

Analyzing Internet Security Protocols

Alec Yasinsac, Justin Childs
Florida State University Computer Science Dept.
yasinsac@cs.fsu.edu, childs@cs.fsu.edu

Abstract

In this paper, we show how a novel tool for analyzing classical cryptographic protocols can be used to model and analyze the more complex Internet security protocol families. We discuss the modifications that were necessary in the tool and how the tool illuminates flaws in the Transport Layer Security (TLS) protocol. We also show how a flaw carried over to TLS from the Secure Sockets Layer protocol (the predecessor of TLS) was illustrated using our tool.

1. Introduction

The seminal paper by Needham and Schroeder [11] triggered the field of cryptographic protocol analysis in 1978. For twenty years, that analysis focused on simple, serial protocols in a structured and well-understood environment (e.g. [2]). This analysis has been very successful in identifying flaws in well-known and widely used protocols. In addition, much was learned about the nature of cryptographic protocols and characteristics that reflect strengths and weaknesses [1], [13]. Automatic analysis tools, such as the NRL Protocol Analyzer, developed by Meadows [9] and the AAPA2 developed by Brackin [3], have extended these ideas to software tools that can aid in the analysis of cryptographic protocols.

In the past five years, these simple protocols have given way to more complex protocol suites, suitable for E-Commerce applications and general use on the Internet. These protocols have characteristics that make use of early analysis methods difficult or impossible. For example, early analysis of cryptographic protocols by formal methods is limited to protocols where the actions of individual parties cannot be changed by actions within the protocol.

Many security protocols in use or proposed for widespread use, such as Secure Electronic Transaction (SET), Internet Key Exchange (IKE), Secure Sockets Layer (SSL), and TLS, offer optional sub-protocols that are agreed upon by protocol participants during protocol

execution. Interactions between the sub-protocols may allow the protocol to be subverted. If the principals in the protocol can be confused as to what protocol is being run, an intruder can increase the potential number of interactions between the principals.

These new interactions may reveal an unforeseen weakness in the protocol. Wagner and Schneier [14] discovered a weakness of this type in the Secure Socket Layer (SSL) protocol. In order to fully analyze these protocols, interactions between the sub-protocols must be considered. Unfortunately, analyzing sub-protocol interactions can lead to rapid increases in the cost of analysis as the number of possible interactions grows rapidly with the addition of sub-protocols. This makes the analysis difficult and costly. To date, Meadows [9] appears to have had the only success in performing this type of analysis with her work on IKE and SET using the NRL protocol analyzer.

The scarcity of tools examining sub-protocol interactions can be explained in part by the origins of the tools used to analyze security protocols. Until recently, security protocols have only contained sequential control. Hence, analysis tools had no reason to create ways to model branching operations. The Cryptographic Protocol Analysis Language Evaluation System (CPAL-ES) is based on a formal programming method, so a branch operator was included from the outset. This inclusion allowed the tool to represent and analyze protocols that include sub-protocols. If not for the problem of state explosion, which is an obstacle to extending all tools, the CPAL-ES system was ready to analyze branching protocols before the extension in this work were made.

Despite these obstacles, from work done by Wagner and Schneier that found a vulnerability caused by the interaction of two sub-protocols [14]. Another paper by Kelsey, Schneier, and Wagner proved the existence of an insecure protocol match for any other protocol [8]. It is apparent that tools that can analyze sub-protocol interactions have a target-rich environment in which to discover flaws in security protocols.

At HASE-99, we showed how CPAL-ES [15] can be combined with BAN Logic to analyze common cryptographic protocols. In this paper, we show how analysis of sub-protocol interactions can be accomplished with CPAL-ES and illustrate the technique through the analysis of TLS. Complex protocols such as TLS are difficult to analyze and design securely with their large size. CPAL-ES manages this complexity by allowing the actions of the protocol with the respect to its goals to be formally represented by a precondition. This becomes more useful as interaction among sub-protocols makes it even more difficult to keep track of effects of one sub-protocol upon another. Extensions made to the tool in order to perform this analysis are described. We will also describe the results obtained through this analysis and the specification methods used to elicit them.

This paper begins with a description of the TLS protocol in section 2. We review the CPAL-ES analysis tool in section 3, discuss our specification of TLS in section 4, describe the analysis of TLS and its results in section 5, and comment on the extensions made to CPAL-ES in section 6. We give our conclusions and summarize our future work in section 7.

