A Modular Correctness Proof of IEEE 802.11i and TLS

Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, John C. Mitchell
Electrical Engineering and Computer Science Departments,
Stanford University, Stanford, CA 94305-9045

ABSTRACT

The IEEE 802.11i wireless networking protocol provides mutual authentication between a network access point and user devices prior to user connectivity. The protocol consists of several parts, including an 802.1X authentication phase using TLS over EAP, the 4-Way Handshake to establish a fresh session key, and an optional Group Key Handshake for group communications. Motivated by previous vulnerabilities in related wireless protocols and changes in 802.11i to provide better security, we carry out a formal proof of correctness using a Protocol Composition Logic previously used for other protocols. The proof is modular, comprising a separate proof for each protocol section and providing insight into the networking environment in which each section can be reliably used. Further, the proof holds for a variety of failure recovery strategies and other implementation and configuration options. Since SSL/TLS is widely used apart from 802.11i, the security proof for SSL/TLS has independent interest.

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Security

Keywords: IEEE802.11i, TLS, Protocol Composition Logic

1. INTRODUCTION

Security is an obvious concern in many wireless networks, because intruders can potentially access a network without physically entering the buildings in which it is used. While intended to provide security, the Wired Equivalent Privacy (WEP) [1] protocol lacks good key management and suffers from significant cryptographic problems [7], to the extent that FBI agents have publicly demonstrated that they can break a 128-bit WEP key in about three minutes [9]. For these reasons, the *IEEE Task Group i* has developed the 802.11i Standard [2], ratified in June 2004, to provide confidentiality, integrity, and mutual authentication. 802.11i provides authentication protocols, key management protocols, and data confidentiality protocols that may execute concurrently over a network in which other protocols are also used.

In this paper, we present a formal correctness proof of the 802.11i protocols using *Protocol Composition Logic (PCL)* [17, 18, 11, 14, 12, 13, 25]. In previous work, PCL has been proved sound for protocol runs that use any number of principals and sessions, over both symbolic models and (for a subset of the logic at present) over more traditional cryptographic assumptions [10], which implies security for an unbounded number of participants and sessions. Therefore, while there are previous studies [20, 21] using finite-state analysis to find errors in 802.11i with bounded configurations, we believe that this is the first complete proof of 802.11i for an unbounded number of participants and sessions. Furthermore, the formal proof for the TLS protocol has independent interest since TLS is widely used independent of 802.11i (e.g. [3]).

Our proof consists of separate proofs of specific security properties for 802.11i components - the TLS authentication phase, the 4-Way Handshake protocol and the Group Key Handshake protocol. Using a new form of PCL composition principle, formulated as *staged composition* in this paper, we combine the component proofs to show that any staged use of the protocol components achieves the stated security goals. It follows that the components compose securely for a range of failure recovery control flows, including the improvements proposed in [21]. The general result also proves security for other configurations presented in the 802.11i Standards document, including the use of a Pre-Shared Key (PSK) or cached Pair-wise Master Key (PMK). In addition to devising a new composition principle for PCL, we also extend the logic to handle local memory associated with reusing generated nonces. The memory feature is needed to prove correctness of an unusual feature of the improved 4-Way Handshake protocol [21] that involves reusing a nonce to avoid a Denial of Service (DoS) attack.

An advantage of PCL is that each proof component identifies not only the local reasoning that guarantees the security goal of that component, but also the environmental conditions that are needed to avoid destructive interference from other protocols that may use the same certificates or key materials. These environment assumptions are then proved for the steps that require them, giving us an invariant that is respected by the protocol. In formulating the proof, we identify the precise conditions that will allow other protocols to be executed in the same environment without interfering with 802.11i. Moreover, our proof provides certain insights

CCS'05, November 7–11, 2005, Alexandria, Virginia, USA.

into the component protocols. For example, one proof step in showing authentication for the 4-Way Handshake protocol reveals that the *authenticator* (the access point) must be a different principal from the *supplicant* (typically a laptop); without this separation a reflection attack is possible. While this condition is entirely reasonable in standard enterprise installations, it could be violated in an ad hoc network configuration and cause vulnerabilities. Briefly, our analysis yields the following suggestions for implementers:

- To prevent a reflection attack on the 4-Way Handshake and Group Key Handshake, no principal can be both an authenticator and supplicant.
- For TLS security, if a key associated with the CA-issued certificate is used in other protocols, all uses must conform to conditions enumerated in this paper.
- Failure recovery can roll back to the end of any completed component, respecting the invariants discussed in this paper.

Our results suggest that PCL is suitable for compositional analysis of large protocols, yielding assurance and guidelines for implementation and deployment. Among the methods and security studies carried out in recent years, such as [35, 8, 23, 37, 26, 29, 28, 27, 31, 4], the closest to our study appear to be Paulson's investigations [32, 33] of TLS and SET [36]. Some advantages of our approach over Paulson's inductive method are a compositional proof method and a higher level of abstraction. Paulson's method involves direct reasoning about an inductively defined set of protocol execution traces, built from the protocol specification and attacker actions. PCL, however, has an axiomatic system that abstracts away arguments about traces and eliminates the need to reason explicitly about attacker actions. This feature, along with the Floyd-Hoare style specification, make proofs in PCL concise and readable. Because the semantic soundness of PCL shows that each of the axioms and inference rules are correct for arbitrary protocol runs in the presence of a symbolic attacker, the PCL proof system provides the same semantic guarantees as Paulson's method, without requiring a set of lemmas to be reproved for each protocol that is studied.

The rest of the paper is organized as follows. Section 2 briefly describes the IEEE 802.11i Standard and the Protocol Composition Logic (PCL). Sections 3, 4 and 5 present the analysis of the 4-Way Handshake, the TLS protocol, and the Group Key Handshake, respectively. Section 6 describes the staged composition principle which takes into account the structure of complicated control flows and proves the safe composition of various components of 802.11i. Finally, Section 7 concludes the paper.

2. OVERVIEW

2.1 Overview of 802.11i

The IEEE 802.11i Standard [2] defines data confidentiality, mutual authentication, and key management protocols intended to provide enhanced security in the MAC layer of a wireless network. This set of protocols together defines a "Robust Security Network Association"(RSNA). The RSNA establishment procedure involves three entities called the *Supplicant* (the wireless station), the *Authenticator* (the access point), and the *Authentication Server* (typically a RADIUS server).

In this paper, we focus on the mutual authentication and key management protocols. A typical RSNA establishment procedure starts by executing an EAP authentication between the supplicant and the authentication server, typically using EAP-TLS, with the authenticator acting as a relay. After the successful completion of the EAP-TLS session, the supplicant and the authentication server verified each other's identity and agreed on a shared secret. Then the authentication server moves the secret to the authenticator; the authenticator and supplicant derive a shared Pair-wise Master Key (PMK) from this secret. Afterwards, the authenticator and the supplicant execute a session of the 4-Way Handshake protocol, from which a fresh Pair-wise Transient Key (PTK) is derived to secure subsequent data traffic. Note that, in practice, the authentication server can be implemented either in a single device with the authenticator, or through a separate server. In the latter case, it is assumed that the link between the authentication server and the authenticator is physically secure. Therefore, while modelling the protocol, it is safe to make a simplifying assumption that the authentication server and the authenticator are the same principal.

While the typical run described above is relatively straightforward, the complete specification is much more complicated due to additional optional components and alternative configurations. For example, 802.11i can adopt other EAP methods for authentication, such as password-based authentication, instead of EAP-TLS. Moreover, in the case of multicast applications, the authenticator can also distribute a fresh group key to all supplicants in a group after the PTK has been established.

In this paper, we focus on the complete RSNA establishment procedure that consists of four components: TLS, 4-Way Handshake, Group Key Handshake, and data sessions. These components are designed to be executed sequentially; however, in order to prove security properties of this procedure, we also have to consider all other possible executions. For example, 4-Way Handshakes can be periodically re-run to refresh the PTK. Failure of one component also leads to other possible execution sequences. In the original 802.11i specification, the entire sequence is restarted if one component fails, as shown in Figure 1(a). As observed in [21], this failure recovery mechanism is quite inefficient and may be improved as shown in Figure 1(b). We therefore formulate our proof in a way that demonstrates the desired security properties for both control flow graphs.

2.2 Overview of Proof Method

We use Protocol Composition Logic (PCL) [17, 11, 14, 13] to prove correctness of the 802.11i protocols. This section contains a brief discussion of PCL relevant to this analysis.

