,j}
[at b2_{i,j} dest {a2_{i,j}}]>.
                        (nb_{i,j}; yT_{i,j}, ymac_{i,j}).
                        decrypt ymac_{i,j} as {{|nb_{i,j},
yT_{i,j}|}:Hash;}:K_{i,j} [at b3_{i,j} orig {a3_{i,j}}] in
0 )
)
|
/* Credentials of illegitimate principals */
|_{i in X union ZERO} |_{j in ZERO} <K_{i, j}, K_{j, i},
id_{i, j},id_{j, i}, Hash>.0
)
```

Bibliography

[1] **Johnston, D. and Walker, J.**, 2004. Overview of IEEE 802.16 security, *IEEE Security & Privacy Magazine*, vol. 2, Issue: 3, 40 – 48.

[2] **IEEE Std 802.16e-2005**, 2006. Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1, *IEEE*, New York, USA.

[3] **Lowe, G.**, 1996. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR, *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, Springer-Verlag, 147-166.

[4] **Lowe, G.**, 1995. An attack on the needham-schroeder public-key authentication protocol, *Information Processing Letters*, 56(3), 131-133.

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

[6] **Mitchell, J. C., Mitchell, M. and Stern, U.**, 1997. Automated Analysis of Cryptographic Protocols Using Murphi, *IEEE Symposium on Security and Privacy*. IEEE Press, 141-151.

[7] **Shmatikov, V. and Stern, U.**, 1998. Efficient finite-state analysis for large security protocols, *IEEE Computer Security Foundations Workshop*, 106-115.

[8] **Corin, R. and Saptawijaya, A.**, 2006. A logic for constraint-based security protocol analysis, *IEEE Symposium on Security and Privacy*, 155 – 168.

[9] **Tanenbaum, A. S.**, 2003. Computer Networks, Fourth Edition, Prentice Hall,
New Jersey.

[10] **Burrows, M., Abadi, M. and Needham, R. M.**, 1989. A logic of authentication, *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 426(1871), 233-271.

[11] **Diffie, W. and Hellman, M.E.**, 1976. New Directions in Cryptography, *IEEE*
Transactions on Information Theory, vol. IT-22, 644-654.

[12] **Dolev, D. and Yao, A.C.**, 1983. On the security of public key protocols, *IEEE*
Transactions on Information Theory, IT-29(12), 198-208.

[13] **Bodei, C., Buchholtz, M., Degano, P., Nielson, H.R. and Nielson, F.**, 2004.
Static Validation of Security Protocols, *Journal of Computer Security*, 347-390.

[14] **Lin, P. and Lin, L.**, 1996. Security in enterprise Networking: A quick tour, *IEEE Communications Magazine*, January, 56.

[15] **Bella, G., Massacci, F. and Paulson, L.C.**, 2003. Verifying the set registration
protocols, IEEE *Journal on Selected Areas in Communications*, 21(1), 77-87.

[16] **Behm, P., Benoit, P., Faivre, A. and Meynadier, J.M.**, 1999. Meteor: a successful application of B in a large project, *Lecture Notes in Computer Science*, 1708/1999, 369.

[17] **Baier, C. and Katoen, J.P**., 2006. Principles of model checking, Preprint.

[18] **Bodei, C., Buchholtz, M., Degano, P., Nielson, F. and Nielson, H.R.**, 2003. Automatic validation of protocol narration, *Proceedings of the 16th Computer Security Foundations Workshop (CSFW 03)*, 126-140.

[19] **Milner, R., Parrow, J. and Walker, D.**, 1992. A calculus of mobile processes,
Information and Computation, 100(1), 1-40.

[20] **Nielson, F., Nielson, H.R. and Hankin, C.**, 1999. Principles of Program Analysis, Springer-Verlag, New York..

[21] **Abadi, M. and Gordon, A. D.**, 1999. A calculus for cryptographic protocols: The spi calculus, *Information and Computation*, 148(1), 1-70.

[22] **IEEE Std 802.16-2004**, Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems, *IEEE*, New York, USA.

[23] **Andova, S., Cremers, C., Gjosteen, K., Mauw, S., Mjolsnes, S. F., and Radomirovica, S.**, 2006. Framework for compositional verification of security protocols, *preprint submitted to Elsevier*.

[24] **Yüksel, E., Soytürk, M., Ovatman, T. and Örencik, B.**, 2005. Security Problem in Wireless Local Area Networks, *National Symposium on Network and Information Systems Security*, 23-30.

[25] **Datta, A., He, C., Mitchell, J. C., Roy, A. and Sundararajan, M.**, 2005. 802.16eNotes, *IETF Liasons*.

[26] **Buchholtz, M., Nielson, H.R. and Nielson, F.**, 2004. A calculus for control flow analysis of security protocols, *International Journal on Information Security*, 2(3-4), 145-167.

[27] **Hansen, S., Skriver, J. and Nielson, H.R.**, 2005. Using static analysis to validate the SAML single sign-on protocol, *Workshop on Issues in the Theory of Security*, 27-40.

[28] **Milner, R.**, 1999. Communicating and mobile systems: the π-calculus, *Cambridge University Press*, fifth edition.

[29] **Abadi, M.**, 1999. Security protocols and specifications, *Proceedings of the Second International Conference on Foundations of Software Science and Computation Structure*, pages 1–13.