International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:12, 2008

Authentication Analysis of the 802.11i Protocol

Zeeshan Furqan, Shahabuddin Muhammad, and Ratan Guha

***Abstract*—**IEEE has designed 802.11i protocol to address the security issues in wireless local area networks. Formal analysis is important to ensure that the protocols work properly without having to resort to tedious testing and debugging which can only show the presence of errors, never their absence. In this paper, we present the formal verification of an abstract protocol model of 802.11i. We translate the 802.11i protocol into the Strand Space Model and then prove the authentication property of the resulting model using the Strand Space formalism. The intruder in our model is imbued with powerful capabilities and repercussions to possible attacks are evaluated. Our analysis proves that the authentication of 802.11i is not compromised in the presented model. We further demonstrate how changes in our model will yield a successful man-in-the-middle attack.

***Keywords*—**authentication, formal analysis, formal verification, security.

I. Introduction

A Security protocol is a sequence of messages between two or more parties in which encryption is used to provide authentication or to distribute cryptographic keys for new conversations [1]. It is important to guarantee the correctness of security protocols to ensure the desired working of businesses on the internet. This guarantee becomes vital in sensitive domains such as defense, e-commerce, etcetera. History has proven security protocols to be vulnerable to attacks despite circumspect design and meticulous review by experts. In order to foil such vulnerabilities, one would need to employ more sophisticated modes of analyses. The application of formal methods to security protocols refers to mathematics or logic based techniques for the specification, development, and verification of these protocols. For security protocols, this implies modeling of both the communicating parties and the potential penetrator. Formal verification of security protocols is a challenging task due to the presence of intricate reasoning behind the correctness of these protocols. In formal verification, the network is assumed to be hostile as it contains intruders with the capabilities to encrypt, decrypt, and so forth. Formal analysis allow us to do a thorough analysis of the different paths which an intruder can take under a set of environmental assumptions [2].

While the standards for wireless local area network (WLAN) are still being finalized, we prove the authentication of the proposed 802.11i protocol. We formally represent the proposed 802.11i protocol using the Strand Space Model (SSM) [3]. We then present the high level abstraction of a protocol run. We consider a penetrator, empower its capabilities, and evaluate the authentication of the proposed protocol. Given the constraints and suggestions of the proposed architecture, we state convincingly that any attempt to defy the authentication mechanism of the 802.11i protocol will not be successful. We further describe a situation where modifications to our model will lead to a successful intrusion. Our choice of SSM as a verification framework is based on its simplicity, elegance, precision of results, and ease of developing simple and powerful proofs even without automated support.

This paper is organized as follows. We first present related work in Section 2. Section 3 deals with the basics of Strand Space formalism. We present the proposed 802.11i and analyze its authentication using SSM in Section 4. The conclusion is followed in Section 5.

Zeeshan Furqan is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, email: zfurqan@cs.ucf.edu

Shahabuddin Muhammad is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, email: muhammad@cs.ucf.edu

Ratan K. Guha is Professor with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, email:guha@cs.ucf.edu

II. Related Work

The protocol security problem is undecidable [4], [5]. This undecidability is the reason why the analysis tools are not always successful. A security protocol is required to achieve its goals in the presence of saboteurs. Designers should foresee all the possible attacks on the protocol under development. History has shown that the protocols are subject to nonintuitive attacks which are not easily apparent even to careful designers [6]. These attacks are not due to the flawed underlying cryptographic algorithm; instead they are consequences of arbitrary forwarding and combinations of other simple operations. Formalism has been applied to a wide range of security protocols in order to verify the security properties. Dolev and Yao [7] and Dolev et al. [8] contributed the early work in the application of formal methods to verify security protocols. The work of Dolev and Yao [7] is significant as it proposed the earliest formal model and encompassed the capabilities of the penetrator.

Once a formal model of a system has been established; model checking can be utilized to establish the accuracy of the system. A model checker is a tool that explores the state space of the model to determine if there are any paths through the space that corresponds to a successful attack. The tools proposed in [9], [10], [11], [12], [13], [14] are few examples of model checkers. Model checking approach for verifying cryptographic security protocols suffers from the state space explosion problem. Clarke et al. [15] addressed this problem by applying partial order reduction techniques. Model checkers have the disadvantage of being able to search only a finite number of states. In [16], a set of conditions are presented under which checking a small number of sessions would be sufficient to prove the secrecy of a key.