Modelling protocols. A protocol is defined by a set of roles, each specifying a sequence of actions to be executed by an honest agent. Protocol roles are represented using a simple "protocol programming language" based on *cords* [17]. Figure 2 shows a simple two message protocol in the informal arrows-and-messages notation and the formal programs for roles of the same protocol using the cords notation. Program **Init** describes the actions of a *thread* X executing the initiator role in the protocol with *agent* $\hat{Y}$ as the responder. The

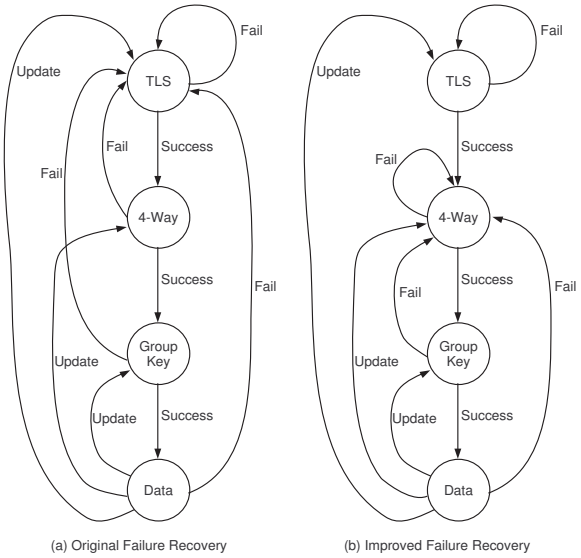

Figure 1: The 802.11i Control Flow

possible protocol actions include nonce generation, signatures and encryption, communication steps, and decryption and signature verification via pattern matching. Programs can also depend on input parameters (typically determined by context or the result of set-up operations) and provide output parameters to subsequent operations.

Protocol logic and the proof system. The syntax of the logic and informal descriptions of the logical predicates is given in [18, 13]. Most protocol proofs use formulas of the form $\theta[P]_X\phi$, which means that starting from a state where formula θ is true, after actions P are executed by the thread X, the formula ϕ is true in the resulting state. Formulas ϕ and ψ typically make assertions about temporal order of actions (useful for stating authentication) and/or the data accessible to various principals (useful for stating secrecy).

The proof system extends first-order logic with axioms and proof rules for protocol actions, temporal reasoning, knowledge, and a specialized form of invariance rule called the *honesty rule*. The honesty rule is essential for combining facts about one role with inferred actions of other roles. Intuitively, if Alice receives a response from a message sent to Bob, the honesty rule captures Alice's ability to use properties of Bob's role to reason about how Bob generated his reply. In short, if Alice assumes that Bob is honest, she may use Bob's role to reason from this assumption.

Compositional proof method. Following the modular design of the protocol, we extensively use the compositional approach developed in [14, 13] and prove properties of the whole protocol by combining proofs of its parts.

We separately prove security guarantees of the form $\Gamma \vdash \theta[P]_X\phi$ for 4-Way Handshake, TLS, and the Group Key Handshake in Sections 3, 4, and 5 respectively. Here Γ includes invariants of the specific protocol component. Assuming that these invariants are satisfied, the respective components possess the stated properties. For each component, invariants are proved using the honesty rule, showing that

$$\begin{array}{rcl} X \to Y & : & x \\ Y \to X & : & x, SIG_Y(X,x) \end{array}$$

$$\begin{array}{rcl} \textbf{Init} & = & (X,\hat{Y})[\texttt{new } x; \texttt{send } \hat{X},\hat{Y},x; \\ & & \texttt{receive } \hat{Y},\hat{X},x,s; \\ & & \texttt{match } s/SIG_Y(X,x);]_X \\ \textbf{Resp} & = & (Y)[\texttt{receive } \hat{X},\hat{Y},x; \\ & & \texttt{send } \hat{Y},\hat{X},SIG_Y(X,x);]_Y \end{array}$$

Figure 2: Arrows-and-messages vs cords

the component is secure in isolation.

In order to prove properties of the complete protocol, we combine guarantees provided by the different components. For example, the 4-Way Handshake provides authentication assuming that the Pair-wise Master Key established by TLS is a shared secret between the supplicant and the authenticator. The combined protocol consisting of a TLS session followed by a 4-Way Handshake therefore provides authentication. Technically, sequential composition involves matching a *postcondition* of one protocol to a *precondition* of the other, as well as checking that the two protocols satisfy each other's invariants. Ensuring that the protocol components compose safely given the error handling mechanisms in Figure 1 requires an extension of the existing composition theorems. The new composition theorem as well as its application to 802.11i is presented in Section 6.

We do not present an analysis of the data confidentiality protocol, used once keys are established, in this paper. Since the current logic is based on an execution model based on idealized cryptography, a correctness proof of the data transfer protocol is unlikely to be very informative. However, it could be prudent to study the data confidentiality protocol in another manner.

3. 4-WAY HANDSHAKE

In this section, we prove security properties of the 4-Way Handshake protocol and the modified 4-Way Handshake proposed in [20, 21]. The 4-Way Handshake generates the Pairwise Temporary Key (PTK) for data confidentiality protocols and the Group Key Handshake Protocol, using a pre-established secret shared between the authenticator and the supplicant. The pre-established secret, called the Pair-wise Master Key (PMK), may be set up via mutual authentication protocols (de facto EAP-TLS), or it may be pre-configured as a Pre-Shared Key (PSK).

3.1 Modelling 4-Way

During the handshake, the authenticator and supplicant generate fresh nonces, then derive a fresh PTK based on the shared PMK, the nonces, and their MAC addresses. They authenticate the key material generated using keyed hashes. The authenticator and supplicant roles of the 4-Way Handshake are described formally using the programming language introduced in the previous section, which are listed in Table 1. We describe the authenticator program **4WAY** : **AUTH** in details below.

This program has three input parameters - the authenticator and the supplicant identifiers, and the PMK represented by the variable *pmk*. The first action executed

$$
\begin{array}{rcl}
\mathbf{4WAY:AUTH} & = & (X, \hat{Y}, pmk) \\
 & & [\texttt{new } x;\ \texttt{send } \hat{X}, \hat{Y}, x, \text{“msg1”}; \\
 & & \texttt{receive } \hat{Y}, \hat{X}, z;\ \texttt{match } z/y, \text{“msg2”}, mic1; \\
 & & \texttt{match } HASH_{pmk}(x,y)/ptk;\ \texttt{match } mic1/HASH_{ptk}(y, \text{“msg2”}); \\
 & & \texttt{send } \hat{X}, \hat{Y}, x, \text{“msg3”}, HASH_{ptk}(x, \text{“msg3”}); \\
 & & \texttt{receive } \hat{Y}, \hat{X}, w; \\
 & & \texttt{match } w/\text{“msg4”}, mic2;\ \texttt{match } mic2/HASH_{ptk}(\text{“msg4”})]_X \\
\mathbf{4WAY:SUPP} & = & (Y, pmk) \\
 & & [\texttt{receive } \hat{X}, \hat{Y}, z;\ \texttt{match } z/x, \text{“msg1”}; \\
 & & \texttt{new } y;\ \texttt{match } HASH_{pmk}(x,y)/ptk; \\
 & & \texttt{send } \hat{Y}, \hat{X}, y, \text{“msg2”}, HASH_{ptk}(y, \text{“msg2”}); \\
 & & \texttt{receive } \hat{X}, \hat{Y}, w; \\
 & & \texttt{match } w/x, \text{“msg3”}, mic;\ \texttt{match } mic/HASH_{ptk}(x, \text{“msg3”}); \\
 & & \texttt{send } \hat{Y}, \hat{X}, \text{“msg4”}, HASH_{ptk}(\text{“msg4”})]_Y
\end{array}
$$

Table 1: 4-Way Handshake Program

by the authenticator X involves generating a fresh nonce x using the action `new` . Then X sends out $Message$ 1 to Y, which contains the nonce x and the string "msg1". In practice, message indicators are represented by sequences of bits, but we use strings here for readability. Authenticator X waits to receive a response from Y and then checks whether the received message is the expected $Message$ 2 using a `match` action, which verifies the Message Integrity Code (MIC) based on the *ptk* derived. If $Message$ 2 is valid, X sends out $Message$ 3 including the nonce x, the string "msg3" and the MIC, and waits for the response. Once a valid $Message$ 4 is received and verified, X completes the 4-Way Handshake and subsequently uses the derived *ptk*.

Note that in the 802.11i specifications, the PTK is divided into several parts: KCK (Key Confirmation Key) for computing MIC, KEK (Key Encryption Key) for encrypting the group key, and TK (Temporary Key) for protecting data packets. For expository convenience, we use *ptk* to refer to all of these parts.

3.2 Security Properties

The desired security properties for the 4-Way Handshake as identified by the standard (see Section 8.4.8 in [2]) are:

1. Confirm the existence of the PMK at the peer.
2. Ensure that the security association key (PTK) is fresh.
3. Synchronize the installation of session keys into the MAC.
4. Transfer the GTK from the Authenticator to the Supplicant.
5. Confirm the selection of cipher suites.

