Multi-protocol Attack: A Survey of Current Research

Nalin V. Subramanian
Iowa State University

Josh Dehliger
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports
Part of the Information Security Commons, and the OS and Networks Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/194

This Article is brought to you for free and open access by the Computer Science at Iowa State University Digital Repository. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Multi-protocol Attack: A Survey of Current Research

Abstract
Traditionally, verification methods for security protocols typically assume that the protocols are used in isolation of other protocols (i.e., there is only a single protocol using a network at a given time). However, in practice it is unrealisitc to assume that a security protocol runs in isolation in an insecure network. A multi-protocol attack is an attack in which more than one protocol is involved. The verification methods for security protocols that assume a single protocol on a network will fail to verify a protocol's resistance/vulnerability to multi-protocol attacks. Further, multiple security protocols that are verified to be correct in isolation can be susceptible to multiprotocol attacks when used over the same network. However, the verification of security properties for multiple protocols existing on the same network is difficult since security properties are not compositional. This paper surveys some of the recent approaches and contributions into the verification of security properties in the context of multiple protocols being run in an insecure network and the efforts to prevent multi-protocol attacks.

Keywords
Multi-protocol attack, chosen-protocol attack, protocol compositionality, security, verification, specification

Disciplines
Information Security | OS and Networks

This article is available at Iowa State University Digital Repository: http://lib.dr.iastate.edu/cs_techreports/194
Multi-Protocol Attacks: A Survey of Current Research

Nalin V. Subramanian  
Department of Computer Science  
Iowa State University  
nvsubram@iastate.edu

Josh Dehlinger  
Department of Computer Science  
Iowa State University  
dehlinge@iastate.edu

Abstract

Traditionally, verification methods for security protocols typically assume that the protocols are used in isolation of other protocols (i.e., there is only a single protocol using a network at a given time). However, in practice it is unrealistic to assume that a security protocol runs in isolation in an insecure network. A multi-protocol attack is an attack in which more than one protocol is involved. The verification methods for security protocols that assume a single protocol on a network will fail to verify a protocol’s resistance/vulnerability to multi-protocol attacks. Further, multiple security protocols that are verified to be correct in isolation can be susceptible to multi-protocol attacks when used over the same network. However, the verification of security properties for multiple protocols existing on the same network is difficult since security properties are not compositional. This paper surveys some of the recent approaches and contributions into the verification of security properties in the context of multiple protocols being run in an insecure network and the efforts to prevent multi-protocol attacks.

Keywords: Multi-protocol attack, chosen-protocol attack, protocol compositionality, security, verification, specification.

1. Introduction

Identifying and verifying the security properties and assumptions that may affect the security of protocols in insecure network is one of the most difficult design problems. Also to note, keys involved in the protocol are mainly assumed to be secured via some other protocol. It is very realistic to expect in real world scenarios that multiple users will be likely to have the same public/private key pairs for lot of their interactions with different systems.

Traditionally, verification methods for security protocols assume that the protocols are used in isolation of others (i.e., there is only a single protocol using a network at a given time, known as a mono-protocol system). Yet, in practice it is unrealistic to assume that a security protocol will run in isolation in an insecure network. Many protocols may be secure when executed in isolation, but there are possibilities in which they may lose their secure properties when executed in parallel with other protocols. Further, [12] claims:

Given a (provably) correct security protocol, there exists another correct security protocol, such that their composition is incorrect.

However, the verification of security properties for multiple protocols existing on the same network is difficult since security properties are not compositional. That is, any verification for multi-protocol systems must ensure that each protocol should not interact in a harmful manner with the other protocols used in the system [17].

A protocol used to break secure protocols is termed the chosen protocol, and this attack is called a chosen protocol attack. A multi-protocol attack is an attack in which more than one protocol is involved. A multi-protocol attack typically interleaves messages using two different protocols to target one of them. The protocol under attack is compromised by one of the following [1]:

- An incidental collision with another protocol
- A deliberately tailored protocol

A multi-protocol attack can affect users using the same computer or smart card. Note, in literature and throughout this paper, the terms “multi-protocol attack” and “chosen protocol attack” are used interchangeably.

This paper surveys and presents some of the recent approaches and contributions into the verification of security properties in the context of multi-protocol attacks.
The rest of this paper is organized as follows: Section 2 defines a multi-protocol attack as per the research literature. After discussing generalities, several specific examples of a multi-protocol attack are given in Section 3. In Sections 4 and 5, several proposed specification and verification techniques addressing multi-protocol attacks are discussed. Section 6 provides a brief summary of the proposed preventative steps for a multi-protocol attack. Section 7 discusses various aspects addressed in the literature that appear to render the chosen protocol attack. Finally, Section 8 provides some concluding remarks.

