Formal Verification of Hash-based Authentication Protocol in WiMAX Networks

Beth N. Komu[1], Mjumo Mzyece[2] and Karim Djouani[3]
Department of Electrical Engineering/French South African Institute of Technology
Tshwane University of Technology, Private Bag X680, Pretoria 0001, South Africa
Tel: +27 (0) 12 382 4191, Fax: +27 (0) 12 382 5294
Email: {bethkomu@gmail.com[1], mzyecem@tut.ac.za[2], djouanik@tut.ac.za[3]}

Abstract-**The Initial Network Entry procedure is the first stage in establishing a connection in an IEEE 802.16 (WiMAX) network. The process involves the transmission of unencrypted management messages, which constitutes a major security flaw that is exploited by the Man-in-the-Middle (MITM) attack. This security defect necessitates the implementation of appropriate security protocols. Research has shown that developing secure protocols is a difficult task as is evident from the presence of flaws in published protocols such as the Needham-Schroeder public key authentication protocol that had a bug that remained undetected for 17 years. With the use of formal verification techniques, bugs can be discovered very early in the design of the system and automatically validated, thereby increasing confidence in their use. In this paper, we analyse a security protocol proposed to mitigate the MITM attack at the initial network entry point in WiMAX referred to as Secure Initial Network Entry Protocol (SINEP), and model the protocol and an intruder process with MITM capabilities in Process/Protocol Meta-language (PROMELA) formalism. We then use Linear Temporal Logic (LTL) to define the attributes the protocol should satisfy and carry out verification by use of the SPIN model checker. Using our intruder model, we conclude that the protocol fulfils its confidentiality objective by concealing the most valuable messages in the protocol run.**

Index Terms—Formal Modelling, Formal Verification, Initial Network Entry, LTL, Man-in-the-Middle, SPIN, WiMAX

I. INTRODUCTION

The Initial Network Entry procedure in the Worldwide Interoperability for Microwave Access (WiMAX) network consists of security-sensitive unprotected Medium Access Control (MAC) management messages being exchanged between the Subscriber Station (SS) or Mobile Station (MS) and the Base Station (BS). These messages are exchanged for purposes of ranging, capabilities negotiations, authentication and key exchange and registration. Unfortunately, the Advanced Encryption Standard (AES) implemented by Mobile WiMAX only protects the data messages after the Initial Network Entry procedure, leaving the MAC management messages to be sent in the clear and thus exposing them to the MITM attack [1].

The MITM attack strategically positions itself between the SS/MS and the BS and passively listens to an insecure channel of communication, in this case, the Initial Network Entry procedure. It then creates detailed profiles of the victim SS inclusive of its security settings and associations with the serving BS, imitates the legitimate stations and then modifies the management messages exposing the network to other destructive attacks like replay attacks, masquerade attacks and denial-of-service (DoS) attacks. Eventually, the MITM attack manages to fool the legitimate SS/MS and the BS into operating as if they are still communicating with each other although the intruder controls the communication process.

The vulnerability at the Initial Network Entry procedure in WiMAX makes the implementation of a security protocol vital. Security protocols, also referred to as cryptographic protocols, aim at providing some security-related objective in potentially hostile environments such as the Internet and in wireless networks like WiMAX. Many security protocols have been found to be error-prone especially because of the difficulty of foreseeing all the possible attacks. Consequently, a lot of research is being conducted on the use of formal verification techniques to discover subtle attacks in protocols that may otherwise be difficult to uncover using traditional methods based on human inspection and testing.

Model checking [2] is an automatic formal approach for verifying finite state systems and has so far been successfully used to check for correctness in communication protocols, software, and hardware comprising complex sequential circuit designs. This, in turn, significantly increases the level of confidence in employing such systems. Model checking is gaining popularity due to its automatic system checking process, speed of verifying systems and high efficiency.