The definitions below formalize two security properties called *key secrecy* and *session authentication*. Items 1, 2, 3, and 5 are captured by *session authentication*; moreover, *session authentication* can be asserted only when the *key secrecy* is guaranteed. We discuss item 4 in Section 5. These conditions state the security guarantees for the authenticator precisely. The guarantees for the supplicant are analogous, but omitted due to space constraints. Informally, the formula $\phi_{4way,sec}$ below states that only the authenticator and the supplicant possess the PTK.

Definition 3.1 (4-way key secrecy).
The 4-Way Handshake is said to provide key secrecy if $\phi_{4way,sec}$ holds, where

$$
\begin{array}{l}
\phi_{4way,sec} ::= \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\
\quad ((\mathsf{Has}(\hat{Z}, ptk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y})) \wedge \\
\quad \mathsf{Has}(\hat{X}, ptk) \wedge \mathsf{Has}(\hat{Y}, ptk)
\end{array}
$$

The formula below formalizes a standard notion of authentication called *matching conversations* [6]. It guarantees that the two principals have consistent views of protocol runs. It follows that they agree on terms such as the cipher suite and the freshly generated PTK.

Definition 3.2 (4-way authentication).
The 4-Way Handshake is said to provide session authentication for the authenticator if $\phi_{4way,auth}$ holds, where

$$
\begin{array}{l}
\phi_{4way,auth} ::= \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\
\quad \exists Y. ActionsInOrder(\\
\quad \mathsf{Send}(X, \hat{X}, \hat{Y}, Message\ 1), \\
\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, Message\ 1), \\
\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, Message\ 2), \\
\quad \mathsf{Receive}(X, \hat{Y}, \hat{X}, Message\ 2), \\
\quad \mathsf{Send}(X, \hat{X}, \hat{Y}, Message\ 3), \\
\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, Message\ 3), \\
\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, Message\ 4), \\
\quad \mathsf{Receive}(X, \hat{Y}, \hat{X}, Message\ 4))
\end{array}
$$

The main result of this section is Theorem 1, which states the security guarantees for the authenticator. A proof of this theorem appears in Appendix B. A similar guarantee holds for the supplicant. We omit the theorem and the proofs for the supplicant due to space constraints.

Theorem 1 (4-Way Authenticator Guarantee).
(i) On execution of the authenticator role, key secrecy *and* session authentication *are guaranteed if the formulas in Table 2 hold. Formally,*

$$
\begin{array}{l}
\Gamma_{4way,1} \wedge \Gamma_{4way,2} \vdash \\
\theta_{4way}[4WAY{:}AUTH]_X \phi_{4way,auth} \wedge \phi_{4way,sec}
\end{array}
$$

$$\theta_{4way} := \mathsf{Has}(\hat{X}, pmk) \wedge \mathsf{Has}(\hat{Y}, pmk) \wedge \mathsf{NonceSource}(Y, pmk, ENC_{\hat{X}}(pmk))$$
$$\Gamma_{4way,1} := \mathsf{Computes}(\hat{X}, HASH_{pmk}(x,y)) \supset \neg(\mathsf{Send}(\hat{X}, m) \wedge \mathsf{Contains}(m, HASH_{pmk}(x,y))$$
$$\Gamma_{4way,2} := (\mathsf{Honest}(\hat{X}) \wedge \mathsf{Receive}(X, Message\ 1) \supset \neg\mathsf{Send}(X, Message\ 3)) \wedge$$
$$(\mathsf{Honest}(\hat{X}) \wedge \mathsf{Send}(X, Message\ 1) \supset \neg(\mathsf{Send}(X, Message\ 2) \wedge \mathsf{Send}(X, Message\ 4)))$$

Table 2: 4-Way Handshake Precondition and Invariants

(ii) $\Gamma_{4way,1}$ *is invariant of the 4-Way Handshake;* $\Gamma_{4way,2}$ *is an assumption on the environment. Formally,*

$$4WAY \vdash \Gamma_{4way,1}$$

The theorem states that starting from a state in which the precondition holds, if the authenticator role is executed, then in the resulting state the desired authentication and secrecy properties are guaranteed. The precondition θ_{4way} listed in Table 2 requires that the PMK is only sent out under encryption. The authenticator deduces its security properties based on the actions that it performs, the properties of certain cryptographic primitives and knowledge of the behavior of an honest supplicant. (By definition, an honest principal behaves in accordance with the protocol.) In the case of the 4-Way Handshake, the expected behavior of an honest principal is captured by the formulas $\Gamma_{4way,1}$ and $\Gamma_{4way,2}$ listed in Table 2. The first statement of the theorem states that, assuming these formulas hold, the security property is guaranteed.

Invariants are generally proved by induction over programs using the *Honesty Rule*. However, the detailed proofs are omitted here due space limitations. The second part of the theorem states that $\Gamma_{4way,1}$ is an invariant of the 4-Way Handshake protocol. The formula $\Gamma_{4way,2}$ states that no principal performs the roles of both the supplicant and the authenticator. This is a reasonable assumption for a typical 802.11i deployment. As noted in [21], security of 802.11i may be compromised if this condition is violated.

3.3 Operating Environment

In this section, we discuss the characteristics of an operating environment in which the 4-Way Handshake Protocol provides security guarantees. The first goal is to identify the class of protocols that may run concurrently with the 4-Way Handshake without degrading its security; the second goal is to identify other application scenarios in which the 4-Way Handshake may be deployed safely. Our analysis provides insight in both directions.

As discussed above, in order to provide the *key secrecy* and *session authentication* properties, the 4-Way Handshake must be executed in an environment where the formulas listed in Table 2 are satisfied. $\Gamma_{4way,1}$ states that the supplicant and the authenticator derive the PTK locally and do not reveal it. It is possible that protocols executing concurrently with the 4-Way Handshake may share state and have access to these shared secrets. In this case, our formulas require that these protocols do not reveal these secrets. Note that the shared PMK should also be kept secure; we will discuss that in the TLS invariants.

$\Gamma_{4way,2}$ states the requirement that an honest principal does not execute both the authenticator and the supplicant roles. It expresses this by requiring that a principal that performs actions corresponding to one role does not perform actions corresponding to the other. If $\Gamma_{4way,2}$ does not hold, a simple reflection attack can be demonstrated [21]. It is highly unlikely that this condition would be violated in a wireless LAN environment, as we do not expect a laptop to play the role of an authenticator, or an access point to play the role of a supplicant. However, 802.11i may be deployed in an ad-hoc network environment, in which case it is conceivable that nodes play both roles and violate this assumption.

3.4 Improved 4-Way Handshake

The 4-Way Handshake, described in the previous section suffers from a DOS vulnerability [20, 21]. The vulnerability results from the lack of any authentication in $Message$ 1, which allows the attacker to block the supplicant role. It is possible to work around this flaw by allowing an arbitrary number of sessions, but this may result in a memory exhaustion attack since the supplicant must store all the nonces that it generates for various sessions. A modification that involves nonce re-use is discussed in [20, 21] and has been adopted by the 802.11i standards committee. The modified supplicant program is listed in the pseudo-code that follows, while the authenticator program is the same as the one in Table 1.

Algorithm 3.1: MOD-4-WAY:SUPP(Y, pmk)

```
new y;
repeat
 receive X̂, Ŷ, z;
 if match z/x, "msg1";
   then
    match HASH_pmk(x,y)/ptk;
    send Ŷ, X̂, y, "msg2", HASH_ptk(y, "msg2");
until
  match z/x, "msg3", HASH_ptk(x, "msg3");
send Ŷ, X̂, "msg4", HASH_ptk("msg4");
```

It is easy to see that the modified protocol cannot be blocked by the attacker. Also re-using the nonce until one 4-Way Handshake completes allows the supplicant to avoid storing state, which prevents memory exhaustion. Since the suggested fix involves nonce reuse, and nonces are generally used to provide freshness guarantees, it is not obvious that the *authentication* property is preserved under this modification. The main result of this section is the following theorem.

