A Secure Simplification of the PKMv2 Protocol in IEEE 802.16e-2005

Ender Yüksel[1], Hanne Riis Nielson[2], Christoffer Rosenkilde Nielsen[2], and Mehmet Bülent Örencik[1]

[1] Department of Computer Engineering, Istanbul Technical University, 34469 Istanbul, Turkey

[2] Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads bldg 321, DK-2800 Kongens Lyngby, Denmark

yuksele@itu.edu.tr riis@imm.dtu.dk crn@imm.dtu.dk

bulent.orencik@bte.mam.gov.tr

Abstract. Static analysis is successfully used for automatically validating security properties of classical cryptographic protocols. In this paper, we shall employ the same technique to a modern security protocol for wireless networks, namely the latest version of the Privacy and Key Management protocol for IEEE 802.16e, PKMv2. This protocol seems to have an exaggerated mixture of security features. Thus, we iteratively investigate which components are necessary for upholding the security properties and which can be omitted safely. This approach is based on the LySa process calculus and employs the corresponding automated analysis tool, the LySaTool.

Keywords: Protocol Validation, Process Calculi, Static Analysis, Authentication, IEEE 802.16e

1 Introduction

Security in wireless networks is of great concern as the wireless medium faces different threats from wired networks. Thus, in order to provide secure communication, these threats must be taken into account in the design of security protocols for wireless networks. However the standard for wireless metropolitan area networks, IEEE 802.16, incorporated a pre-existing standard called Data Over Cable Service Interface Specifications, designed for wired networks. Therefore the standard failed to protect the IEEE 802.16 link [1] and significant changes in its Privacy and Key Management protocol (PKMv1) were required.

The latest standard, IEEE 802.16e-2005 [2], includes a new version (PKMv2) of the protocol that caters for the shortcomings of the first version. Derivation of the authorization key is now derived by the contribution of both parties using well known standards such as RSA and EAP, where it initially was done only by the base station (BS). Additionally, BS is extended with a certificate, allowing for mutual authentication with the mobile station (MS), which was missing in

PKMv1. Finally, nonces are incorporated in order to avoid replay attacks. These corrections, though thought to benefit the security of the protocol, have also intensively complicated it. This motivates an investigation of which extensions that are really necessary and which that can be omitted without compromising the protocol.

Formal analysis of cryptographic protocols is normally concerned with whether a given protocol satisfies a number of security criteria such as correct establishment of a secret shared key or authentication the principals involved. This has been a very active area of research over the last decades, and the tools, that have been constructed based on the theoretical development, have successfully located many hitherto unknown flaws. One of the most well-known example is the Lowe's attack [3, 4] of the Needham Schroeder public key protocol [5] using the process algebra Communicating Sequential Processes (CSP) and the Failures-Divergences Refinement which is the model checker for CSP [6]. Similar examples are obtained by Shmatikov and Stern [7] using Murphi, and Corin et al. [8] using symbolic traces and Pure-past Security - Linear Temporal Logic successfully.

This paper builds on this development, but with a different focus of interest. Relying on a well-established verification tool, the LySaTool, we shall iteratively attempt to remove components of the PKMv2 protocol and investigate the influence it has on the security properties. Our analysis shows that not only is the PKMv2 SA-TEK 3-Way Handshake secure, but that it can even be simplified by removal of some redundant fields without compromising the overall security protocol.

2 PKMv2 in IEEE 802.16e-2005

The second version of the Privacy and Key Management (PKMv2) protocol of IEEE 802.16 is described in IEEE 802.16e-2005 and aims to fix the bugs in the former version. In the first part of the protocol, an Authorization Key (AK) is generated using RSA or EAP or both. After that, each party generates a Key Encryption Key (KEK) using their AKs. 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, SA-TEK 3-Way Handshake, which lets MS to gather TEKs from BS. In the handshake, TEKs are encrypted by KEKs. The process can be seen in Fig. 1.

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 1.

The PKMv2 SA-TEK 3-Way handshake sequence proceeds as shown in Table 2.

