Automatic Vulnerability Checking of IEEE 802.16 WiMAX Protocols through TLA+

Prasad Narayana, Ruiming Chen, Yao Zhao, Yan Chen, Zhi (Judy) Fu†, and Hai Zhou
Northwestern University, Evanston IL, USA
†Motorola Labs, Schaumburg IL, USA

***Abstract*— Vulnerability analysis is indispensably the first step towards securing a network protocol, but currently remains mostly a best effort manual process with no completeness guarantee. Formal methods are proposed for vulnerability analysis and most existing work focus on security properties such as perfect forwarding secrecy and correctness of authentication. However, it remains unclear how to apply these methods to analyze more subtle vulnerabilities such as denial-of-service (DoS) attacks. To address this challenge, in this paper, we propose use of TLA+ to automatically check DoS vulnerability of network protocols with completeness guarantee. In particular, we develop new schemes to avoid state space explosion in property checking and to model attackers' capabilities for finding realistic attacks. As a case study, we successfully identify threats to IEEE 802.16 air interface protocols.**

1. Introduction

Security of network protocols is critical in ensuring availability and provisioning of network services to customers. Exploiting and attacking a vulnerable network protocol can cause devastating effects to networks and service providers. For example, a simple black hole attack to a routing protocol caused much of the Internet to halt for 20 minutes to more than two hours in April 1997. Fully aware of the importance of network protocol security, for years, industry and research community have conducted vulnerability analysis as an indispensable first step towards securing a network protocol.

Currently, vulnerability analysis of network protocols is a manual and lengthy process, mostly based on human heuristic and reasoning. In IETF, it is now required to include a "Security Consideration" section in every network protocol draft or RFC containing heuristic threat analysis and recommended security solutions. Manual threat analysis, which may or may not involve security experts, represents human best effort that, which "hopefully" can uncover major potential attacks. There are no rigorous processes or evaluation criteria to ensure completeness or thoroughness of the manual threat analysis.

However, in many cases it is indeed difficult for the human mind to thoroughly analyze potential threats of a protocol. Nowadays, protocols are increasingly complex and a protocol standard can be hundreds or even a thousand pages long (for some wireless standards). Providing mechanisms for interactions of different components, a network protocol can be intrinsically a complex and concurrent system. In the vulnerability analysis, we need to further consider the concurrent interaction of an attacker. Checking a concurrent system works correctly is an extremely difficult task, let alone the robustness of a concurrent system under the attacks of an adversary. As observed by Owicki and Lamport [1], "There is rather a large body of sad experiences to indicate that a concurrent program can withstand very careful scrutiny without revealing its errors. The only way we can be sure that a concurrent program does what we think it does is to prove rigorously that it does it." In other words, formal methods are essential for establishing the security and robustness of a network protocol.

It is highly desirable to have a formal method to automatically check vulnerabilities of any general protocol with completeness and correctness guarantees. Previous vulnerability checking by formal methods mainly focus on security protocols and security properties such as perfect forwarding secrecy and correctness of authentication, using various languages and frameworks. For example, Lowe [2] used CSP and FDR, Shmatikov and Stern [3] used Murϕ, and Corin *et al.* [4] used symbolic traces and PS-LTL. However, non-security network protocols are mostly ignored. More importantly, it remains unclear how to apply these methods to analyze more subtle vulnerabilities such as the denial-of-service (DoS). For example, in the latest related work [4], they admit that they can only model some very primitive DoS attacks. A few other formal analysis on DoS attacks [5], [6] mostly focus on resource exhaustion attacks and ignore protocol malfunction attacks where attackers cause protocol execution into a wrong state so as to prevent users from accessing certain resources.

Thus, in this paper, we propose to use TLA+ [7] as the formal framework for checking the vulnerability of network protocols. In the TLA+ language, we specify a network protocol, as well as an attack model and the security properties to be checked. Then TLC will be run to search the entire protocol state space and output any possible attacks it can find. This can ensure completeness of our analysis. This can identify not only attacks, but also faulty situations. By using this process iteratively, TLA/TLC is also very helpful in protocol design and enhancement as follows. Once a faulty or vulnerable design is identified, a fix or modification will be programmed into TLA and TLC will be rerun to verify if the problem is really fixed.