THEOREM 2 (MODIFIED 4-WAY GUARANTEE). *For the authenticator, the* session authentication *and the* key secrecy *guarantees for the modified protocol are identical to Theorem 1.*

Intuitively the authenticator guarantee continues to hold because nonce re-use occurs within the *repeat - until loop*

TLS : Client	=	$(X, \hat{Y}, VerSUx)[$
		`new` Nx; `send` $\hat{X}, \hat{Y}, Nx, VerSUx$;
		`receive` $\hat{Y}, \hat{X}, Ny, VerSUy, cert$; `match` $cert/SIG_{\hat{CA}}(\hat{Y}, Ky)$;
		`new` $secret$;
		`send` $\hat{X}, \hat{Y}, SIG_{\hat{CA}}(\hat{X}, Vx)$,
		$SIG_{Vx}(handShake1), ENC_{Ky}(secret), HASH_{secret}(handShake1,$ *"client"*$)$;
		`receive` $\hat{Y}, \hat{X}, hash$; `match` $hash/HASH_{secret}(handShake2,$ *"server"*$)$; $]_X$
TLS : Server	=	$(Y, VerSUy)[$
		`receive` $\hat{X}, \hat{Y}, Nx, VerSUx$; `new` Ny;
		`send` $\hat{Y}, \hat{X}, Ny, VerSUy, SIG_{\hat{CA}}(\hat{Y}, Ky)$;
		`receive` $\hat{X}, \hat{Y}, cert, sig, encsec, hash$; `match` $cert/SIG_{\hat{CA}}(\hat{X}, Vx)$;
		`match` $sig/SIG_{Vx}(handShake1)$; `match` $encsec/ENC_{Ky}(secret)$;
		`match` $hash/HASH_{secret}(handShake1,$ *"client"*$)$;
		`send` $\hat{Y}, \hat{X}, HASH_{secret}(handShake2,$ *"server"*$)$; $]_Y$

Table 3: TLS: Client and Server Programs

of a single session. The formalization of authentication - matching conversations - usually refers to all the actions of a principal in a session. Since an attacker can inject a forged *Message* 1 an arbitrary number of times, and the supplicant will respond to it, we exclude certain messages from the matching conversation. Authentication in this form is implied by our proof. The supplicant guarantees are identical to the original 4-Way Handshake.

4. TLS

The TLS/SSL [15] protocol provides end-to-end security and is widely deployed on the Internet in various security and e-commerce systems. IEEE 802.11i suggests EAP-TLS, which is an encapsulated version of TLS protocol, to mutually authenticate the supplicant and the authenticator, and to derive a shared secret key (PMK). In addition to proving TLS secure in isolation, we improve previous analysis of TLS [32, 29, 28, 27] by identifying conditions under which other protocols may run concurrently without introducing vulnerabilities. Identifying such conditions appears valuable given the wide deployment of SSL/TLS. Note that we use the terms *client* and *server* for TLS protocol participants, as in the TLS documentation [15]. When TLS is used with 802.11i, the *client* corresponds to the *supplicant* and the *server* can reside in the *authenticator*.

4.1 Modelling TLS

TLS has many possible modes of operation. We restrict our attention to the mode where both the server and the client have certificates, since this mode satisfies the mutual authentication property requirement of the 802.11i Standard.

The **TLS : Client** and **TLS : Server** programs are described in Table 3, where $VerSU$ represents the protocol version and cipher suite, K_y is the server's public key, Vx is the client's verification key. We use the `match` action to check signatures, verify keyed hashes and perform decryption. Note that the term *handShake1* and *handShake2* represent the concatenation of all the terms sent and received by a principal up to the point it is used in the program.

4.2 Security Properties

The properties that TLS [15] ought to satisfy include:

1. The principals agree on each other's identity, protocol completion status, the values of the protocol version, cryptographic suite, and the *secret* that the client sends to the server. For server $\hat{Y}$, communicating with client $\hat{X}$, this property is formulated in Definition 4.1.

2. The *secret* that the client generates should not be known to any other principal other than the client and the server. For server $\hat{Y}$ and client $\hat{X}$, this property is formulated in Definition 4.2.

Definition 4.1 (TLS Authentication).
TLS is said to provide session authentication for the server role if $\phi_{tls,auth}$ holds, where

$$\begin{aligned}\phi_{tls,auth} ::= {} & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\ & \exists X.ActionsInOrder(\\ & \mathsf{Send}(X, \hat{X}, \hat{Y}, m1), \\ & \mathsf{Receive}(Y, \hat{X}, \hat{Y}, m1), \\ & \mathsf{Send}(Y, \hat{Y}, \hat{X}, m2), \\ & \mathsf{Receive}(X, \hat{Y}, \hat{X}, m2), \\ & \mathsf{Send}(X, \hat{X}, \hat{Y}, m3), \\ & \mathsf{Receive}(Y, \hat{X}, \hat{Y}, m3), \\ & \mathsf{Send}(Y, \hat{Y}, \hat{X}, m4))\end{aligned}$$

and m1, m2, m3, m4 represent the corresponding TLS messages in Table 3.

Definition 4.2 (TLS Key Secrecy).
TLS is said to provide secrecy if $\phi_{tls,sec}$ holds, where

$$\begin{aligned}\phi_{tls,sec} ::= {} & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\ & \mathsf{Has}(\hat{X}, secret) \wedge \mathsf{Has}(\hat{Y}, secret) \wedge \\ & (\mathsf{Has}(\hat{Z}, secret) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y})\end{aligned}$$

We use the proof system to prove guarantees for both the client and the server. Due to space constraints, we list only the guarantee for the authenticator in Theorem 3. The

$$\Gamma_{tls,1} := \neg\exists m.\mathsf{Send}(X,m) \wedge (\mathsf{Contains}(m, HASH_{secret}(handShake1, \text{“server”})) \vee \mathsf{Contains}(m, HASH_{secret}(handShake2, \text{“client”})) \vee \mathsf{Contains}(m, SIG_{Vx}(handShake1)))$$

$$\Gamma_{tls,2} := \mathsf{Honest}(\hat{Y}) \wedge \mathsf{Send}(Y,m) \wedge \mathsf{ContainsOut}(m, secret, ENC_{Ky}(secret)) \supset (\neg\mathsf{Decrypts}(Y,m') \wedge \mathsf{Contains}(m', secret)) \vee (\mathsf{Receives}(Y,m'') < \mathsf{FirstSend}(Y, secret) \wedge \mathsf{ContainsOut}(m'', secret, ENC_{Ky}(secret)))$$

Table 4: TLS Invariants

client guarantee is similar. The secrecy of the exchanged key material in TLS is established by combining local reasoning based on the client's actions with global reasoning about actions of honest agents. Intuitively, a client that generates the secret only sends it out either encrypted with an honest party's public key or uses it as a key for a keyed hash (this is captured by the predicate NonceSource). Furthermore, no honest user will ever decrypt the secret and send it in the clear. Specifically, an honest party can send the secret in the clear only if it receives it in the clear first (this is captured by the TLS invariant $\Gamma_{tls,2}$). Secrecy follows directly from these two facts.

THEOREM 3 (TLS SERVER GUARANTEE). *(i) On execution of the server role,* key secrecy *and* session authentication *are guaranteed if the formulas in Table 4 hold. Formally,*

$$\Gamma_{tls,1} \wedge \Gamma_{tls,2} \vdash [TLS{:}Server]_X \phi_{tls,auth} \wedge \phi_{tls,sec}$$

(ii) The formulas in Table 4 are invariants of TLS. Formally, TLS $\vdash \Gamma_{tls,1} \wedge \Gamma_{tls,2}$

4.3 Operating Environment

We now characterize the class of protocols that compose safely with TLS. As in Section 3.3, we relate *invariants* in Table 4 to deployment considerations.

$\Gamma_{tls,1}$ states that messages of a certain format should not be sent out by any protocol that executes in the same environment as TLS. One set of terms represent keyed hashes of the handshake, where the key is the shared secret established by a TLS session; another set refers to signatures on the handshake messages. A client running a protocol that signs messages indiscriminately could cause the loss of the authentication property. Such an attack would only be possible if the client certificate used by TLS were shared with other protocols and $\Gamma_{tls,1}$ were violated by them.

$\Gamma_{tls,2}$ rules out an undesirable sequence of actions that may allow an intruder to learn the shared secret. Intuitively, if an honest principal is tricked into decrypting a term containing the secret using its private key, after which it sends out the contents of the encryption, the *secrecy property* of TLS is lost. Clearly, if principals use an exclusive public/private key pair for TLS , such an attack is not possible. However, since another protocol (or another stage of 802.11i) may uses the same public/private key pair as TLS, it is important to check that these formulas are invariants of any other protocol.

5. GROUP KEY HANDSHAKE

In multicast applications, the authenticator may distributes a Group Temporary Key (GTK) to supplicants. *Messages 3,4* of the 4-Way Handshake may optionally set up this key distribution. The authenticator then runs the Group Key Handshake protocol periodically to update the GTK. In this section we prove the correctness of the Group Key Handshake.