Most research work done on the security of Mobile WiMAX did not establish any security vulnerabilities due to the extra security capabilities integrated in Privacy and Key Management protocol version 2 (PKMv2) and thus assumed that the Initial Network Entry point is secure. A few authors have analysed and unmasked security holes in the Initial Network Entry process in WiMAX and subsequently proposed various security protocols aimed at mitigating the MITM attack. These protocols are believed to be correct and resistant to malicious manipulation. In this paper, we focus on one of the few security protocols proposed to preclude

the existing vulnerability at the Initial Network Entry point in WiMAX and perform protocol verification using the SPIN model checker to check for existing flaws in the protocol.

The rest of the paper is structured as follows. Section II gives an overview of the proposed protocol to thwart the MITM attack. In Section III, an overview of related work on model checking inclusive of that applying the SPIN model checker is presented. Section IV presents a PROMELA-based model of the proposed protocol, its properties specifications, the modelled intruder process and the respective verification results. Finally, Section V concludes the paper.

II. SINEP PROTOCOL

Tao Han *et al.* in [3] propose a security model called SINEP that enhances the security level during the Initial network entry procedure in WiMAX. The protocol is based on the Diffie-Hellman (DH) key exchange protocol but introduces a mutual entity authentication algorithm using hash functions so as to mitigate the MITM attack present in the implementation of the basic version of the DH key exchange protocol. Cryptographic hash functions provide integrity or authenticity to data by mapping the data to a short bit string known as a hash value [4].

The SINEP protocol incorporates a challenge-response entity authentication mechanism with the exchange of stations' public keys, illustrated in Figure 1. This model assumes that every SS in the network has a unique International Subscriber Station Identity (ISSI) which it uses to yield a Temporary Subscriber Station Identity (TSSI) used as its identity. In turn, the SS uses its TSSI to generate the hash value H (TSSI), which is assumed to be known by the BS. The model uses H (TSSI) in the protocol run rather than TSSI as the input parameter of the hash authentication function to avoid identity theft of the legitimate BS by an adversary.

An SS willing to communicate with a BS in the WiMAX network sends a claim to the BS, which in turn sends a challenge (Nb) to the claiming SS. The SS computes the result corresponding to the challenge received and sends it together with its public key (PKSS), its own challenge (Na) and prime numbers (r, s) necessary for the implementation of the DH protocol, to the serving BS. The result generated is a hash-based cascade of the Nb, PKSS and the hashed identity of the SS, H (TSSI), H (H (TSSI) ||Nb||PKSS)). Upon receipt, the BS computes the hash value using the same inputs and compares it with the SS's hash value. If the values match, the BS sends an acceptance token as proof of authentication, its own public key (PKBS) and the result to the challenge (Na), to the SS. In case of a mismatch, the BS disconnects from the respective SS. Likewise, the result generated by the BS is a hash-based cascade of Na, PKBS and the hash value, H (TSSI), H (H (TSSI||Na||PKBS)).

On receiving the result message from the BS, the SS verifies the hash value and if a match is found, it sends back an acceptance token to the BS as proof of successful authentication. Finally, successful mutual entity authentication is achieved and a shared encryption key is generated and used to encrypt the vulnerable management messages. Similarly, if there is no match between the SS's hash value and the one sent by the BS, the SS disconnects from that particular BS.

