Chapter 8

A Correctness Proof of a Mesh Security Architecture

Doug Kuhlman, Ryan Moriarty, Tony Braskich, Steve Emeott and Mahesh Tripunitara

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/49051

1. Introduction

We discuss our proof of security properties of a standards-track protocol suite for authentication and key establishment using a formal verification technique. Our technique is Protocol Composition Logic (PCL) [15] (see Section 2.1). Our setting is the IEEE 802.11 Mesh Networking task group, known as 802.11s, which was formed to define extensions to IEEE 802.11 [1] to support wireless mesh networking [25]. A goal of the task group is to secure a mesh by utilizing existing IEEE 802.11 security mechanisms and extensions.

The Mesh Security Architecture (MSA) proposal [4–7] to 802.11s consists of a definition of a key hierarchy and a suite of protocols to enable security in a wireless mesh network. The proposal includes detailed information to implement MSA within the framework defined by 802.11s, including key derivation, protocol execution, and message formatting. The suite of protocols encompasses all the necessary components to create and maintain a mesh of nodes.

We describe the following three major contributions in this chapter:

- We conduct a comprehensive assessment of all 10 protocols (averaging 4 messages and 8 components) of the MSA proposal from a security standpoint and proven its correctness. We present an overview of the protocol suite and the main insights from the proof. The full details are generally unenlightening; a companion technical report [28] complements this chapter.

  As this is one of few instances of the proof of correctness of a substantial, standards-track protocol suite of which we are aware, we feel that this is an important contribution.
- PCL has been used to prove the correctness of the IEEE 802.11i protocol suite [26]. However, 802.11s presents new challenges that have necessitated extensions to PCL for us to be able to carry out our correctness proof. We present these extensions and details from the MSA proposal that illustrate their necessity (see Section 3). We believe that the extensions are general enough to be useful in other work in protocol verification.

- In the course of carrying out our proof, we discovered two security issues with protocols in the proposal. We discuss these issues and our suggestions for changes to address them. Our suggestions have since been incorporated into the proposal. As we point out in Section 5, our proof would not have been possible without these changes.

The remainder of this chapter is organized as follows. In Section 2, we provide a background on PCL, 802.11s and the MSA proposal. In Section 3 we present our additions to PCL; for each addition we illustrate its need via components from the protocol suite we have analyzed. We provide an overview of the proof in Section 4. In Section 5, we discuss our recommendations for changes to the original design of the protocol suite in the MSA proposal based on our proof efforts. We conclude with future work and general conclusions in Section 6.

2. Preliminaries

In this section, we provide some background and motivations for our work.

2.1. Overview of proof method

We use Protocol Composition Logic (PCL) to prove the correctness and security of the Mesh Security Architecture. We provide a brief overview of PCL in this section. PCL has been used for a security analysis of 802.11i [26], Kerberos [32], and the Group Domain of Interpretation (GDOI) protocol [29].

2.1.1. Terminology

Protocols in PCL are modeled using a particular syntax. A *role* specifies a sequence of actions performed by an honest party. A matching set of roles (two, in this chapter) define a *protocol*. A particular instance of a role run by a specific principal is a *thread*. Possible actions inside a thread include nonce generation, signature creation, encryption, hash calculation, network communication, and pattern matching (which includes decryption and signature verification). Each thread consists of a number of *basic sequences*, each of which has pre- and post-conditions. A basic sequence is a series of actions, which may not include a blocking action (like receive) except as the first action. Pre- and post-conditions are assertions expressed as logic predicates that must be true before and after a protocol run, respectively. Each basic sequence may have pre- and post-conditions as well, allowing for additional reasoning about certain actions.

2.1.2. Notation

We use the following notation in this chapter. Our notation is consistent with previous work on PCL except for the extensions that we propose in this chapter (see Section 3 for a discussion of the extensions).

$X, Y, Z, \ldots$ are used to denote threads.

$\hat{X}, \hat{Y}, \hat{Z}, \ldots$ denote the principals associated with the corresponding threads.

send, receive, new, ... are *actions*. Actions are things that principals do in a thread.

MKHSH, **TLS:CLNT**, **4WAY**, ... denote protocols. We use the convention of protocol:role to note both the protocol and the associated role that a principal plays in an instance of the protocol; for example, in **TLS:CLNT**, **CLNT** denotes that it is the client's portion of the **TLS** protocol.

$pmk_{X,Z}$, gtk_X, ... denote cryptographic keys. We use subscripts to indicate the principal(s) with whom a key is associated.

θ, Φ, Γ, ... are used to denote logic formulae that express pre- or post-conditions, or *invariants*.

Has(), KOHonest(), SafeMsg(), ... are logic predicates that are used in assertions (pre- and post-conditions, and invariants).

Many of the predicates follow a $Pred(actor, action)$ format. Thus, $\text{Has}(X, m)$ means that thread X has information m. Similar predicate formats follow for Send, Receive, New, and Computes. Other predicates can be more complicated. $\text{Honest}(\hat{X})$ means that the principal ($\hat{X}$) running the thread is honest. $\text{KOHonest}(s, \mathcal{K})$ essentially means that all principals with access to any key $k \in \mathcal{K}$ or to the value s are honest. $\text{Contains}(m, t)$ is equivalent to $t \subseteq m$ and means that information t is a subterm of m.

2.1.3. Proof methodology

The proof methodology of PCL is described by Durgin et al. [21, 22] and Datta et al. [12–18, 26, 32]. We use the standard syntax of $\theta[P]_X\Phi$. This means that with preconditions θ before the run of actions P by thread X, the result (postcondition) Φ is proven to hold. θ is always used to denote a precondition, Φ a postcondition, and Γ an invariant.

The proof system is built on three fundamental building blocks. The first is a series of first-order logical axioms [15]. A first-order logical axiom is a natural logical assumption (e.g., creation of a value implies possession of that value). The second is a series of cryptographic/security axioms [15, 22, 26]. Cryptographic axioms provide formal logic equivalents of standard cryptography (e.g., possession of a key and a value provides possession of the encryption of the value with that key). These assume idealized cryptographic functionality which most cryptographic primitives do not achieve in practice. For example, the hash of two different values is assumed to never be the same.

The third building block is the fundamental principle of *honesty*. Honesty imposes certain restrictions on roles – that they follow protocol descriptions correctly and do not send out particular information assigned to that role. Honesty is a special type of rule that allows an instance of a thread to reason about the actions of another, corresponding thread that participates in the same protocol. The actions of an attacker are not assumed to be honest. We do, however, assume that the attacker does not violate an assumption, condition or invariant (e.g., the possession of a private key) that is necessary for a protocol to run to completion. This notion of an attacker model is the same as that considered in previous work that uses approaches based on mathematical logic to verify protocols (c.f. [26]).

All but one of the axioms on which we depend have been proposed previously [12, 15, 16, 26]; space constraints preclude the presentation of a comprehensive list of all PCL axioms in

this chapter. We provide a few frequently used axioms in Figure 1. We do, however, point out that we need an additional axiom: a node which generates a signature over some (previously-defined) information has that information and the key with which the signature is generated. The existence of information m outside of the computation is important to eliminate concerns about existential signature forgery.

SIG1: $m \wedge \text{Computes}(X, SIG_k(m)) \supset \text{Has}(X, m) \wedge \text{Has}(X, k)$.

(Computes() and Has() are predicates, $\supset$ can be read as "implies," $\wedge$ is logical conjunction and $a < b$ indicates a occurred temporally before b.)

AA1	$\phi[a]_X \mathsf{a}$
AA4	$\phi[a_1; a_2; \ldots; a_k]_X \mathsf{a}_1 < \mathsf{a}_2 \wedge \ldots \wedge \mathsf{a}_{k-1} < \mathsf{a}_k$
AN2	$\phi[\text{New } x]_X \text{Has}(Y, x) \supset Y = X$
AN3	$\phi[\text{New } x]_X \text{Fresh}(X, x)$
ARP	$\text{Receive}(X, p(x))[\text{match } q(x)/q(t)]_X$
	$\text{Receive}(X, p(t))$
FS1	$\text{Fresh}(X, t)[\text{send } t']_X$
	$\text{FirstSend}(X, t, t')\forall t \subseteq t'$
FS2	$\text{FirstSend}(X, t, t') \wedge \mathsf{a}(Y, t'') \supset \text{Send}(X, t') < \mathsf{a}(Y, t'')$, where $X \neq Y \wedge t \subseteq t''$
HASH1	$\text{Computes}(X, HASH_K(x)) \supset \text{Has}(X, x) \wedge \text{Has}(X, K)$

Figure 1. Some PCL Axioms Used in MSA Proofs

The methodology of PCL has proven very successful in dealing with large-scale architectures. A recent paper by Cremers looked at the soundness of the various axioms of PCL [11]. For the problem of preceding actions, we have consistently used implicit pre- and post-conditions at the basic sequence level, leading to a tighter joining of actions. Another issue arises with the **HASH3** axiom. We propose a straight-forward generalization of the **HASH3** axiom, following earlier work on signatures. We define a new axiom, which is sound.

HASH3′: $\text{Receive}(X, HASH_K(x)) \supset \exists Y.\text{Computes}(Y, HASH_K(x)) \wedge \text{Send}(Y, m) \wedge \text{Contains}(m, HASH_K(x))$.

2.1.4. Composing proofs

An important feature of PCL is that with it, we can reason about how protocols interact. As this chapter covers an entire architecture, it is imperative that the large number of individual protocols be proven secure not only independently, but also working together in conjunction as a complete system. To this end, we extensively use the methodology of protocol composition developed by Datta et al. [15]. We discuss this in Section 4.3. Alternate composition methods are available, in certain circumstances [10].