5.1 Modelling Group Key Handshake

The programs for the Group Key Handshake are listed in Table 5. The authenticator sends *GrpMessage1* containing the GTK, and the supplicant confirms receipt of the GTK by sending *GrpMessage2*. The authenticator encrypts the GTK under the Key Encryption Key (KEK) (represented by term *ptk*) and sends it to the supplicant. The authenticator monotonically increases the sequence number for every key exchange message sent to prevent replay attacks. MICs are used to provide authentication and message integrity. The sequence number comparison `isLess` (a,b) is used by the supplicant to check that $a < b$; the expression $Succ(a)$ represents a number greater than a.

5.2 Security Properties

The properties we prove for the Group Key Handshake are listed below.

1. The Supplicant is assured that the GTK received in the current Group Key Handshake was sent by the Authenticator, and was generated by the Authenticator after the GTK that the supplicant holds from a previous Group Key Handshake or 4-Way Handshake. This is called the *key ordering* property, and is formalized in Definition 5.1.

2. The Authenticator is assured that the principals with knowledge of the GTK must have executed a 4-Way Handshake with the Authenticator. This is called the *key secrecy* property, and formalized in Definition 5.2.

DEFINITION 5.1 (GROUP KEY ORDERING). *For a supplicant* $\hat{Y}$*, the Group Key Handshake is said to provide key ordering if* $\phi_{gk,ord}$ *holds, where*

$$\phi_{gk,ord} ::= \mathsf{Honest}(\hat{X}) \supset (\mathsf{Send}(X, \hat{X}, \hat{Y}, SeqNo1, ENC_{ptk}(gtk1)) \wedge \mathsf{Send}(X, \hat{X}, \hat{Y}, SeqNo2, ENC_{ptk}(gtk2)) \wedge \mathsf{isLess}(SeqNo1, SeqNo2) \supset \mathsf{FirstSend}(X, \hat{X}, \hat{Y}, SeqNo1, ENC_{ptk}(gtk1)) < \mathsf{FirstSend}(X, \hat{X}, \hat{Y}, SeqNo2, ENC_{ptk}(gtk2)))$$

DEFINITION 5.2 (GROUP KEY SECRECY). *For an authenticator* $\hat{X}$*, the Group Key Handshake is said*

$$
\begin{array}{lcl}
\mathbf{GK:AUTH} & = & (X, \hat{Y}, CurrSeqNo, ptk, gtk)[\\
 & & \texttt{match } Succ(CurrSeqNo)/NewSeqNo; \\
 & & \texttt{send } \hat{X}, \hat{Y}, NewSeqNo, \text{“grp1”}, ENC_{ptk}(gtk), \\
 & & HASH_{ptk}(NewSeqNo, \text{“grp1”}, ENC_{ptk}(gtk)); \\
 & & \texttt{receive } \hat{Y}, \hat{X}, z; \\
 & & \texttt{match } z/NewSeqNo, \text{“grp2”}, HASH_{ptk}(NewSeqNo, \text{“grp2”})]_X \\
\mathbf{GK:SUPP} & = & (Y, \hat{X}, OldSeqNo, OldGTK, ptk)[\\
 & & \texttt{receive } \hat{X}, \hat{Y}, z; \\
 & & \texttt{match } z/NewSeqNo, \text{“grp1”}, ENC_{ptk}(gtk), \\
 & & HASH_{ptk}(NewSeqNo, \text{“grp1”}, ENC_{ptk}(gtk)); \\
 & & \texttt{isLess } OldSeqNo, NewSeqNo; \\
 & & \texttt{send } \hat{Y}, \hat{X}, NewSeqNo, \text{“grp2”}, HASH_{ptk}(NewSeqNo, \text{“grp2”});]_Y
\end{array}
$$

Table 5: Group Key Handshake Programs

$$
\begin{array}{l}
\theta_{gk} := \mathsf{Has}(\hat{X}, ptk) \wedge \mathsf{Has}(\hat{Y}, ptk) \wedge (\mathsf{Has}(\hat{Z}, ptk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}) \wedge \\
\quad (\mathsf{Send}(X, m) \wedge \mathsf{Contains}(m, SeqNo) \supset \mathsf{isLess}(SeqNo, Succ(CurrSeqNo))) \\
\Gamma_{gk,1} := \mathsf{Honest}(\hat{X}) \wedge (\mathsf{Send}(X, \hat{X}, \hat{Y}, m') < \mathsf{Send}(X, \hat{X}, \hat{Y}, m'')) \wedge \\
\quad \mathsf{Contains}(m', SeqNo1) \wedge \mathsf{Contains}(m'', SeqNo2) \supset \mathsf{isLess}(SeqNo1, SeqNo2) \\
\Gamma_{gk,2} \equiv \Gamma_{tls,2} \\
\Gamma_{gk,3} := \mathsf{Honest}(\hat{X}) \wedge \mathsf{Send}(X, GrpMessage\ 1) \supset \neg \mathsf{Send}(X, GrpMessage\ 2)
\end{array}
$$

Table 6: Group-Key Protocol Precondition and Invariants

to provide key secrecy if $\phi_{gk,sec}$ holds, where

$$
\begin{array}{l}
\phi_{gk,sec} ::= \\
\mathsf{Honest}(\hat{Z}_1) \wedge \mathsf{Honest}(\hat{Z}_2) \ldots \mathsf{Honest}(\hat{Z}_n) \supset \\
\quad ((\mathsf{Has}(\hat{Z}, gtk) \wedge \hat{Z} \neq \hat{X}) \supset \\
\quad \hat{Z} = \hat{Z}_1 \vee \hat{Z} = \hat{Z}_2 \cdots \vee \hat{Z} = \hat{Z}_n)
\end{array}
$$

Note that $\hat{Z}_1,\ \hat{Z}_2,\ \ldots\ \hat{Z}_n$ lists the set of supplicants that the authenticator intends to send the GTK. Each member of this set shares a PTK with the authenticator, established by a 4-Way Handshake. Though the 802.11i standard is not very clear with regards to the supplicant guarantee, two types of properties are conceivable - key freshness and key secrecy. However, key freshness cannot be achieved for the supplicant since a message could be lost; hence, we consider key ordering instead, which is a weaker requirement. Obviously the authenticator knows the key ordering since it generates the keys. Key secrecy can be guaranteed only for the authenticator because the supplicants do not have knowledge of the other supplicants in the group. Theorem 4 states the guarantee for the supplicant and the authenticator.

Theorem 4 (Group Key Guarantee).
(i) After execution of the supplicant role, key ordering *is guaranteed if the formulas in Table 6 hold. Formally,*

$$
\begin{array}{l}
\Gamma_{gk,1} \wedge \Gamma_{gk,2} \wedge \Gamma_{gk,3} \vdash \\
\theta_{gk}[GK{:}SUPP]_Y \phi_{gk,ord}
\end{array}
$$

(ii) After execution of the authenticator role, key secrecy *is guaranteed if the formulas in Table 6 hold. Formally,*

$$
\begin{array}{l}
\Gamma_{gk,1} \wedge \Gamma_{gk,2} \wedge \Gamma_{gk,3} \vdash \\
\theta_{gk}[GK{:}AUTH]_X \phi_{gk,sec}
\end{array}
$$

(iii) $\Gamma_{gk,1}$ and $\Gamma_{gk,2}$ are invariants of the Group Key Handshake; $\Gamma_{gk,3}$ is an assumption on the environment. Formally, GK $\vdash \Gamma_{gk,1} \wedge \Gamma_{gk,2}$

5.3 Operating Environment

In this section, we identify the class of operating conditions under which the Group Key Handshake provides security. Table 6 lists formulas required for the Group Key Handshake to function correctly. As in Sections 3.3, and 4.3, we regard these formulas as specifications for a safe operating environment.

$\Gamma_{gk,1}$ states that the authenticator should increase the sequence counter monotonically to guarantee the key ordering property for the supplicant. As pointed out in [20], the sequence numbers play an important role in the Group Key Handshake, but are redundant in the 4-Way Handshake. Our proof here indicates that sequence numbers in the 4-Way Handshake are redundant with respect to the key ordering properties of the Group Key Handshake as well. This is intuitively true because the Group Key stage uses key material established by the 4-Way stage, thus guaranteeing that actions of the Group Key protocol are done after actions of the 4-Way.

As in the case of $\Gamma_{tls,2}$, $\Gamma_{gk,2}$ states that the supplicants and the authenticator should not decrypt a message with the PTK, and send out the result of the decryption. This could conceivably happen if the data protection protocol uses the same encryption key as the Group Key Handshake. Fortunately in 802.11i the PTK is divided into several parts, where the GTK is encrypted by KEK and the data confidentiality protocol uses a different part TK (Temporal Key). Our analysis highlights the need for the key hierarchy in 802.11i. Obviously, the *key secrecy* property fails if any member of the group reveals the PMK, PTK, or GTK.

