

Formal Verification of the IEEE 802.11i WLAN Security Protocol

Elankayer Sithirasenan¹, Saad Zafar¹, and Vallipuram Muthukkumarasamy²

¹*Institute for Integrated and intelligent Systems*

²*School of Information and Communication technology*

Griffith University, Queensland, Australia

{E.Sithirasenan, S.Zafar, V.Muthu}@griffith.edu.au

Abstract

With the increased usage of wireless LANs (WLANs), businesses and educational institutions are becoming more concerned about wireless network security. The latest WLAN security protocol, the IEEE 802.11i assures rigid security for wireless networks with the support of IEEE 802.1X protocol for authentication, authorization and key distribution. In this study we investigate the integrity of the security model developed by us based on 802.11i Robust Security Mechanism (RSN), strengthening our desire towards establishing a secure wireless network environment. We have used the Symbolic Analysis Laboratory (SAL) tools to formally verify the Behavior Tree models. This paper presents the several Linear Temporal Logic (LTL) formulas established to prove the credibility of our model. We also discuss probable software issues that could arise during implementation. By examining all possible execution traces of the security protocol we have proved our implementation model to be complete and consistent.

1. Introduction

The first wireless security solution for 802.11 based networks, the Wireless Equivalency Protocol (WEP) [1], received a great deal of coverage due to various technical failures in the protocol [2]. Standard bodies and industry organizations are spending enormous amount of money and time in developing and deploying next-generation solutions that address growing wireless network security issues. The 802.11i IEEE standard [3] provides much-improved authentication, authorization, and encryption capabilities. The Wi-Fi Protected Access (WPA) standard [4], a subset of the 802.11i, created by the Wi-Fi Alliance, addresses the weaknesses of 802.11 data privacy by incorporating Temporal Key Integrity Protocol (TKIP), a much stronger implementation of

the RC4 encryption algorithm, plus a sophisticated keying system that ties together the data privacy and authentication functions. IEEE 802.1X [5] was introduced to specifically address the authentication functions in the network environment. The IEEE 802.1X standard enhances the IEEE 802.11i standard with its powerful authentication, authorization and key management functions.

However, the strong security mechanisms introduced by the standards bodies and other organizations can still be in vain if not implemented properly. Software Engineers must be able to interpret and comprehend the standards effectively. A naïve implementation of the security protocols can lead to the same security breaches as in the case of technical flaws. As such, the primary aim of this study is to analyze the Behavior Tree (BT) models developed using the GSE methodology [6] for the IEEE 802.11i RSN [7]. We have used the SAL [8] model checker to formally verify the models. In this process, we have developed a number of theorems to verify the integrity of the protocol. The results obtained assures that the GSE methodology delivers a complete and consistent set of implementation models enabling Software Engineers to successfully implement the security protocol with lesser software defects.

Section 2 of this paper presents an overview of the related work in the field. The IEEE 802.11i protocol details are described in Section 3. A brief explanation on modeling is outlined in Section 4. Details of the formal verification technique used in this research are given in Section 5. In Section 6 we analyze the protocol and finally Section 7 concludes the paper.

2. Related Work

The basic idea of using state graph search to verify network and communication protocols is quite old, dating back to at least 1978 [9]. In recent decades, model checking has made significant progress in tackling the verification of complex, concurrent systems [10]. Tools such as SMV [11], SPIN [12], and Murphi [13] have been used to verify hardware and software protocols by exhaustively searching the state space. By caching states and employing sound state reduction techniques, these tools can detect non-trivial bugs.

Gray and McLean propose encoding the entire protocol in terms of temporal logic [14]. Much like symbolic model checking, they describe the model by giving formulas that express the possible relationships between variable values in the current state and variable values in the next state. This makes their framework more formal than the others, but much more cumbersome as well. They provide a simple example and prove a global variant for this example. The few sub cases they consider are very straightforward but their technique demands very long proofs even for the extremely simple example they present. They argue that their technique could be automated but provide no tools for their system.

Woo and Lam propose much more intuitive model for authentication protocols [15]. Their model resembles sequential programming with each participating principle being modeled independently. There is an easy and obvious translation from the common description of a protocol as a set of messages to their model. Their models are also intuitive because they consider all possible execution traces instead of considering just the set of words obtainable by the intruder. They are concerned with checking of what they call secrecy and correspondence properties. The secrecy property is expressed as a set of words that the intruder is not allowed to obtain. The correspondence properties can express things of the form if principal A finishes a protocol run with principal B, then principal B must have started the protocol run with A. However, they do not provide a general logic in which to formalize security properties, nor do they provide an automated tool.