2.2. Overview of the MSA proposal for 802.11s

The 802.11s task group is working to develop a mesh networking protocol that sets up auto-configuring, multi-hop paths between wireless stations to support the exchange of data packets. A goal of the task group is to utilize existing IEEE 802.11 security mechanisms [1], with extensions, to secure a mesh in which all the stations are controlled by a single logical administrative entity from the standpoint of security [25]. The 802.11s task group continues to refine its draft specification through the resolution of comments received during a review of the specification that began in late 2006 [4–7].

A mesh network is a collection of network nodes, each of which can communicate with the others. Several kinds of nodes are specified in the MSA proposal. One is a *Mesh Point* (MP), a member of the mesh that can communicate with other nodes. Each mesh has at least one *Mesh Key Distributor* (MKD) which is an MP that is responsible for much of the key management within its *domain* (a MKD's domain is the set of nodes with which it has a secure connection). The MKD also provides a secure link to an external authentication server (e.g., a RADIUS [30] server). A *Mesh Authenticator* (MA) is an MP which has been authorized by the MKD to participate in key distribution within the mesh. A *Candidate MP* is an entity that wishes to join the mesh but is not yet an MP.

Differences from 802.11i Part of the MSA proposal is very similar to the 802.11i protocol suite [1]. In 802.11i, connections are established between authenticators and supplicants in a server-client topology. An authenticator is connected to a backbone infrastructure, and each supplicant may use an Extensible Authentication Protocol (EAP) [3] method such as EAP-TLS [34] to authenticate with the infrastructure. Each supplicant then uses a four-message handshake to secure a session with an authenticator, allowing subsequent use of its resources. The authenticator also maintains a broadcast key that is given to each of its successful supplicants. These protocols were examined in [26] and proven to be secure.

In addition to the 802.11i functionality, the MSA proposal allows the mesh to be a peer-to-peer network. Nodes in an MSA mesh may play different roles at different times. Thus, the proof of security of the 802.11i 4-way handshake [26], which assumes limitations on the messages a node can send, does not hold. The peer-to-peer nature also poses some difficulties with timing. The 802.11i proofs adopt *matching conversations* [2] as the authenticity property. As we discuss in Section 3.1.1, the notion of matching conversations imposes a rather strict ordering of messages in a protocol run, and is too rigid for our purposes. In MSA, we must provide for the case that both parties simultaneously start instances of a protocol and messages are not necessarily well-ordered. Thus, the proofs from [26] do not carry over directly.

The key hierarchy Each node in Figure 2 represents a key in principal X's key hierarchy. An edge from key k_1 to k_2 shows that k_2 is either derived from or delivered using k_1 (that is, k_1 *protects* k_2, as knowledge of k_1 is required to obtain k_2). The edge's label is the protocol that is used to derive or deliver the key. The subscript in a key (for example, the subscript X in $pmkmkd_X\{X, T\}$) is used to denote the principal(s) associated with the key. Principals listed in curly brackets are the honest entities that may possess the key. The subscripts are ordered (i.e., $pmk_{X,Y}$ is different from $pmk_{Y,X}$).

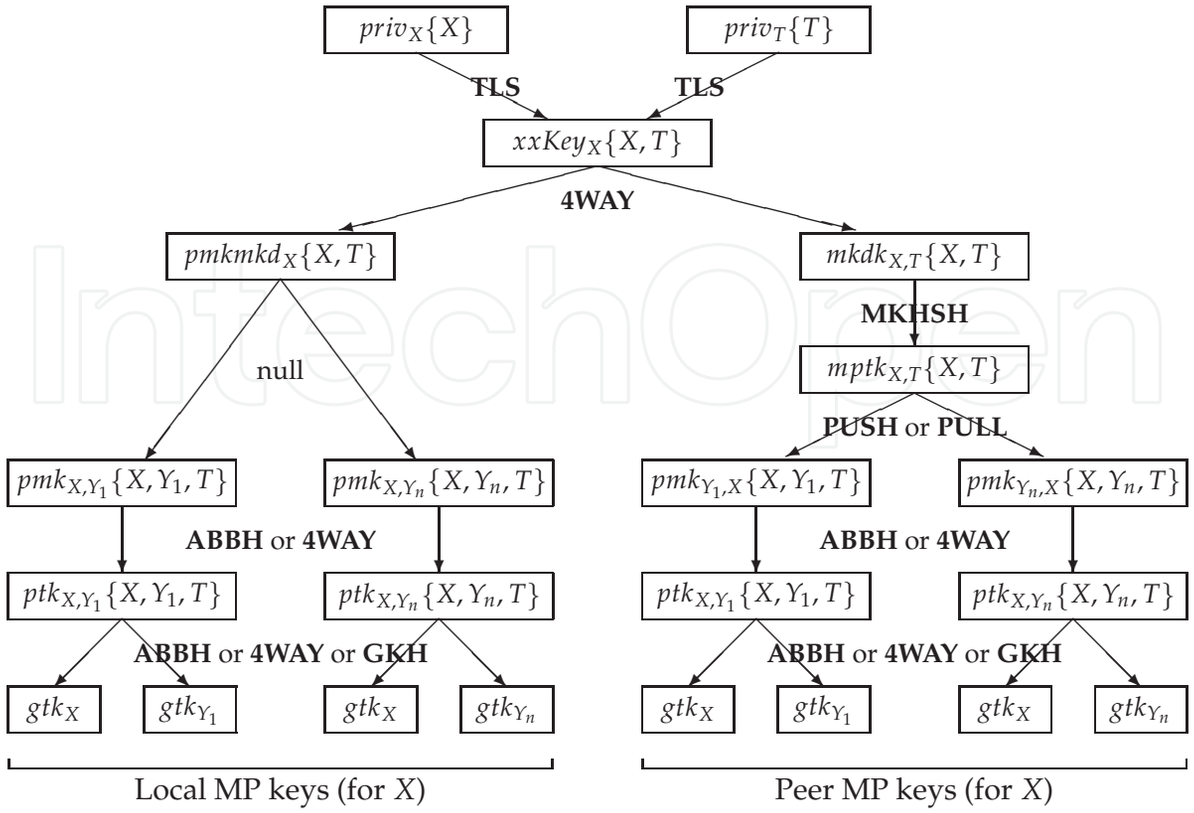

Figure 2. The key hierarchy of the MSA proposal

Key derivation (one-way) functions are utilized rather than key generation for efficiency. The MSA suite's use of key derivation also provides the potential for certain protocols to complete successfully when neither principal has connectivity to the rest of the mesh. This results in a key hierarchy, with each node being associated with several keys. The key hierarchy is an excellent avenue for understanding and summarizing the various protocols in the MSA proposal, and for demonstrating which keys protect other keys [6]. The complete descriptions of the protocols in PCL and prose are in the companion technical report [28].

We start our progression through the key hierarchy and the protocols at the top of Figure 2. Let X be a Candidate MP and let T be the MKD. The *MSA Authentication Protocol* allows X to join the mesh and become an MP, and consists of three stages: *Peer Link Establishment* (**PLE**), **TLS** [19], and a *Four-Way Handshake* (**4WAY**). X either has a shared $xxKey_X$ with T or it shares public key credentials with T. If it shares public credentials with T, then T and X run **TLS** to derive the $xxKey_X$; otherwise, **TLS** is omitted. To derive the $pmkmkd_X$, X needs a nonce; it is delivered to X from T when X runs **4WAY** with an MA (which we denote Y, noting that Y may be T). Subsequently, X can derive the $pmkmkd_X$, $pmk_{X,Z}$ for any Z and the $mkdk_{X,T}$. When X completes **4WAY** with Y, X will have derived $ptk_{X,Y}$, received the gtk_Y from Y, and delivered gtk_X to Y.

At this point, X is a Mesh Point (MP) but is not yet a Mesh Authenticator (MA). To become an MA, X needs to run the *Mesh Key Holder Security Handshake* (**MKHSH**) with T, and derive the $mptk_{X,T}$, which is a session key between X and T. This enables X to run the **PUSH** and

PULL protocols with T. **PUSH** is started by T to tell X to retrieve $pmk_{Z,X}$ for some Z. **PULL** is started by X to request $pmk_{Z,X}$ from T.

The 802.11s task group has expressed interest in developing an Abbreviated Handshake (**ABBH**) [9, 35]. **ABBH** is used by an MP or an MA X to derive $ptk_{X,Z}$, and exchange gtk_X and gtk_Z with another MA or MP Z. Without an **ABBH**, the method of exchanging these credentials is to have the MP or MA run the full MSA Authentication Protocol with the other MA or MP. In this chapter we discuss a candidate **ABBH**, which has been presented to 802.11s [8], and its proof of security and composability with the rest of the MSA architecture. The full **ABBH** is presented in the full paper [28] and comprises two variants. One denoted is **ABBH.INIT** and the other **ABBH.SIMO**, depending on the timing of the first messages. We explore **ABBH.SIMO**, denoted simply **SIMO**, in more depth in Section 3. The **ABBH.INIT** protocol follows more conventional timing rules, but as seen in Figure 2, the **ABBH.SIMO** allows more complication.