2. Protocol Interactions and Multi-Protocol Attacks

Protocol interactions occur when some information from a protocol, $P$, can allow an adversary to mount an attack over another protocol, $Q$. For example, a replay attack over one protocol using the messages from another protocol, assuming the protocols are of the same form. Protocol interactions reduce to the subset of possible attacks on the protocol, as in the man-in-the-middle attack above. Ways in which two different protocols may interact include:

- Third party may observe the messages in $P$, and attack using $Q$
- A user, $C$, impersonating another user, $B$, to a third user, $A$, using $P$ to get information. In parallel, $C$ uses the information as $B$ with $C$ using protocol $Q$
- $C$ interacts as a legitimate user with $B$ using protocol $P$, and simultaneously attack $A$ as $B$ in protocol $P$. As long as the attack on either $P$ or $Q$ is made possible, the protocols interact

A chosen-protocol is an existing (or tailor-made) protocol designed (may be specifically) to interact with some already-running secure protocol, called the target protocol. The chosen protocol, the protocol used to attack the target protocol, remains secure in isolation, but allows an attack on the target protocol. To build such a protocol, if there are no restrictions on the allowed steps, consider a target protocol that uses a private key to sign and decrypt. Then, any protocol that gives the attacker a decryption oracle and a signing oracle will compromise the target protocol as long as both the chosen and target protocols share the same key.

There is no fixed definition of what a reasonable chosen protocol should be, but this does not prove such a protocol could not exist. Any protocol that is not susceptible to message-based attacks is highly likely to be immune to a multi-protocol attack, but still it is not complete. The following provides some realistic scenarios for multi-protocol attacks:

1. Different products and protocols may use the same cryptographic keys
2. Infiltration of lower-security products’ protocols may not be as carefully overseen as those of higher-security protocols
3. Custom-modified protocols used in commercial products may be used to install a chosen protocol for this class of attack: such protocols may end up being adopted later on as a widely-used standard

3. Multi-Protocol Attack Examples

This section describes four examples of a multi-protocol attack including: attacks against public-key signatures, attacks against a symmetric-key authentication protocol, attacks against the Distributed Authentication Security Service (DASS) public-key protocol, and attacks against combined services.

3.1. Attacks Against Public-Key Signatures

Based on the draft SSL-3 protocol [21], a target protocol, as shown in Figure 1, is constructed. Using an RSA certificate, a server, $B$, authenticates itself to establish master encryption key pairs ($K$-public, $K$'-private) for subsequent communication. The protocol then proceeds as follows:

1. Client $A$ sends hello message to server $B$ that contains a list of possible cryptographic and compression routines along with an initial random number.
2. Server $B$ responds with a server hello message that contains information similar to the client hello message, except that it indicates specific cryptographic and compression routines to be used.
3. Server $B$ then sends a copy of its certificate(s) (e.g., X.509 certificates) to client $A$.
4. In this version of the protocol, $B$ then sends to $A$ a plain text copy of a temporary RSA public key (i.e., a modulus and exponent pair) and a signed hash of that key.
5. The server, $B$, then sends a server hello done message to $A$. 
6. The client, A, sends a secret message (i.e., a pre-master key) encrypted with the server’s temporary RSA public key.
7. The server responds with a server finished message that is a hash of the exchanged information encrypted with the values just negotiated.

Message 1. \( A \rightarrow B : \text{Client Hello} \)
Message 2. \( B \rightarrow A : \text{Server Hello} \)
Message 3. \( B \rightarrow A : \text{Server Certificate} \)
Message 4. \( B \rightarrow A : K.\{H(K)\}_{PKS(B)} \)
Message 5. \( B \rightarrow A : \text{Server Hello Done} \)
Message 6. \( A \rightarrow B : \{M\}_K \)
Message 7. \( B \rightarrow A : \{\text{Server Finished}\}_K \)

Figure 1. Proposed SSL-Like Target Protocol

The client uses the server’s certificate and the signed hash to authenticate the origin of the temporary public key. Thus, an attacker who tries to replay an old Message 4 from the server will not be able to read Message 6 from the client.

Message i. \( A \rightarrow B : A.B.\{M\}_{PKS(B)} \)
Message ii. \( B \rightarrow A : B.A.\{H(M)\}_{PKS(B)} \)

Figure 2. Tailored Protocol

Figure 2 shows a protocol, which may be designed by an application programmer or provided by a network service, to develop a certified message receipt mechanism, or it could be a protocol deliberately tailored to break other authentication protocols. This protocol consists of the following two steps:

1. \( A \) sends some data value, \( M \), to \( B \) encrypted by \( B \)’s RSA public key
2. \( B \) responds to \( A \) with its signature (in terms of hash) which is encrypted by \( B \)’s RSA private key \( PKS(B) \) to authenticate receipt of the first message

It is acceptable to believe that similar forms of messages can occur in different secure protocols. Note that the tailored protocol obtains a signature of a server for any of the given data. The attacker can use this protocol to break the SSL-like protocol. For example, as shown in Figure 3, an attacker can obtain the server’s signature over any of its temporary RSA public keys using a tailored protocol. An attacker can replay the same message in an SSL-like protocol masquerading as the valid server, since Message 4 is used to authenticate the server’s message. Since the attacker can now use the temporary RSA key it created, it can see any messages sent by \( A \) to a server, where the attacker acts as a valid server.