Dolev and Yao [16] proposed an approach to model network protocols by defining a set of states and a set of transitions that takes into account an intruder, the messages communicated back and forth, and the information known by each of the components. This state space is then traversed to check if some particular state can be reached or if some state trace can be generated. They developed an algorithm for determining whether or not a protocol is secure in their model. However, their model is extremely limited. They only consider secrecy issues, and they model only encryption, decryption, and adding, checking, or deleting a component name.

The drawback of traditional model checkers is that the system to be verified must be modeled in a particular description language, requiring a significant amount of manual effort that can easily be error prone. In contrast the process adopted by us produces graphical models that are derived and integrated from the original requirements. Our method can find conceptual errors much early in the development phase rather than being detected after the software is implemented. The BT models derived using our methodology is complete with appropriate traceability to original requirements. These models when formally verified prove to be consistent with lesser defects.

Static analysis has also gained ground in recent years in detecting bugs in software. Tools such as ESC [17], ESP [18], and the MC Checker [19] have been used to check source code for errors that can be statically detected with minimal manual effort. While static techniques are good for finding specific set of errors it does not discover implementation defects or issues in requirements. Conventional modeling techniques build software models to satisfy the system requirements [20], whereas our methodology derives the models from the requirements [21]. Hence, our BT models represent the exact replica of the given specifications. The hierarchical and graphical models produced by our method are admired for its simplicity, traceability, ability to detect defects and its control of complexity [22]. The security protocol modeled using GSE method provides a complete and consistent system model with minimum design defects. The formal verification performed further strengthens the integrity of our models by traversing every possible executions trace that is not observable by manual human review.

3. The 802.11i Security Protocol

Let us first take a brief look at the IEEE 802.11i standard. The standard defines two classes of security framework for IEEE 802.11 WLANs: RSN and pre-RSN. A station is called RSN-capable equipment if it is capable of creating RSN associations (RSNA). Otherwise, it is pre-RSN equipment. The network that only allows RSNA with RSN-capable equipments is called a RSN security framework. The major difference between RSNA and pre-RSNA is the 4-way handshake. If the 4-way handshake is not included in the authentication / association procedures, stations are said to use pre-RSNA.

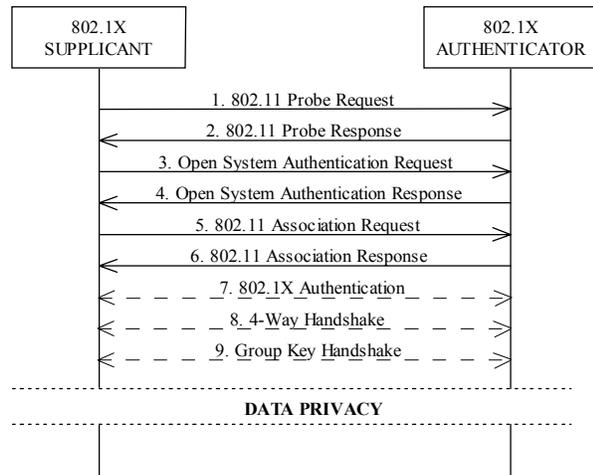


Figure 1. RSN Association

Fig. 1 shows an example RSNA establishment between a supplicant (STA) and the authenticator (AP) in an Extended Service Set (ESS). It assumes no use of pre-shared key. Flows 1-6 are the IEEE 802.11 association and authentication process prior to attaching to the authenticator. During this process, security information and capabilities could be negotiated using the RSN Information Element (IE). The Authentication in Flows 3 and 4 refer to the IEEE 802.11 open system authentication. After the IEEE 802.11 association is completed, the IEEE 802.1X authentication indicated in Flow 7 is initiated. If the supplicant and the authentication server authenticate each other successfully, both of them independently generate a *Pairwise Master Key (PMK)*. The authentication server then transmits the PMK to the authenticator through a secure channel (for example, IPsec or TLS).

The 4-way handshake then uses the PMK to derive and verify a *Pairwise Transient Key (PTK)*, guaranteeing fresh session key between the supplicant and the authenticator. This is indicated in Flow 8. Thereafter, the group key handshake is initiated as indicated by Flow 9. The group key handshake is used to generate and refresh the group key, which is shared between a group of stations and APs. Using this key, broadcast and multicast messages are securely exchanged in the air.

In the next section we present a brief note on modeling the above-described RSN. The modeling from requirements analysis to the final design models was carried out using the GSE methodology.