We had simplified the protocol, making use of the work of John Mitchell [14] (that was used in his security review together with IETF EAP Work Group), made the necessary changes that are necessary for LySa and obtained the follow-

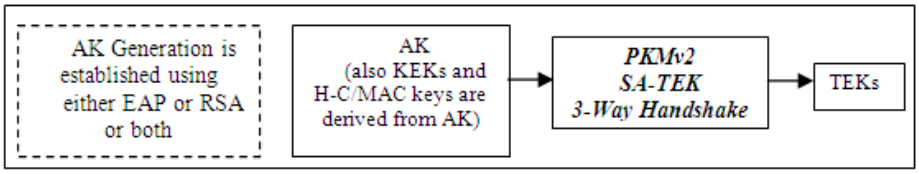

Fig. 1. The PKMv2 Process

Table 1. The PKMv2 SA-TEK 3-Way Handshake Protocol Fields

Attribute	**Content**
MS_Random	Random number received from MS
BS_Random	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, optionalonhandoveretc.
FrameNo	The frame number that old PMKs and AKs should be discarded
SA_Descriptors	Descriptors of the SA, only for initial entry
SecNegParam	Confirms messages security capabilities
HMAC/CMAC	Message Authentication Codes

Table 2. The PKMv2 SA-TEK 3-Way Handshake Protocol Narration

1. SA-TEK-Challenge
BS $\rightarrow$ **MS:** $BS\_Random, KeySeqNo, AKID, [KeyLifeTime], H-C/MAC$
2. SA-TEK-Request
MS $\rightarrow$ **BS:** $MS\_Random, BS\_Random, KeySeqNo, AKID, SecurityCapabilities, SecNegParam, PKMConfSettings, H-C/MAC$
3. SA-TEK-Response
BS $\rightarrow$ **MS:** $MS\_Random, BS\_Random, KeySeqNo, AKID, [SA-TEKUpdate], FrameNo, [SADescriptors], SecNegParam, H-C/MAC$

ing protocol narration in Table 3. A, B, id, na, nb, S, T, K stands for BS, MS, AKID, BS_Random, MS_Random, SecurityCapabilities + SecNegParam +PKM-ConfSettings, SA-TEKUpdate and AK, respectively.

The first message, named as PKMv2 SA-TEK-Challenge, includes a random number generated by A (na) and the id of the authorization key (id) and protected by the message authentication code (MAC).

The second message is the PKMv2 SA-TEK-Request which includes na and id received in the first message in addition to the random number generated by B, nb, security configuration details S and the message authentication code for the remaining fields.

Table 3. PKMv2 SA-TEK 3-Way Handshake Simplified Protocol Narration

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

Upon reception A checks the id, MAC and the na of and if any of these values are invalid, it discards the message. Otherwise, it checks the security capabilities provided by the B and if the properties does not match it reports this inconsistency to the higher layers.

If the second message is successfully validated by A then message 3 which is named as the PKMv2 SATEK-Response is sent to B. This message has the TEKs T.

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

This simplification of the protocol is verified using Murphi model checker in [14].

3 LySa Calculus

To analyze the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol we need to formalize it in LySA calculus. LySA [10] is a process calculus based on the π-calculus [16] and incorporates cryptographic operations using ideas from the Spi-calculus [13]. However, 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 is 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.

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 4.

The tuples of terms $E_1, \ldots, E_k$ 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 [13] is shown in Table 5.

Table 4. LySA Terms

$E ::= x$		variable
	n	name
	k^+/k^-	public and private keys
	$\{E_1, \ldots, E_k\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}\,]$	symmetric encryption
	$\{\!\{E_1, \ldots, E_k\}\!\}^{\ell}_{E_0}[\mathsf{dest}\ \mathcal{L}\,]$	asymmetric encryption

Table 5. LySA Processes

$P ::= \mathsf{0}$		nil
	$P_1 \mid P_2$	parallel
	$!P$	replication
	$(\nu\, n)\, P$	restriction (name)
	$(\nu_{\pm}\, m)\, P$	restriction (key pair)
	$\langle E_1, \ldots, E_k \rangle P$	output
	$(E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P$	input
	$\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \ldots, E_j;\ x_{j+1}, \ldots, x_k\}^{\ell}_{E_0}[\mathsf{orig}\ \mathcal{L}\,]\ \mathsf{in}\ P$	symmetric decrypt.
	$\mathsf{decrypt}\ E\ \mathsf{as}\ \{\!\{E_1, \ldots, E_j;\ x_{j+1}, \ldots, x_k\}\!\}^{\ell}_{E_0}[\mathsf{orig}\ \mathcal{L}\,]\ \mathsf{in}\ P$	asymmetric decrypt.