Similarly, an attack can be raised over a modified Agora protocol [9], which is commonly used as an electronic payment protocol [9].

<table>
<thead>
<tr>
<th>Proposed SSL-Like Protocol</th>
<th>Tailored Protocol 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message i. ( A \rightarrow B : A.B.{M=K}_{PKS(B)} )</td>
<td>Message i. ( A \rightarrow B : B.A.{H(M=K)}_{PKS(B)} )</td>
</tr>
<tr>
<td>Message ii. ( B \rightarrow A : B.A.{H(M=K)}_{PKS(B)} )</td>
<td>Message ii. ( B \rightarrow A : B.A.{H(M=K)}_{PKS(B)} )</td>
</tr>
</tbody>
</table>

Message 1. \( A \rightarrow E_B : \text{Client Hello} \)
Message 2. \( E_B \rightarrow A : \text{Server Hello} \)
Message 3. \( E_B \rightarrow A : \text{Server Certificate} \)
Message 4. \( E_B \rightarrow A : K'.\{H(K')\}_{PKS(B)} \)
Message 5. \( E_B \rightarrow A : \text{Server Hello Done} \)
Message 6. \( A \rightarrow E_B : \{M\}_K \)
Message 7. \( E_B \rightarrow A : \text{Server Finished} \)

Figure 3. attack over SSL-like protocol

3.2. Attacks Against Symmetric-Key Authentication Protocols

Consider the symmetric key based simple secure protocol MAP1:

1. \( A \rightarrow B : R_A \)
2. \( B \rightarrow A : \{B.A.R_A.R_B\}_a \)
3. \( A \rightarrow B : \{A.R_B\}_a \)

Also consider the two tailored protocols EVE1, EVE2:

**Protocol EVE1**

1. \( A \rightarrow B : R_A \)
2. \( B \rightarrow A : \{A.B.R_A.R_B\}_a \)
3. \( A \rightarrow B : \{A.R_B\}_a \)

**Protocol EVE2**

1. \( A \rightarrow B : \{A.B.R_A.R_B\}_a \)
2. \( B \rightarrow A : \{A.R_B\}_a \)

Below, we can see how the EVE1 protocol can be exploited to attack the MAP1 protocol at one side in the following:

**Protocol MAP1**

1. \( A \rightarrow E_B : R_A \)
2. \( E_B \rightarrow A : R_A \)
3. \( A \rightarrow E_B : \{B.A.R_A.R_B\}_a \)

**Protocol EVE1**

1. \( E_B \rightarrow A : R_A \)
2. \( A \rightarrow E_B : \{B.A.R_A.R_B\}_a \)
3. \( A \rightarrow E_B : \{A.R_B\}_a \)
Similarly, we can see how the EVE2 protocol can be exploited to attack the MAP1 protocol at the other side in the following:

\[
\text{Protocol MAP1} \quad \text{Protocol EVE2}
\]

1. \(E_A \rightarrow B : R_A\)
2. \(B \rightarrow E_A : [B \cdot A \cdot R_A : R_B]_a\)
3. \(E_A \rightarrow B : [A \cdot R_B]_a\)

### 3.3. Attacks Against the Distributed Authentication Security Service Public-Key Protocol

The Distributed Authentication Security Service (DASS) is a commercial protocol for mutual authentication and key exchange developed by Digital Equipment Corporation and marketed in a product called SPX [18].

**Protocol E:** DASS (the Target Protocol) [18].

A chosen protocol is built by adding some additional functionality to the system. In this case, to have Trent generate random public keys, a new protocol must be added as needed.

**Protocol F:** Protocol for Requesting a Temporary Public Key (the Chosen Protocol)

The attack then proceeds as follows:

1. Alice forms
   - \(R_0\) = a random number the same length as an ID, and sends to Trent
   - \(M_0 = \text{Request for } PK; ID_A; R_0\).
2. Trent generates a new public key, \(PK_T\), forms
   - \(K_1\) = a random encryption key, and sends back
   - \(M_1 = PKE_{PK_A}(K_1); E_{K_1}(SK_T); SIGN_{SK_T}(R_0; PK_T)\).
3. Mallory now encrypts a random session key under Bob's public key, including the timestamp, key lifetime, and Alice's ID. All this is exactly as appears in the third message of DASS.
4. Bob sends \(ID_A\) to Trent
5. Mallory intercepts this request. She sends back \(SIGN_{SK_T}(R_0; PK_T)\), recovered from the second message in the chosen protocol.
6. Bob decrypts this message, and uses \(PK_T\) to verify the signature on the first message sent to Bob. Bob is now convinced he shares a secret symmetric key with Alice, when in fact, he shares it with Mallory instead.