4. Modeling

The modeling was carried out in stages. First, the RSN requirements were assembled from the IEEE 802.11i specifications. Thereafter, the Requirements Behavior Trees (RBTs) were developed. Several incompleteness and uncertainties were detected while developing the RBTs for the RSN. Such issues were resolved using appropriate domain expertise. Once all ambiguities were resolved, we integrated the RBTs to derive at the Design Behavior Tree (DBT). While integrating we came across integration problems due to inconsistencies in pre/post conditions of RBTs. In such situations, we used proper domain knowledge to assume suitable pre/post conditions enabling viable integration of the requirements (RBTs). The DBT was used to develop the other models that are useful in the analysis.

Component behavior projection

By manual inspection of the DBT we derived the component projection models for the supplicant and the authenticator. We did this by simply ignoring the component-states of all components other than the one we are currently projecting. The resulting connected behavior tree for a particular component defines the behavior of the component that we will need to implement and encapsulate in the final component-based implementation. The projected component behavior for the Supplicant (STA) and the Authenticator (AP) are shown in Fig. 2.

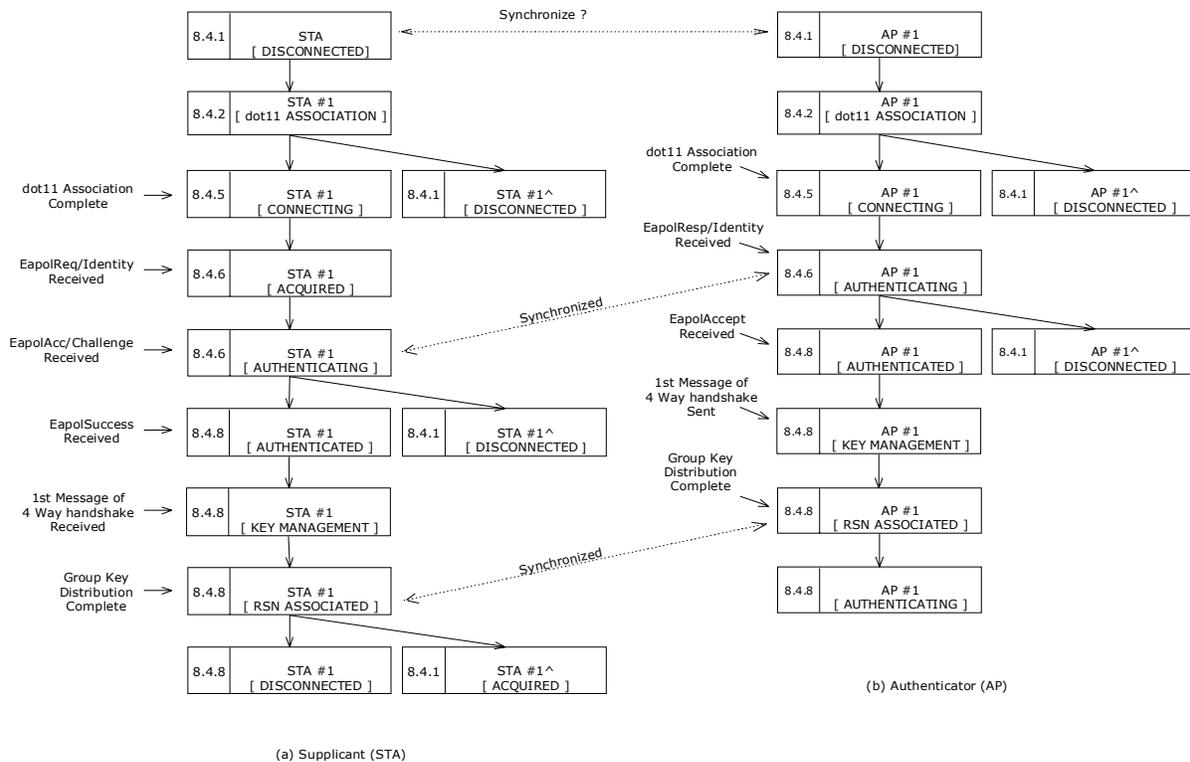


Figure 2. Supplicant (STA) Projection Model

In the component projection, both the STA and the AP are initially at the DISCONNECTED state, which means the port is disconnected. From this state the STA begins the IEEE 802.11 (dot11) association by sending a ProbeReq signal. This dot11 association state of the STA is indicated as dot11 ASSOCIATION in the STA projection model. At any instance if the STA is unable to establish common security parameters with the AP, the STA will decline connection reverting it to the DISCONNECTED state. Once dot11 association is complete both the STA and the AP transfer into the CONNECTING state enabling the port filters. When the filters are ON all non-IEEE 802.1X data traffic is blocked from the uncontrolled port of the authenticator.