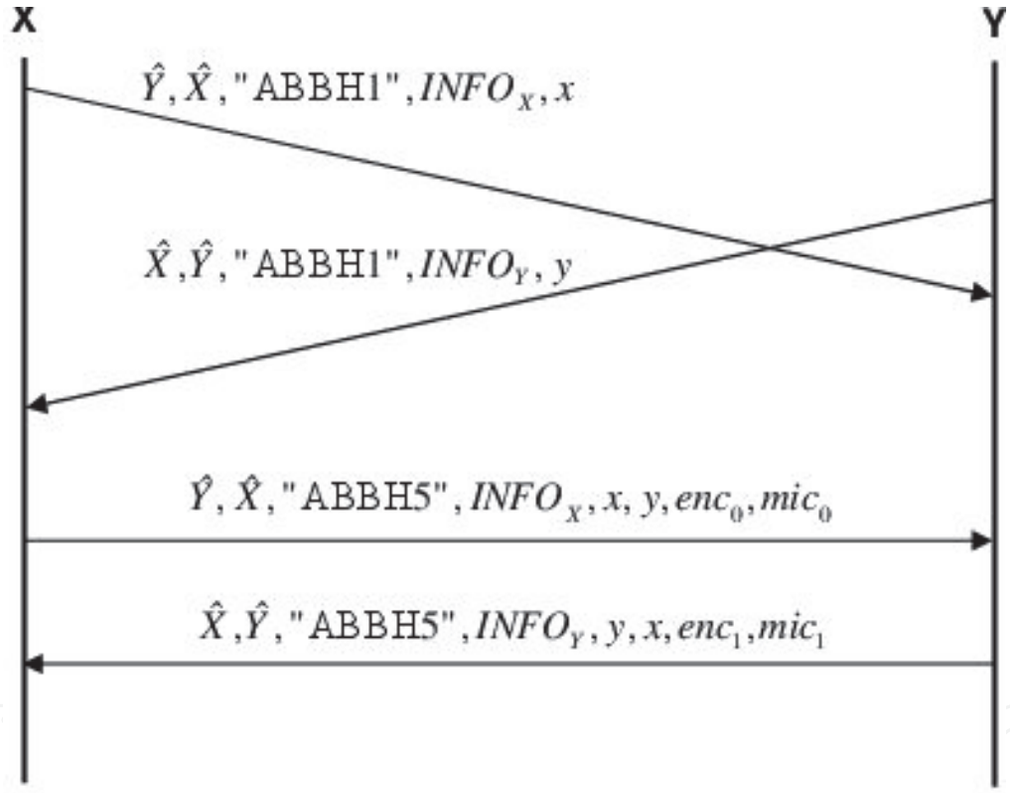

Figure 2. A SIMO Abbreviated Handshake

Additional MSA protocols include the *Group Key Handshake* (**GKH**) and the **DEL**ete protocol (for key management). **GKH** is used by X to update its group key (gtk_X) at Z. The protocol only works with nodes with which X maintains a security association (i.e., shares $ptk_{X,Z}$). **DEL** is started by T to tell X to delete a particular $pmk_{Z,X}$.

We note that each protocol message has a unique identifier. These identifiers must be unique amongst all protocols at a node, so that no other protocol at a node can use those unique identifiers.

Further work has been done on the security of the 802.11s suite of protocols. See [23] for additional details.

3. Additions to PCL and proof methodology

In modeling the MSA protocol suite in PCL, we found a number of situations for which the current language model had no support. We provide a motivating example from MSA and discuss our proposed additions to the language. None of the additions modify the existing language, so all previous proofs and work should not need re-examination. We also broadened the proof methodology slightly.

Many of the additions can be explained by looking at the abbreviated handshake protocol with a simultaneous open (**SIMO**). The purpose of this protocol is to establish a linkage between two nodes which are already part of a mesh. Therefore, the two nodes already have authenticated to the MKD and need only authenticate to each other and establish a fresh, unknown session key. One possible instantiation of this protocol is presented in Figure 3. Values x and y are nonces generated by X and Y respectively. $INFO_X$ contains additional information about node X's configuration. The *enc* values are broadcast keys encrypted with session keys derived from x, y and a shared key. The *mic* values provide integrity and authentication verification. We note that the messages labeled "ABBH5" do not have to occur in the listed order. Node X can receive message 5 before or after it sends its message 5. Note, too, that X might receive its message 5 after it sends its own message 5, even if node Y sends its message 5 before X does.

The thread for this protocol is symmetric (though it does not have to be) and is presented in Figure 4. Some additions to PCL were used in the thread description, which are fully described below. The precondition $\theta_{ABBH,1}$, invariant $\Gamma_{\mathbf{ABBH},1}$ and a sample security goal $\Phi_{\mathbf{SIMO},AUTH}$ are also presented. This protocol is useful in demonstrating the necessity of the additions, as well as providing a sample of how the addition is used in the proof of the MSA proposal.

3.1. Flexible temporal ordering

The temporal ordering of actions in the original PCL definition is too strict for our applications. In the **SIMO** protocol presented in Figure 4, the order of sending and receiving the message labeled "ABBH5" is nondeterministic. Once the initial messages have been exchanged, the final messages could be sent/received in either order. The change to PCL is realized as an addition to the language. The proposed modification does not change any other aspect of PCL; therefore all previous proofs are still valid.

We add an *action group* and redefine the notion of a strand. We define an action group as: (action group) $g ::= (a; \ldots; a)$, where a is an action as defined in [15]. We also redefine a strand as: (strand) $s ::= [g(; \text{or } :) \ldots (; \text{or } :)g]$. Thus a strand is now composed of an arbitrary number of action groups separated by colons or semicolons. The idea behind the action group is that the actions in an action group must be done in the order they appear. However, the action groups within a strand separated by a colon (:) can be done in any order and action groups separated by a semicolon (;) must be done in the order they appear. Note that any strand defined previous to this addition to the language can still be defined exactly the same

$\mathbf{ABBH.SIMO} = (X, \hat{Y}, INFO_X, gtk_X)$
[New x; send $\hat{Y}, \hat{X}$, "ABBH1", $INFO_X, x$;
receive $\hat{X}, \hat{Y}$, "ABBH1", $INFO_Y, y$;
match select$(INFO_X, INFO_Y)/CS, pmkN$;
match retrieve$(pmkN)/pmk$;
match HASH$_{pmk}(x, y)/ptk_{X,Y}$;
(match ENC$_{ptk_{X,Y}}(gtk_X)/enc_0$;
match HASH$_{ptk_{X,Y}}(\hat{Y}, \hat{X}$, "ABBH5", $INFO_X, x, y, enc_0, INFO_Y)/mic_0$;
send $\hat{Y}, \hat{X}$, "ABBH5", $INFO_X, x, y, enc_0, mic_0$) :
(receive $\hat{X}, \hat{Y}$, "ABBH5", $INFO_Y, y, x, enc_1, mic_1$;
match enc_1/ENC$_{ptk_{X,Y}}(gtk_Y)$; match mic_1/HASH$_{ptk_{X,Y}}(\hat{X}, \hat{Y}$, "ABBH5", $INFO_Y, y, x, enc_1, INFO_X))]_X$

$\boldsymbol{\theta}_{\mathbf{ABBH,1}} :=$ Has$(X, pmk_{X,Y}) \wedge$ Has$(Y, pmk_{Y,X}) \wedge$ (Has$(X, mptk_{X,T}) \vee$ Has$(Y, mptk_{X,T})$)

$\boldsymbol{\Gamma}_{\mathbf{ABBH,1}} :=$ Honest$(\hat{X}) \wedge$ Send$(X, m) \wedge$
(Contains$(m,$ Hash$_{ptk}((\text{"ABBH2"}, \hat{Y}, \hat{Z}))) \vee$ Contains$(m,$ Hash$_{ptk}((\text{"ABBH3"}, \hat{Y}, \hat{Z}))) \vee$
Contains$(m,$ Hash$_{ptk}((\text{"ABBH4"}, \hat{Y}, \hat{Z}))) \vee$ Contains$(m,$ Hash$_{ptk}((\text{"ABBH5"}, \hat{Y}, \hat{Z})))) \supset$
$\hat{Z} = \hat{X}$

$\boldsymbol{\Phi}_{\mathbf{SIMO,AUTH}} :=$
KOHonest$(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
(Send$(X,$SIMO1X$) <$ Receive$(Y,$SIMO1X$)) \wedge$ (Send$(Y,$SIMO1Y$) <$ Receive$(X,$SIMO1Y$)) \wedge$
(Send$(Y,$SIMO5Y$) <$ Receive$(X,$SIMO5Y$)) \wedge$ (Send$(X,$SIMO1X$) <$ Receive$(X,$SIMO1Y$) <$
(Send$(X,$SIMO5X$) \wedge$ Receive$(X,$SIMO5Y$)) \wedge$ (Send$(Y,$SIMO1Y$) <$ Receive$(Y,$SIMO1X$) <$
Send$(Y,$SIMO5Y$)$)

Figure 4. Protocol Description of the Abbreviated Handshake Simultaneous Case

way by defining each action group to be one action and by setting all the separators inside a strand to ';'.

We update Axiom **AA4** [15] to reflect this addition to the language. The original version is **AA4**: $\top[a; \ldots; b]_X a < b$. We redefine it to be

AA4: $\top[a : b; \ldots; c : d]_X a \wedge b < c \wedge d$

where a, b, c and d are action groups. Thus nothing about the temporal order of a compared to b or c compared to d is indicated. We also include a new axiom

AA5: $\top[(a; \ldots; b)]_X a < b$

where a and b are actions, to deal with the temporal ordering of action groups. If each action group is exactly one action and only semicolons are used in the new strands our **AA4** becomes exactly the **AA4** previously defined and **AA5** is redundant.

A consequence of the above addition is that protocols whose definition includes ':' have an additional complication in the determination of basic sequences. Recall that a basic sequence is defined as any actions before a receive. With the : notation, two different sets of actions may

occur before a receive, corresponding to the potential temporal ordering of the action groups. Thus we must ensure that invariants and preconditions hold over all possible basic sequence orderings and compositions.