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 $(E_1, \ldots, E_j\ ;\ x_{j+1}, \ldots, x_k).P$ means that a k-tuple of values $(E'_1, \ldots, E'_k)$ 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 bound to the variables $x_{j+1}, \ldots, x_k$. 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 5. 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,

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

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

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

means that the decryption in Q always succeeds, binding x to y. In both examples, the remainder processes P' and Q' are only executed if the decryptions succeed and of course P and Q have the implicit check that the length of the tuples are the same.

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-point*s defining its position in the process, and with *assertion*s specifying the origin and destination of encrypted messages. Crypto-points ℓ are from some enumerable set $\mathcal{C}$ and added to state where the encryptions and decryptions occur. The LySA term for encryption:

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

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

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

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

The actual semantics have been omitted for lack of space, but are present in [10].

A LySa process may generate a large number of names which would cause infinite sets of names to be recorded. These sets are partitioned into finitely many equivalence classes for the efficiency of the analysis. A canonical value is a representative for each of these equivalence classes. For each name n, there is a canonical representative $\lfloor n \rfloor$ and extended homomorphically to terms, $\lfloor E \rfloor$ is the term where all names and variables are replaced by their canonical versions. Since it allows us to analyze an infinite number of principals, canonical value is an important analysis element [15].

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

We are now ready to model the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake protocol in LySA. We have the protocol narration in Table 3. We extend the narration to distinguish between inputs and corresponding outputs and also make clear which checks must be performed [10]. Then we translate the narration into LySA by dividing the narration into two processes, one for each principal. Notice that the checks to be performed 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 section. The LySA model of the PKMv2 SA-TEK 3-Way Handshake is given in Table 6.

In PKMv2, a keyed MAC is used to verify the integrity of messages. The message is hashed along with the key and then encrypted with the MAC key. Since the hash functions are one way functions, they can be modelled by using a public name for the encryption key and with no corresponding key for decryption. Therefore, the message is encrypted by asymmetric encryption first. After that symmetric encryption is applied. More details about LySa implementation can be found on [17] and [18].

Table 6. PKMv2 LySa Model

$(\nu\, K)\,(\nu\, id)\,($
$!\ (\nu\, na)\,\langle id, na, \{\!|id, na|\!\}_{Hash}\}_K[\mathsf{at}\ a1\ \mathsf{dest}\ \{b1\}\,]\rangle.$
$\quad (na, id;\ xnb, xS, xmac).$
$\quad \mathsf{decrypt}\ xmac\ \mathsf{as}\ \{\!|na, id, xnb, xS|\!\}_{Hash};\ \}_K[\mathsf{at}\ a2\ \mathsf{orig}\ \{b2\}\,]\ \mathsf{in}$
$\quad (\nu\, T)\,\langle na, nb, id, T, \{\!|na, nb, id, T|\!\}_{Hash}\}_K[\mathsf{at}\ a3\ \mathsf{dest}\ \{b3\}\,]\rangle.0$
$|$
$!\ (id;\ yna, ymac).$
$\quad \mathsf{decrypt}\ ymac\ \mathsf{as}\ \{\!|id, yna|\!\}_{Hash};\ \}_K[\mathsf{at}\ b1\ \mathsf{orig}\ \{a1\}\,]\ \mathsf{in}$
$\quad (\nu\, nb)\,(\nu\, S)\,\langle yna, id, nb, S, \{\!|yna, id, nb, S|\!\}_{Hash}\}_K[\mathsf{at}\ b2\ \mathsf{dest}\ \{a2\}\,]\rangle.$
$\quad (na, nb, id;\ yT, ymac).$
$\quad \mathsf{decrypt}\ ymac\ \mathsf{as}\ \{\!|na, nb, id, yT|\!\}_{Hash};\ \}_K[\mathsf{at}\ b3\ \mathsf{orig}\ \{a3\}\,]\ \mathsf{in}\ 0$
$)$

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 main components of the analysis are:

The variable environment ρ, an over-approximation of the potential values of each variable that may be bound to.
The network component κ, an over-approximation of the set of messages that can be communicated over the network
The error component ψ, the set of error messages of the form (ℓ, ℓ') indicating that something encrypted at ℓ was unexpectedly decrypted at ℓ'.