To enable verification, formal languages and/or mathematical notations are often used for which subsequently a proof can

be worked out [17]. Burrows et al. [18] devised a logic based on belief. This logic, abbreviated as BAN logic, introduced a wide range of security objects, formal notations to represent security concepts, formulae describing the relationship between principals and data, and inference rules that helped derive new rules from existing rules. Since then, legions of new methods based on logics began to occupy the attention of researchers. In [19], BAN logic was extended to exclude universal assumptions and include some new notions like recognizability. Syverson and Oorschot [20] applied belief logic to include protocols such as Diffie-Hellman [21]. Paulson [22] constructed Isabelle by inductively defining possible traces exhibited by the communicating agents. Automated theorem provers, such as [23], have been employed to overcome the computational difficulty associated with the theorem provers. The tools presented in [24], [25] are special purpose theorem provers designed for the analysis of security protocols. Type checking [26] is a new approach toward security protocol analysis. This approach has a potential disadvantage of defining security violations in terms of type inconsistencies. Hence, the security requirements must be considered when the specifications are being written. Another approach to security protocol verification is to develop a computational model. A formal system is also developed and a proof of soundness is established for the system using the computational model [27]. Fabrega et al. [3] proposed a graph-theoretic approach, Strand Space Model (SSM), based on [7]. SSM is an elegant approach [28], [29], [30] which has been employed to develop special purpose tools such as [31].

III. Strand Space Model (SSM)

In [3], a *strand* is defined as a sequence of events that a participant may engage in a protocol. A strand represents either a protocol execution by a legitimate party (*regular strand*) or by a penetrator (*penetrator strand*). A *strand space* is a set of strands, consisting of strands of various legitimate protocol parties, together with the penetrator strands. In SSM, each participant of the security protocol is represented by a strand and the individual run of each participant is captured by the traces of these strands. The correctness claims for the protocol are then expressed in terms of connections between different kinds of strands. In the following lines, we describe basic SSM terminologies that will be used in Section 4.

The set of actions that a participant may take during the execution of a protocol includes actions such as send (denoted by +) and receive (denoted by -). Let M be the set of possible messages that can be exchanged by all the participants in a protocol. A *signed term* is a pair $\langle \sigma, u \rangle$ with $\sigma \in \{+, -\}$ and $u \in M$. A signed term $\langle +, u \rangle$ represents the sending of a message u and is typically written as $+u$. Similarly, $-u$, represents the reception of a message u. The set of finite sequence of signed terms is represented as $(\pm M)^*$. A strand space over M consists of a set Σ, whose elements are called strands, together with a trace mapping $tr : \Sigma \rightarrow (\pm M)^*$. The trace mapping tr associates each strand in Σ with a sequence of signed terms.

A $node$ is a pair $\langle s, i \rangle$ with $s \in \Sigma$ and i is an integer with $1 \leq i \leq |tr(s)|$. The set of all the nodes is donated by N. For a node $n = \langle s, i \rangle$, where $tr(s) = \langle \sigma_1, u_1 \rangle ... \langle \sigma_k, u_k \rangle$, $term(n)$ is defined as $\langle \sigma_i, u_i \rangle$. There is an $edge$ $n_1 \rightarrow n_2$ if and only if $term(n_1) = +a$ and $term(n_2) = -a$ for some $a \in M$. The edge $\rightarrow$ represents a potential causal link between two strands. When $n_1 = \langle s, i \rangle$ and $n_2 = \langle s, i+1 \rangle$ are nodes, then there is an edge $n_1 \Rightarrow n_2$. The edge $\Rightarrow$ indicates that n_1 is an immediate causal predecessor of n_2. In a similar fashion, $n' \Rightarrow^+ n$ implies that n' preceedes n (not necessarily immediately) on the same strand. A term t occurs in $n \in N$ if and only if $t \subset term(n)$. The node $n \in N$ is an entry point for the term t if $term(n) = +t$ and whenever $n' \Rightarrow^+ n$, $term(n') \notin term(n)$. A term t *originates* on $n \in N$ if and only if n is an entry point for t. A term t *uniquely originates* if and only if t originates on a uniques $n \in N$. It can be seen that N, together with both sets of edges $n_1 \rightarrow n_2$ and $n_1 \Rightarrow n_2$, is a directed graph $\langle N, (\rightarrow \bigcup \Rightarrow) \rangle$.

A *bundle* represents a full protocol exchange. It consists of a number of strands linked together where one strand sends a message and another strand receives the same message. Intuitively, a bundle is a portion of a strand space that is large enough to represent at least a full protocol exchange. A bundle has a natural causal precedence relation through which inductive arguments are carried out. A bundle is a finite acyclic subgraph that captures the natural causal precedence relation among nodes as defined by the edges $\rightarrow$ and $\Rightarrow$. For a given strand space Σ, let $B = \langle N_B, (\rightarrow_B \bigcup \Rightarrow_B) \rangle$ be a subgraph of $\langle N, (\rightarrow \bigcup \Rightarrow) \rangle$. The graph B is a bundle if:

1) B is finite,
2) if $n_2 \in N_B$ and $term(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_B n_2$,
3) if $n_2 \in N_B$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_B n_2$,
4) B is acyclic.

The bundle-height of a strand is the largest i such that $\langle s, i \rangle \in$ bundle.