Next the IEEE 802.1X (dot1x) authentication begins. The dot1x authentication is initiated by the STA issuing an EapolStart signal to the AP. In reply the AP sends the EapolReq/Identity to the STA. On receiving this message the STA transit to the ACQUIRED state. STA then issues the EapolResp/Identity to the AP advertising its identity. AP transits to AUTHENTICATING state on receiving the response from the STA. The AUTHENTICATING state of the AP is coupled with the Authentication

Server (RADIUS or DIAMETER). Thereafter, the STA transits into AUTHENTICATING state no sooner it receives the EapolAcc/Challenge message from the AS. At this stage depending on the EAP method used the number and the type of messages interchanged between the supplicant and the authenticator may vary. However, at any instance if the supplicant is unable to establish its identity the authenticator declines connection with the STA, thereby pushing the STA to the DISCONNECTED state.

The dot1x authentication completes with the installation of the pairwise master keys (PMK) on both STA and the AP. At this stage both the AP and STA reach the AUTHENTICATED state. This state initiates the 4-way handshake. During the 4-way handshake the PTKs are installed on both the STA and the AP. Once the PTKs are installed the group key handshake begins. The group key handshake transfers the groupwise transient key (GTK) to the STA enabling it to receive broadcast/multicast messages. Once the GTK is installed both STA and the AP stops filtering enabling normal network traffic. At this point both AP and the STA become RSN-ASSOCIATED.

We have given importance to the projection model in this report since it is used to derive some of the temporal logic expressions discussed in section 6. Further details of the modeling process, relevant BT models and the derivation of the projection models can be found in [7].

5. Model Checking

First, we used a special toolset [23] developed by the ARC Centre for Complex Systems to automatically translate the BT model into formal notations like *Communicating Sequential Processes* (CSP) and SAL [24] [25]. The static analysis of the translated specification is possible using different analysis tools available for CSP and SAL. In this paper we have translated the authentication and key distribution models into SAL specification.

SAL is an integrated environment of static analysis tools that includes tools for model checking and theorem proving. In the SAL environment the systems are specified using a description language for state transition. The system properties of interest are calculated in SAL based on the expressed transition system in this description language. In the SAL environment a number of tools provide support for *abstraction, program analysis, theorem proving, and model checking*.

A detailed description of the translation of BT to SAL specification can be found in [25]. However, for the purpose of our study we provide a brief overview of this translation. Since, in BT the concurrent systems are expressed as state transitions, the translation rules for a subset of BT notation is relatively straight forward. Hence, the BT is represented in a single SAL transition system *module*. The components and their states are declared as *state types* in the module. The BT events that are marked with INPUT tags are translated as *input* variables. A set of special variables called $pc1...pcN$ (program counters) is used to keep track of concurrent state transitions in the tree. An atomic action can be either manually specified or automatically generated from set of BT state transitions between two external (observable) events.

For example, consider the BT shown in Fig. 3. In the SAL specification of the BT, there are two program counters $pc1$ and $pc2$. Initially, the program counter $pc1$ is set to 1, which acts as a guard to the first action (A1). In this action, the aP state changes from its

initial state to $aP'=disconnected$ and the program counter is set to $pc1'=2$. The next action A2 is guarded by the condition $pc1=2$. In this action all the state transitions from $aP'=authenticated$ to $timer'=start$ take place and are considered atomic. At this point the tree splits into two branches (A3 and A4). The action A3 represents a thread running in concurrent to the thread A4 (specified by the '||' symbol). Since, the action A3 starts off as a concurrent thread, therefore, a new program counter $pc2$ is used as a guard for this action. This program counter, which was initialized to 0, is set to 1 in action A2. As a result whenever the event $timer=hundredMilSec$ occurs and when the counter $pc2$ is set to 1, then action A3 is triggered. In parallel the action A4 may also occur which is guarded by an input variable $sTA_EapolKeyStaReceive$ and $pc1=4$. This value of $pc1$ is also incremented in action A2.