3.1.1. Generalized matching conversations and generalized mutual authentication

The proofs of mutual authentication used in many previous work that use PCL for protocol verification have adopted the notion of matching conversations [2] for the authenticity property. This is natural as these protocols are "turn-by-turn" protocols in which one a participant receives a message and then responds to the message, which is received by the other party who responds to it, and so on. However, some of the peer-to-peer protocols analyzed in this chapter can never correspond to matching conversations as the order in which messages are sent and received is flexible, as a functional requirement. We generalize the properties of matching conversations and define two new notions which we call maximal conversations and generalized matching conversations. We feel these definitions are of general interest beyond this work. Recall the definitions of conversation and matching conversation from [2].

We define the *maximal conversation* for a participant A. We first determine the maximal possible temporal ordering. To do this we consider all legal orderings in an ideal world (one with no adversarial interference) from the view of a participant A in a protocol. Given this maximal temporal ordering, we note the existence of messages for which A can never confirm reception. We take the maximal temporal ordering and remove any send or receive for which A cannot confirm reception in the ideal world – the remaining actions represent the maximal conversation for participant A.

We now define *generalized matching conversations* for a participant A. We say A has generalized matching conversations, if in every run of the system, every action in the maximal conversation for participant A has a corresponding action at participant A (e.g., A does all its actions) and at the appropriate other participant in the system. For two-participant protocols (like all those in this chapter), this means that the maximal conversation for participant A has messages exactly matching the other participant's maximal conversation, with the strictest time ordering possible.

We now define *generalized mutual authentication*. In a world in which an adversary has access to every message and can act on them within the restraints of the proof system (symbolic or computational), generalized mutual authentication means that generalized matching conversations for every participant implies acceptance and acceptance implies generalized matching conversations for every participant. For the purpose of this chapter we wish to keep the definition of generalized mutual authentication general. We explore all these definitions in detail in separate work.

When our definition is applied to a "turn-by-turn" two-party protocol it becomes exactly the definition from [2]. In every other instance our definition requires that the ordering of actions be maximal with respect to what is possible in the ideal world. As this definition imposes maximal temporal ordering on a protocol, this definition is at least sufficient for mutual authentication. Most protocols in the MSA are turn-by-turn and thus the [2] definition

suffices for those cases. The three exceptions are **SIMO** (which is a peer-to-peer protocol and has some timing flexibility), **PLE** (which is not a cryptographic protocol in itself and requires no temporal ordering), and **PUSH** (which is a composition of two protocols).

We note that the generalized matching conversations property encompasses the matching record of runs property [20]. Also, this property guarantees all desired properties from [27] and implies all the possible authentication definitions in [24].

3.1.2. Generalized matching conversations For ABBH.SIMO

We apply the generalized matching conversations definition to **SIMO**, which is a protocol for establishing a secure connection between two nodes already in a mesh.

Let X be the principal from whose view we seek to establish the proof of generalized matching conversation and Y be the other principal. SIMO1X and SIMO5X represent X's messages, and similarly for Y's messages. We need to determine the maximal timing relations in the ideal world (no adversaries) when only **SIMO** is run. X cannot confirm whether Y has received SIMO5X even in the ideal world, because it may be the last message sent. Therefore, SIMO5X is not part of X's maximal conversation. Note that every message must be sent by the correct party before it is received by the other party in an ideal world. Now we simply list what actions must happen after other actions and omit receives after sends that are irrelevant (e.g., $\text{Send}(X, \text{SIMO1X}) < \text{Receive}(Y, \text{SIMO1X})$).

$$\text{Send}(X, \text{SIMO1X}) < \text{Receive}(X, \text{SIMO1Y}) < (\text{Send}(X, \text{SIMO5X}) \wedge \text{Receive}(X, \text{SIMO5Y}))$$
$$\text{Send}(Y, \text{SIMO1Y}) < \text{Receive}(Y, \text{SIMO1X}) < \text{Send}(Y, \text{SIMO5Y})$$

This temporal ordering is inherently maximal for X's view of an arbitrary run of **SIMO**, so it satisfies the definition of generalized matching conversations for X (Y's view is similar). The enforcement of the order of the send messages within a node can be accomplished by waiting for acknowledgements from the MAC layer before proceeding. If X has not sent its message 1, it initiates the corresponding thread for **ABBH.INIT**, not for **ABBH.SIMO**, so this ordering is maximal.

3.2. Modeling information exchange

In the full paper [28], we provide detailed PCL equivalents of the protocols presented in the MSA submissions. Such detailed examinations are necessary to prove protocol correctness. For example, the presence of $INFO_Y$ in mic_0 in **SIMO** (Figure 4) is not intuitively obvious but is essential to the security of the protocol. Modeling the protocol at a higher level of abstraction would have missed this subtle requirement.

Real protocol implementations such as MSA require more than simple key agreement. Additional information must be exchanged and agreed on before secure communication can happen. Examples of information of this type are basic network functions (e.g., bandwidth selections) and security information (e.g., cipher suite selection). The two principals in the protocol must agree in each case, and an attacker must not be able to influence the selection. That is, the agreed-upon value in all protocol runs must match the agreed-upon value in an

ideal world with no adversary. The peer-to-peer nature of certain protocols such as **SIMO** do not allow pre-defined protocol roles of principals to always allow one principal to make this selection and dictate the choice to the other. The two principals must independently choose matching values from two lists.

A new construct, *INFO* is used to capture this. The information principal X contributes to a protocol is $INFO_X$. It contains ordered lists of acceptable selections for one or more fields. The contents of $INFO_X$ may vary for different protocols. In MSA, the **PLE**, **ABBH**, and **MKHSH** protocols require the use of $INFO$.

The Select() action We have added a new action, Select(), to PCL. Two principals X and Y must make identical but independent selections of link and protocol options from exchanged information $INFO_X$ and $INFO_Y$. The Select() action deterministically retrieves information from two lists, independent of the order of the lists (i.e., select$(INFO_X, INFO_Y) =$ select$(INFO_Y, INFO_X)$). During Peer Link Establishment and Abbreviated Handshake protocols, Select() determines the key to be used based on information each principal sends about the keys it has cached and whether it is an MA capable of retrieving the key from the MKD. Thus, the function ensures that a key is either locally cached or may be retrieved from the MKD if the protocol is to continue. The Select() action is used in other contexts as well, such as to determine which principal initiates the 4-way handshake, or which pairwise cipher suite to use after completing a protocol.

This level of detail is necessary to provide protection against downgrade attacks (wherein the attacker chooses the protocol selection suite) and other attacks where public information can be subverted by an attacker to weaken the final strength of the protocol. Additionally, modeling the interactions at the lower level, demonstrated in the description of SIMO in Figure 4, allows us to provide additional guarantees against attack vectors which may be non-obvious to a lay implementer.

Without modeling at this detailed level, a nearly-equivalent SIMO protocol, which only creates a keyed hash across the information it sends, would appear viable and secure. The cryptographic components would be identical. However, without node $\hat{X}$ including $INFO_Y$ in its *mic* (and equivalent for $\hat{Y}$), attack vectors become possible. In particular, the strong requirement that the messages sent exactly match the messages received no longer holds. This loss directly leads to potential manipulation of the $INFO_X$ and $INFO_Y$ fields by an attacker. Not only can an attack user such manipulation to mount a straightforward denial-of-service attack, but the attacker could also compromise the selection of the shared cipher suite, a dangerous form of a downgrade attack.

3.3. Calling one protocol in another

For many of the protocols in MSA, the protocol may instantiate another protocol partway through its run. This second protocol must complete before the first protocol can continue. For example, in a key exchange protocol such as **SIMO**, if both parties that try to establish a session key do not have the other party's pairwise master key cached locally (e.g., X does not have the current $pmk_{Z,X}$), then one of the parties must pause its protocol run and run a key

pull protocol. Furthermore, the second protocol could potentially be triggered in the middle of a basic sequence.

This is new ground for PCL and we have devised a system and proof (see section 4.3) that enables us to frame this complex action in PCL and develop sound proofs. Essentially, we define the inception of functions that may need to run a separate protocol to be basic sequences, as they may involve blocking actions (like receive). Then, before and after these actions we define basic sequence pre- and post-conditions that must be satisfied for a successful completion of the protocols. The idea of basic sequence pre- and post-conditions were give in [26] to enable staged composition and remain standard in the language [15], although they have not been previously used in this way to enable mid-protocol composition.

The retrieve action We have added a new action, retrieve, to PCL. The retrieve action provides the key to the strand, after key selection is complete. The retrieve action takes a key name (*pmkN*, corresponding to a specific *pmk*) as its input. If the principal that executes the function does not have the key locally cached on disk, but is an MA (and has a connection with the MKD), retrieve initiates the **PULL** protocol. If the key is not on disk and the principal is not an MA, retrieve fails and the protocol that called it aborts. As the retrieve action may or may not perform a key pull, we create a break in the basic sequence directly before and directly after the retrieve.

The retrieve action has inherent pre- and post-conditions as it is a series of one or more basic sequences (e.g., a protocol). As a precondition, retrieve must have the *pmk* cached locally or it must have the $mptk_{X,T}$. Thus the precondition is $\text{Has}(X, mptk_{X,T}) \vee \text{Has}(X, pmk)$ where *pmk* matches the input *pmkN*. The postcondition is simply $\text{Has}(X, pmk)$. The retrieve function itself has security requirements only if the principal must perform a key pull, when it inherits the requirements of the **PULL** protocol.

In Figure 4, retrieve is used to get the selected *pmk*. This provides two potential paths of execution through the protocol, one which runs a key pull mid-protocol and one which simply fetches some stored memory (equivalent to a match action).

4. Overview of the proof

In this section, we provide an overview of our proof efforts by highlighting three aspects of it. In Section 4.1 we discuss our approach to proving key secrecy in the MSA proposal. In Section 4.2 we present additional security goals and a theorem that culminates our proof efforts. Finally, in Section 4.3, we discuss our approach to protocol composition.