The set $T \subseteq M$ is the set of texts (representing the atomic messages). The set $K \subseteq M$ contains cryptographic keys disjoint from T. The term $\{g\}_k \in M$ represents the encryption of the term $g \in T$ using $k \in K$. The *subterm* relation is inductively defined as the smallest relation such that:

1) any term $m \in M$ is a subterm of itself,
2) a term $m \in M$ is a subterm of $\{g\}_k$ if m is a subterm of g,
3) a term $m \in M$ is a subterm of gh if m is a subterm of g or h.

The detailed explanation of the concepts introduced in this section can be found in [3]. We illustrate an example of these concepts with their application to 802.11i in figure 1.

Protocol authenticity can be verified by using the agreement property explained in [32]. A protocol guarantees agreement to a participant B (say, as the responder) for certain data items x, if each time a participant B completes a run of the protocol as responder using x, apparently with A, then there is a unique run of the protocol with the principal A as initiator using x, apparently with B.

IV. Modelling 802.11i in the Strand Space Formalism

IEEE defined a series of specifications for wireless local area networks as IEEE 802.11. 802.11b uses Wired Equivalent Privacy (WEP) protocol to address security. The susceptibility to attacks within WEP motivated protocol designers to redesign the protocol. 802.11i addresses the concerned issues. The communication in 802.11i takes place among three parties: $peer$ (or station), $authenticator$ (or access point), and $RADIUS$ (or authentication) $server$. Communication between the peer and the authenticator is EAP over LAN (EAPOL) [33] and between the authenticator and the RADIUS server is EAP over RADIUS [34].

The dynamics of the 802.11i can be explained in four phases: discovery, authentication, key management, and data transfer. We restrict ourselves to the authentication phase in which a peer sends a start message to the authenticator. The latter then queries the peer about its identity and the peer replies with an answer. The authenticator forwards this reply to the RADIUS server. The RADIUS server evaluates the request and determines whether the peer is a legitimate user or not. For decision purposes, the server raises a challenge and passes this challenge to the authenticator. Next, the authenticator passes the challenge of the RADIUS server to the peer. The peer then responds to the challenge with an answer which is forwarded to the RADIUS server by the authenticator. The authenticator acts like a pass through server for the entire communication. The RADIUS server evaluates the challenge response and sends either an acceptance or a rejection to the peer via the authenticator. This process is also called a handshake. From questions, answers, challenges and responses, we mean different kinds of nonces, keys, and secrets that are required for authentication. Note, that in some implementations, the RADIUS server is implemented in-line with the authenticator, and the authentication involves only the peer and the authenticator.

Intuitively, a successful authentication means that the peer and the authenticator verify the identity of each other and generate some shared secret for future data transmissions. In figure 1, we illustrate an abstract model of the 802.11i's authentication. After the handshakes, the peer and the RADIUS server have authenticated each other and generate a common secret called the Master Session Key (MSK). The peer uses this MSK to derive a Pairwise Master Key (PMK). The Authentication, Authorization and Accounting (AAA) key material on the RADIUS server is transferred to the authenticator to derive the same PMK in the authenticator. The handshake is executed in order to establish a Robust Security Network Association (RSNA). The handshake confirms the existence of the PMK, the liveness of the peers and the selection of the cipher suite. After a successful handshake, a fresh Pairwise Transient Key (PTK) for each subsequent session is generated. Providing lower level details of the above mentioned keys and protocols are beyond the purview of this paper. In this paper, we analyze the handshake after a shared PMK is achieved and before the data communication begins.

We abstract away the lower-level communication details involved in the 802.11i protocol and model it using SSM. In order to prove the authentication property of a protocol, we do not need to focus on the intricate implementation details of the protocol. Instead, we concentrate on the higher level abstraction and emphasize on the necessary actions to avoid possible attacks by the illegitimate parties. A protocol needs to ascertain that the messages are intact and that no secret is divulged during its execution. Our focus is to prove the authentication property in which each participant involved in the communication should be certain that the messages are coming from the legitimate participants. In terms of Strand Space formalism, encrypted messages should originate from the regular strands. We make use of the ideal cryptography and algebra freeness assumptions provided in Section IV-A. A sample run of the 802.11i in terms of SSM is depicted in figure 1.

In figure 1, we represent the message exchange between the peer, the authenticator, and the RADIUS server, after a shared PMK is agreed upon between the authenticator and the peer. P, A, R, and M represent the peer, the authenticator, the RADIUS server and the message respectively. Note that P, A, R, are the abstractions of the ids of the peer, the authenticator and the RADIUS server respectively. M is the abstraction of the rest of the communication message. The term, $\{MPARc_1\}_{K_{PA}}$, represents a message that is encrypted with a shared secret between P and A (K_{PA}) and where c_1 represents a challenge. The challenges (represented as $c_1, c_2, ...$), answers (represented as $x_1, x_2, ...$) and the challenge response (represented as cr) are the abstractions of combinations of different nonces, shared secrets and keys. This abstraction facilitates the formal analysis as it hides the lower level communication and cryptographic details. These lower level communication and cryptographic details are not essential for the authentication, although they could improve the security in some sense. We present our assumptions in the following Section followed by our theoretical model in Section IV-B.