Like the 4-Way Handshake, there is a restriction on a principal to not play both the authenticator and the supplicant roles. This is represented by the formula $\Gamma_{gk,3}$. This is an assumption about the operating environment which should be carefully considered in implementations.

6. COMPOSITION

The components of 802.11i are intended to be used in a specific order: TLS, 4-Way Handshake, Group Key Handshake, and the data confidentiality protocols. Since this is an example of sequential composition of protocols, previously proved composition theorems [14, 13], allow us to prove the correctness of the composite protocol by combining the independent proofs. However, 802.11i provides an error-handling strategy that reacts to failures and timeouts by restarting execution at the beginning. Furthermore, more efficient error-handling strategies have also been proposed (see Figure 1). Finally, 802.11i allows modes of operation that use Pre-Shared Keys instead of the TLS stage or reuse previously cached PMK's. The previously published sequential composition theorems do not cover these more complicated situations.

In this section, we prove that the various components of 802.11i compose safely, for a general class of error-handling mechanisms, and for the modes of operation mentioned above. Section 6.1 is a technical section that introduces a new composition theorem and may be safely skipped by readers primarily interested in 802.11i. Section 6.2 applies this theorem to assert the correctness of 802.11i.

6.1 Staged Composition

The sequential composition theorem [14, 13] allows us to combine the proofs of sub-protocols into a proof of a composed protocol. However, sequential composition only works for control flows where the components execute one after the other in sequence. An examination of the control flow graphs in Figure 1 indicates that the intended sequence of execution of the sub-protocols forms a chain and the error-handling strategy introduces a set of backward arcs. This makes the sequential composition theorem stated in [14, 13] inadequate for our purpose. We introduce the notion of Staged Composition that extends sequential composition to handle control flow graphs with an arbitrary set of backward edges. This is formally stated as Theorem 5.

Recall that a protocol is a set of roles. For instance, the 4-Way Handshake consists of two roles – the authenticator and the supplicant. Each role is a sequence of protocol steps (`send` , `receive` , `new` and `match` actions), possibly depending on an input parameter list and providing an output parameter list. In a correct execution, we expect the role to run to completion. However, in reality the execution might be interrupted due to unexpected failures. Therefore, we partition a role into atomic steps, during each of which the execution does not block and cannot be interrupted. Since only `receive` actions require one thread to wait for another, roles are broken into atomic steps at receive points. The following definition captures the intuition that a protocol role may terminate at the end of any of its atomic protocol steps.

Definition 6.1 (Role-Prefix).
The set of role-prefixes of a role R, $RolePref(R) := \bigcup_{0\le i\le k} \{(\overrightarrow{x})[b_0 \dots b_i]_X\langle\overrightarrow{x}\rangle\} \bigcup\{(\overrightarrow{x})[]_X\langle\overrightarrow{x}\rangle\} \bigcup\{R\}$, *where role* $R = (\overrightarrow{x})[b_0 \dots b_k]_X\langle\overrightarrow{y}\rangle$.

A role in the composed protocol corresponds to a possible execution path in the control flow graph. A backward edge in a control flow graph from role R_i to role R_j causes control to proceed from the end of an atomic protocol step of R_i to the beginning of R_j. The following definition captures the set of possible executions of a role in a composed protocol.

Definition 6.2 (Staged Role).
The set of staged roles $RComp(\langle R_1, R_2 \dots R_n\rangle)$ *is defined as the sequential composition* $r_{a_1}; r_{a_2}; \dots; r_{a_l}$, *where* $a_k \in \{1, 2, \dots, n\}$, $r_{a_k} \in RolePref(R_{a_k})$, *with* $a_1 = 1$. *Furthermore, if* $r_{a_j} = R_{a_j}$ *then* $a_{j+1} = a_j + 1$; *otherwise* $a_{j+1} \le a_j$.

Given a chain of roles $\langle R_1, R_2 \dots R_n\rangle$, execution always starts at R_1; progress down the chain happens on successful completion of a role, with error-handling causing control to flow to the beginning of an earlier block in the chain. Failure of a role R_{ak} can happen at multiple points as characerized by the set $RolePref(R_{a_k})$ defined above. Finally, a protocol is simply a set of roles.

Definition 6.3 (Staged Composition).
A protocol Q is in the set of staged compositions $SComp(\langle Q_1, Q_2 ... Q_n\rangle)$ *of sub-protocols* $Q_1, Q_2 ... Q_n$, *if each role of Q is in* $RComp(\langle R_1, R_2 ... R_n\rangle)$, *where* R_i *is a role of protocol* Q_i.

Theorem 5 below states the conditions that need to hold for the composite protocol to be secure under staged composition. The first condition states that each protocol guarantees certain security properties assuming certain invariants are true. This follows from the correctness proofs of individual components. The second condition captures the intuition that the protocols running in the system do not cause vulnerabilities in each other. The third condition ensures that the precondition of a role R_i is discharged by the postcondition of R_{i-1}, which is required for sequential composition without error flows. Finally, the fourth condition requires that the precondition of a sub-protocol are preserved by the protocols steps of all the sub-protocols later in the chain. This ensures secure composition in the presence of arbitrary error handling mechanisms.

Theorem 5 (Staged Composition Theorem).
Given protocols $Q_1, Q_2 ... Q_n$, *if*

$$
\begin{aligned}
&(i) \quad \forall i,\ \Gamma_i \vdash \theta_i[P_i]_X\varphi_i \\
&(ii) \quad \forall i, j, \quad Q_i \vdash \Gamma_j \\
&(iii) \ \forall i,\ \varphi_i \supset \theta_{i+1} \\
&(iv) \quad \forall B \in \bigcup_{j\ge i} ProtocolSteps(Q_j),\ \theta_i[B]\theta_i
\end{aligned}
$$

then $SComp(\langle Q_1, Q_2 ... Q_n\rangle) \vdash \theta_1[P; P_i]_X\phi_i$, *where* $P; P_i \in SComp(\langle Q_1, Q_2, \dots, Q_n\rangle)$ *and* $P_i \in Q_i$.

Intuitively, we treat branches introduced by the error handling flows as non-deterministic choices, which reduces staged composition to the previously studied notion of sequential composition [14, 13]. The set defined in Definition 6.2 represents all possible linearizations of the executions of a role in a composed protocol. We omit the proof of Theorem 5 due to space constraints.

6.2 Composition of 802.11i

We now apply the compositional proof method to the 802.11i protocol components by showing that the four conditions required by Theorem 5 are satisfied.

1. We start with the properties of the components proved independently in sections 3, 4, 5. This corresponds to asserting condition (i) from Theorem 5.

$$
\begin{aligned}
&TLS \vdash \Gamma_{tls,1} \wedge \Gamma_{tls,2}\\
&\Gamma_{tls,1} \wedge \Gamma_{tls,2} \vdash\\
&[\mathit{TLS{:}Server}]\phi_{4way,auth} \wedge \phi_{4way,sec}\\
&4WAY \vdash \Gamma_{4way,1}\\
&\Gamma_{4way,1} \wedge \Gamma_{4way,2} \vdash \theta_{4way}\\
&[\mathit{4WAY{:}AUTH}]\phi_{4way,sec} \wedge \phi_{4way,auth}\\
&GK \vdash \Gamma_{gk,1} \wedge \Gamma_{gk,2}\\
&\Gamma_{gk,1} \wedge \Gamma_{gk,2} \wedge \Gamma_{gk,3} \vdash\\
&\theta_{gk}[\mathit{GK{:}AUTH}]\phi_{gk,sec} \wedge \phi_{gk,ord}
\end{aligned}
$$

2. Next we prove that each component respects the invariants of the other components. This is done by induction over initial segments of the roles of the components. These conditions hold because components use different keys, and use MICs (Message Integrity Codes) that contain sufficient identity to be distinguished from each other.

$$
\begin{aligned}
TLS &\vdash \Gamma_{4way} \wedge \Gamma_{gk}\\
4WAY &\vdash \Gamma_{tls} \wedge \Gamma_{gk}\\
GK &\vdash \Gamma_{tls} \wedge \Gamma_{4way}
\end{aligned}
$$

3. We prove that the precondition of the 4-Way authenticator role is implied by the postcondition of the TLS server role and the precondition of the Group Key authenticator role is implied by the postcondition of 4-Way authenticator role. At this point, we have the safe sequential composition of the components of 802.11i.

$$
\begin{aligned}
&\phi_{tls} \supset \theta_{4way}\\
&\phi_{4way} \supset \theta_{gk}
\end{aligned}
$$

4. Finally, the precondition of the 4-Way Handshake, which involves not sending out the PMK, is preserved by all protocol steps of the 4-Way and the Group Key Handshake, allowing restarts of the 4-Way Handshake to be secure. Similarly, the protocol steps of the Group Key Handshake do not send out the PTK ensuring restarts of the Group Key Handshake are secure.