4.1. Key secrecy in MSA

Key secrecy is a critical security requirement. Some previous work [26] has proven key secrecy as a protocol postcondition. We show that proving key secrecy as a postcondition is insufficient by providing an example of a protocol which has key secrecy as a postcondition (i.e., upon protocol completion, key secrecy can be proven) but is insecure, because key secrecy can be lost. The Insecure Key Transfer Protocol in Figure 5 illustrates this point. If we assume protocol completion from the point of view of RESP we can prove that the secret key is

Inputs and Parties:
- Two parties: INIT and RESP.
- Shared input: confirmation key (ck).
- INIT private input: INIT public key (PK_{INIT}).
- RESP private input: secret key (sk).
- Goal: Has$(Z, sk) \supset Z = \text{INIT} \vee Z = \text{RESP}$

Insecure Key Transfer Protocol:

1. INIT sends PK_{INIT} to RESP.
2. RESP receives PK_{INIT}; encrypts sk under PK_{INIT}, computes the keyed hash of the encryption with key ck; and sends $(\{sk\}_{PK_{INIT}}, HASH_{ck}(\{sk\}_{PK_{INIT}}))$ to INIT.
3. INIT receives $(\{sk\}_{PK_{INIT}}, HASH_{ck}(\{sk\}_{PK_{INIT}}))$, verifies the keyed hash; decrypts sk; computes the keyed hash of sk and PK_{INIT} with the ck and sends $HASH_{ck}(sk, PK_{INIT})$ to RESP.
4. RESP verifies the signature.

Figure 5. Insecure Key Transfer

distributed correctly, as the validity of INIT's public key is established once RESP receives the third message. However RESP uses the public key in the second message before the validity of the public key can be established. Thus if the protocol aborts after the RESP sends the second message, it may be the case that the public key sent in message 1 is an adversary's public key. It is therefore possible for the adversary to intercept the secret key. While this protocol is contrived, in larger protocols with complex security goals, it may be the case that a subtle insecurity such as this goes unnoticed. Thus, for certain assertions related to secrecy, we advocate showing that they hold at every critical point in a protocol.

We prove the security of MSA's key hierarchy using the work of Roy et al. [31–33]. We present the key secrecy postconditions relevant to the MSA key hierarchy in Figure 6. We prove that these conditions hold at every point during any protocol execution of MSA, as long as the indicated principals are honest. We also claim the new axiom

SAF5: SafeMsg$(HASH_s(M), s, \mathcal{K})$

Informally, this states that a keyed hash of a message does not reveal the key.

It seems natural to prove key secrecy in an inductive manner, locally for each thread and role. But that's not sufficient in the proposed MSA system, because key information is not generally held purely locally and other nodes with the information may be abusing it. Key secrecy must be maintained locally, of course, but it requires a global proof.

The techniques of [32] would note the deficiencies of the protocol in Figure 5, because the first message sent by RESP could not be proven to be a SafeMsg, inductively. We utilize this notion and extend the SafeNet concept across the entire suite of MSA protocols, to show that no protocol violates the key secrecy of any other protocol. Doing this step before examining other desired protocol security goals (postconditions) provides for more elegance and correctness in the other proofs.

$\boldsymbol{\theta}_{TLS,SI,1} :=$
$\text{KOHonest}(xxKey_X, \{priv_X, priv_T, xxKey_X\}) \supset \text{Has}(Z, xxKey_X) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{T}$

$\boldsymbol{\theta}_{4WAY,SI,1} :=$
$\text{KOHonest}(pmkmkd_X, \{xxKey_X\}) \supset \text{Has}(Z, pmkmkd_X) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{T}$

$\boldsymbol{\theta}_{4WAY,SI,2} :=$
$\text{KOHonest}(mkdk_{X,T}, \{xxKey_X\}) \supset \text{Has}(Z, mkdk_{X,T}) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{T}$

$\boldsymbol{\theta}_{PPD,SI,1}, \theta_{MKHSH,SI,1} :=$
$\text{KOHonest}(mptk_{X,T}, \{mkdk_{X,T}\}) \supset \text{Has}(Z, mptk_{X,T}) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{T}$

$\boldsymbol{\theta}_{PPD,SI,2} :=$
$\text{KOHonest}(pmk_{X,Y}, \{pmkmkd_X, mptk_{Y,T}\}) \supset \text{Has}(Z, pmk_{X,Y}) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y} \vee \hat{Z} = \hat{T}$

$\boldsymbol{\theta}_{ABBH,SI,1}, \theta_{4WAY,SI,3}, \theta_{GKH,SI,1} :=$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset \text{Has}(Z, ptk_{X,Y}) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y} \vee \hat{Z} = \hat{T}$

$\boldsymbol{\theta}_{ABBH,SI,2}, \theta_{4WAY,SI,4}, \theta_{GKH,SI,2} :=$
$\text{KOHonest}(gtk_X, \{ptk_{X,Y_1}, \ldots ptk_{X,Y_n}\}) \supset \text{Has}(Z, gtk_X) \supset \text{Has}(Z, ptk_{X,Y_i})$

Figure 6. MSA Key Secrecy Conditions

We introduce new notation $\vdash_U$. The meaning of $P \vdash_U \theta$ is that postcondition θ must hold at every intermediate point of the relevant protocols in program set P. That is, if the terms in θ are defined and bound at the end of a basic sequence in P, then θ holds.

Theorem 1. *Let MSA represent all the protocols in the Mesh Security Architecture and $\theta_{SI,ALL}$ represent all of the key secrecy conditions in Figure 6. Then $\theta_{SI,ALL}$ are satisfied by MSA. That is,*

$$MSA \vdash_U \theta_{SI,ALL}$$

Proof sketch: This theorem is proven in two steps. The first step is a massive induction over all the basic sequences of all the protocols that could be run by any participant in a mesh. This induction guarantees that all messages sent are "safe," in that critical information is protected by keys. In the key secrecy goals of Figure 6, the critical information protected is another key, lower in the hierarchy. From this, we argue the invariant nature of multiple SafeNet axioms over the entire MSA protocol suite, limiting various goals to the protocols where the terms are instantiated/defined. Then, we use the **POS** and **POSL** axioms [32] to state who can potentially have access to other keys. By proceeding in this way through the entire key hierarchy, we establish all the necessary key secrecy goals, at any point in a run where the keys may be defined. The full proof is generally unenlightening and we do not provide it. We stress that this proof does not depend on any of the analysis done in proceeding sections. It is simply induction over all basic sequences and application of secrecy axioms. This proves key secrecy is maintained by all protocols in MSA.

This theorem guarantees that the parties listed in Figure 6 are the only principals with those keys. This proves that an attacker could not learn any key in the entire hierarchy from the MSA protocols.

4.2. Goals and correctness result

We present important security postconditions (goals) below. For each goal, we point out the kinds of protocols to which it applies. Goals are customized for each protocol; formal instances of each kind of goal we discuss below are in Appendix 6. We keep our discussions in this section more informal for clarity.

AUTH: Authentication as realized by the generalized matching conversations property (see Section 3.1.1). In practice, this confirms peer liveness and peer possession of a particular key. This goal applies to all protocols in the MSA proposal. This goal is expressed as:

$\Phi_{AUTH} := \exists Y.ActionsInOrder(\text{Send}(X, \hat{X}, \hat{Y}, msg_1), \text{Receive}(Y, \hat{X}, \hat{Y}, msg_1),$
$\text{Send}(Y, \hat{Y}, \hat{X}, msg_2), \cdots, \text{Receive}(X, \hat{Y}, \hat{X}, msg_n))$

KF: Key freshness as realized by a freshly-generated nonce from each party as a term in the agreed-upon key. This goal applies only to protocols which create a joint (session) key.

$\Phi_{KF} := \text{KOHonest}(k, \mathcal{K}) \supset (\text{New } (\hat{X}, x) \wedge x \subseteq k \wedge \text{New } (\hat{Y}, y) \wedge y \subseteq k) \wedge$
$\text{FirstSend}(X, x, \hat{X}, x, m) \wedge \text{FirstSend}(Y, y, \hat{Y}, y, m)$

KA: Key agreement as realized by the Has predicate. This ensures that both parties have the session key. This goal applies to only those protocols that establish a session key.

$\Phi_{KA} := \text{KOHonest}(k, \mathcal{K}) \supset \text{Has}(X, k) \wedge \text{Has}(Y, k)$

KD: Transfer of secret information (key delivery) as realized by the key secrecy goals and the Has predicate. This applies only to those protocols which transmit keys (either a group transfer key (*gtk*) or a pairwise master key (*pmk*)).

$\Phi_{KD} := \text{KOHonest}(k, \mathcal{K}) \supset \text{Has}(X, k) \wedge \text{Has}(Y, k)$

INFO: Authentic exchange of non-secret information and authenticated selection of sub-elements as realized in detailed protocol description and validated return information. This applies only to protocols which must exchange non-security information and agree on parameters.

$\Phi_{INFO} := \text{KOHonest}(k, \mathcal{K}) \supset \text{Select}(INFO_X, INFO_Y) = CS, pmkN \wedge$
$\text{Has}(X, CS, pmkN) \wedge \text{Has}(Y, CS, pmkN)$

Our goals are extensions and clarifications of the goals adopted by He et al. [26], which in turn are adapted from the list of desired security properties for 802.11i [1]. No security goals have been explicitly specified for the general 802.11s protocol suite; however, we anticipate that the security goals for 802.11i are meaningful for 802.11s as well, provided they are adapted appropriately. Furthermore, we feel that the goals we present above have intrinsic intuitive appeal. We recommend that these goals, in addition to the key secrecy goals discussed in Section 4.1, be formally adopted by the 802.11s task group.