The reasons behind of our selection of TLA+ is as follows. First, TLA+ is a language based on normal mathematics, not on any specific programming language. It gives us the advantage of specifying the protocol at the appropriate abstraction level. We can easily focus on the details that is relevant and abstract away the irrelevant parts. Second, TLA+ has a uniform mathematical language to specify both a system and its properties. There is no need to establish the semantics for a system. Finally, TLA+ has tools that support the analysis

and checking of the specifications. The most important one is the model checker TLC, which will be used in our automatic vulnerability checking.

There are several challenges for applying TLA+ to automatically check for network protocol vulnerabilities.

- How to avoid state space explosion in the protocol specification and property checking?
- How to model attacker capabilities to find realistic attacks?

To the best of our knowledge, we are the *first* to apply TLA+ to examine the general DoS vulnerabilities of network protocols. To address the aforementioned challenges, we make the following contributions.

- We propose several techniques to reduce the size of state space. When specifying protocol, we (1) combine similar states, (2) replace random variables with constants with some additional properties to simulate the effects of randomness, and (3) use symmetric principals to reduce the extra states caused by nondeterminacy of protocols. When modeling attackers, we add more constraints to the attackers to exclude uninteresting attacks.
- We propose dynamic modification of attacker model which will lead to a complete robustness proof or a report of a realistic attacks.

We apply our formal method to check potential vulnerabilities in the IEEE 802.16 standards [8]. The IEEE 802.16 technology (popularly called as WiMAX), with enormous backing from the industry, is positioned to lead the wireless broadband network space to build the high-speed Wireless Metropolitan Area Networks (MAN). Security is crucial for its functioning and growth, and many security problems associated with WLAN IEEE 802.11 make it more than a necessity to have careful threat analysis on 802.16. However, IEEE 802.16 standards have not been thoroughly scrutinized due to fast evolution of the protocol and vastness of the standard (close to a thousand pages).

We studied two key processes of the 802.16 standard - initial ranging and authentication. Through automated analysis, we found that one potential DoS scenario exists in initial ranging process. The authentication process as given by PKMv2 is invulnerable to any attack based on our attacker model. Note that both conclusions are based on the assumptions that the translation and abstraction process from the protocol spec to TLA is correct.

2. Background and Related Work

2.1. TLA and TLC

TLA (the Temporal Logic of Actions) is a logic designed by Lamport [9] for specifying and reasoning about concurrent systems, and TLA+ [7] is a complete specification language based on it.

TLA is a program logic that expresses both programs and properties with a single language. TLA+ is based on normal mathematics with a simple extension to handle dynamic state changes. All normal mathematics in a static world is expressible in TLA+. It also employs primed variables such as x' to represent the variables in the next state (relative to the current state with unprimed variables). Without using any programming language, TLA frees the specifications from using a limited set of operations and constructs, and forces the control flow to be explicitly stated. One obvious benefit is that a program $Prog$ satisfying a property $Prop$ can be expressed as a predicate $Prog \Rightarrow Prop$. Furthermore, all important concepts in programming can be expressed formally: nondeterminacy is disjunction; program composition is conjunction, etc. More importantly, the freedom from the restrictions of basic constructs in a specific programming language allows TLA to enjoy a rich hierarchy of abstraction levels in mathematics. For example, a large sequential program can always be specified by one action in TLA.

TLA is a slightly extended version of the simplest form of temporal logic [10]. Formulas can be built from elementary formulas using only logical operators ($\neg, \vee, \wedge, \ldots$) and the one temporal operator $\Box$, which means "forever". There are two types of elementary formulas: ones of the form $[\mathcal{A}]$, where $\mathcal{A}$ is an *action*, and *state predicates*. An action is a Boolean-valued expression containing primed and unprimed variables, such as $x' = x + 1$, and a state predicate only has unprimed variables, such as $x + y = 0$. The canonical form of a TLA+ specification is

$$\exists\, y \,:\, Init \wedge \Box[Next]_{\langle x,y \rangle} \wedge Liveness \tag{1}$$

where x and y are tuples of variables, $Init$ is the initial state predicate, $Next$ is an action, and $Liveness$ is a conjunction of fairness conditions on actions. This formula essentially represents a dynamic system where there exist values for the variables y (possibly different values for different states in the system) such that $Init$ holds in the initial state, every successive pair of states satisfies relation $Next$ or leaves x and y unchanged (a stuttering step), and $Liveness$ is satisfied.