The details of the analysis and the proofs of the soundness of the analysis can be found in [11, 12].

In practice, the protocols - especially the ones in wireless networks - are executed in medium with malicious attackers. In this study, LySa processes are analyzed in parallel with Dolev-Yao attacker [9] which can perform operations like sending/receiving messages and encryption/decryption same as a legitimate principal.

We have new canonical name and variables (see section 3 on page 6) 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 $\mathcal{C}$ 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 descriptions of the Dolev-Yao conditions are:

- The attacker initially has the knowledge of the canonical name $n_\bullet$ and all free names of the process P but he can improve his knowledge by eavesdropping all messages sent on the network.
- 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 (ℓ,$\ell_\bullet$) should be added to the error component ψ which means that something encrypted at ℓ was actually decrypted by the attacker at $\ell_\bullet$.
- 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 ($\ell_\bullet$,ℓ) 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.
- The attacker can send messages on the network using his knowledge and thus forge new communications.

This conditions enable the attacker to establish the attack scenarios including eavesdropping, modification and replay. The soundness of Dolev-Yao condition is proved in [10].

The flow of the analysis is shown in the Fig.2. First of all, we have a protocol narration as we had in section 2. Then we extend the narration and convert into LySA model, as we did in section 3. We also have our attacker model which is covered in this section. The LySA model is analyzed in parallel with the attacker model and is processed by the LySA-tool which implements the analysis. The results of the analysis are used to validate destination/origin authentication and confidentiality properties of the protocols. If no violation is detected, namely the error component ψ is empty, than it is guaranteed that the protocol satisfies the destination/origin authentication properties. Besides, the potential values that are learned by the attacker ($\rho(z_\bullet)$) helps us in validating the confidentiality properties. Since the analysis is an over-approximation there may occur false positives.

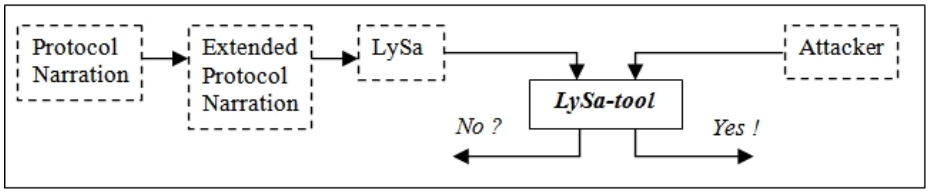

Fig. 2. The flow of the analysis

4.1 Scenarios

We shall analyze the PKMv2 SA-TEK 3-Way Handshake in scenarios with a number n of As, $A_1, \ldots, A_n$, and Bs, $B_1, \ldots, B_n$. As mentioned previously, A is the abbreviation for the base station and B for the mobile station.

We use the pair $(i, j))$ to refer to the instance of the protocol where A_i is communicating with the B_j. Thus we add the two indices i, j to all variables, constants and crypto-points of the model of the protocol in Table 6 and obtain the LySA code that allows the analysis to distinguish between the various instances. We then introduce the index 0 to refer to the attacker and the resulting analysis scenario takes the form

$$|_{i=1}^{n}||_{j=0}^{n} P_{A_{i,j}} \quad | \quad |_{i=0}^{n}|_{j=1}^{n} P_{B_{i,j}}$$

Here we use $P_{A_{i,j}}$ and $P_{B_{i,j}}$ to denote the processes needed at the A_i and the B_j principals, respectively, in order to perform a mutual handshake. Notice that the scenario describes that not only are all principals ready to interact with all other honest principals, but the attacker is also allowed to act as a legitimate principal. The analysis is carried out for $n = 2$ which models two groups of As and Bs.

4.2 Validating the Protocol

In order to improve the PKMv2 SA-TEK 3-Way Handshake, first we have to verify the base protocol which is given in Table 3. The result of our static analysis is: *no violations possible*. This means that the protocol is secure and the attacker could not violate the authentication properties. This result is similar to the work in [14] which is established using model checking using Murphi. Now, we can make our modifications convenient with our experiment logic.