In the following Theorem, we introduce some notation ($\rightsquigarrow$) for ease of exposition. $TLS \rightsquigarrow AUTH, KD$ means $\Gamma_{\mathbf{TLS},\{1,2,\}} \vdash \theta_{\mathbf{TLS}}[\mathbf{TLS} : \mathbf{CLNT}]_X \Phi_{\mathbf{TLS},\{AUTH,KD\},CLNT}$ and the corresponding goal for the other node, namely $\Gamma_{\mathbf{TLS},\{1,2,\}} \vdash \theta_{\mathbf{TLS}}[\mathbf{TLS} : \mathbf{SRVR}]_X \Phi_{\mathbf{TLS},\{AUTH,KD\},SRVR}$. These state that, with the proper invariants, the protocol from each perspective provably satisfies the security goals *AUTH* and *KD* ,

particular to **TLS**. Similar expansions have been made for each protocol and the details are in our full paper [28].

With the changes that we discuss in Section 5, we are able to prove the component-wise correctness of each of the protocols of the MSA proposal.

Theorem 2. *The following are true, with the notation described above.*

(i) $TLS \rightsquigarrow AUTH, KD$
(ii) $4WAY \rightsquigarrow AUTH, KF, KA, KD, INFO$
(iii) $MKHSH \rightsquigarrow AUTH, KF, KA, KD, INFO$
(iv) $GKH \rightsquigarrow AUTH, KD$
(v) $PUSH \rightsquigarrow AUTH, KD$
(vi) $PULL \rightsquigarrow AUTH, KD$
(vii) $DEL \rightsquigarrow AUTH$
(viii) $ABBH \rightsquigarrow AUTH, KF, KA, KD, INFO$

This theorem was one of the major driving forces behind the work. It asserts that the full protocol suite in the MSA proposal, with a rather complex key hierarchy, is secure. Each protocol achieves the maximal security goals for its type. Appendix 6 contains a proof of part of (viii) and provides a feel for the proof methodology. The proof of Theorem 2 depends on the PCL additions of Section 3.

4.3. Composition

The MSA architecture allows for significant variation in how protocols compose together [4]. Once an established state is reached, many protocols (which may have been run previously to reach the established state) may be chosen. Reaching an established state may take a variety of paths, depending on the authentication mechanism (TLS or pre-shared key) used. Error-handling strategies will cause protocols to restart, or, potentially, different protocols to be run. This introduces a complex state diagram and complexities of composition.

While *staged composition* proofs have been presented previously [26, 31], the presentation in each case has differed. Staged composition allows arbitrary back arrows and paths through possible protocol execution paths. This allows for protocol restarts, lost connections, and other real-world considerations about the order in which protocols are run. We provide a slightly different presentation of similar ideas in Section 4.3.1. Readers primarily interested in the proof of MSA may skip this section and proceed to Section 4.3.2 where the overall MSA security theorem is presented.

4.3.1. Consistent composition

The concept of branches within protocols or between protocols has not been explicitly mentioned in previous PCL composition theorems. We require this functionality, to denote how a particular staging can be accomplished within the MSA framework. One of our motivations is to allow such possibilities as are represented in Figure 7. After basic sequence A, either sequence B or sequence C may follow. Sequence D follows C and both B and D lead to E. The consistent composition theorem provides the requirements under which such

branches will still compose. This also provides for all manner of if/then functionality within PCL, if it can be properly created in semantics and the various results of the if/then statement are properly modeled in terms of basic sequence breaks. We believe this fills a gap in the span of PCL.

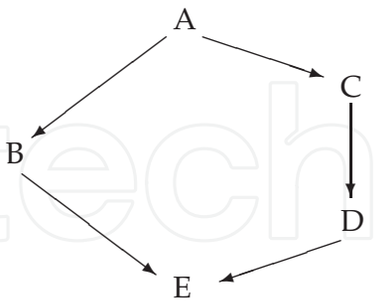

Figure 7. Branches in PCL

We utilize the definitions of role-prefix, staged role, and staged composition from [26], suitably augmented for the retrieve action. Informally, role-prefix defines which sets of basic sequences can lead to a particular basic sequence. A staged role is a particular, legitimate sequence of basic sequences leading to a particular execution point. And staged composition allows for sequential implementation with arbitrary returns to earlier execution points, with the branching of retrieve potentially following different paths on each iteration.

We use θ_{P_i} to indicate the precondition for basic sequence P_i. Additionally, to add simplicity to our exposition, we use Γ to denote the conjunction of all invariants within a staged composition of protocols. That is, Γ is the totality of all the invariants from each of the protocols Q_i that make up a composition of protocols Q. This allows us to state the following theorem succinctly.

Theorem 3. *Let Q be a staged composition of protocols $Q_1, Q_2, \ldots, Q_n$ and $P; P_i \in SComp(\langle Q_1, Q_2, \ldots, Q_n \rangle)$ and $P_i \in Q_i$. Then $Q \vdash \theta_{P_0}[P; P_i]_X \theta_{P_{i+1}}$, if for all $RComp(\langle P_1, P_2, \ldots, P_n \rangle) \in Q$, all of the following hold:*

(Invariants)
(i) $\forall i. \forall S \in BS(Q_i). \vdash \theta_{P_i} \wedge \Gamma[S]_X \Gamma$
(Preconditions)
(ii) $Q_1 \otimes Q_2 \otimes \cdots \otimes Q_n \vdash \forall i. \theta_{P_i}[P_i]_X \theta_{P_{i+1}}$
(iii) $\forall i. \forall S \in \bigcup_{j \geq i} BS(P_j). \theta_{P_i}[S]_X \theta_{P_i}$
(iv) $Q_1 \otimes Q_2 \otimes \cdots \otimes Q_n \vdash Start(X) \supset \theta_{P_1}$

Theorem 3 states the conditions under which a particular run through a set of actions reaches its ultimate goal. The "Invariants" condition requires that no basic sequence violate any invariant of any basic sequence, with its proper preconditions, and invariants holding before the basic sequence. The "Preconditions" conditions require that each basic sequence's postconditions imply the next basic sequence's preconditions, that no basic sequence ever violate any preceding basic sequence's preconditions, and that the start state is valid.

We point out that Theorem 3 is dependent on basic sequences as its fundamental building block. The protocols themselves, while useful distinctions in understanding and modeling

the system, are not critical. In particular, the Q_i's could be single basic sequences and the entire theorem still holds. This allows us to model at the level of basic sequences. This level of granularity has been suggested before [26], but we make it explicit.

This allows, for example, the behavior of the retrieve action that we discuss Section 3.3. Retrieve allows two different paths through a larger staged composition. In one path, a locally stored value is returned. In the other path, an entire protocol is run. As protocols compose consistently at the granularity of basic sequences in the initial protocol, retrieve fundamentally denotes alternate methods of staging the composition. In all protocols that use retrieve, the invariants and various preconditions in the protocol are proven against all possible stagings of the retrieve action.

4.3.2. Composition in MSA

We wish to apply Theorem 3 to the protocols of the MSA proposal. We view the protocols of staged composition as the protocols given previously. As mentioned, we consider arbitrary breaks at the basic sequence level, for mid-protocol composition as well as overall composition. We need to prove that all protocols within MSA (comprising **PLE**, **TLS**, **4WAY**, **MKHSH**, **GKH**, **PULL**, **PUSH**, **DEL**, and **ABBH**, (both **ABBH.INIT** and **ABBH.SIMO**) satisfy the necessary conditions for composition.

Theorem 4. *Let Q be a specific composition of protocols from MSA and $RComp(\langle P_1, P_2, \ldots, P_n\rangle) \in Q$ and $\Gamma = \Gamma_{TLS,\{1,2\}} \wedge \Gamma_{4WAY,1} \wedge \Gamma_{MKHSH,1} \wedge \Gamma_{GKH,\{1,2\}} \wedge \Gamma_{PPD,\{1,2\}} \wedge \Gamma_{ABBH,1}$. Then:*

(i) $\forall i. \forall S \in BS(Q_i). \vdash \theta_{P_i} \wedge \Gamma[S]_X \Gamma$
(ii) $\Phi_{4WAY} \vdash \theta_{MKHSH} \wedge \theta_{GKH}$
$\Phi_{MKHSH} \vdash \theta_{PUSH} \wedge \theta_{PULL} \wedge \theta_{DEL}$
$\Phi_{MKHSH} \vdash \theta_{ABBH}$
$\Phi_{ABBH} \vdash \theta_{GKH}$
(iii) $\forall i. \forall S \in \bigcup_{j \geq i} BS(P_j). \theta_{P_i}[S]_X \theta_{P_i}$
(iv) θ_{P_1}

Proving that all the protocols securely compose is a lengthy induction process, which we omit owing to space constraints. We briefly discuss the meaning of the various subpoints. No portion of any protocol in MSA violates the invariants (i) or changes the preconditions (iii) of any MSA protocol. All nodes in a mesh start with the correct information, by assumption (iv). Point (ii) gives the protocols which guarantee certain subsequent protocols can be completed with other legitimate nodes, via pre and post condition matching.

This theorem states that, given any MSA protocol, if the MKD and the players in the protocol are honest (that is, they conform to the protocol specification), then the security of that protocol is ensured, regardless of what other protocols may be running in the system. By extension, a mesh of honest nodes guarantees our security goals; the Mesh Security Architecture is sound.

5. Modifications to MSA

Our analysis of the protocols and key hierarchy of the MSA proposal indicate that it was largely well-designed. We have two recommendations that have been incorporated into the

802.11s draft (as of March 2008) and are necessary for Theorem 2 to hold; otherwise, the protocols are insecure.