A. Assumptions

We lay out our assumptions in this Section. Our set of assumptions is in accordance with the assumptions set forth by other researchers in the related area.

1) In a protocol environment, each participant is associated with a unique ID. Moreover, for each ID there is a key (or a pair of public-private keys in case of asymmetric cryptosystem) associated with it.
2) In the present work, we do not consider encrypting a message more than once. Earlier work has shown that double or multiple encryptions do not serve a better purpose than a single encryption. Instead, sometimes it is detrimental to encrypt a message more than once [7].
3) In literature of security protocols, a penetrator is generally assumed to be a legitimate participant of the network. The legitimate participant becomes a penetrator when he behaves in an undesired manner.
4) We assume ideal cryptography where all participants share limited computational and cryptanalytic abilities.

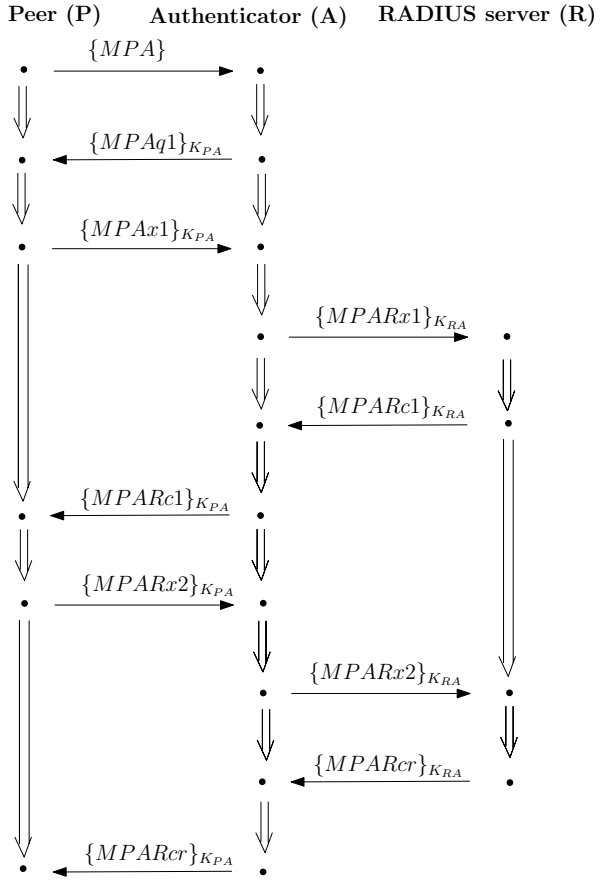

Fig. 1. Strand Space representation of the 802.11i protocol.

5) For analysis purposes, we confine ourselves to a single penetrator. It is shown that increasing the number of penetrators does not increase the probability of a successful attack. A single penetrator can be modelled such that it emulates any number of penetrators [6].
6) We assume our algebra to be freely generated as is assumed in [3].

B. Theoretical Model

We present our theoretical model in this section. Figure 1 represents a bundle B in a strand space Σ where the participants have disjoint ids. The nodes figure 1 are represented by little circles.

We represent the peer, the authenticator, and the RADIUS server by P, A and R respectively. M represents the set of messages exchanged among the participants. We define T_{name} $\in$ {P, A, R } and $N \in$ {P, R}. K is the set of shared keys and is represented as K_{ij} where {i,j} $\in T_{name}$. $K_{pen} \in K$ is the set of keys held by the penetrator. $M_K \in M$ is the message M encrypted with the key K. The set of questions asked by the authenticator is denoted by the set q = {q1, q2, q3,...}. The set $c = \{c1, c2, c3, ...\}$ represents the challenges raised by the RADIUS server. The challenge response c_r indicates a success or a failure. The set $x = \{x1, x2, x3, ...\}$ is the set of answers given by the peer.

The set of queries is represented by $Q = c \bigcup q$. The single arrow, $\rightarrow$, indicates that a message is sent to another participant. The double arrow, $\Rightarrow$, connects two successive nodes on the same strand. The peer strand is represented as S_{peer} and is equal to $Peer[P, M, R, A, c_i, q_j, x_k, K_{PA}]$. This strand is a symbolic representation of infinite instantiations of the peer strand. From figure 1, the trace of the peer strand can be represented as follows:
$\langle +\{PAM\}, -\{MPAq_1\}_{K_{PA}}, +\{MPAx_1\}_{K_{PA}},$
$-\{MPARc_1\}_{K_{PA}}, +\{MPARx_2\}_{K_{PA}}, -\{MPARc_r\}_{K_{PA}}\rangle$.