5 Simplifying the Protocol

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.

We based our model on the simplified version of the IEEE 802.16 PKMv2 SA-TEK 3-Way Handshake . After that we developed our LySA model in section 3.1. We start with our base protocol model and try to simplify the model by removing components and analyzing with attacker to find flaws.

We made the experiments systematically, and the road map of the experiment can be seen in Fig. 3. 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.

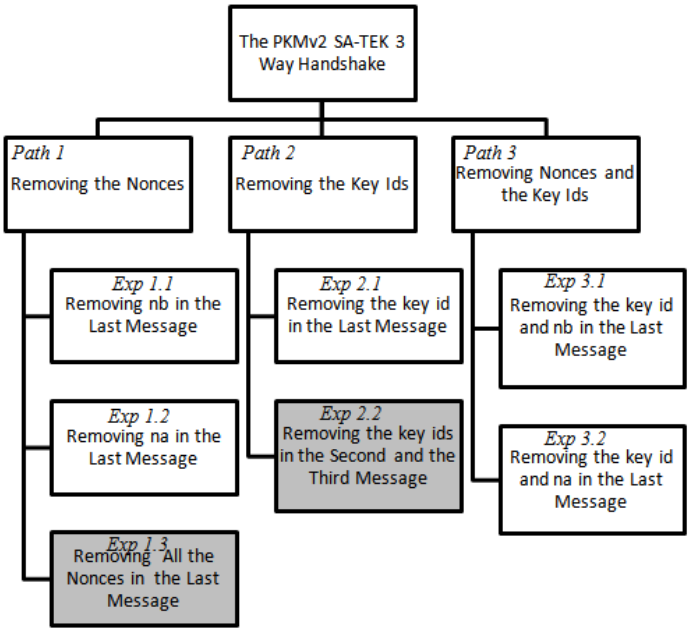

Fig. 3. Experiment Road Map

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. 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.

5.1 Experiment 1.1

This experiment is the first part of the first path, removing the nonces. We remove *nb* in the last message. The protocol narration of this modification is given in Table 7 in the appendix section. The result of the analysis is: *no violations possible*. This means that the protocol is still secure and the attacker still could not violate the authentication properties even though we did not 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.

5.2 Experiment 1.2

In this experiment, we remove the other nonce, *na* from the base protocol. The protocol narration of this modification is given in Table 8 in the appendix section. The result of the analysis is: *no violations possible*. This means that the protocol is still secure and the attacker still could not violate the authentication properties even though we did not 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.

5.3 Experiment 1.3

In the last part of the first path, we remove both nonces from the last message of the base protocol. The protocol narration of this modification is given in Table 9 in the appendix section. This time we find violation of authentication properties. The result is given as:
$\psi = (a1_{11}, b3_{11}), (a3_{11}, b1_{11}), (a1_{21}, b3_{21}), (a3_{21}, b1_{21}), (a1_{12}, b3_{12}),$
$(a3_{12}, b1_{12}), (a1_{22}, b3_{22}), (a3_{22}, b1_{22})$
Sample trace for $(a1_{11}, b3_{11})$ can be shown as:

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

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 could not decrypt in a reply attack. In the third message, he replayed the MAC's, namely he used the MAC of message one in message-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.

5.4 Experiment 2.1

This experiment is the first part of the second path, removing the key ids. We start with removing the *id* only in the last message. The protocol narration of this modification is given in Table 10 in the appendix section. The result of the analysis is: *no violations possible*. This means that the protocol is still secure and the attacker still could not violate the authentication properties even though we did not use the key id in the last message.