5.1. Include mesh nonce in 4WAY

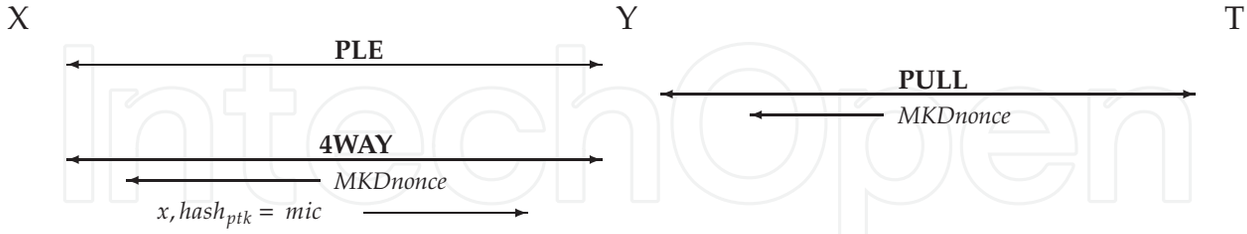

Figure 8. MSA Authentication. The text above a double-headed arrow (e.g., **4WAY**) is a protocol, and text below (e.g., *MKDnonce*) is some data that is sent as part of the protocol.

The draft specification of **4WAY** during MSA authentication does not properly provide key freshness. The proposal has the key generation nonce (*MKDnonce*) provided by the MKD used both to derive the *pmkmkd* (see Figure 2) and as the nonce to derive the session key (*ptk*). This is shown in Figure 8.

This enables an attack that proceeds as follows. At some point, a legitimate node (X) disconnects from the mesh. The attacker then starts MSA authentication with the same MA with which X connected before. The rogue node does **PLE** (claiming to be X) and then continues to the **4WAY** protocol, where the *MKDnonce* is the same. The rogue node re-uses the nonce X used and now the same *ptk* is derived. The attacker may then utilize some information recorded from the legitimate conversation or otherwise abuse the mesh. If **TLS** is used and not a pre-shared key, then this particular attack no longer works.

The solution adopted by 802.11s is to modify the derivation of *pmkmkd* so that it does not require an *MKDnonce*, so that 4WAY is responsible only for transporting nonces used to derive the *ptk*. The *MKDnonce* was removed as it did not provide significant benefit to the architecture and was not required for our key freshness goal. At this point, key freshness (for the *ptk*) can be proven and the attack outlined above is thwarted.

5.2. Include MAC address in the Group Key Handshake protocol (GKH)

In the original proposal, **GKH** does not provide authentication. Recall from section 4.2 that the *AUTH* goal requires matching conversations between two different nodes. In the proof of this property, it became apparent to us that the proposal did not protect against a reflection attack.

MAC addresses were contained in the **GKH** message headers (to facilitate transport of the messages) but were not incorporated in the calculation of the message integrity code (mic) included in each **GKH** message. **GKH** messages are protected using the *ptk*, a pairwise key known only to two parties, but either party may initiate the **GKH**. Owing to this symmetry, the first **GKH** message could be reflected back to the sender, and would be accepted as valid because of the presence of a valid mic. This reflection attack could change the security state at the MP that sends the first message of **GKH**, such as by installing a stale *gtk* or installing its own *gtk* as if it were its neighbor's *gtk*.

The proposed modification includes the explicit identification of sender and receiver in the protected portion of the message, and updates the processing of **GKH** messages to verify this information upon reception. This prevents the replay attack because the sender and receiver MAC addresses would not match if a reflection attack is attempted.

6. Conclusions and future work

We have proven the security of the MSA, under standard assumptions. We provided and justified a few recommendations that were incorporated (as of March 2008) into the 802.11s draft standard, which is still being developed. We also hope that providing a security proof during the design and review process will lead to additional efforts in that regard. We feel that protocol design is important and an analysis of a system should be done before implementation, not after. In the process of this analysis, we made a number of contributions to PCL.

The most important contribution, from our perspective, is the ability to handle simultaneity, with the introduction of action groups and associated axioms and proof techniques. The definition of generalized authentication using generalized matching conversations is also required for simultaneous peer-to-peer protocols. The select and retrieve actions were also designed to extend naturally to examinations of other architectures.

This chapter also takes a deeper dive into the details of the protocols than is often undertaken. While examining only the security components (nonces, keys, etc.) simplifies analysis, it also leaves a gap. Our experience leads us to believe that gaps in analysis are often dangerous, as they lead to assumptions about security, implementation difficulties, and unforeseen attack vectors. Some level of abstraction is necessary, but adding a model for authenticated information exchange is critical for many applications.

This chapter opens opportunities for applying PCL to other peer-to-peer protocols, where ordering may not be as strict as in server-client models. Other protocol systems, particularly those on standard-track, would be natural candidates for additional analysis.

Finally, we provide a new, more general composition theorem, which explicitly allows for mid-protocol composition and branching. As it is not unusual for protocols to intermix, explicitly allowing multiple potential paths through basic sequences is important, and should naturally extend to other situations.

Acknowledgements

The authors would like to thank Anupam Datta and Arnab Roy for their helpful comments and suggestions.

Author details

Doug Kuhlman
Motorola Mobility 600 US Hwy 45 E1-40Y Libertyville, IL 60048

Ryan Moriarty[1]
Computer Science Department
University of California at Los Angeles

Tony Braskich, Steve Emeott and Mahesh Tripunitara
Motorola, Schaumburg, IL 60196

Appendix A: SIMO Security

A.1. Security goals for SIMO

Here we detail the five PCL security goals for the SIMO abbreviated handshake protocol (the *AUTH* goal was presented in Figure 4 and is repeated here). These directly correspond to the security goals detailed in Section 4.2. Unlike the generic goals presented there, these are the specific instances for the SIMO protocol.

Goals SIMO:

$\mathbf{\Phi}_{SIMO,AUTH} :=$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$(\text{Send}(X, \text{SIMO1X}) < \text{Receive}(Y, \text{SIMO1X})) \wedge (\text{Send}(Y, \text{SIMO1Y}) < \text{Receive}(X, \text{SIMO1Y})) \wedge$
$(\text{Send}(Y, \text{SIMO5Y}) < \text{Receive}(X, \text{SIMO5Y})) \wedge (\text{Send}(X, \text{SIMO1X}) < \text{Receive}(X, \text{SIMO1Y}) <$
$(\text{Send}(X, \text{SIMO5X}) \wedge \text{Receive}(X, \text{SIMO5Y})) \wedge (\text{Send}(Y, \text{SIMO1Y}) < \text{Receive}(Y, \text{SIMO1X}) <$
$\text{Send}(Y, \text{SIMO5Y}))$

$\mathbf{\Phi}_{SIMO,KF} :=$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$(\text{New } (\hat{X}, x) \wedge x \subseteq ptk_{X,Y} \wedge \text{New } (\hat{Y}, y) \wedge y \subseteq ptk_{X,Y}) \wedge$
$\text{FirstSend}(X, x, \hat{X}, x, \text{SIMO1X}) \wedge \text{FirstSend}(Y, y, \hat{Y}, y, \text{SIMO1Y})$

$\mathbf{\Phi}_{SIMO,KA} :=$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Has}(X, ptk_{X,Y}) \wedge \text{Has}(Y, ptk_{X,Y})$

$\mathbf{\Phi}_{SIMO,KD} :=$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \wedge$
$\text{Receive}(Y, SIMOX5) \supset \text{Has}(X, gtk_Y) \wedge \text{Has}(Y, gtk_X)$

$\mathbf{\Phi}_{SIMO,INFO} :=$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$SELECT(INFO_X, INFO_Y) = CS, pmkN \wedge \text{Has}(X, CS, pmkN) \wedge \text{Has}(Y, CS, pmkN)$

A.2. Proof security goals, SIMO

Proof sketch generalized authentication, SIMO

We only need to show the proof from a single point of view as the roles are symmetric. Let principal X be the principal from whose view we are establishing the proof from and let Y be the other principal. As the proof assumes X has completed the protocol successfully, we know that SIMO1X was sent before SIMO5X and SIMO1Y was received before SIMO5Y. Thus to complete the proof we must show that Y sent exactly SIMO1Y before SIMO5Y and received

[1] Funded by Motorola while working on this project

exactly SIMO1X before sending SIMO5Y. We can determine the MIC in SIMO5Y could have only been sent by Y if X, Y and T are honest. Since all the variables used in the protocol are contained in the MIC of SIMO5Y, we know that X and Y share identical variables. Now using the honesty of Y we are sure that Y sent SIMO1Y and received SIMO1X before sending SIMO5Y and that it was sent exactly as X received it. Again if Y is honest since X and Y share variables, then Y must have received SIMO1X exactly as X had sent it. This gives us generalized authentication.