The steps explained above allow us to apply Theorem 5, and prove the 802.11i authenticator guarantee. A similar process can be applied to the supplicant. Theorem 6 states the result of our analysis.

THEOREM 6 (802.11I GUARANTEE).
(i) The security properties of the components listed in sections 3.2, 4.2, 5.2 are guaranteed in all modes of IEEE 802.11i, if the assumptions in Table 2, 4, 6 are satisfied.
(ii) Suppose no principals play both the authenticator and supplicant role, 802.11i satisfies assumptions in Table 2, 4, 6.

Note that most of our effort was on proving individual components correct. Steps 2, 3, and 4 took considerably less effort. This shows that it is convenient to apply the compositional method to the 802.11i protocol suite. Furthermore, our proof implies the correctness of other deployment modes in 802.11i, which are different from that in section 2.1. First, 802.11i may be run without the TLS stage, using Pre-Shared Keys (PSK) instead. When a PSK is shared by an authenticator-supplicant pair, the precondition requirement of the 4-Way Handshake is satisfied, which implies the safety of this mode. Second, supplicants may also run the 4-Way Handshake using a PMK cached from an eariler execution of the composite protocol. This corresponds to restarting exection from the 4-Way Handshake stage. Since our proof holds for an arbitrary set of such backward arcs, using a cached PMK is safe.

7. CONCLUSIONS

We present a formal correctness proof for the IEEE 802.11i and TLS protocols using Protocol Composition Logic (PCL). The proof of IEEE 802.11i and TLS supports many design decisions of the 802.11i committee and reinforces conclusions of our previous studies [20, 21]. For example, the PCL proof demonstrates the need for separate keys for supplicant and authenticator to prevent a reflection attack, supports our previous intuition that various sequence numbers were unnecessary, shows that the protocol remains secure when a nonce-reuse mechanism is adopted in the 4-Way Handshake to reduce a vulnerability of Denial of Service [20, 21], and supports the adoption of optimized error-recovery strategies. Developing a single correctness proof for a class of error-recovery strategies requires a new composition principle for PCL, which is formulated and proved semantically sound in this paper.

The compositional nature of our protocol logic distinguishes our effort from other methods with similar foundations, such as Paulson's Inductive method [32] and Meadows' NRL protocol analyzer [23], which have been applied to protocols of similar scale and structure. In fact, Meadows previously identified composability of cryptographic protocols as a significant concern in establishing correctness [24]. A compositional approach is useful both for managing scale when working with a large protocol and in understanding how a single protocol interacts with its environment. In order to achieve compositionality, each individual proof involves a set of protocol invariants that must be satisfied in any environment where the protocol runs. These conditions must not only be satisfied by other subprotocols of 802.11i, but also by other protocols using the same certificates or other critical data which are run in parallel with 802.11i. Our proof therefore provides useful deployment information beyond the correctness of the protocols.

Our high-level logic with provable soundness over arbitrary symbolic runs is intended to combine the relative readability and ease of use of BAN-style logics [8] while being based on the semantic protocol execution model of Paulson's method [32]. Although we constructed all proofs manually, the proof system is completely rigorous and amenable to future automation. For more details about the PCL proof system and the actual proofs described in this paper, we refer the reader to our Protocol Composition Logic web site. Finally, the axioms and rules used in the current proof have been proved sound for a symbolic model of protocol execu-

tion and attack. We hope that in the future a computational semantics of PCL, such as suggested in [10], can be developed for the entire proof system used here, providing a correctness proof of 802.11i under standard cryptographic assumptions.

8. REFERENCES

[1] IEEE Standard 802.11-1999. Local and metropolitan area networks - specific requirements - part 11: Wireless LAN Medium Access Control and Physical Layer specifications. 1999.

[2] IEEE P802.11i/D10.0. Medium Access Control (MAC) security enhancements, amendment 6 to IEEE Standard for local and metropolitan area networks part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications. April 2004.

[3] Verified By Visa. 2004. https://usa.visa.com/personal/security/vbv/.

[4] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols The Spi Calculus. In *Proceedings of Fourth ACM Conference on Computer and Communications Security*, 1997.

[5] W. A. Arbaugh, N. Shankar, and J. Wang. Your 802.11 network has no clothes. In *Proceedings of the First IEEE International Conference on Wireless LANs and Home Networks*, pages 131–144, 2001.

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryprtology - Crypto'93 Proceedings*. Springer-Verlag, 1994.

[7] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001.

[8] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[9] H. Cheung. FBI Teaches Lesson in how to break into Wi-Fi networks. Information Week Network Pipeline, 2005.

[10] A. Datta, A. Derek, J. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *Int'l Colloquium on Automata, Languages, and Programming (ICALP)*, 2005.

[11] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*, pages 109–125. IEEE, 2003.

[12] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proceedings of 17th IEEE Computer Security Foundations Workshop*, pages 30–45. IEEE, 2004.

[13] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 2005.

[14] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proceedings of 19th Annual Conference on Mathematical Foundations of Programming Semantics*, volume 83 of *Electronic Notes in Theoretical Computer Science*, 2004.

[15] T. Dierks and C. Allen. The TLS Protocol — Version 1.0. IETF RFC 2246, January 1999.

[16] W. Diffie, P. C. V. Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[17] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pages 241–255. IEEE, 2001.

[18] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11:677–721, 2003.

[19] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 1–24, 2001.

[20] C. He and J. C. Mitchell. Analysis of 802.11i 4-way handshake. In *Proceedings of the Third ACM International Workshop on Wireless Security (WiSe'04)*, 2004.

[21] C. He and J. C. Mitchell. Security analysis and improvements for IEEE 802.11i. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, 2005.

[22] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW'97)*, page 31, Washington, DC, USA, 1997. IEEE Computer Society.

[23] C. Meadows. A model of computation for the NRL protocol analyzer. In *Proceedings of 7th IEEE Computer Security Foundations Workshop*, pages 84–89. IEEE, 1994.

[24] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. *Lecture Notes in Computer Science*, 2052:21–36, 2001.

[25] C. Meadows and D. Pavlovic. Deriving, attacking and defending the GDOI protocol. In *ESORICS*, pages 53–72, 2004.

[26] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[27] J. C. Mitchell. Finite-state analysis of security protocols. In *Computer Aided Verification*, pages 71–76, 1998.

[28] J. C. Mitchell, M. Mitchell, and U.Stern. Automated analysis of cryptographic protocols using murphi. In *IEEE Symp. Security and Privacy*, pages 141–153, 1997.

[29] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0. In *Proceedings of Seventh USENIX Security Symposium*, pages 201–216, 1998.

[30] V. Moen, H. Raddum, and K. J. Hole. Weakness in the temporal key hash of WPA. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8:76–83, 2004.

[31] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[32] L. C. Paulson. Inductive analysis of the Internet protocol TLS. *ACM Transactions on Computer and System Security*, 2(3):332–351, 1999.

[33] L. C. Paulson. Verifying the SET protocol. In *Proceedings of International Joint Conference on Automated Reasoning*, 2001.

[34] A. W. Roscoe. Intensional specifications of security protocols. In *Proceedings of the Ninth IEEE Computer Security Foundations Workshop (CSFW'96)*, page 28, Washington, DC, USA, 1996. IEEE Computer Society.

[35] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and A. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley Publishing Co., 2000.

[36] SETCo. SET Secure Electronic Transaction specification: Business description, 1997.

[37] F. J. Thayer-Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171, Oakland, CA, May 1998. IEEE Computer Society Press.

[38] J. R. Walker. Unsafe at any key size; an analysis of the WEP encapsulation. In *IEEE Document 802.11-00/362*, 2000.

APPENDIX

A. PCL PROOF SYSTEM EXTENSIONS

The details of the programming language, protocol logic and proof system in PCL have been described in previous work [13, 18]. Here we only explain the extra axioms introduced to capture the properties of the keyed hash function, which is used to compute the Message Integrity Code (MIC) and derive a fresh Pair-wise Temporary Key (PTK) in 802.11i.