5.5 Experiment 2.2

In this experiment we remove the id fields in both the last and the second message. The protocol narration of this modification is given in Table 11 in the appendix section. Now we have found violation of authentication properties. The result is given as:
$\psi = (b2_{11}, b3_{11}), (a3_{11}, a2_{11}), (b2_{21}, b3_{21}), (a3_{21}, a2_{21}), (b2_{12}, b3_{12}), (a3_{12}, a2_{12}),$
$(b2_{22}, b3_{22}), (a3_{22}, a2_{22}), (a3_{10}, a2_{10}), (a3_{20}, a2_{20}), (b2_{02}, b3_{02}), (b2_{01}, b3_{01})$
We found traces for specific types of violation. Sample trace for $(b2_{11}, b3_{11})$ can be shown as:

1. $\mathbf{A}_1 \rightarrow \mathbf{B}_1$ **:** id_{11}, na_{11}, MAC{ id_{11}, na_{11}}$_{K_{11}}$
2. $\mathbf{B}_1 \rightarrow \mathbf{A}_1$ **:** na_{11}, nb_{11}, S_{11}, MAC{ na_{11}, nb_{11}, S_{11}}$_{K_{11}}$
2'. $\mathbf{B}_1 \rightarrow \mathbf{M}(\mathbf{A}_1)$ **:** na_{11}, nb_{11}, S_{11}, MAC{ na_{11}, nb_{11}, S_{11}}$_{K_{11}}$
3. $\mathbf{M}(\mathbf{A}_1) \rightarrow \mathbf{B}_1$ **:** na_{11}, nb_{11}, S_{11}, MAC{ na_{11}, nb_{11}, S_{11}}$_{K_{11}}$

This result shows that we cannot remove both *id*s in the protocol.

5.6 Experiment 3.1

Now we construct the third path of the experiments by taking the successfully validated experiments of the first two path. Therefore, this path is called removing the nonces and the key ids. In this first experiment of this path, we combine the experiment 1.1 and 2.1 so we remove the key *id* and *nb* in the last message. The protocol narration of this modification is given in Table 12 in the appendix section. The result of the analysis is: *no violations possible*. This means that the protocol is still secure and the attacker still could not violate the authentication properties even though we did not 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.

5.7 Experiment 3.2

In this experiment of this path, we combine the experiment 1.2 and 2.1 so we remove the key id and *nb* in the last message. The protocol narration of this modification is given in Table 13 in the appendix section. The result of the analysis is: *no violations possible*. This means that the protocol is still secure and the attacker still could not violate the authentication properties even though we did not use neither the key id nor nb 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 MS), and also key ids. Now we have seen the limits of the protocol and removed the redundant fields.

5.8 Further Experiments

The experiments above present enough evidence to support our proposal but we went further to see if we could fix the flaws that appeared when we removed fields from the base protocol. The problems occur in the MAC part, and we claim that this can be fixed by adding sequence numbers inside the MACs. Therefore, we applied sequence number revision to the experiments with erroneous results which can be seen as shaded in Fig. 3. The results have empty error components which mean that we have fixed the flaws. After that we took another way and decided to test the affect of the order of the fields in the messages. As can be seen in Fig. 4, we just swapped the appropriate fields in the last message in order to get the same order with the second message. The result is very interesting, now we have violations in the base code. Besides, we can easily find traces for those violations which means that the results are not false positive, but real flaws. We fixed this situation with the sequence numbers and had no violations afterwards.

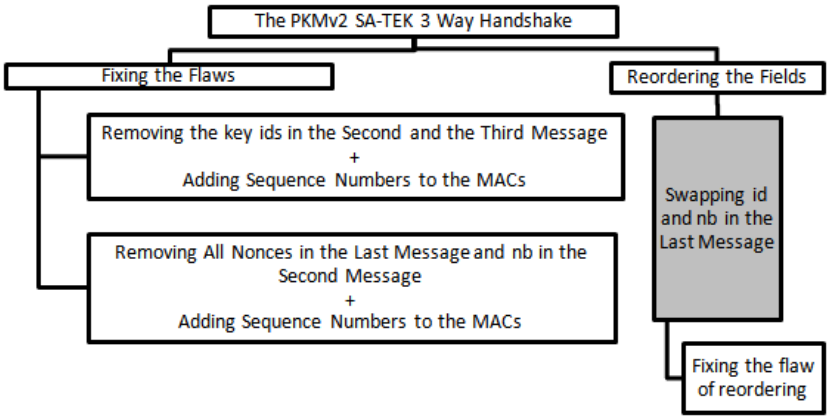

Fig. 4. Further Experiments

6 Conclusion

The contribution of this paper is two-fold. First, we establish the security of the Privacy and Key Management protocol PKMv2 of the latest version of the WiMAX standard, the IEEE 802.16e-2005. Second, we show how the protocol can be simplified without violating the security properties.