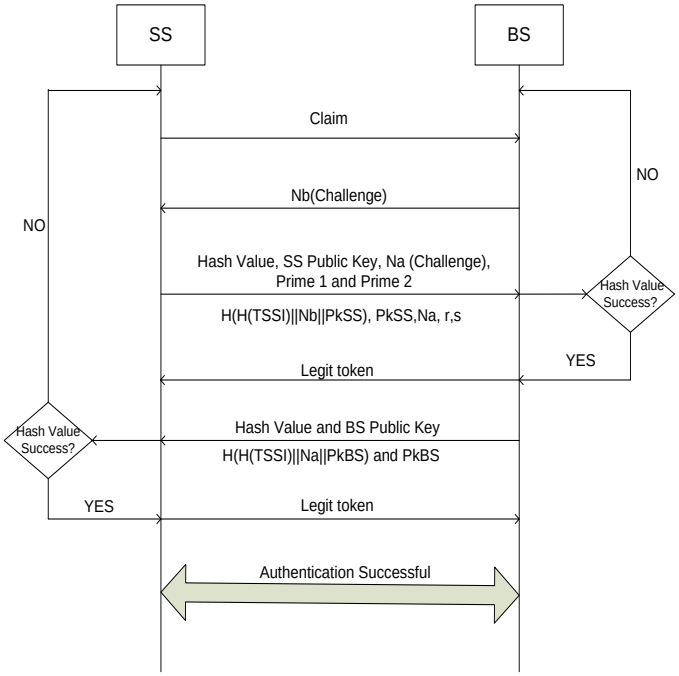

Figure 1: Entity authentication of principals using hash functions in the SINEP protocol

In this model, it is assumed that it is only the legitimate BS and the legitimate SS that have knowledge of the H (TSSI) and hence the cryptographic hash function used to compute the hash values sent in the protocol run. Therefore, an attacker in the network is not able to bring forth the TSSI generated by the SS and the correct result to the given challenges and is thus isolated as an intruder to the network.

III. RELATED WORK

A. Model Checking

Different tools for model checking have been successfully implemented to expose vulnerabilities that may still exist in security protocols implemented especially in networks like WiMAX. The authors of [5] applied the Scyther tool to expose vulnerabilities in the PKMv1 and PKMv2 authentication protocols implemented in WiMAX. They succeeded in discovering a breach in information confidentiality in both protocols. Similarly, the authors of [6] proposed the use of TLA+ to specify the properties of protocols in the authentication and ranging processes in WiMAX and used TLC as the model checking tool to check for denial-of-service (DoS) vulnerabilities. They identified some possible DoS attacks at the initial ranging process but failed to detect a DoS flaw in the PKMv2 authentication protocol using their attacker model. Additionally, the authors of [7] successfully implemented the Strand space verification technique to uncover the MITM attack in cryptographic authentication protocols.

B. *PROMELA and the SPIN Model Checker*

SPIN [8], one of the most powerful general-purpose model checkers, has been extensively used to identify defects in control systems, software systems and security protocols. PROMELA is the description language for SPIN used to implement concurrently executing processes in a protocol. In cryptographic protocols, the protocol participants and the behaviour of the intruder process are formalized in PROMELA, while the properties that the protocol should satisfy are specified in Linear Temporal Logic (LTL). Consequently, the protocol is executed in the presence of the modelled intruder process and in case a violation is detected, a counter-example is generated, aiding in the diagnosis of the error and in the improvement of the overall protocol. In [9] and [10], the authors implemented, analysed and verified the Needham-Schroeder public key authentication protocol using the SPIN model checker and unearthed a Lowe's attack that compromised the authentication and secrecy provided by the protocol. Likewise, the authors of [11] extended the approach used in [9] to analyse and check for correctness in the Helsinki protocol and were successful in discovering authentication vulnerabilities in the protocol.

IV. METHODOLOGY AND RESULTS

The SPIN model checker tool was employed to check for vulnerabilities in the SINEP protocol and the iSPIN graphical user interface used for visualization of simulation and verification runs.

A. *The PROMELA Equivalent of the SINEP Protocol*

The SS and the BS in the proposed protocol are first implemented as processes in PROMELA and their protocol sessions and respective assumptions modelled without interference from an intruder station as illustrated in Figure 2.