Similarly, the authenticator strand S_{auth} is represented as $Auth[P, M, R, A, c_i, q_j, x_k, K_{NA}]$. We represent the trace of the authenticator strand as:
$\langle -\{MPA\}, +\{MPAq_1\}_{K_{PA}}, -\{MPAx_1\}_{K_{PA}},$
$+\{MPARx_1\}_{K_{RA}}, -\{MPARc_1\}_{K_{RA}}, +\{MPARc_1\}_{K_{PA}},$
$-\{MPARx_2\}_{K_{PA}}, +\{MPARx_2\}_{K_{RA}}, -\{MPARc_r\}_{K_{RA}}\rangle$.

The Radius strand S_{RADIUS} is equal to $Radius[P, M, R, A, x_k, c_j, K_{NA}]$. We represent the trace of the RADIUS strand as:
$\langle -\{MPARx_1\}_{K_{RA}}, +\{MPARc_1\}_{K_{RA}},$
$-\{MPARx_2\}_{K_{RA}}, +\{MPARc_r\}_{K_{RA}}\rangle$

After formally representing 802.11i, we begin by proving the authentication property of the protocol. We make use of the agreement property to claim that if the peer completes its protocol run with a unique set of parameters, it can be inferred that the authenticator and the RADIUS server must also have completed their part of protocol run using the same set of parameters. Authentication is guaranteed in both directions, that is, the peer authenticates the authenticator and the authenticator authenticates the peer. The RADIUS server is assumed to be authentic in 802.11i. Section 4.2 focuses on the proof where the peer authenticates the authenticator.

C. Peer's Authentication

A bundle represents a $unique$ run of a protocol. Unique refers to a specific set of parameters used in only one protocol run in the entire strand space. Consider a strand space Σ in which bundle B represents a protocol run. SSM defines authentication in terms of bundle-height of strands of legitimate parties. If a strand $S_{peer} \in \Sigma$ represents a peer strand in a bundle B, then the RADIUS and the authenticator strands should also lie in the same bundle B to ensure that the peer, the authenticator and the RADIUS server are involved in the same session of the protocol. The presence of the peer (S_{peer}), the authenticator (S_{auth}) and the RADIUS server (S_{RADIUS}) in the same bundle B guarantees that the peer is talking to the legitimate authenticator and the RADIUS server and not with any masquerading agent.

We represent the peer strand $S_{peer} = Peer[P, M, R, A, c_i, q_j, x_k, K_{NA}]$, the authenticator strand $S_{auth} = Auth[P, M, R, A, c_i, q_j, x_k, K_{NA}]$, and the Radius server strand $S_{RADIUS} = Radius[P, M, R, A, x_k, c_j, K_{NA}]$.

Theorem 1: If S_{peer} has the bundle-height of 6 then for a unique set of questions and answers, there will be authenticator and RADIUS strands of bundle-height 10 and 4 respectively.

Proof 1: (Sketch) Proof of theorem 1 will imply that both the authenticator and the RADIUS server have completed their protocol run with matching variables in the same session with the peer.

The trace of the strand $S_{peer} \in Peer[P, M, R, A, c_i, q_j, x_k, K_{NA}]$, as can be seen in figure 1, is given below:

$\langle +\{MPA\}, -\{MPAq_1\}_{K_{PA}}, +\{MPAx_1\}_{K_{PA}}, -\{MPARc_1\}_{K_{PA}}, +\{MPARx_2\}_{K_{PA}}, -\{MPARc_r\}_{K_{PA}} \rangle$

We need to prove that the term $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in the bundle B. This proof will help in determining the bundle-height of the authenticator.

Theorem 1.1: $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in the bundle B.

Proof 1.1: (Sketch) Consider a bundle B in Σ. We assume that $K_{NA} \notin K_{pen}$. Intuitively, this assumption implies that the penetrator does not possess the secret keys of any other participant. We need to prove that any term encrypted with K_{NA} must originate only on the regular nodes in a bundle. Since K_{NA} $(= \{K_{PA}, K_{RA}\}) \notin K_{pen}$, we just need to show that any regular node does not generate the key K_{NA}. The traces of the peer (S_{peer}), the authenticator (S_{auth}), and the RADIUS (S_{RADIUS}) server show that no key is a subterm of any term of these traces.

We need to prove that K_{PA} is generated on a regular node. We will prove this by showing that K_{PA} is not generated on a penetrator node. We consider the case in which the penetrator generates this secret. As we assumed that $K_{PA} \notin$ set of penetrator keys, we discuss the possible set of actions that a penetrator can take in the following lines.

$Message$: The penetrator strand of this type is $\langle +term \rangle$. The $Message$ strand has only one positive node. This means that the penetrator emits a term without previously obtaining it from anywhere. In our case, the term is equal to $\{MPARc_r\}_{K_{PA}}$. Since the penetrator can not originate a term encrypted with the key K_{PA}, this implies that Σ lacks the message strand.

$Flushing$: The flushing represents a strand $\langle -term \rangle$ which indicates that the penetrator received a term from somewhere and then flushed it. We claim that $\{MPARc_r\}_{K_{PA}}$ is not originated by the penetrator because an originating node is always a positive node. We do not need to worry about any strand with only the negative nodes because a negative node does not imply origination.