The PKMv2 SA-TEK 3-Way Handshake protocol was developed as a response to the identification of various flaws in the earlier version. Several security elements were added, not only to fix known flaws, but also to cater for other

possible flaws. Our analysis shows that the resulting protocol is indeed secure, but also that it was over-secured by introducing redundant fields.

Our approach is based on the LySa calculus and the corresponding analysis tool, the LySaTool. The LySa framework provides an intuitive protocol description and an automated verification of the security properties authentication and confidentiality. Therefore LySa is a suitable choice for investigating protocols such as the classical security protocols [15], large-scale systems [18] and even voting protocols [12].

As we mentioned in the static analysis section, an error found by analysis does not always imply that the protocol actually has a flaw. A successful run on the analysis on the other hand, guarantees that the protocol does not violate the security properties. Thus, our experiments that yielded no violations are versions of the protocol that are guaranteed secure, whereas the experiments that returned violations would require further investigation to determine if these actually corresponded to an attack.

The success of our approach shows that is a viable method for iteratively simplifying a secure protocol. Only alterations that does not introduce possible attacks are kept and tested in combinations with other safe alterations. During development, we carried out many experiments, but only the ones we found of most interest were included in the paper. The result is a simplified version of PKMv2 SA-TEK 3-Way Handshake protocol, guaranteed to be secure, but without various redundant fields.

References

1. Johnston, D., Walker, J.: Overview of IEEE 802.16 Security. IEEE Security & Privacy Magazine Vol. 2, Issue: 3, (2004) 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
3. Lowe, G.: 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 (1996) 147–166
4. Lowe, G.: An attack on the needham-schroeder public-key authentication protocol. Information Processing Letters **56(3)** (1995) 131–133
5. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Communications of the ACM. **21(12)** (1978) 993–999
6. Mitchell, J. C., Mitchell, M., Stern, U.: Automated Analysis of Cryptographic Protocols Using Murphi. IEEE Symposium on Security and Privacy. (1997) 141–151
7. Shmatikov, V., Stern, U.: Efficient finite-state analysis for large security protocols. Communications of the ACM. (1998) 106–115
8. Corin, R., Saptawijaya, A., Etalle, S.: A logic for constraint-based security protocol analysis. IEEE Symposium on Security and Privacy. (2006) 155–168
9. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory. **29(12)** (1983) 198–208

10. Bodei, C., Buchholtz, M., Degano, P., Nielson, H.R., Nielson, F.: Static Validation of Security Protocols. Journal of Computer Security. (2004) 347–390
11. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Nielson, H.R.: Automatic validation of protocol narration. Proceedings of the 16th Computer Security Foundations Workshop. (2003) 126–140
12. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer-Verlag
13. Abadi, M., Gordon, A. D.: A calculus for cryptographic protocols: The spi calculus. Information and Computation. **148(1)** (1999) 1–70
14. Datta, A., He, C., Mitchell, J. C., Roy, A., Sundararajan, M.: 802.16e Notes. IETF Liasons. (2005)
15. Buchholtz, M., Nielson, H.R., Nielson, F.: A calculus for control flow analysis of security protocols. International Journal on Information Security. **2(3-4)** (2004) 145–167
16. Milner, R.: Communicating and mobile systems: the pi-calculus. Cambridge University Press, fifth edition.
17. Nielsen, C.R., Andersen, E.H., Nielson, H.R.: Static Validation of a Voting Protocol. Nordic Journal of Computing . (2006) 98–116
18. Hansen, S.M., Skriver, J., Nielson, H.R.: Using Static Analysis to Validate the SAML Single Sign-On Protocol. Proceedings of the 2005 Workshop on Issues in the theory of Security . (2005) 27–40

A Protocol Narrations

Table 7. PKMv2 without nb in message 3

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

Table 8. PKMv2 without na in message 3

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

Table 9. PKMv2 without na and nb in message 3

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

Table 10. PKMv2 without id in message 3

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

Table 11. PKMv2 without ids in message 2 and 3

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

Table 12. PKMv2 without id and nb in message 3

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

Table 13. PKMv2 without id and na in message 3

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