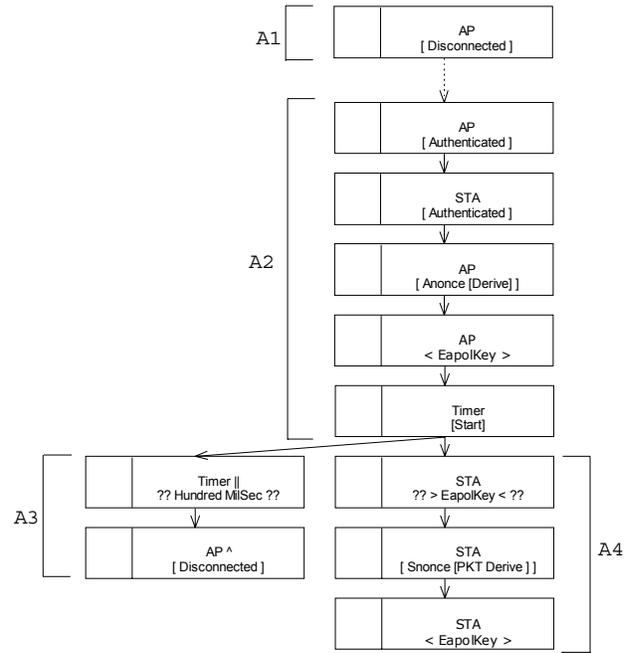


Figure 3. Translation of BT with multiple paths

The SAL code for the BT illustrated in Fig. 6 is shown below:

```

...
[
A1: pc1=1
    -->aP'=apDisconnected;
    pc1'=2;
    ...
]

```

```

A2: pc1=2
    -->aP'=apAuthenticated;
    sTA'=stAuthenticated;
    aP__Anonce'=derive;
    aP__EapolKey'=apSend;
    timer'=start;
    pc2'=1;
    pc1'=4;
[]

A3: pc2=1 AND timer=hundredMilSec
    -->pc2'=0;
    aP'=apDisconnected;
    ...
    pc1'=2;
[]

A4: pc1=4 AND sTA__EapolKeyStaReceive
    -->sTA__EapolKey=staReceive
    pc1'=5;
    sTA__Snonce'=ptkDerive;
    sTA__EapolKey'=staSend;
[]
...

```

Once the system has been specified in the SAL environment language, a number of analyses can be performed [27] on the system specification. The sal-sim tool is a SAL simulator which is used to show different execution paths. The evaluation of different execution paths is usually a desirable first step to assess the correctness of the specification. The SAL specification can also be checked for deadlocks using dead-lock-checker tool.

The useful properties of the system can be specified using linear temporal logic (LTL) or computation tree logic (CTL). These properties can then be model checked using sal-smc and sal-bmc tools. In the next section we discuss the actual model checking performed on the authentication and key distribution models.

6. Analysis

To formally verify the integrity of the security protocol we had to formulate a number of requirements to develop the necessary temporal logic expressions. Firstly, we used the projection model discussed in section 4 and established the behavioral requirements to verify the integrity of our model. Once our model was proven to be complete and consistent we then analyzed the security issues of the protocol. In this view we used the security issues highlighted in Clause 8 (Security) of the **IEEE 802.11i standard** together with the domain expertise derived from real world

experience and formulated requirements to verify the integrity of the security protocol against possible security threats. In this paper we discuss the two main security issues highlighted in the association and authentication processes in the standard.

The following linear temporal logic notations are used in the analysis:

- G(p) (read “always p”), states p is always true
- F(p) (read “eventually p”), states that p will be eventually true
- U(p, q) (read “p until q”), states p holds until a state reached where q holds.

6.1. Association Process

As identified in **Clause 8.4.1** permitting APs to advertise their SSIDs can lead to malicious associations. Furthermore, as shown in Fig. 2, during the dot11 association both supplicant and the authenticator operate independently without any form of software synchronization. Therefore, in a situation where the supplicant or the authenticator is allowed to make presumptions about the characteristics of other participating components can lead to malicious associations or Identity-Theft [28]. Hence, in case of a re-association request from a roaming STA, we first transit the STA into DISCONNECTED state before it is allowed to associate with the new AP. This makes the RSN more reliable so that session-hijack attacks [29] can be avoided. In our model the following LTL formula was proved to be false endorsing our decision:

$$U((sTA = rsnAssociated), ((sTA = staDisconnected) \vee (sTA = staAuthenticating)))$$

Although this is a possible state transition in an RSN, we have deliberately DISCONNECTED the STA to make the RSN more reliable, i.e. an attacker cannot pretend as an Authenticating STA. Further, to strengthen the process defined in **Clause 8.4.2**, we force the STA to DISCONNECT from the AP if it is not RSN capable. Similarly an AP DISCONNECTS itself if it sees an STA that is RSN incapable.