HASH1	$\mathsf{Computes}(X, HASH_K(x)) \supset$
	$\mathsf{Has}(X, x) \wedge \mathsf{Has}(X, K)$
HASH2	$\mathsf{Computes}(X, HASH_K(x)) \supset$
	$\mathsf{Has}(X, HASH_K(x))$
HASH3	$\mathsf{Receive}(X, HASH_K(x)) \supset$
	$\exists Y.\mathsf{Computes}(Y, HASH_K(x))$
	$\wedge\mathsf{Send}(Y, HASH_K(x))$
HASH4	$\mathsf{Has}(X, HASH_K(x)) \supset$
	$\mathsf{Computes}(X, HASH_K(x)) \vee$
	$\exists Y, m. \quad \mathsf{Computes}(Y, HASH_K(x))$
	$\wedge\mathsf{Send}(Y, m) \wedge \mathsf{Contains}(m, HASH_K(x))$
$Define$	$\mathsf{Computes}(X, HASH_K(a)) \equiv$
	$\mathsf{Has}(X, K) \wedge \mathsf{Has}(X, a)$

B. PROOF OF THE 4-WAY HANDSHAKE GUARANTEE

In Section 3, we state the security of the 4-Way Handshake in the authenticator side as Theorem 1, which claims both the security properties, including *key secrecy* and *session authentication*, and the *invariants* of the 4-Way Handshake. Here we describe the details of the proof.

The *invariants* are verified by induction over basice sequences. When proving *key secrecy* of the 4-Way Handshake, we need to show that the secrecy of the key established by TLS is preserved. This is accomplished by proving that the NonceSource predicate is preserved after each step, and that the $\Gamma_{tls,2}$ invariant is satisfied by the 4-Way Handshake. The details of the proof can be found in our website. In the proof of *key secrecy*, we proved that the following property holds after each step of the 4-Way Handshake.

SPMK $\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \mathsf{Has}(\hat{Z}, pmk) \supset \hat{Z} = \hat{X} \vee \hat{Y}$

Using this formula, we will describe the proof details for *session authentication* as an example to show the general proof structures, stated as $\theta_{4way}[\mathbf{4WAY} : \mathbf{AUTH}]_X \phi_{4way,auth}$.

Recall that *session authentication* is represented as matching conversations; when the 4-Way Handshake is executed, the authenticator $\hat{X}$ can reason as follows:

1. Since the authenticator is honest, obviously it knows that actions of itself are in order; i.e., Send $Message$ 1, Receive $Message$ 2, Send $Message$ 3,Receive $Message$ 4 are matched, pepresented in line (1) of the proof;
2. Since the authenticator received and verified *Message 4*, there must be some entity $\hat{Z}$ who computes and sends out *Message 4* at previous stages, which implies $\hat{Z}$ must know the *ptk* used in the MIC to protect the message, indicated in line (2) and (3);
3. According to the invariants of the 4-Way Handshake and the properties of keyed hash, $\hat{Z}$ must be either the authenticator $\hat{X}$ itself or the supplicant $\hat{Y}$ since these are the only two parties who have the *ptk* in the system, described in line (4)-(6);
4. The authenticator knows it does not send out *Message 4* by itself; thus, it must be the supplicant who have computed and sent out *Message 4*. Furthermore, this occurs before the authenticator receives this *Message 4*, as shown in line (7)-(9);
5. The authenticator assumes that the honest supplicant acts honestly obeying the protocol. Therefore, the supplicant must has a sequence of Receive and Send actions in order, i.e., Receive $Message$ 1, Send $Message$ 2, Receive $Message$ 3, Send $Message$ 4, as in line (10);
6. The remaining task is to match the sequence of actions between the authenticator side and the supplicant side. Since the supplicant must have recevied and verified *Message 3* before sending out *Message 4*, the authenticator can similarly reason as in step 2, 3, and 4, and conclude that it must have sent out *Message 3* before the supplicant actually received it, as shown in line (11)-(15);
7. Due to the freshness of the nonce generated by the supplicant, the authenticator can only receive *Message 2* after the supplicant sends it. Similarly due to the freshness of the nonce generated by the authenticator, the supplicant can only receive *Message 1* after the authenticator send it, shown in line (16)-(19).

Based on these arguments, all the actions are matched as in line (20). Hence, the authenticator can conclude that the security property of *session authentication* is guaranteed in the 4-Way Handshake.

$\mathbf{AA1}, \mathbf{ARP}, \mathbf{AA4}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) <$

$\mathsf{Receive}(X, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$

$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$

$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (1)

$\mathbf{ARP}, \mathbf{HASH3}$ θ_{4way}

$[\texttt{receive } \hat{Y}, \hat{X}, z;$

$\texttt{match } z/\text{“}msg4\text{”}, mic2; \texttt{ match } mic2/HASH_{ptk}(\text{“}msg4\text{”})]_X$

$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})) \supset$

$\quad \exists Z.\mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge \mathsf{Send}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge$

$\quad (\mathsf{Send}(Z, HASH_{ptk}(\text{“}msg4\text{”})) <$

$\quad \mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})))$ (2)

$\mathbf{HASH1}$ $\mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \equiv \mathsf{Has}(\hat{Z}, ptk) \wedge \mathsf{Has}(\hat{Z}, \text{“}msg4\text{”})$ (3)

$\mathbf{HASH4}$ $\mathsf{Has}(\hat{Z}, ptk) \equiv \mathsf{Has}(\hat{Z}, HASH_{pmk}(x, y)) \supset$

$\quad \mathsf{Computes}(Z, HASH_{pmk}(x, y)) \vee$

$\quad (\exists Y, m.\mathsf{Computes}(Y, HASH_{pmk}(x, y)) \wedge$

$\quad \mathsf{Send}(Y, m) \wedge \mathsf{Contains}(m, HASH_{pmk}(x, y)))$ (4)

$(4), \Gamma_{4way,2}$ θ_{4way}

$[\texttt{receive } \hat{Y}, \hat{X}, z;$

$\texttt{match } z/\text{“}msg4\text{”}, mic2; \texttt{ match } mic2/HASH_{ptk}(\text{“}msg4\text{”})]_X$

$\mathsf{Has}(\hat{Z}, ptk) \equiv \mathsf{Has}(Z, HASH_{pmk}(x, y)) \supset \mathsf{Computes}(Z, HASH_{pmk}(x, y))$ (5)

$(5), \mathbf{HASH1}, \mathbf{SPMK}$ θ_{4way}

$[\texttt{receive } \hat{Y}, \hat{X}, z;$

$\texttt{match } z/\text{“}msg4\text{”}, mic2; \texttt{ match } mic2/HASH_{ptk}(\text{“}msg4\text{”})]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \supset$

$\quad \mathsf{Has}(\hat{Z}, ptk) \supset \mathsf{Has}(\hat{Z}, pmk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}$ (6)

$\mathbf{AA1}, \Gamma_{4way,3}$ θ_{4way}

$[\texttt{new } x; \texttt{ send } \hat{X}, \hat{Y}, x, \text{“}msg1\text{”};]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) \supset \hat{Z} \neq \hat{X}$ (7)

$(2), (6), (7)$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$

$\quad \exists Z.\mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge \mathsf{Send}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge \hat{Z} = \hat{Y}$ (8)

$(2), (8)$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$

$\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})) <$

$\quad \mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (9)

$(9), \phi_{\mathbf{HONESTY}}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$

$\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) <$

$\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$

$\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$

$\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (10)

$(10), \mathbf{HASH3}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) \supset$

$\quad \exists Z.\mathsf{Computes}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge$

$\quad \mathsf{Send}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge$

$\quad (\mathsf{Send}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) <$

$\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})))$ (11)

$\mathbf{HASH1}, (5), (6)$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Computes}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \supset \mathsf{Has}(\hat{Z}, ptk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}$ (12)

$\Gamma_{4way,3}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{Y}) \wedge \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) \supset \hat{Z} \neq \hat{Y}$ (13)

$(11), (12), (13)$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$

$\quad \exists Z.\mathsf{Computes}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge$

$\quad \mathsf{Send}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge \hat{Z} = \hat{X}$ (14)

$(11), (14)$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$

$\quad \mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$

$\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”}))$ (15)

$\mathbf{FS1}, \mathbf{AN3}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{Y}) \supset \mathsf{FirstSend}(Y, y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”}))$ (16)

$(16), \mathbf{FS2}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Send}(Y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$

$\mathsf{Receive}(X, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”}))$ (17)

$\mathbf{FS1}, \mathbf{AN3}$ θ_{4way}

$[\texttt{new } x; \texttt{send } \hat{X}, \hat{Y}, x, \text{“}msg1\text{”};]_X$

$\mathsf{FirstSend}(X, x, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”})$ (18)

$(18), \mathbf{FS2}$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) < \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”})$ (19)

$(1, 9, 10, 15, 17, 19)$ θ_{4way}

$[\mathbf{4WAY} : \mathbf{AUTH}]_X$

$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$

$\quad \mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) < \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) <$

$\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$

$\quad \mathsf{Receive}(X, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$

$\quad \mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$

$\quad \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$

$\quad \mathsf{Send}(Y, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})) <$

$\quad \mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (20)