Tee: Its trace has a strand of the form $\langle -term, +term, +term \rangle$. The tee strand shows that the penetrator received a term and then forwarded that term twice. As the penetrator received a term in the first node, the term could not have originated by the penetrator. So $\{MPARc_r\}_{K_{PA}}$ was not originated on a Tee strand.

$Concatenation$: Its strand is of the form $\langle -term1, -term2, +term1term2 \rangle$. The penetrator received two terms, concatenated them to form a new term and then forwarded the concatenated term. Considering our algebra to be free, no new term can be obtained by simply concatenating other terms. Hence, no new term is generated at the positive node of the concatenation strand. Instead, the penetrator is sending a concatenated term. So $\{MPARc_r\}_{K_{PA}}$ is not originated on a concatenation strand.

$Separation$: The trace of separation is $\langle -term1term2, +term1, +term2 \rangle$. Since penetrator is getting both the terms, term1 and term2, from the previous node, the separation strand lacks any positive originating node.

Key: This strand emits the key $\langle +K \rangle$. The algebra freeness assumption guarantees that a key cannot be equal to any encrypted message. So $\{MPARc_r\}_{K_{PA}}$ cannot be generated by the key penetrator strand.

$Encryption$: Its strand can be written as $\langle -K, -term, +\{term\}_K \rangle$. The trace of encryption states that the $\{term\}_K$ is equal to $\{MPARc_r\}_{K_{PA}}$. Using algebra freeness, if two encrypted terms are equivalent, the only possibility is that the term is equal to $\{MPARc_r\}$ and K is equal to K_{PA}. This means that the penetrator receives the key K_{PA} in its first node. We assume that any legitimate participant never sends any secret without encryption. Therefore, encryption strand does not produce $\{MPARc_r\}_{K_{PA}}$.

$Decryption$: The decryption strand has the trace $\langle -K^{-1}, -\{term\}_K, +term \rangle$. As the positive term is obtained by decrypting the previous node, no positive node serves as an originator.

After ruling out all the possible penetrator strands, it is safe to conclude that $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in B. Equivalently, this proof can also be extended for $\{MPARc_r\}_{K_{RA}}$. We do not present the proof here because it is similar to the proof 1.1. However, we will prove that $\{MPARc_r\}_{K_{RA}}$ originates on the RADIUS strand. We present this proof in the following lines.

Theorem 1.2: $\{MPARc_r\}_{K_{RA}}$ originates on the RADIUS strand.

Proof 1.2: (Sketch) We have proved that $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in proof 1.1. Since c_r is a subterm of $\{term\}_{K_{PA}}$, it implies that c_r originates on a regular node. If c_r is a subterm of $\{term\}_{K_{RA}}$, then it is clear from the traces of regular strands that this term originates only on the RADIUS strand. We know that c_r can not originate on the penetrator strand. Hence, $\{MPARc_r\}_{K_{RA}}$ belongs to the RADIUS strand where M is any message, $\{PAR\} \in T_{name}$, and c_r is the challenge response.

The term $\{MPARc_r\}_{K_{RA}}$ is present on the last node on the RADIUS strand. According to the bundle property, a bundle must contain all the previous nodes that collectively makes bundle-height equal to 4.

Since all the messages are encrypted with symmetric keys and we assume that K_{PA} and K_{RA} do not belong to K_{pen}, we can say that no penetrator can sit in the middle and behave as a regular participant by simply forwarding the messages to the legitimate parties. The keys K_{PA} and K_{RA} are shared between the peer and the authenticator, and the authenticator and the RADIUS server respectively. Hence, all the messages arriving at any regular node must be originated on the legitimate strands. The peer receives the term $\{MPARc_r\}_{K_{PA}}$ in its last message of the peer strand that makes the peer height equal to 6. Since this message occurs in the last node of the authenticator strand, according to the bundle property it

must contain all the previous nodes that makes the bundle-height of the authenticator equal to 10. Similarly, the RADIUS server strand possesses the term $\{MPARc_r\}_{K_{RA}}$ on its 4th node which makes the the RADIUS server height equal to 4. Thus authenticator height is equal to the peer height plus the RADIUS server height, i.e, 4 + 6 = 10. This also implicitly guarantees that the authenticator is not rogue because it has to have the secret keys to forward messages between the peer and the RADIUS Server.

D. Authenticator's Authentication

Consider a strand space Σ with the bundle B representing a protocol run. We proceed by stating that if a bundle contains a strand $S_{auth} \in \Sigma$ then the peer and the RADIUS strands will agree with the authenticator.