TLC is a tool implemented by Y. Yu *et al.*to find the errors in TLA+ specifications. TLC transforms the finite state machine specified by TLA+ into a directed graph with the vertices representing states, and the edges representing transitions. There are two ways to use TLC: simulation and model checking. The simulation mode builds the graph like the depth-first-search and the maximal number of the depth is specified by users, while the model-checking mode builds the graph like the propagation of the wavefront. Thus, the model-checking mode checks all the reachable states, while the simulation mode may miss some reachable states because of the limit of the depth.

2.2. Related Work on Protocol Analysis with Formal Methods

In addition to the work mentioned in Section 1, C.-F Yu *et al.* proposed a formal specification and verification method for prevention of DoS in the absence of failures and integrity violations [11]. Mahimkar *et al.* used game-based formal methods to study availability-related security properties [12]. Meadows's classical work [5], [6] introduced a cost-based framework for analysis of denial of service in network protocols. In her framework, the success of an attack depends on the cost of the attackers and the tolerance functions of defenders, which provides a smooth tradeoff between these factors and the amount of attacks. However, all these works mainly target resource exhaustion DoS attacks. While this stands for a large portion of traditional DoS attacks, there also exist some protocol malfunction attacks as described

in Section 1. On the other hand, TLA+ has much richer expressive power. It can detect *both* of the exhaustive attacks and malfunction attacks.

3. Protocol Vulnerability Checking by TLA+

3.1. General Flow

In TLA+, a network protocol, as any other concurrent reactive system, can be specified as a state transition system:

$$Protocol \triangleq Init \wedge \Box[Next]_x \wedge Fair.$$

If a protocol is specified as a combination of multiple principals, such as a BS (Base Station) and some SSes (Subscriber Stations), its TLA+ specification can be done accordingly as:

$$Protocol \triangleq BS \wedge (\forall\, i \in N : SS(i))$$

where BS and $SS(i)$ are specified as systems with their own initial conditions and state transition actions. The communication between these components are done by shared variables. However, if you want to also take the communication channels and mechanisms into consideration, they can be modeled and specified as other components that will be conjuncted with the above spec.

The difference between vulnerability checking and correctness verification is that in vulnerability checking we generally assume that the protocol is correct in an ideal environment but may have problems under attacks. In order to check for vulnerability, we need to model hostile attackers. It can be shown that given an appropriate capability, one attacker is at least as powerful as a set of cooperating attackers. Therefore, we need only to specify an attacker by $Attacker$ in TLA+. The issues and considerations for attacker specification will be discussed in the next section.

We also need to formally state what are the requested properties of the protocol. Usually, all the correctness properties (in the traditional correctness verification) should be checked to show that the protocol is robust under attacks. Security properties such as secrecy and authentication are only meaningful when there is an attacker (they are trivial true without it). Different from previous work focusing on formal checking of security properties, the properties we focus on in this paper are mainly correctness properties, especially non-existence of DoS. More discussions will be provided in Section 3.4.

Assume the requested property is specified in TLA+ as $Property$, the robustness of protocol under attack is stated as:

$$Protocol \wedge Attacker \Rightarrow Property.$$

The general flow of our approach is to first specify the protocol, the attacker, and the property, and then to use TLC (which is a model checker for TLA+) to automatically prove the above formula. The benefit of such an approach is that the checking is totally automatic and if there is any violation, a trace will be produced by the TLC. Such a trace can be used to either weaken the attacker (in the case where the attack trace is unrealistic) or correct the protocol (in the case we find a vulnerability).

Figure 1 shows the flow of our approach. In each case, the following procedure was followed:

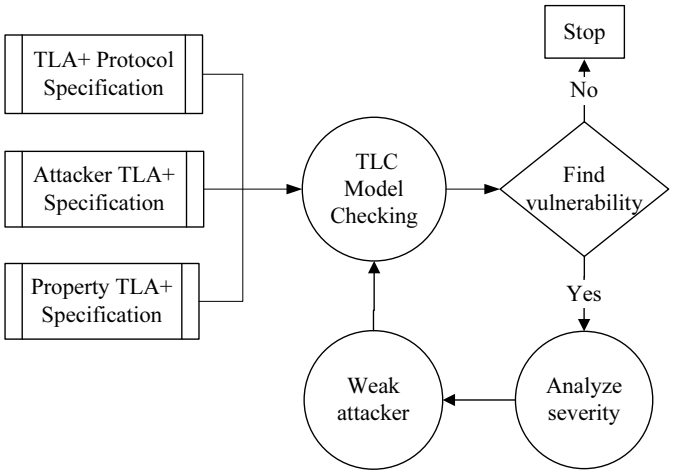

Fig. 1. TLA modeling process.