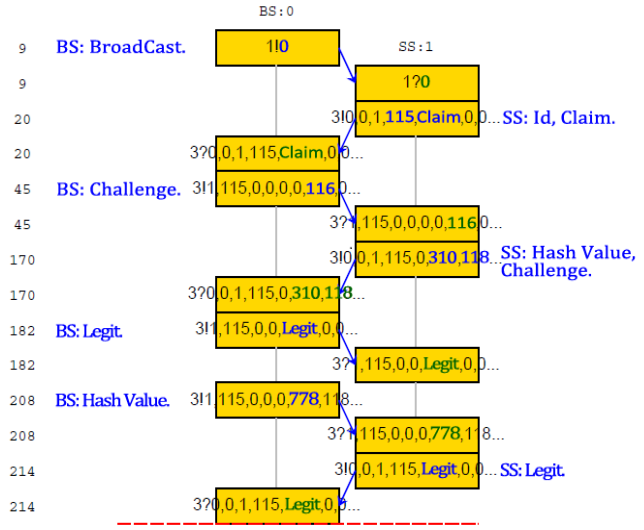

Figure 2: Entity authentication of principals in the SINEP protocol in PROMELA

In the above PROMELA model, the TSSI, which is the identity of the SS as specified in the SINEP protocol, is randomly generated and in this case set to "115". On the other hand, the identity of the BS is represented by the process identifier "0". The protocol run in the PROMELA model in Figure 2 proceeds successively as described in Figure 1. The cryptographic hash function implemented herein is based on modular squaring, of the nature $H = X^2 \bmod n$, where X is either the TSSI or the computed result, while n is a product of the primes (r, s). Moreover, it is assumed that the cryptographic hash function is only known by both the SS and the BS. The respective challenges in the protocol run, both the SS's and BS's public keys and the prime numbers, are assigned different values in each protocol run by invoking a random function. The Legit token in the respective principals' processes is an indicator that successful authentication has been achieved. The frame transmitted in the protocol assumes the structure in Figure 3.

Destination		**Source**		**Payload** (Message type and Integers)
MAC Address	Identity	MAC Address	Identity	

Figure 3: SINEP's Protocol Frame Structure

B. *Defining Correctness Claims*

LTL, first introduced by Pnueli in [12], is employed to describe the attributes the SINEP protocol should satisfy. An LTL formula is composed of both temporal operators and propositional logic operators otherwise known as Boolean connectives. Examples of propositional logic operators and their respective meanings implemented in SPIN include: (!–NOT), (&&-AND), (||-OR), (- > -Implies) and (<-> -equivalent). On the other hand, examples of temporal operators supported by SPIN include: ([ ]-Always), (<>-Eventually), (O-Next) and (U-Until). The SPIN model checker automatically translates stated LTL formula(s) into PROMELA Never Claims based on the algorithm described in [13]. Never claims is the SPIN terminology for Buchi Automata used to specify model behaviour that should never occur and are represented as the negation of the LTL formula specified for the model. The LTL formulas representing the properties that should hold in the SINEP protocol are represented as inline specifications within the PROMELA model as follows.

1) *Authentication Checking*

The fact that the SS and the BS correctly authenticate each other in the challenge-response process of the model in Figure 1 is expressed with the following predicates: a BS commits to a session with the SS if and only if the SS is correctly authenticated by the BS, expressed by the variable *BSCommit*. For the SS to be authenticated, first, it must have accepted to share a protocol instance with the BS by sending a claim message, defined by the *SSRunning* variable in the implementation of the protocol and the computed hash value as the response to the BS challenge must be correct. A similar predicate expresses the reciprocal property, that a BS was correctly authenticated by the SS, expressed by the variable *SSCommit*. For the BS to be authenticated it must have received a challenge from the SS, defined by the *BSRunning* variable in the implementation of the protocol and the computed hash value corresponding to the SS challenge must be correct. To check for authentication of SS

to BS and vice versa, the precedence property using LTL is expressed as:

[] ((! SSCommit) U (! BSRunning) U (! BSCommit) U (SSRunning)) *(1)*

2) Secrecy Checking

The confidentiality of the result generated from the respective challenges in the protocol run is validated by checking that the original message, which is the result, cannot be derived by the intruder from the implementation of the intercepted hash value and the hash function. Therefore, two scenarios are described in the intruder process. In the first scenario, it is assumed that the intruder process is aware of both the H (TSSI) and the hash function (Hfunc), since the authors of [3] are not clear on how the BS gets to know H (TTSI). Thereby, the respective Boolean variables, namely *KHTSSI* and *KHfunc*, are initialized to *true* within the intruder process and the variables *KSSResult* and *KBSResult* representing knowledge by the intruder of the SS result and the BS result respectively, are initialized to *false*. *KSSResult* and *KBSResult* become *true* once the intruder manages to revert the hash function using the hash value it intercepts. In the second scenario, considered by the authors in [3], it is assumed that the intruder process neither knows the H (TSSI) nor the Hfunc, therefore setting both the *KHTSSI* and *KHfunc* variables within the intruder process to *false*. Similarly, the *KSSResult* and *KBSResult* variables are initialized to *false*. In both situations, to verify the confidentiality of the respective results computed, the following LTL formula is applied:

[] ((! KSSResult) && (! KBSResult)) *(2)*

C. The PROMELA Model with the Intruder Process Interference

To introduce the intruder process with MITM capabilities to the already modelled protocol, the general concept used in [14] was adopted taking into account three assumptions. It is assumed that the intruder station is strategically positioned between legitimate communicating stations in the WiMAX network, that the intruder station has two wireless network cards and that the intruder already knows the general flow of the protocol. This process is illustrated in Figure 4.

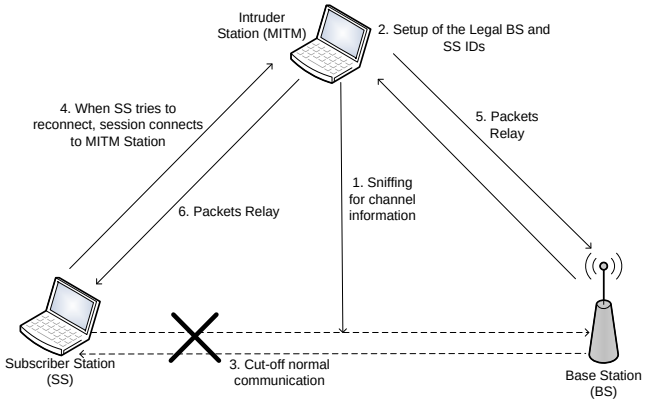

Figure 4: The MITM attack intrusion into the network

By first listening to an already established connection between communicating parties, the intruder process intercepts the frames being transmitted in the execution of the protocol, retrieves the legitimate stations' identities and maps them to its MAC address, increasing its initial knowledge necessary to launch attacks at step 2. After successful mapping, the intruder process attempts to disconnect or break the connection between the SS and BS, forcing the SS to go back to the claiming stage of Figure 1. The intruder takes advantage of this opportunity and poses as the legitimate BS, redirecting data traffic to itself from the SS in step 4. Similarly, to the BS, the intruder poses as the legitimate SS, forwarding traffic to and from the BS in step 5. Eventually, the intruder process manages to control the communication process unobserved by the SS and the BS, and thus a MITM attack is launched.

The modelled protocol with the intruder process is illustrated in Figure 5.