$$G((aP_Rsn = inCapable) \Rightarrow F(sTA = staDisconnected))$$

$$G((sTA_Rsn = inCapable) \Rightarrow F(aP = apDisconnected))$$

As defined in **Clause 8.4.3** both STA and AP disassociate with each other if they don't mutually agree on a common security policy. In Fig. 4, we have illustrated a malicious association scenario making use of this condition. The BT model shows an intruder, InAP (shaded boxes), reading messages from a legitimate AP and appropriately issuing an Association Request to the legitimate AP with a wrong RSN IE (as if it is sent by the associating STA). The legitimate AP disassociates itself assuming the STA is not RSN capable. In the mean time the intruder pretends as the legitimate AP and associates with the mistaken STA.

$G (\text{inAP_dot11Request}=\text{d11Send}) \Rightarrow$
 $(\text{sta_Rsn} = \text{inCapable}) \text{ AND } (\text{aP} = \text{apDisconnected})$

If this type of malicious association takes place the intruder can simply walkthrough the legitimate STA via the entire process of authentication and key distribution acquiring all the necessary security information and characteristics of the legitimate STA. This type of an association cannot be stopped by software means and requires some form of synchronization at lower layers of communication.

6.2. Authentication Process

Next, let us focus on the authentication process. As shown in Fig. 5 this process is initiated by the STA issuing the EapStart message. The STA eventually communicates with the Authentication Server (AS) via a secure channel to validate its credentials. As said in **Clause 8.4.6**, if the AS does not recognize the challenge sent by the STA, the STA has to eventually reach the disconnected state as proved by the following LTL formula.

$G ((\text{aS_AccChallenge} = \text{stachReject}) \Rightarrow F (\text{sta} = \text{staDisconnected}))$

This situation also implies that STA and AP will never reach a state where the controlled port becomes authorized. The following LTL formula proves that the control port never reaches the authorized state if STA does not prove its identity.

$G ((\text{aS_AccChallenge} = \text{stachReject}) \Rightarrow$
 $(\text{control_Port} = \text{pUnauthorized}))$