1) Convert the English language/flow-chart/finite state machine specification to TLA+ specification;
2) Model the attacker and specify it in TLA+;
3) Specify the properties we want to check;
4) Run TLC model-checker to check protocol vulnerability;
5) Analyze the violating trace to see whether the vulnerability is realistic and document it if so;
6) Weaken attacker if vulnerability is not realistic and repeat steps 2 through 5 until robustness is proved.

Next, we will discuss each step in detail.

3.2. Protocol Specification

The first phase of vulnerability analysis is identification of critical parts of the standard and assessment of their vulnerability levels. This is just to prepare for formal modeling and verification phases and could be skipped altogether if the target Standard is limited in scope and size. Once the target parts are identified, the next step is to appropriately model them based on their functionalities.

The process of converting a standard to TLA+ specification can be either straightforward or quite involving based on the format of the standard - a finite state machine gets easily translated to a TLA+ behavior (which is a sequence of states, where a state is an assignment of values to variables) whereas a pure English language specification might have to be first converted to a pseudo code or a flow-chart or a simple sequence of states and transitions before it can be formally specified using TLA+.

The first step of protocol specification is to identify principals or entities in the model. If it's a communication process being modeled, the communicating entities, *e.g.*, clients and servers, become principals. The behavior of each principal can then be modularized using TLA+, with each module comprising of a series of predicate formulae and next-state relations. Each principal begins with an initial state and moves through various states based on different sets of input triggers. Triggers can be messages received, timeouts, exceptions etc. The TLA+ specification would then be comprised of a set of such modules with each having a series of states and transitions.

The next step is to specify the flow of control between principals. The simplest way is to follow a round robin sequence, where each principal does its bit before passing the control to the next one. More sophisticated specifications would require complex flow control mechanisms involving asynchronous interfaces. A note of caution: the more complex the flow is, the harder it becomes to analyze output traces; also, this can lead to significant expansion in state space. Hence, it is worthwhile to explore ways to simplify control flow.

There are some significant challenges posed by this conversion exercise as below.

Challenge: Vagueness in English specification and the correctness in its translation to TLA+. English, being a natural language, cannot precisely and concisely specify system behaviors. Such specifications are often ambiguous. Furthermore, the "correctness" of translation will always be questioned.

Solutions: With a natural language specification as a starting point, there is no hope of a good solution: any translation involves some interpretation and re-invention. Note that this vagueness problem for natural language based specification also exists for manual verification. If it is a newly developed standard, it might prove worthwhile to approach the standards committee to get answers. Consulting product implementation teams might be beneficial at times. A better solution to this problem is for a protocol to be designed and specified in a formal language. In this aspect, we believe that TLA+ is a good candidate of choice; there are many positive experiences from both industry and academia [13], [14].

Challenge: State space explosion. This is one of the most common issues faced during formal verification process. Too many states and transitions coupled with the presence of randomness in the system can cause the model-checking to never stop or take an unreasonable amount of time and space to complete.

Solutions: We propose several schemes to address this problem. First, we combine similar states without loss of functionality into one state, thereby reducing the number of reachable states to minimum required.

Secondly, we replace some random variables with constants with some additional properties to simulate the effects of randomness to bring down the number of states in the state space considerably. This scheme may not be generally applicable to all random numbers in a protocol. But we believe it can be used for most of them. For the parts of 802.16 protocols studied in Section 4, we found all random numbers can be replaced with constants plus some additional properties. For example, nonce is widely used in the network security protocol to avoid replay attack. If we model the nonces in a straightforward manner as random variables, we will have very large or even infinite states. Our solution is to represent a nonce as a set of constants that cannot be guessed by the attacker. Such a change greatly reduces the state space. Another usage of random number is to ensure the liveness of a key.

Thirdly, the nondeterminacy in the protocol also gives a huge state space. For example, if a protocol has multiple equivalent principals A, B, *etc.*, principal A taking an action before B will generate a different state from B taking the action before A. In TLC, we define these principals *symmetric*, which will treat the different states as one common state.

While all these changes are effected to overcome state space explosion, care should be taken the keep the overall behavior of the system unchanged.