### 3.4. Attacks on Combined Services

This section provides an example of a multi-protocol attack over combined services that use different protocols but still interact. Figure 4 and 5 provides an example of such an attack. In figure 4, the Protocol P be any protocol used for authenticating partners and generating fresh shared secret. Service 1, is an extended protocol as shown. Provided Protocol P is secure, we can also prove that Service 1 protocol is correct. Let the Service 2 be another protocol which reuses the same protocol P, but we extend it by sending a fresh session identifier \(x\) (generated by P) and some message \(m\).

If we run Service 1 in parallel with Service 2, the combined protocols are broken as shown in Figure 5. In this attack, the intruder simply re-routes the initial messages from Service 1 to Service 2. After this initial phase, \(a\) is halfway into Service 1, and \(b\) is halfway into Service 2. Therefore \(b\) will now use the random value \(x\) as a session identifier, effectively revealing it to the
intruder. Then, when $a$ uses this value $x$ as a session key for the secret $Y$, the intruder can decrypt it. Thus the security claim of Service 1 is violated.

Expressing the specifications, properties, and requirements in a concise way is an important research challenge. Once such requirements in the context of protocols can be specified, small protocols can be considered as building blocks for larger composed protocols. Another level of abstraction can be introduced once a set of small protocols are properly specified to serve as building blocks [7].

5. Model Checking and Verification Techniques for Multi-Protocol Systems

Very little work has specifically examined the formal verification of security properties in multi-protocol systems [17]. This is mainly because the problem of protocol composability has been presented as one of the most challenging issues faced in formal methods (which are applied to protocol verification of multi-protocol systems) [16]. This section discusses recent work in formal methods in modeling checking and other suggested verification techniques to analyze multi-protocol attacks in multi-protocol systems.

5.1 Model Checking

Panti et al. [17] extended a verification methodology that was originally used for security protocols in isolation to analyze multi-protocol systems using the NuSMV model checker. To model a multi-protocol system as a collection of finite state machines in NuSMV, the following must be described in the model:

- The behaviors toward the given protocols of participating non-malicious agents
- The behaviors of the malicious agents that acts to compromise the protocols
- The communication channel used by all the agents to exchange the messages

Each non-malicious agent, $X$, in the multi-protocol system can participate using different protocols, $P_i$, and play different roles using the same protocol or different protocols, $P_j$. For example, $X$, may play a customer role in a session of $P_i$ and be a merchant in a session of $P_j$. If trusted third parties are involved (e.g., authentication servers, payment gateways, etc.), they must only act in a third party role using a single protocol. The roles are all classified in one of the following two categories:

- Initiators ($init$) – those roles that start a protocol execution
Responders (resp) – those roles that are engaged in a protocol execution by an initiator

Modeling a malicious agent, \( I \), requires the identification and documentation of relevant assumptions such as:

- \( I \) is considered a legitimate member of the system
- \( I \) has the same capabilities as the non-malicious users of the multi-protocol system
- \( I \) can send and receive messages according to all protocols in \( P_i \)
- \( I \) can be as powerful as possible (to allow for the verification of the protocol assuming a worst case scenario)
- \( I \) may have complete control over the communication channel:
  - Ability to eavesdrop on all the messages sent over the communication channel from any agent to any agent within the system
  - Ability to store messages eavesdropped from the communication channel in its memory
  - Ability to remove a message sent over the communication channel from any agent to any agent within the system
  - Ability to send messages to agents of the system impersonating any agent of the system (even itself)

For a multi-protocol system to be verified using such a model-checking approach, the following security properties are vital [17]:

- **Authentication** – the parties convince each other of their identities
- **Confidentiality** – nobody but the authorized parties must know a term
- **Integrity** – a term must not be altered by an unauthorized agent

To formalize these three security properties, [17] introduces a correspondence property to generalize and capture these properties for the use of formal verification. The correspondence property is as follows:

If an event, \( E \), occurs \( n \) times, then another event, \( F \), must have occurred at least \( n \) times.

and can be represented as a CTL property in NuSMV as follows:

\[
AG(x.\text{counter}, \geq Y.\text{counter})
\]

Authentication is represented as the following properties, expressed in CTL in NuSMV:

\[
AG(x.\text{begin}(P_i, \text{init}, Y) \geq Y.\text{end}(P_i, \text{resp}, X)) \\
AG(x.\text{begin}(P_i, \text{resp}, Y) \geq Y.\text{end}(P_i, \text{int}, X))
\]

Confidentiality is represented as the following properties, expressed in CTL in NuSMV:

\[
AG(x.\text{send}(P_i, I, \text{term}) \geq I.\text{receive}(P_i, X, \text{term})) \\
AG(x.\text{send}(P_i, Y, \text{term}) \geq Y.\text{receive}(P_i, X, \text{term}))
\]

Integrity for a confidential term is expressed in CTL in NuSMV as

\[
AG(x.\text{send}(P_i, I, \text{term}) \geq I.\text{receive}(P_i, X, \text{term})) \\
AG(x.\text{send}(P_i, Y, \text{term}) \geq Y.\text{receive}(P_i, X, \text{term}))
\]

However, integrity for terms transmitted in clear text is only expressed as

\[
AG(x.\text{send}(P_i, Y, \text{term}) \geq Y.\text{receive}(P_i, X, \text{term}))
\]

in CTL in NuSMV.