Theorem 2: Assume a typical run of the protocol in which the RADIUS server challenges the peer. If $S_{auth} \in Authenticator[P, A, M, R, c_i, q_j, x_k, K_{NA}]$ *of B-height of at least 9, then there are regular strands such that:*

1. $S_{peer} \in Peer[P, A, M, R, c_i, q_j, x_k, K_{PA}]$ *of B-height at least 5.*

2.$S_{radius} \in Radius[P, A, M, R, c_i, x_j, K_{RA}]$ *of B-height = 4.*

Proof 2: (Sketch) The trace S_{auth} can be written as: $\langle -\{MPA\}, +\{MPAq_1\}_{K_{PA}}, -\{MPAx_1\}_{K_{PA}}, +\{MPARx_1\}_{K_{RA}}, -\{MPARc_1\}_{K_{RA}}, +\{MPARc_1\}_{K_{PA}}, -\{MPARx_2\}_{K_{PA}}, +\{MPARx_2\}_{K_{RA}}, -\{MPARc_r\}_{K_{RA}}\rangle$.

We do not take the trivial case in which the RADIUS server does not challenge the peer. We assumed that $K_{RA} \notin K_{pen}$. By using a proof similar to proof 1.1, we can infer that $\{MPARc_r\}_{K_{RA}}$ originates on a regular node in the bundle. Using proof 1.2, we can say that the term $\{MPARc_r\}_{K_{RA}}$ originates on the RADIUS server strand. Using theorems 1.1 and 1.2, we can conclude that the bundle-height of the RADIUS server strand is equal to 4. Similarly, using theorem 1.1 we can conclude that the term $\{MPAx_2\}_{K_{PA}}$ lies on a regular peer strand. Since $\{MPAx_2\}_{K_{PA}}$ is the last node on the peer strand, bundle property makes the bundle-height of the peer equal to 5.

The term $+\{MPA\}$ is not necessarily an encrypted message. Even if some intruder tries to impersonate a legitimate user, the reply from the authenticator is encrypted by a shared key between the authenticator and the legitimate user. The intruder will not be able to reply to the authenticator with the term $+\{MPAx_1\}_{K_{PA}}$. We need to emphasize the fact that the authenticator can not guarantee bundle-height of its strand to exceed 9 because any penetrator can sit in the middle and throw away (flush) messages among regular parties. The basic notion is the guarantee that if a message encrypted with a secret key is received by a participant, then that message must have originated on a regular strand. However, we can not guarantee that a message sent by a legitimate party will always be received.

E. Case Analysis

We present a scenario where peer-authenticator communication is not supported by a shared secret. In a case in which the authenticator forwards messages to the peer after communicating with the RADIUS server, we expect a man-in-the-middle attack by a penetrator lying in between the peer and the authenticator. The penetrator possesses a strand of the form Tee with a trace $\langle -term, +term, +term\rangle$. In this strand, the penetrator gets a message from a legitimate party and forwards it without changing it. The penetrator may apply multiple strands to form a complex attack. In the scenario presented above, any penetrator lying in the middle can pose itself as a legitimate participant to the peer, the authenticator, or to both.

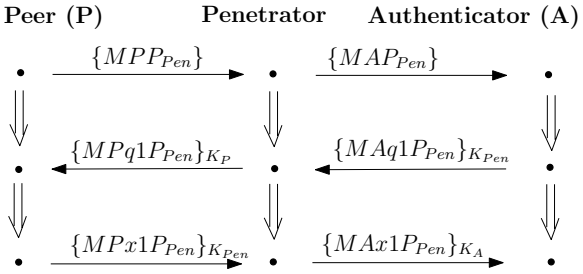

Fig. 2. Case analysis of 802.11i.

The authenticator acts as a gateway between the peer and the RADIUS server. It first decrypts the incoming messages using its shared secret with the source and then forwards the outgoing messages after encrypting it using a shared secret with the destination. In figure 2, a penetrator P_{pen} behaves as an authenticator and engages the peer into a protocol run. If the penetrator starts communicating with the authenticator, a hostile situation may exist. Consider a public key system in which the penetrator starts running the protocol by simply forwarding the messages between the peer and the authenticator. In this way, a penetrator may be able to get all the secret questions answered and gets authenticated. The rest of the protocol proceeds as normal. The penetrator behaves as an authenticator for the peer and as a peer for the authenticator as shown in figure 2.

V. Conclusion

Formal verification of security protocols is important to guarantee the correctness of these protocols. While the standards for wireless LAN protocol are being finalized, we have modelled the 802.11i protocol using the Strand Space formalism. We have used an algebraic framework and strong penetrator capabilities to evaluate the authentication of the 802.11i protocol. Given the penetrator capabilities as modelled in terms of SSM, we have proved the authentication of the 802.11i protocol. We have extended our work to demonstrate a situation where changes from the proposed model leads to a successful man-in-the-middle attack. In this paper, we have confined ourselves to the authentication property. Security properties such as secrecy, integrity, etcetera will be the part of our future research.

ACKNOWLEDGMENT

This work was partially supported by ARO under grant DAAD19-01-1-0502. The views and conclusions herein are those of the authors and do not represent the official policies of the funding agencies or the University of Central Florida.