3.3. Attacker Model and Specification

The focus of our study was on the usage of over-the-air communication channel used by the two communicating entities - the BS and the SS and hence, we use the following *attacker capability model* similar to Dolev-Yao model [15] in our analysis. Basically attackers can:

- Eavesdrop on and store messages.
- Replay old messages.
- Inject or spoof unprotected messages.
- Corrupt messages on the channel by causing collisions.

We also assume the ideal cryptography, which means unforgeable signatures, safe encryption and safe digest. For example, SHA-1 message digest (used in 802.16) is collision resistant and hence, cannot be calculated without the possession of the secret key. Also, it is a secure one-way function and hence, attacks cannot reverse-engineer to get the key.

Here the challenge is *how to find realistic attacks under such attacker model*. Our solution is to start with an attacker model that is very strong. When TLC model-checks the robustness of the protocol under such an attacker, it may yield traces that are unrealistic attacks. We then weaken the attacker model. As an example, armed with the capability to corrupt messages on the channel, the attacker can continuously corrupt a response message from the BS. Since we are more interested in other realistic attacks, we will put more restrictions on the attacker behavior to weaken it. *This dynamic modification of attacker model will end up with a complete robustness proof or a report of a realistic attack.*

3.4. Property Specification in TLA+

Formally specifying the targeting property that we want the protocol to satisfy is a critical and important issue. It will help us to understand unambiguously what is the real requirement, and it will enable us to mathematically prove that it is satisfied by the protocol. Some traditional correctness properties are easy to specify–simply because they have been studied for very long time. For example, the absence of deadlock in a system with the action $Next$ can be given by $\Box$ENABLED $\langle Next \rangle_x$. However, it is not always easy to specify some properties. The secrecy property may be specified as

$$Secrecy \triangleq \Box(Sec \notin Attacker.Knowledge)$$

However, it does not exclude the situation where partial information of Sec is learned by the attacker. The DoS attack is such a subtle property that we are focusing on in this paper.

The attacker in DoS attacks occupies some/all resources such that some/all normal parties cannot get the resources. A generic TLA+ formula for the "non-existence of DoS" property is as follows:

$$DoSProperty \triangleq$$
$$\forall msg \in Network : Party[msg.source].sentmsg = msg$$

which means that all the messages in the network should be sent from normal parties. This is a very strong property:

the attacker cannot occupy any resource. In reality, even if the attacker occupies some resources, the normal parties can still get the service by several retries. A more reasonable property is to *directly check if normal parties can reach their objective final state.*

- If the final state is a fixed state, the property is:

$$DoSPropertyNew \triangleq$$
$$\Diamond \Box \, (\forall \, i \in PartySet : Party[i].state = ObjState)$$

- Sometimes, there is no such a fixed final state. Instead, the parties can reach the objective state infinite times:

$$DoSPropertyNew \triangleq$$
$$\Box \Diamond (\forall \, i \in PartySet : Party[i].state = ObjState)$$

Another important issue here is that it is possible that normal parties cannot reach the objective final state even without the attacker. For example, if the signal is very weak, a cell phone might not get the service. Since the TLC exhaustively searches all the spaces, it eventually will find a trace to this kind of state although the probability of occurrence of this trace may be very low. So we always need to exclude this kind of trace from the specification of normal parties.

3.5. Model Checking with TLC

When we have TLA+ specifications for the protocol, the attacker, and the property, the model checker TLC can be used to check that the protocol still satisfies the property even under the attacker. Like any other model checker, TLC requests that the system have finite states. Here the system includes both the protocol and the attacker. Since many protocols are naturally infinite-state, again, it is a big challenge to reduce the state space to a small finite one. In addition to the schemes in Section 3.2, we add more constraints to the attackers to exclude uninteresting attacks. For instance, we restrict the maximal number of messages that the attacker can corrupt. The attack where the attacker continuously corrupts messages can be easily detected in reality, so we ignore such scenarios.

4. CASE STUDIES

An SS, when powered up, has to perform certain initialization activities to get it ready to carry user data (voice and data) over the 802.16 communication link. These initialization activities are listed below.

a) Scan for downlink channel and establish synchronization with the BS
b) Obtain transmit parameters (from UCD message)
c) Perform ranging
d) Negotiate basic capabilities
e) Authorize SS and perform key exchange
f) Perform registration
g) Establish IP connectivity
h) Establish time of day
i) Transfer operational parameters
j) Set up connections

Based on the criticality of function and the probability of vulnerability, initial ranging and authentication processes were chosen for TLA modeling as discussed below. The detailed protocol and property specifications of TLA for the vulnerability checking are online at [16].