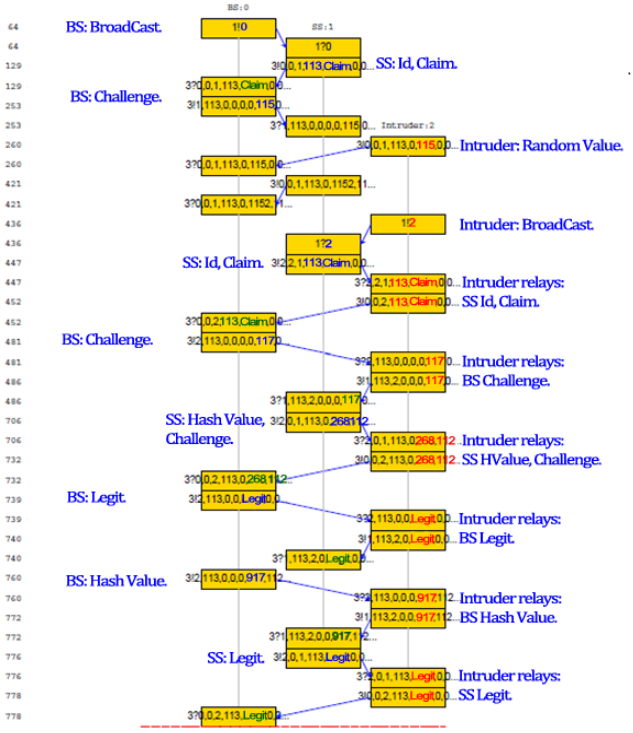

Figure 5: Protocol run instance where the Intruder process succeeds in penetrating the network

At step 260 in the above model, the intruder process whose identity is "2" sends a message with a random number "115" as an attempt to disrupt the protocol run between the SS and the BS. This random number is taken by the BS as the response (hash value) corresponding to the challenge it had sent earlier at Step 253. This message from the intruder process has the identity (TSSI) of the SS and arrives at the BS prior to the hash value from the legitimate SS. As a result, the BS verifies it and finds out it is incorrect, consequently disconnecting from the legitimate SS. At this point, the intruder process has successfully broken the connection between the SS and the BS. At Step 447, the SS selects the intruder process posing as a BS as it tries to reconnect and sends a claim message. In turn, the intruder process forwards the claim to the legitimate BS with the identity of the SS. The BS then sends a challenge to the intruder posing as the SS at step 481, which it forwards

to the SS posing as the BS. This process continues until the protocol ends with all the messages passing through the intruder process before being forwarded to the respective recipients. The BS and the SS authenticate each other oblivious of the fact that the protocol was controlled by another process the entire time.

D. Formal Verification Results

Formal Verification of the SINEP protocol is performed on a laptop with a 2.0 GHz Intel Pentium Processor and 1.99 GB of RAM. The protocol is first verified with the stated LTL properties using an exhaustive search that includes a Depth First Search (DFS) algorithm of the state space and the use of the partial order reduction strategy aimed at reducing the number of states that need to be visited so as to verify the stated properties. During the verification process, SPIN executes the generated never claim from the specified LTL formulas in Section III with the given PROMELA model looking for a match between the claim and the model. A match can either correspond to the detection of an acceptance cycle within the never claim or termination of the never claim, signifying a violation of the LTL property and leading to the generation of a counter-example for the LTL formula. The results of the verification process are displayed below.

1) Authentication Property:

```
$ pan -a
warning: for p.o. reduction to be valid the never claim must be
stutter-invariant
(never claims generated from LTL formulae are stutter-invariant)
pan:1: acceptance cycle (at depth 464)
pan: wrote ReMT2ndProtocol.pml.trail

(Spin Version 6.0.1 -- 16 December 2010)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             + (AuthProperty)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 956 byte, depth reached 540, errors: 1
    13852 states, stored (27522 visited)
     5899 states, matched
    33421 transitions (= visited+matched)
      189 atomic steps
hash conflicts:         154 (resolved)

Stats on memory usage (in Megabytes):
    10.704       total actual memory usage

pan: elapsed time 0.375 seconds
```

Figure 6: Verification Output with LTL Authentication Property

By executing an exhaustive search for the verification of the model with the LTL authentication property stated in Section III, an acceptance cycle was detected as seen in Figure 6, meaning that there exists a state in the model that is visited infinitely often, resulting in a violation of the stated authentication property. On replaying the error trail file generated by SPIN, a cycle exists when the intruder process is attempting to break the normal protocol run between the SS and BS by sending the wrong hash value. Nonetheless, eventually, the modelled intruder process was able to break the normal protocol run between the BS and the SS by directing messages from the SS and BS to itself and relaying them to the intended recipients as seen in Figure 5. By doing this, the intruder process is successful in penetrating the network consisting of the SS and the BS and acts as a "middle man".