REFERENCES

[1] R. M. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, Dec. 1978.
[2] C. Meadows, "Formal methods for cryptographic protocol analysis: Emerging issues and trends," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 44–54, Jan. 2003.
[3] F. T. Fabrega, J. Herzog, and J. Guttman, "Strand spaces: Proving security protocols correct," *Journal of Computer Security*, vol. 7, no. 1, pp. 191–230, 1999.
[4] N. Heintze and J. Tygar, "A model for secure protocols and their compositions," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 16–30, Jan. 1996.
[5] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov, "A meta-notation for protocol analysis," in *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, June 1999.
[6] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using FDR," in *Tools and Algorithms for the Construction and Analysis of Systems, 2nd International Workshop TACAS'96*, ser. LNCS 1055. Springer Verlag, Mar. 1996, pp. 147–166.
[7] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, pp. 198–208, Mar. 1983.
[8] D. Dolev, S. Even, and R. karp, "On the security of ping-pong protocols," *Information and Control*, vol. 55, no. 1-3, pp. 57–68, 1982.
[9] J. K. Millen, S. C. Clark, , and S. B. Freedman, "The interrogator: protocol security analysis," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 274–288, Feb. 1987.
[10] R. Kemmerer, "Using formal methods to analyze encryption protocols," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 4, pp. 448–457, 1989.
[11] D. Longley and S. Rigby, "An automatic search for security flaws in key management schemes," *Computers and Security*, vol. 11, no. 1, pp. 75–90, 1992.
[12] J. C. Mitchell, M. Mitchell, and U. Stern, "Automated analysis of cryptographic protocols using Murϕ," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997, pp. 141–151.
[13] C. A. Meadows, "Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.
[14] D. Rosenzweig, D. Runje, and W. Schulte, "Model-based Testing of Cryptographic Protocols," in *Proceedings of the IST/FET International Workshop on Trustworthy Global Computing*. Springer-Verlag, Apr. 2005.
[15] E. Clarke, S. Jha, and W. Marrero, "Partial Order Reductions for Security Protocol Verification," in *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*. Springer-Verlag, 2000, pp. 503–518.
[16] G. Lowe, "Towards a completeness results for model checking security protocols," *Journal of Computer Security*, vol. 7, no. 2, 1999.
[17] R. Gumzej, M. Colnaric, and W. Halang, "Temporal feasibility verification of specification PEARL designs," in *Proceedings of the Seventh IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*. IEEE Computer Society Press, May 2004, pp. 249–252.
[18] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, Feb. 1990.
[19] L. Gong, R. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1990, pp. 234–248.
[20] P. Syverson and P. V. Oorschot, "On unifying some cryptographic protocol logics," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1994, pp. 14–28.
[21] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
[22] L. C. Paulson, "The inductive approach to verifying cryptographic protocols," *Journal of Computer Security*, vol. 6, pp. 85–128, 1998.
[23] S. Brackin, "Evaluating and Improving Protocol Analysis by Automatic Proof," in *Proceedings of the 11th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1998, pp. 138–152.
[24] J. Heather and S. Schneider, "Towards automatic verification of authentication protocols on an unbounded network," in *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2000.
[25] E. Cohen, "TAPS: a first-order verifier for cryptographic protocols," in *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2000, pp. 144–158.
[26] M. Abadi, "Secrecy by typing in security protocols," *Journal of the ACM*, vol. 46, no. 5, pp. 749–786, Sept. 1999.
[27] M. Abadi and P. Rogaway., "Reconciling two views of cryptography (the computational soundeness of formal encryption)," *Journal of Cryptology*, vol. 15, no. 2, pp. 103–127, Jan. 2002.
[28] J. Y. Halpern and R. Pucella, "On the relationship between strand spaces and multi-agent systems," *ACM Transactions on Information and System Security*, vol. 6, no. 1, pp. 43–70, 2003.
[29] Q. Ji, S. Qing, Y. Zhou, and D. Feng, "Study on strand space model theory," *Journal of Computer Science and Technology*, vol. 18, no. 5, pp. 553–570, 2003.
[30] C. Caleiro, L. Vigano, and D. Basin, "Relating strand spaces and distributed temporal logic for security protocol analysis," *Logic Journal of the IGPL*, vol. 13, no. 6, pp. 637–664, 2005.
[31] D. Song, S. Berezin, and A. Perrig, "Athena: a novel approach to efficient automatic security protocol analysis," *Journal of Computer Security*, vol. 9, pp. 47–74, 2001.
[32] G. Lowe, "A hierarchy of authentication specification," in *Proceedings of the 1997 IEEE Computer Society Symposium on Research in Security and Privacy*, 1997, pp. 31–43.
[33] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. E. Levkowetz, "Extensible authentication protocol EAP. RFC 3748," June 2004.
[34] C. Rigney, A. Rubens, W. Simpson, and S. Willens, "Remote authentication dial in user service (RADIUS). RFC 2138," Apr. 1997.