4.1. Initial Ranging

Initial ranging process is the first step in which an SS communicates with a BS via message exchanges. In this process, an SS acquires correct timing offset and power adjustments such that the SS's transmissions are aligned and received within the appropriate reception thresholds. These adjustments are critical to successful communication over the air-link between the BS and the SS located at reasonably long distances from each other. Subsequent phases in the network entry and initialization process, and eventually the 'actual' data communication can happen only if the initial ranging is successful. The request-response communication between the SS and BS happens until the BS is satisfied with the ranging parameters. If the SS is unable to satisfy the BS, after a predetermined number of retries, the BS orders the SS to move to another channel and initiate the ranging process in that channel. Figure 2 shows the basic message exchange during ranging process.

In the TLA model of the initial ranging process, we set the correctness property as follows.

$$\begin{array}{rrcl} \exists \, i \in ContentionSlots : & \wedge & slot[i] \neq \langle\,\rangle \\ & \wedge & slot[i].type = \text{“REQ”} \\ \Rightarrow slot[i].source.pendrequest & = & slot[i] \end{array}$$

which means that there is at least one request in the allocated slots with the corresponding SS having sent the same request, which is our definition of "success of service".

We also need to consider that the attacker may affect the behavior of the normal parties. We enforce SSs to go to the "Done" state without the attacker, so we can set another correctness condition as follows:

$$\Box \Diamond (SSstate = \text{“Done”}).$$

Without the enforcement, SSs may go to the "Stop" state even without the attacker. For example, the power of the SSs is always too weak. So without the enforcement, we cannot tell if the "Stop" is caused by attack.

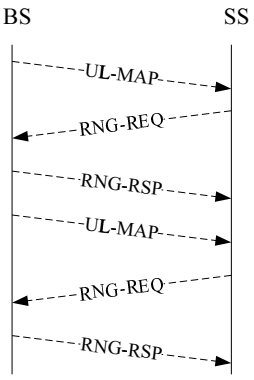

Fig. 2. Initial ranging process.

TLC model checking did find some possible DoS attacks. For example, The RNG-REQ is transmitted during the initial ranging slot as advertised by the BS. Initial ranging slot is a contention-based slot, which means every SS can put RNG-REQ message in the same slot. Binary truncated exponent back-off algorithm is designed to avoid possible collisions. By intentionally sending RNG-REQ frames in all the initial ranging slots, a rogue SS can stop all new SSes connecting to the BS. We also notice that in 802.16 OFDMA MAC layer, CDMA is introduced to mitigate the chance of collision between legitimate users, which makes this DoS attack harder.

On the other hand, TLC model checking exhaustively searches all possible cases that lead SS to unexpected states,

and thus may generate some impractical attack possibilities. For example, the other DoS attack found by TLC is to corrupt all the RNG-RSP messages. But since the attacker cannot predict when RNG-RSP message will be sent out by BS, this kind of attack is not practical.

Finally, TLC finds no more vulnerabilities in this initial ranging process. Note that since we assume that the attacker can change the MAC layer, they can simply ignore the backoff or retransmission and DoS attack the ranging requests. Thus we did not model these parts in TLA+.

4.2. Authentication

In the authentication phase, both the SS and BS mutually authenticate each other and exchange keys for use in data encryption. This process is formulated as a key management protocol called as PKM (Privacy Key Management) protocol. Our focus was on the improved second version called PKMv2.

The working of PKMv2 is directed by two underlying state machines running in the SS - the Authentication State Machine, responsible for handling mutual authentication and the TEK (Traffic Encryption Key) State Machine, responsible for handling TEK exchanges and key refresh. PKMv2 provides for two authentication procedures - RSA-based and EAP-based authentication. Without the loss of generality, we chose the RSA-based procedure for our model. Additionally, PKMv2 employs a SATEK-three-way-handshake for the BS and the SS/MS (Mobile Station) to exchange the security capabilities in general, and the actual TEK's during handover.

Figure 3 gives the high-level view of PKMv2. The PKMv2 works in three phases. In the first phase, the SS and the BS exchange signed messages containing their RSA certificates to mutually authenticate each other. Once this is successfully completed, they perform the SATEK-three-way-handshake procedure to exchange security capabilities. Finally, they exchange key request and response messages to share Traffic Encryption Keys. As each of the keys used by them has a specific lifetime, the above process repeats often.