Panti et. al. [17] used such a model to verify these security properties over a series of multi-protocol systems, shown in Table 1, including analyzing the interactions in the following groups: the Neelham-Schroeder public-key protocol (NS) and the Signed Receipt protocol; the Agora protocol [9] and the Age Verification protocol; the SET Purchase protocol, the SET Authorization protocol, the SET Payment Phase protocol, and the 2KP Payment Phase protocol.

Their modeling approach was able to identify known security vulnerabilities in the interaction of the NS and Signed Receipt protocols as well as the Agora and Age Verification protocols [9]. In addition, it discovered an unknown vulnerability in the interaction of the Agora and Age Verification protocols. For the SET Purchase protocol, the SET Authorization protocol, the SET Payment Phase protocol, and the 2KP Payment Phase protocols, their modeling approach verified the previously described security properties.

Also of significance, this approach produces results, as the authors claim, in an encouragingly efficient manner. In the best case, their approach needed about one hour to verify the Agora and Age Verification protocols that had nearly 27 million states. In the worst case, their approach required approximately three days to verify a system with a state-space having nearly \( 10^{28} \) states.
5.2 Verification Techniques

Cremers and Mauw [7] proposed a generic canonical model for the analysis of security protocols. Their model gives static requirements for valid protocols and is parametric with respect to the matching function and the intruder’s network capabilities. Specifically, to analyze a multi-protocol system, this work simply adds additional role descriptions to the model for each additional protocol or for each additional protocol role. The security properties to be verified in a multi-protocol system are defined as local claims and local constants are utilized and bound to runs, assisting in the construction of proofs to the security properties.

Like [7], [17], describes the behavior of a protocol in terms of its roles: either an initiator or a responder. A multi-protocol system contains a number of communicating agents (entities) that may take on one or more roles. A role performed by an agent is called a run. Thus, an agent, $A$, may perform two initiator runs and one responder run in parallel. Each run executed by an agent of the system is meant to achieve some security goal. For example, the agent may desire the confidential exchange of a message to another agent during a run. While an agent is pursuing a security goal in a run, an intruder may try to oppose such a security goal. An intruder’s capabilities determine its strength in attacking a protocol run. Using this general description, [7] constructs a global description of a security model consisting of the following components:

- **Protocol specification** – describes the behavior of each of the roles in the protocol, most often as a sequential list of events
- **Agent model** – a model of the agents that execute the roles of the protocol based on a closed world assumption (i.e., the description of an honest agent that only includes the behavior of the agent as described in the protocol specification)
- **Communication model** – describes how the messages between the agents are exchanged (in an asynchronous manner)
- **Threat model** – a parameter in the semantics of the model (to accommodate all types of network communications and threats, e.g., wireless communication) based on Dolev and Yao’s network threat model [19]
- **Cryptographic primitives** – idealized mathematical constructs (e.g., encryption) using a black-box approach to only describe the relevant properties
- **Security requirements** – expressed as safety properties

Note, the work described in [19] only considers secrecy and authentication as security properties to validate.

$$\begin{align*}
\text{create} & : \text{run} = ((\text{init}, p, a), s) \rightarrow \text{run}(p), r \neq \text{send}(p), (M, BS, BR, F) \vdash (M, BS, BR, F, (\text{run})) \\
\text{send} & : \text{run} = ((\text{init}, \text{send}(p), a), s) \rightarrow F, (M, BS, BR, F) \vdash (M, BS, BR, F, (\text{init}, s, \text{run})) \\
\text{read} & : \text{run} = ((\text{init}, \text{read}(p), m), a) \rightarrow F, m \in BR, (M, BS, BR, F) \vdash (M, BS, BR, F, (\text{init}, s, \text{run})) \\
\text{claim} & : \text{run} = ((\text{init}, \text{claim}(e, c), a), s) \rightarrow F, (M, BS, BR, F) \vdash (M, BS, BR, F, (\text{init}, s, \text{run}))
\end{align*}$$

Figure 6. SOS Rules

Figure 6 details the formal derivation rules for a system. The create rule details that a new run can only be created if its run identifier has yet to be used. The send rule states that if a run executes a send event, the sent message is added to the output buffer and the executing run can proceed to the next event. The read rule determines when an input event can be executed. The claim rule express that an enabled claim event can always be executed.

Figure 7 details the some of the formal intruder rules used. The transmit rule describes the transmission of a message from the output buffer to the input buffer without interference from an intruder. The deflect rules says that an intruder with deflection capabilities can delete any message from the output buffer. The inject rule describes that the injection of any message inferable from the intruder’s knowledge into an input buffer. The eavesdrop message details how an intruder can learn a message during the transmission between two agents. The jam rule states that an intruder can read the deflected messages and add it to its knowledge.
6. Prevention Design Techniques

This section points out the viable environment for attack and prevention techniques. The three conditions that enable these attacks are [1], [5]:

1. Use of private/public key in more than one protocol or various services which use different protocols.
2. Chosen protocol (tailored or authentic) should be installed on the target machine and should have access to cryptographic services used by the target protocol.
3. Presence of matching patterns: this result in type flow attacks.