Generalized Authentication:
$\mathbf{AA1}, \mathbf{ARP}, \mathbf{AA4}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{Send}(X, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x) < \text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y) <$
$(\text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1) \wedge \text{Send}(X, \hat{Y}, \hat{X}, \text{“ABBH5”}, INFO_X, x, y, enc_0, mic_0))$ (1)

$\mathbf{ARP}, \mathbf{HASH3}', \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X \text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1) \supset$
$\exists Z.\text{Computes}(Z, \text{HASH}_{ptk_{X,Y}}(\hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, INFO_X)) \wedge$
$\text{Sends}(Z, \text{HASH}_{ptk_{X,Y}}(\hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, INFO_X)) <$
$\text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1)$ (2)

$\boldsymbol{\theta}_{ABBH,SI,1}, HASH1$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Computes}(Z, \text{HASH}_{ptk_{X,Y}}(\hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, INFO_X)) \supset$
$\text{Has}(Z, ptk_{X,Y}) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y} \vee \hat{Z} = \hat{T}$ (3)

$2, 3, \mathbf{AA1}, \boldsymbol{\Gamma}_{\mathbf{ABBH,1}}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Send}(Z, \text{HASH}_{ptk_{X,Y}}(\hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, INFO_X)) \supset \hat{Z} = \hat{Y}$ (4)

$2, 4, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Computes}(Y, \text{HASH}_{ptk_{X,Y}}(\hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, INFO_X)) \wedge$
$\text{Send}(Y, \text{HASH}_{ptk_{X,Y}}(\hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, INFO_X))$ (5)

$5, \mathbf{HASH1}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{Has}(Y, ptk_{X,Y}) \wedge \text{Has}(Y, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1)$ (6)

$5, 6, \phi_{\mathbf{HONESTY}}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y) < \text{Receive}(Y, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x) <$
$\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1)$ (7)

$2,7,\boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1) < \text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1)$ (8)

$\mathbf{FS1}, \mathbf{AN3}, \theta_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{FirstSend}(X, x, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x)$ (9)

$9, \mathbf{FS2}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{Send}(X, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x) < \text{Receive}(Y, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x)$ (10)

$\mathbf{FS1}, \mathbf{AN3}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{Honest}(\hat{Y}) \supset \text{FirstSend}(Y, y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y)$ (11)

$7, 11, \mathbf{FS2}, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y) < \text{Receive}(Y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y)$ (12)

$1, 7, 8, 10, 12, \boldsymbol{\theta}_{\mathbf{ABBH,1}}$
$[\mathbf{ABBH} : \mathbf{SIMO}]_X$
$\text{KOHonest}(ptk_{X,Y}, \{pmk_{X,Y}, pmk_{Y,X}\}) \supset$
$(\text{Send}(X, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x) < \text{Receive}(Y, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x)) \wedge$
$(\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y) < \text{Receive}(Y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y)) \wedge$
$(\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1) < \text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1)) \wedge$
$(\text{Send}(X, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x) < \text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y) <$
$(\text{Receive}(X, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1) \wedge \text{Send}(X, \hat{Y}, \hat{X}, \text{“ABBH5”}, INFO_X, x, y, enc_0, mic_0))) \wedge$
$(\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH1”}, INFO_Y, y) < \text{Receive}(Y, \hat{Y}, \hat{X}, \text{“ABBH1”}, INFO_X, x) <$
$\text{Send}(Y, \hat{X}, \hat{Y}, \text{“ABBH5”}, INFO_Y, y, x, enc_1, mic_1))$ (13)

7. References

[1] 802.11-2007, I. S. [2007]. Local and metropolitan area networks – specific requirements – part 11: Wireless LAN medium access control and physical layer specifications.

[2] Bellare, M. & Rogaway, P. [1993]. Entity authentication and key distribution., *CRYPTO*, pp. 232–249.

[3] Blunk, L., Vollbrecht, J., Aboba, B., Carlson, J. & Levkowetz, H. [2004]. Extensible authentication protocol (EAP), http://tools.ietf.org/html/draft-ietf-eap-rfc2284bis-09.

[4] Braskich, T. & Emeott, S. [2007a]. Clarification and update of MSA overview and MKD functionality text, https://mentor.ieee.org/802.11/documents doc 11-07/2119r1.

[5] Braskich, T. & Emeott, S. [2007b]. Initial MSA comment resolution, https://mentor.ieee.org/802.11/documents doc 11-07/0564r2.
[6] Braskich, T. & Emeott, S. [2007c]. Key distribution for MSA comment resolution, https://mentor.ieee.org/802.11/documents doc 11-07/0618r0.
[7] Braskich, T. & Emeott, S. [2007d]. Mesh key holder protocol improvements, https://mentor.ieee.org/802.11/documents doc 11-07/1987r1.
[8] Braskich, T., Emeott, S., Barker, C. & Strutt, G. [2007]. An abbreviated handshake with sequential and simultaneous forms, https://mentor.ieee.org/802.11/documents doc 11-07/2535r0.
[9] Braskich, T., Emeott, S. & Kuhlman, D. [2007]. Security requirements for an abbreviated MSA handshake, https://mentor.ieee.org/802.11/documents doc 11-07/0770r0.
[10] Cortier, V. [2012]. Secure composition of protocols, *in* S. MÃűdersheim & C. Palamidessi (eds), *Theory of Security and Applications*, Vol. 6993 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 29–32.
[11] Cremers, C. [2008]. On the protocol composition logic PCL, *Proc. of the Third ACM Symposium on Information, Computer & Communication Security (ASIACCS '08)*, ACM Press, Tokyo. To appear.
[12] Datta, A., Derek, A., J.C.Mitchell & B.Warinschi [2006]. Computationally sound compositional logic for key exchange protocols, *Proceedings of 19th IEEE Computer Security Foundations Workshop*, pp. 321–334.
[13] Datta, A., Derek, A., Mitchell, J. C. & Pavlovic, D. [2003a]. Secure protocol composition., *FMSE*, pp. 11–23.
[14] Datta, A., Derek, A., Mitchell, J. C. & Pavlovic, D. [2005]. A derivation system and compositional logic for security protocols, *J. Comput. Secur.* 13(3): 423–482.
[15] Datta, A., Derek, A., Mitchell, J. C. & Roy, A. [2007]. Protocol composition logic (PCL)., *Electr. Notes Theor. Comput. Sci.* 172: 311–358.
[16] Datta, A., Derek, A., Mitchell, J. C. & Warinschi, B. [n.d.]. Key exchange protocols: Security definition, proof method and applications.
URL: *citeseer.ist.psu.edu/datta06key.html*
[17] Datta, A., Derek, A., Mitchell, J. & Pavlovic, D. [2003b]. A derivation system for security protocols and its logical formalization, *16th IEEE Computer Security Foundations Workshop (CWFW-16)*, pp. 109–125.
URL: *citeseer.ist.psu.edu/datta03derivation.html*
[18] Datta, A., Mitchell, J., Roy, A. & Stiller, S. [2011]. Protocol composition logic, *Formal Models and Techniques for Analyzing Security Protocols*, IOS Press.
[19] Dierks, T. & Rescorla, E. [April 2006]. The Transport Layer Security (TLS) Protocol, version 1.1 – RFC 4346, http://tools.ietf.org/html/rfc4346.
[20] Diffie, W., van Oorschot, P. C. & Wiener, M. J. [1992]. Authentication and authenticated key exchanges., *Des. Codes Cryptography* 2(2): 107–125.
[21] Durgin, N., Mitchell, J. & Pavlovic, D. [2001]. A compositional logic for proving security properties of protocols, *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pp. 241–255.
URL: *citeseer.ist.psu.edu/article/durgin02compositional.html*
[22] Durgin, N., Mitchell, J. & Pavlovic, D. [2004]. A compositional logic for proving security properties of protocols, *J. Comput. Secur.* 11(4): 677–721.

[23] Fu, J., Jiang, X., Ping, L. & Fan, R. [2009]. A novel rekeying protocol for 802.11s key management, *Proceedings of the 2009 International Conference on Information Management and Engineering*, ICIME '09, IEEE Computer Society, Washington, DC, USA, pp. 295–299. URL: *http://dx.doi.org/10.1109/ICIME.2009.14*

[24] Gollmann, D. [1996]. What do we mean by entity authentication?, *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Washington, DC, USA, p. 46.

[25] Haasz, J. & Hampton, S. [2006]. Amendment: Mesh networking, http://standards.ieee.org/board/nes/projects/802-11s.pdf.

[26] He, C., Sundararajan, M., Datta, A., Derek, A. & Mitchell, J. C. [2005]. A modular correctness proof of IEEE 802.11i and TLS., *ACM Conference on Computer and Communications Security*, pp. 2–15.

[27] Krawczyk, H. [2003]. SIGMA: The 'SIGn-and-MAc' approach to authenticated diffie-hellman and its use in the IKE-protocols., *CRYPTO*, pp. 400–425.

[28] Kuhlman, D., Moriarty, R., Braskich, T., Emeott, S. & Tripunitara, M. [2007]. A proof of security of a mesh security architecture, http://eprint.iacr.org/2007/364.pdf.

[29] Meadows, C. & Pavlovic, D. [2004]. Deriving, attacking and defending the GDOI protocol, *ESORICS*, pp. 53–72.

[30] Rigney, C., Willens, S., Rubens, A. & Simpson, W. [June 2000]. Remote Authentication Dial In User Service (RADIUS) – RFC 2865, http://tools.ietf.org/html/rfc2865.

[31] Roy, A., Datta, A., Derek, A. & Mitchell, J. C. [2007]. Inductive proof method for computational secrecy, *Proceedings of 12th European Symposium On Research In Computer Security*.

[32] Roy, A., Datta, A., Derek, A., Mitchell, J. C. & Seifert, J.-P. [2006]. Secrecy analysis in protocol composition logic., *Proceedings of 11th Annual Asian Computing Science Conference*.

[33] Roy, A., Datta, A., Derek, A., Mitchell, J. C. & Seifert, J.-P. [2008]. Secrecy analysis in protocol composition logic., *Formal Logical Methods for System Security and Correctness*, IOS Press.

[34] Simon, D., Aboba, B. & Hurst, R. [2007]. The EAP TLS authentication protocol, http://www.ietf.org/internet-drafts/draft-simon-emu-rfc2716bis-11.txt.

[35] Zhao, M., Walker, J. & Conner, W. S. [2007]. Overview of abbreviated handshake protocol, https://mentor.ieee.org/802.11/documents doc 11-07/1998r01.