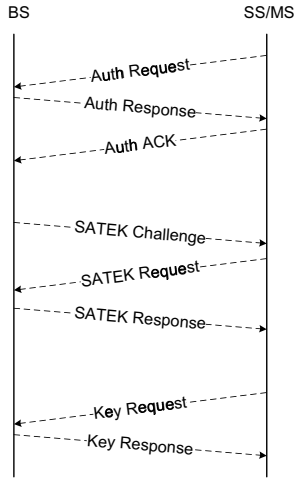

Fig. 3. Overview of Authentication Process using PKMv2.

The finite state machines were converted to TLA+ specification. The BS model was message-driven in that its actions were mainly based on messages it received from the SS/MS with minimal required state information about each SS (such as Key expiry times, the next message it expects to receive from the SS and so on). The model of the attacker mentioned in the Section 3.3.

It is impossible for the attacker to occupy all the resources since the attacker cannot inject its own message if the slot is not empty. We specify that the BS does not send messages such that the SSes go to the Silent state. Each authorization key has a life time, so the SS needs to get authorized from time to time. Thus, SSes should reach the "Authorized" state infinite times. Therefore, we use the following as the correctness condition:

$$\Box\Diamond(SSstate = \text{“Authorized”}). \quad (2)$$

TLC encounters space explosion problem when checking the complicate cases with this kind of liveness condition. In addition to the techniques introduced in Section 3, we restrict the SS can reach "Authorized" state at most a given number of times.

With the above model of the attacker, the TLC model-checking did not yield any suspicious trace. While we cannot conclude that the PKMv2 is invulnerable to any attack, we can safely claim that it is resistant to any attempt using our attacker capability model.

5. Conclusion and Future Work

Our existing work of specifying and validating ranging and authentication part of IEEE 802.l6 standard represents our modest first step towards our aspiration of automatic vulnerability checking of any network protocols with completeness and correctness guarantees. Our future work includes the development of a rigorous process in protocol specification using TLA+ language and modeling of inter-relationships of processes/components. With these enhancements, we will further check vulnerabilities in other parts of 802.16 standards such as mobility support and handoff procedures.

References

[1] S. Owicki and L. Lamport, "Proving liveness properties of concurrent programs," *ACM TOPLAS*, vol. 4, no. 3, pp. 455–495, July 1982.

[2] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR," in *Int'l Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer-Verlag, 1996.

[3] V. Shmatikov and U. Stern, "Efficient finite-state analysis for large security protocols," in *IEEE Computer Security Foundations Workshop*, 1998, p. 106115.

[4] R. Corin and A. Saptawijaya, "A logic for constraint-based security protocol analysis," in *IEEE Symposium on Security and Privacy*, 2006.

[5] C. Meadow, "A formal framework and evaluation method for network denial of service," in *IEEE Computer Security Foundations Workshop*, 1999.

[6] ——, "A cost-based framework for analysis of denial of service in networks," *Journal of Computer Security*, vol. 9, no. 1-2, 2002.

[7] L. Lamport, *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Publishing Company, 2002.

[8] "IEEE Std 802.16-2004: Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems," http://standards.ieee.org/getieee802/download/802.16-2004.pdf.

[9] L. Lamport, "The temporal logic of actions," *ACM TOPLAS*, vol. 16, no. 3, pp. 872–923, May 1994.

[10] A. Pnueli, "The temporal logic of programs," in *IEEE FOCS*, 1977.

[11] C.-F. Yu and V. D. Gligor, "A formal specification and verification method for the prevention of denial of service," in *IEEE Security and Privacy Symposium*, 1988.

[12] A. Mahimkar and V. Shmatikov, "Game-based analysis of denial-of-service prevention protocols," in *IEEE Computer Security Foundations Workshop*, 2005.

[13] B. Batson and L. Lamport, "High-level specifications: Lessons from industry," in *Formal Methods for Components and Objects*, ser. Lecture Notes in Computer Science, no. 2852, 2003, pp. 242–262.

[14] J. E. Johnson, D. E. Langworthy, L. Lamport, and F. H. Vogt, "Formal specification of a web services protocol," in *Int'l Workshop on Web Services and Formal Methods (WS-FM)*, Pisa, Italy, 2004, pp. 23–24.

[15] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, 1983.

[16] "TLA Specification for Ranging and Authentication Process of IEEE Std 802.16-2004," http://list.cs.northwestern.edu/802.16/.