The mitigation of these attacks through additional requirements/specifications placed upon a security system is possibly a viable solution to preventing multi-protocol attacks. The remainder of this section discusses how to thwart these attacks. Kelsey, Schneier, and Wagner [12] presented five design principles for protocols that appear to render the chosen protocol attack impossible, but [3] and [15] claim that these five principles alone do not suffice to ensure the prevention of multi-protocol attack unless there is strong underlying support for such mechanisms. To address condition 1, and principle 1, use of a public key to a specific set of protocols should be limited. To summarize:

- Limiting the scope of the key reduces the impact addressed in the first condition above. Implementation of this restriction is not cost effective, and the commercial market does not restrict the use of certified keys. As long as the application has a handle for the key, it can use it. The API's have to be designed to securely manage key use.

- Unique protocol identifiers in the PKI certificate, along with universal protocol numbering and a secure mechanism should be included for a protocol implementation to prove to the system that it indeed implements the specified protocol. However, this will not be sufficient if the tailored protocol doesn’t follow this step.

- Include a fixed unique identifier in a fixed place in the authenticated protocol.

- A malicious user application can use subroutines in the protocol to implement any tailored protocol. Support for protocol validation in a cryptographic subsystem should be provided. Unfortunately, this forces all protocols to be installed within the cryptographic subsystem.

- Mapping the unique identifier to encryption forces the identifier to be used for successful decryption. Blind signing of protocols can be prevented from being used to decrypt secret messages.

- Cryptographic subsystem support for key limitations, where the user specifies which protocols can use the specified keys. This places a large portion of the burden on the user, who needs to understand the full ramifications of key sharing.

- Use the certified public key to certify new public keys that are generated for a single use (as in [18]). The certified public key is then used in only the key distribution protocol, and each new public key is used only once. A multi-protocol attack may still be possible depending on the content and format of the newly distributed public-key certificate.

- Each run of the protocol should be identified uniquely, in conjunction with the cryptographic subsystem support (ID should be provided by subsystem and not by protocol user); otherwise a protocol can be tailored to use the same identifiers.

- Validation of flaw-free composition of groups of protocols (all possible combination should be considered which could form a complete system, not just 2-protocol compositions)

- Smartcard or user device support.

Multi-protocol attacks will still be possible if these restrictions and conditions are not enforced and validated [15].
7. Discussion – The Feasibility of a Multi-Protocol Attack

Analysis of cryptographic protocols can be described in terms of a single sequence of messages, but they are actually implemented as a suite of sub-protocols. It is necessary to show that the sub-protocols do not interact in a way conducive to multi-protocol or other attacks.

Multi-protocol attacks are not only a theoretical concern. The early version of SSL reveals part of the key, it was not always clear by inspection of the key whether weak or strong cryptography was being used; thus allowing a multi-protocol attack [1], [2]. This serves the example for a confusion of the reconnection protocol with the connection protocol, which can lead to an attack.

Although the problems addressed in [1], [5] and [18] do not necessarily occur as often as other types of flaws [5], [6], they are prevalent enough that it is important to demonstrate the collection of protocols is free from insecure interactions. But it is very hard to show all the possible interactions that are secure in the combination of protocols. [1], [12] have shown it is possible to build an attack protocol for a given protocol. It is necessary at least to demonstrate that a suite of protocols will not accept message sent by another protocol. This has a serious practical impact on the verification of cryptographic protocols [3], [6].

Heintze’s work [11] on composability concentrated on determining under what conditions protocols could be guaranteed to be composable with each other. More recent work [7], [17] has concentrated on taking each state transition that required an input message and determining which transitions could produce output identifying such attacks. The authors have taken initiatives for the specification, verification, and model checking for such kinds of attacks.

Thayer and Guttman [10] used Strand Space to show the secure composition and to analyze the properties. They used generation of unique terms for each protocol used in combination, and proved combination of protocols are secure. The authors extended their work by guaranteeing security by the usage of restricted exceptions. Work on Mixed Strand Space showed the extension towards specific applications. In particular, [8], [13] have shown theoretically that any protocol that is secure in isolation remains secure even in case of composition, provided that all the assumptions and properties specified by each protocol holds good after composition. In [13], authors also suggests that multi-protocol attack will not exists in case of Non-destructive combinations of protocols, which can be achieved by verification of properties of every individual protocols by adding the steps of other protocols (and vice versa) But not all the protocols implemented in composition are verified together, leading to the possibility of multi-protocol attack. As far as the verification of all protocols in a suite is assured to be secure, it is still not completely secure in composition [15].