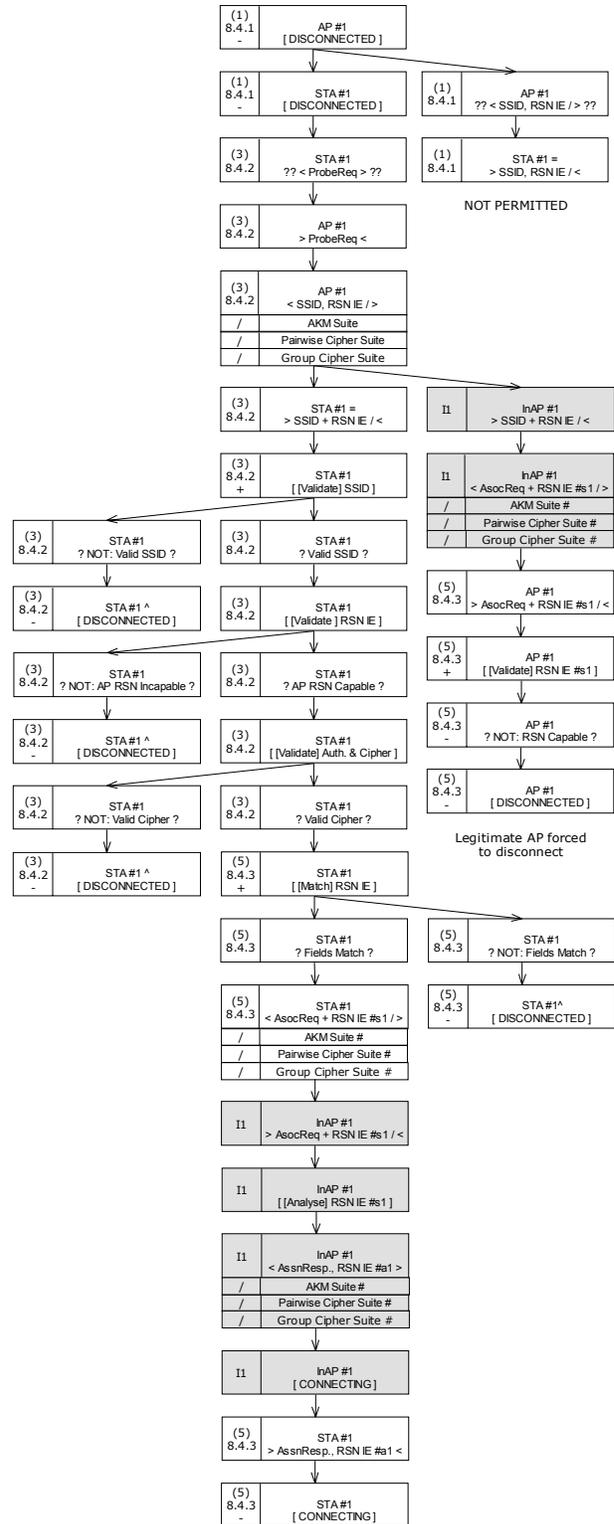


Figure 4. Malicious Association

The purposes of the 4-way handshake are:

- Confirm the existence of the PMK. The second message transfer occurs only if the PMKIDs of STA and AP match each other.

$$G((sTA_Key2 = K2Send) \Rightarrow (STA_PMKID = AP_PMKID))$$

- Ensure that security association keys are fresh. If both STA and AP derive the transient keys they will eventually transit to RSN Associated state.

$$G((sTA_PTK = staptkDerive) \Rightarrow F(sTA = staRsnAssociated))$$

$$G(((aP_PTK = apptkDerive)) \Rightarrow F(aP = apRsnAssociated))$$

- Synchronize the installation of temporal keys. For both STA and AP to reach RSN Associated state transient keys must be installed.

$$G(((aP = apRsnAssociated) \text{ AND } (sTA = staRsnAssociated)) \Rightarrow ((aP_PTK = apptkInstall) \text{ AND } (sTA_PTK = staptkInstall)))$$

Other than the session hijack issue discussed earlier, the IEEE 802.11i security mechanism has another weak point in the key distribution process. In the earlier case we described how the intruder could gain access to the PMK. Having gained access to the PMK, we show how the intruder can perform a session hijack. During the second and third message exchanges both the STA and AP validate the RSN IEs. They both compare the RSN IEs advertised during the dot11 association with that of the RSN IEs transferred during the second and third messages. If the RSN IEs do not match they disassociate. Therefore, an intruder monitoring the key exchanges can deliberately disassociate an STA by issuing a third message with an invalid RSN IE as if it is sent by the AP. Having disconnected the legitimate STA, the intruder will now receive the correct third message from the AP, installs the PTK and become RSN associated. This way the intruder will be in full control of the wireless network.

The formal verification carried out on the BT models confirms the integrity, consistency and completeness of our models. In this process, we first validated all our assumptions using LTL theorems and thereafter we derived theorems to prove the integrity of all state transitions. We have also analyzed possible security issues due to holes in the software implementation of the protocol.

7. Conclusion

Unlike most modeling techniques, the GSE methodology enables issues in the system specifications to be resolved in the early stages of development rather than resolving issues after implementation. In this regard the process used by us to formally verify the protocol is novel and does not suffer from too many false positives, since every scenario checked is a valid execution path due to the inherent qualities of the BT models. Hence, the RSN model developed by us has been proved to be reliable and dependable. All software issues resolved during the design stages have been formally verified. Security threats arising from software defects have been highlighted. On the down side this method does not verify the integrity of the cryptographic aspects of the protocol.

This study was part of an ongoing project towards establishing a safer wireless networking environment. Our next step is to use a state transition tracking mechanism to develop an intrusion prevention and detection system for wireless networks based on these models.

8. References

- [1] IEEE Std. 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.
- [2] A. Stubblefield, J. Ioannidis, A.D. Rubin, "A key recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)", ACM Transactions on Information and System Security, Vol. 7, No. 2, May 2004, pp. 319-332.
- [3] IEEE Std. 802.11i-2004, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 6: Medium Access Control (MAC) Security Enhancements, July 2004.
- [4] Wi-Fi Alliance. Wi-Fi Protected Access, October 2002. URL: [http://www.wi-fi.org/OpenSection/pdf/Wi-Fi Protected Access Overview.pdf](http://www.wi-fi.org/OpenSection/pdf/Wi-Fi%20Protected%20Access%20Overview.pdf)
- [5] IEEE Std. 802.1X-2001, Local and Metropolitan Area Networks – Port-Based Network Access Control, June 2001.
- [6] R.G. Dromey, "From Requirements to Design: formalizing the key steps", Proc. 1st International Conference on Software Engineering and formal methods, September 2003, pp. 2-11.
- [7] E. Sithirasenan, V. Muthukkumarasamy, "IEEE 802.11i WLAN Security Protocol – A Software Engineer's Model", Proc. of 4th AusCERT Asia Pacific Information Technology Security