2) Secrecy Property

First Scenario: KH (TSSI) =1 and KHfunc=1

```
$ pan -a
warning: for p.o. reduction to be valid the never claim must be
stutter-invariant
(never claims generated from LTL formulae are stutter-invariant)
pan:1: end state in claim reached (at depth 1082)
pan: wrote ReMT2ndProtocol.pml.trail

(Spin Version 6.0.1 -- 16 December 2010)
Warning: Search not completed
        + Partial Order Reduction

Bit statespace search for:
        never claim             + (SecrecyProperty)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 940 byte, depth reached 1082, errors: 1
  2025470 states, stored
   337604 states, matched
  2363074 transitions (= stored+matched)
      552 atomic steps

hash factor: 4.14156 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
     2.184       total actual memory usage

pan: elapsed time 30.5 seconds
```

Figure 7: Verification Output with LTL Secrecy Property with the Intruder knowing H (TSSI) and Hfunc.

Second Scenario: KH (TSSI) =0 and KHfunc=0.

```
$ pan -a
warning: for p.o. reduction to be valid the never claim must be
stutter-invariant
(never claims generated from LTL formulae are stutter-invariant)

(Spin Version 6.0.1 -- 16 December 2010)
        + Partial Order Reduction

Bit statespace search for:
        never claim             + (SecrecyProperty)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 940 byte, depth reached 1313, errors: 0
  2984506 states, stored
  5252917 states, matched
  8237423 transitions (= stored+matched)
 27790344 atomic steps

hash factor: 2.81072 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
     2.282       total actual memory usage

pan: elapsed time 88 seconds
```

Figure 8: Verification Output with LTL Secrecy Property without the Intruder knowing H (TSSI) and Hfunc

Due to the size of the model, performing an exhaustive search to verify the secrecy property in both scenarios was impracticable due to system memory limitations. Therefore, the amount of memory to be used was increased from 1024 MB to 2048 MB using the –DMEMLIM directive but still, the search could not be completed due to lack of memory. We resorted to the employment of memory optimization algorithms supported by SPIN to reduce the amount of memory needed to store each state [8]. All the lossless compression techniques including Hash Compact, Collapse Compression and the Minimized Automata technique decreased the memory used in the verification process at the expense of execution time, but failed to complete the search as expected. The lossy compression technique supported by SPIN known as BitState Hashing also referred to as Supertrace, was the method of last resort since a full verification and the lossless compression techniques turned out to be infeasible because of memory limitations.

Using BitState hashing, an error corresponding to the secrecy property stated in Section III was discovered in the first scenario as seen in Figure 7. An end state of the claim was reached meaning that there was a match between the

model and the generated never claim, resulting in a violation of the secrecy property. On replaying the error trail file generated by SPIN, the error is traced back to the fact that, due to the nature of the hash function implemented, there are infinite possibilities to the result used as the input to the hash function and thus, eventually after many trials, there is a possibility that the Intruder process may discover the result to a challenge sent within the protocol run, given that it already knows the hash function and the hash value.

On the other hand, an error was not found in the second scenario of verifying the model with the secrecy property as seen in Figure 8. This is due to the fact that, without knowledge of the H (TSSI) and the Hfunc as specified by the protocol, the intruder process is unsuccessful in decrypting the hash value it acquires from both the SS and the BS even after successively penetrating the network as seen in Figure 5.

Although in some instances our modelled intruder process was able to control the flow of the protocol between the SS and the BS during simulation, based on the verification results achieved and considering the assumptions taken in [3], the SINEP protocol has proved to be resistant to content manipulations by the modelled MITM attack. Therefore, by implementing the SINEP protocol at the initial network entry point in WiMAX, the chances of the security capability of the network being decreased by an intruder will be eliminated.