Table 2. Protocol List

<table>
<thead>
<tr>
<th>Protocols with Multi-Protocol Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilateral Key Exchange</td>
</tr>
<tr>
<td>Boyd key agreement</td>
</tr>
<tr>
<td>Denning-Sacco shared key</td>
</tr>
<tr>
<td>Gong (nonce) v2</td>
</tr>
<tr>
<td>ISO ccitt 509 (BAN)</td>
</tr>
<tr>
<td>Kao-Chow v2</td>
</tr>
<tr>
<td>Kao-Chow v3</td>
</tr>
<tr>
<td>Kao-Chow v3</td>
</tr>
<tr>
<td>KSI</td>
</tr>
<tr>
<td>Needham-Schroeder</td>
</tr>
<tr>
<td>Needham-Schroeder-Lowe</td>
</tr>
</tbody>
</table>

Protocols for which we found no Multi-Protocol Attacks

<table>
<thead>
<tr>
<th>Protocols for which we found no Multi-Protocol Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew Secure RPC (BAN)</td>
</tr>
<tr>
<td>Andrew Secure RPC (Lowe)</td>
</tr>
<tr>
<td>ISO IEC 1779 2-15</td>
</tr>
<tr>
<td>TNM</td>
</tr>
</tbody>
</table>

*Modified version 1                                     |
*Modified version 2 (Clark and Jacobs)

8. Concluding Remarks

Experiments from [6], shown in Table 2, addresses 163 multi-protocol attacks. From a study of the literature, existence of multi-protocol attacks on protocols in large numbers is feasible. Cremers finds attacks that were previously unreported, which leads to a much larger threat than previously assumed [5], [6]. This is not limited to a small subset of the protocols. Out of the 30 protocols examined [6], 23 were found that had security claims that are correct in isolation but were susceptible to multi-protocol attacks.

A survey of the literature also shows that multi-protocol attacks exist now on combinations of protocols for which no attacks were known previously. Composition of protocols which claimed to be secure and have no attack, recently were found to have a 3-protocol attack, as per the experiments conducted in [5]. Without tool support, it becomes harder to find such kinds of attacks. Feasibility of such attacks had been found using the formal models and tools, which helps to run large scale test cases. Single protocols can cause other protocols to be susceptible to malicious
attacks. Proof of correctness for a protocol in isolation is not sufficient to make the protocol secure, nor to justify the claim of security. Proof of correctness in composition should be tested before usage.

9. Acknowledgements

This survey was completed to meet the final project requirement of COM S 610 - Seminar on Security Properties of Software Systems led by Dr. Hridesh Rajan, offered by the Department of Computer Science at Iowa State University during the Fall 2006 semester. The authors would like to thank Dr. Rajan for his guidance throughout the semester. Authors like to thank the support and directions given by Dr. Cas Cremers, ISG, ETH Zurich for understanding the Scyther tools, which helped the authors in checking the existence of multi protocol attack in tailored protocols. Also authors like to thank and appreciate Nate Brenneman, Harish Narayanamma, Youssef Hanna, Santosh Panchapakesan for helping in reviewing and editing the paper.

10. References


Appendix A: Scyther Tool

We have specified the protocols and verified the feasibility of Multi-Protocol Attack using the Scyther Tool [20]. We give a brief description about the tool and results in the Appendix here. Scyther is an automated security protocol verification tool, which can verify protocols with an unbounded number of sessions and nonces. The semantics of protocol execution, security properties and input language are based on [7]. The algorithm used is an extended
version of the method used by the Athena tool by X.D. Song. Scyther tool is used because even when used with a bound on the number of sessions, it can decide (i.e. prove or disprove) a large class of protocols. Multi-protocol analysis is handled in an intuitive way (concatenation of input files). It can verify ordering-related properties such as synchronisation. Multiple key infrastructures (PKIs) can be modeled.