- Conference, Gold Coast, Australia, pp. 39-50, May 2005.
- [8] S. Bensalem, V. Ganesh, Y. Lakhnech, C. Munoz, S. Owre, H. Rue, J. Rushby, V. Rusu, H. Saidi, N. Shanker, E. Singerman, A. Tiwari, "An Overview of SAL", Proc. 5th NASA Langley Formal Methods Workshop, June 2000, pp. 1-10.
- [9] C.H. West, "General technique for communications protocol validation", IBM Journal of Research and Development, 22(4), 1978.
- [10] M.C. Edmund, and M.W. Jeannette, "Formal methods: state of the art and future directions", ACM Computing Surveys, 28(4):626 – 643, 1996.
- [11] K. McMillan, Symbolic Model Checking, Kluwer Academic Publishers, 1993.
- [12] G. J. Holzmann, "The model checker SPIN", Software Engineering, 23(5):279 – 295, 1997.
- [13] D. L. Dill, A. J. Drexler, J. Alan, Hu, and C. Han Yang, "Protocol verification as a hardware design aid", IEEE International Conference on Computer Design: VLSI in Computers and Processors, pages 522 -525, 1992.
- [14] J.W. Gray and J. McLean, "Using temporal Logic to specify and verify cryptographic protocols (progress report)", Proc. of the 8th IEEE Computer Security Workshop, 1995.
- [15] T.Y.C. Woo and S.S. Lam, "A semantic model for authentication protocols", Proc. of the IEEE Symposium on Research in Security and Privacy, 1993.
- [16] D. Dolev and A. Yao, "On the security of public key protocols", IEEE transactions on Information Theory, 29(2):198-208, March 1989.
- [17] L. David, K. Detlefs, M. Rustan, Leino, G. Nelson, and B. J. Saxe, "Extended static checking", 1998.
- [18] Manuvir Das, Sorin Lerner, and Mark Seigle, "ESP: Path-sensitive program verification in polynomial time", Conference on Programming Language Design and Implementation, 2002.
- [19] D.R. Engler, B. Chelf, A. Chou, and S. Hallem, "Checking system rules using System specific, programmer written compiler extensions", Proc. of the Fourth Symposium on Operating Systems Design and Implementation, October 2000.
- [20] J. Jürjens, "Sound Methods and Effective Tools for Model-based Security Engineering with UML", 27th International Conference on Software Engineering, St.Louis, Missouri, USA; 15 - 21 May 2005
- [21] Robert L. Glass, "Is this a revolutionary idea, or not?", Communications of the ACM, 47(11):23 – 25, 2004.
- [22] Sithirasanen, E. Muthukkumarasamy, V. "A Model for Object Based Distributed Processing Using Behavior Trees", Proc. of the Eighth IASTED International Conference on Software Engineering and Applications, Nov. 2004, Cambridge MA, USA, pp 477-482.
- [23] C. Smith, L. Winter, I. Hayes, R.G. Dromey, P. Lindsay, and D. Carrington, "An Environment for Building a System Out of Its Requirements," Tools Track, 19th IEEE International Conference on Automated Software Engineering, Linz, Austria, 2004.
- [24] K. Winter, "Formalising Behavior Trees with CSP," International Conference on Integrated Formal Methods, IFM'04, 2004.
- [25] L. Grunske, A. Peter, P. Lindsay, N. Yatapanage, K. Winter, "An Automated Failure Mode and Effect Analysis Based on High-Level Design Specification with Behavior Trees", International Conference on Integrated Formal Methods, IFM'05, 2005, 129-149
- [26] Bensalem, S., Ganesh, V., Lakhnech, Y., Muñoz, C., Owre, S., Rueß, H., Rushby, J., Rusu, V., Saïdi, H., Shankar, N., Singerman, E., and Tiwari, A., "An Overview of SAL," presented at LFM 2000: Fifth NASA Langley Formal Methods Workshop, 2000
- [27] Moura, L. d., *SAL: Tutorial*, SRI International, 2004
- [28] Arbaugh, W.A. Shankar, N. Wan, J. "Your 802.11 Wireless Network Has No Cloths", IEEE Wireless Communications, Dec. 2002, pp. 44-51.
- [29] Mishra, A. Arbaugh, W.A. "An Initial Security Analysis of the IEEE 802.1X Standard", Critical Infrastructure Grant, National Institute of Standards, February 2002.