V. CONCLUSION

In this paper, the SINEP protocol was modelled using PROMELA and the properties that the protocol must satisfy such as authentication and secrecy specified in LTL. The intruder process with MITM capabilities was modelled and then introduced to the specified protocol for behaviour analysis. Eventually, verification of the protocol was performed using SPIN. The modelled intruder process succeeds in eavesdropping on the communication process between the SS and the BS, impersonating both the SS and BS identities and penetrating the network undetected. However, it fails in retrieving valuable messages in the protocol like the respective results, taking into consideration the assumptions taken by the authors of [3] in the second scenario of the secrecy property. While it cannot be concluded that the SINEP protocol is invulnerable to the MITM attack, it can be safely claimed that it is resistant to any attempt to expose valuable messages in the protocol run using our attacker capability model.

Beth Komu received her bachelor's degree in Computer Science and Engineering 2nd class upper division from Maseno University, Kenya and is currently studying towards her Master of Technology degree at Tshwane University of Technology. Her research interests include Wireless Communications, WiMAX Networks, Security in Wireless Networks, Formal Methods and Radio Network Planning.

REFERENCES

[1] M. Barbeau, "WiMAX/802.16 Threat Analysis," in *Proc. 1st ACM Int. Workshop Quality of service and Security in Wireless and Mobile Networks*, Montreal, Canada, 2005, pp. 8-15.

[2] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*.: MIT Press, 1999.

[3] T. Han, N. Zhang, K. Liu, B. Tang, and Y. Liu, "Analysis of Mobile Wimax Security: Vulnerabilities and Solutions," in *5th IEEE Int. Conf. Mobile Ad hoc and Sensor Syst. (Mass)*, Atlanta, Georgia, 2008, pp. 828-833.

[4] A. Menezes, P. V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*.: CRC Press, Inc, 1996.

[5] A. M. Taha, A. T. Abdel-Hamid, and S. Tahar, "Formal Verification of IEEE 802.16 Security Sublayer Using Scyther Tool," in *IEEE Int. Conf. Network and Service Security (N2S '09)*, Paris, France, 2009, pp. 1-5.

[6] P. Narayana, R. Chen, Y. Zhao, Y. Chen, Z. Fu, and H. Zhou, "Automatic Vulnerability Checking of IEEE 802.16 WiMAX Protocols through TLA+," in *Proc. 2nd IEEE Workshop Secure Network Protocols*, November, 2006, pp. 44-49.

[7] R. K. Guha., Z. Furqan, and S. Muhammad, "Discovering Man-In-The-Middle attacks in authentication protocols," in *MILCOM 2007*, Orlando, FL, October 29-31, 2007.

[8] G. J. Holzmann, *The SPIN Model Checker, The Primer and Reference Manual*.: Addison Wesley, 2003.

[9] P. Maggi and R. Sisto, "Using SPIN to Verify Security Properties of Cryptographic Protocols," in *Proc. 9th Int. SPIN Workshop Model Checking of Software*, Grenoble, France, 2002, pp. 11-13.

[10] A. S. Khan, M. Mukund, and S. P. Suresh. Generic Verification of Security Protocols. [Online]. http://spinroot.com/spin/Workshops/ws05/025_paper.pdf

[11] M. Xiao and J. Li, "The Modelling Analysis of Cryptographic Protocols Using Promela," in *Proc. 6th World Congr. Intelligent Control and Automation (WCICA)*, Dalian, China, 2006, pp. 4321-4325.

[12] A. Pnueli, "The Temporal Logic of Programs," in *Proc. 18th IEEE Symp. Found. Comput. Sci.*, Providence, Rhode Island, 1977, pp. 46-57.

[13] R. Gerth, D. Peled, M. Y Vardi, and P. Wolper, "Simple On-the-fly Automatic Verification of Linear Temporal Logic," in *Proc. 15th IFIP WG6.1 Int. Symp. Protocol Specification, Testing and Verification (PSTV)*, Warsaw, Poland, 1995, pp. 3-18.

[14] H. Hwang, G. Jung, K. Sohn, and S. Park, "A Study on MITM (Man in the Middle) Vulnerability in Wireless Network Using 802.1x and EAP," in *Int. Conf. Inform. Sci. and Security*, Seoul, Korea, 2008, pp. 164-170.