Appendix B: Protocol Specifications

```plaintext
/* ns3.spdl */
* Needham-Schroeder protocol */

// PKI infrastructure
const pk: Function;
secret sk: Function;
inversekeys (pk, sk);

// The protocol description
protocol ns3(I,R) {

role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I, R, {I, ni} pk(R));
    read_2(R, I, {ni, nr} pk(I));
    send_3(I, R, {nr} pk(R));
    claim_i1(I, Secret, ni);
    claim_i2(I, Secret, nr);
    claim_i3(I, Niagree);
    claim_i4(I, Nisynch);
}

role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I, R, {I, ni} pk(R));
    send_2(R, I, {ni, nr, R} pk(I));
    read_3(I, R, {nr} pk(R));
    claim_r1(R, Secret, ni);
    claim_r2(R, Secret, nr);
    claim_r3(R, Niagree);
    claim_r4(R, Nisynch);
}

// An untrusted agent, with leaked information
const Eve: Agent;
untrusted Eve;
compromised sk(Eve);
```

```plaintext
/* ns3.broken.spdl */
* Tailored Needham-Schroeder-Lowe protocol */

// PKI infrastructure
const pk: Function;
secret sk: Function;
inversekeys (pk, sk);

// The protocol description
protocol ns3(I,R) {

role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I, R, {I, ni} pk(R));
    read_2(R, I, {ni, nr} pk(I));
    send_3(I, R, {nr} pk(R));
    claim_i1(I, Secret, ni);
    claim_i2(I, Secret, nr);
    claim_i3(I, Niagree);
    claim_i4(I, Nisynch);
}

role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I, R, {I, ni} pk(R));
    send_2(R, I, {ni, nr, R} pk(I));
    read_3(I, R, {nr} pk(R));
    claim_r1(R, Secret, ni);
    claim_r2(R, Secret, nr);
    claim_r3(R, Niagree);
    claim_r4(R, Nisynch);
}

// An untrusted agent, with leaked information
const Eve: Agent;
untrusted Eve;
compromised sk(Eve);
```

```plaintext
/* nsl3-spdl */
* Needham-Schroeder-Lowe protocol */

// PKI infrastructure
const pk: Function;
secret sk: Function;
inversekeys (pk, sk);

// The protocol description
protocol nsl3(I,R) {

role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I, R, {I, ni} pk(R));
    read_2(R, I, {ni, nr, R} pk(I));
    send_3(I, R, {nr} pk(R));
    claim_i1(I, Secret, ni);
    claim_i2(I, Secret, nr);
    claim_i3(I, Niagree);
    claim_i4(I, Nisynch);
}

role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I, R, {I, ni} pk(R));
    send_2(R, I, {ni, nr} pk(I));
    read_3(I, R, {nr} pk(R));
    claim_r1(R, Secret, ni);
    claim_r2(R, Secret, nr);
    claim_r3(R, Niagree);
    claim_r4(R, Nisynch);
}

// An untrusted agent, with leaked information
const Eve: Agent;
untrusted Eve;
compromised sk(Eve);
```

```plaintext
/* nsl3-broken.spdl */
* Tailored Needham-Schroeder-Lowe protocol,
* broken version (wrong role name in first message) */

// PKI infrastructure
const pk: Function;
secret sk: Function;
inversekeys (pk, sk);

// The protocol description
protocol nsl3(I,R) {

role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I, R, {I, ni} pk(R));
    read_2(R, I, {ni, nr} pk(I));
    send_3(I, R, {nr} pk(R));
    claim_i1(I, Secret, ni);
    claim_i2(I, Secret, nr);
    claim_i3(I, Niagree);
    claim_i4(I, Nisynch);
}

role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I, R, {I, ni} pk(R));
    send_2(R, I, {ni, nr} pk(I));
    read_3(I, R, {nr} pk(R));
    claim_r1(R, Secret, ni);
    claim_r2(R, Secret, nr);
    claim_r3(R, Niagree);
    claim_r4(R, Nisynch);
}

// An untrusted agent, with leaked information
const Eve: Agent;
untrusted Eve;
compromised sk(Eve);
```
Appendix C: Results of Verification

Appendix B gives the protocol specification for the three protocols (Needham-Schroeder, Needham-Schroeder-Lowe, and tailored Needham-Schroeder-Lowe) which has been composed and verified using the Scyther tool. The result shows that the Multi-Protocol Attack is feasible in the composition. Also we have presented here one failed traces per protocol.
Fig. 8. Results of the failed verification.

<table>
<thead>
<tr>
<th>Claim</th>
<th>Status</th>
<th>Comments</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>m01</td>
<td>Secret</td>
<td>Fail</td>
<td>at least 1 attack</td>
</tr>
<tr>
<td>m02</td>
<td>Secret</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>m03</td>
<td>Message</td>
<td>Fail</td>
<td>at least 1 attack</td>
</tr>
<tr>
<td>m04</td>
<td>Message</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>R</td>
<td>m01</td>
<td>Secret</td>
<td>Fail</td>
</tr>
<tr>
<td>m02</td>
<td>Secret</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>m03</td>
<td>Message</td>
<td>Fail</td>
<td>at least 1 attack</td>
</tr>
<tr>
<td>m04</td>
<td>Message</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Secret</td>
<td>Fail</td>
<td>at least 1 attack</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Secret</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Message</td>
<td>Fail</td>
<td>at least 1 attack</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Message</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>R</td>
<td>m00Broken</td>
<td>Secret</td>
<td>Fail</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Secret</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Message</td>
<td>Fail</td>
<td>at least 1 attack</td>
</tr>
<tr>
<td>m00Broken</td>
<td>Message</td>
<td>Fail</td>
<td>1 attack</td>
</tr>
</tbody>
</table>

Fig. 9. Trace for the Multi-Protocol attack on Needham-Schroeder protocol against the claim r4.
Fig. 10. Trace for the Multi-Protocol attack on Needham-Schroeder-Lowe protocol against the claim r3.

Fig. 11. Trace for the Multi-Protocol attack on Tailored Needham-Schroeder-Lowe protocol against the claim r1.