2. The TLS Protocol

The Transport Layer Security protocol (TLS) [6] provides privacy for transmissions between two communicating applications. This protocol standardizes a revision of the Secure Socket Layer 3.0 protocol (SSL) by the Internet Engineering Task Force (IETF) Network Working Group. TLS supports a selection of cryptographic algorithms pertaining to key exchange, encryption, hashing, and digital signatures. A set of *cipher-suites* defines the different combinations of algorithms allowed within one session while a sub-protocol implements the selected cipher-suite.

TLS is based on previous versions of the SSL protocol developed by Netscape. Much of the current consumer commerce performed on the Internet utilizes the SSL protocol. Despite the widespread use, formal analysis of these protocols has been scant. Informal analysis by Wagner and Schneier [13] pointed out several problems with SSL 3.0. Formal analysis of SSL 3.0 has been performed by Mitchell, et al. [10] with analysis of TLS by Paulson [12]. Of the two, Paulson's method was more powerful in that it allowed optional messages and fields within a single protocol. The scope of the formal analysis was limited to examination of individual sub-protocols and, consequently, the analysis is blind to attacks that interleave two sub-protocols.

The TLS protocol is partitioned into two layers, labeled the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides the encryption and message authentication for each message. The TLS Handshake Protocol operates on top of the TLS Record Protocol. A TLS *handshake* supplies the authentication and key exchange operations for the TLS protocol. The security state agreed upon in the handshake is then used by the TLS Record Protocol to provide session security.

Typical security protocols have a single sequence of messages that are sent between the two principals. In TLS, the principals are referred to as the client (C) and the server (S). The client and the server must choose from a set of cipher-suites that can require different sequences of actions and messages within the protocol. For example, Diffie-Helman key agreement requires both principals to exchange public values that are used to compose a private session key, while RSA public key exchange relies on the client alone to choose, encrypt and deliver an appropriate private session key. The cipher-suite defines multiple characteristics such as the key exchange algorithm, the hashing method, and the digital signatures to be used in the handshake. Each characteristic requires different actions within the protocol depending upon which method is chosen.

The TLS protocol offers a choice of several possible protocol characteristics. The Diffie-Helman and RSA key exchange methods comprise the TLS secure key exchange methods. Other characteristics of key exchange include the use of smaller encryption keys for export versions, and a choice between fixed or ephemeral keys. Two signature methods, DSS and RSA can be used with Diffie-Helman key exchange while only RSA signatures are used with RSA key exchange. In addition, key exchange may be anonymous with the use of Diffie-Hellman. The authentication of the client and server is also flexible. This means that in a TLS session there may be no authentication with certificates, only the server is provided a certificate, or that both the server and the client are provided certificates. For hashing requirements, MD5 or SHA may be chosen. In addition to the full handshake, by including an acceptable old session identifier, a session may be resumed quickly by using the previously agreed upon security values. The cipher-suite message also specifies the type of encryption to be used for the session once the handshake is completed.

None of these TLS options are particularly complex on their own. The difficulty is that formal protocol analysis has never been effective on other than small, simple protocols. Using the traditional method of analyzing one protocol at a time without benefit of an "IF" statement

(which many cryptographic protocol analysis tools do not have) would require individual analysis of nearly four hundred separate protocols to ensure that all possible TLS paths were verified. Worse yet, all four hundred of these protocols would be ten to fifteen times larger than the majority of the protocols listed in the canonical security protocol collection by Clark and Jacob [5]. It is both the protocol size and complexity that hampers formal verification of these complex Internet security protocols with existing tools.

3. Overview of CPAL-ES

CPAL-ES utilizes Dijkstra's Weakest Precondition reasoning to analyze security protocols. In this method, stated goals and the actions of an algorithm are analyzed to produce the weakest precondition that will satisfy the given goals. The method was originally proposed as a procedure for developing algorithms and programs. Since security protocols are algorithms, this technique can be used to formally verify the ability of principals in protocols to reach their goals.

Protocols analyzed with CPAL-ES are first specified in the Cryptographic Protocol Analysis Language (CPAL) [15]. The CPAL language allows a formal specification of the protocol to be made and the actions of the principals to be specified accurately. In CPAL protocols are described as a series of actions that transform a principal's local state. A CPAL protocol specification will include actions such as assignments, encryption, decryption, hashing, signing, verifying, function operation, sending and receiving of messages, conditional tests, and assertions. Only the conjunction of send and receive operations allow an interaction between principal's states.

In addition to actions, the goals of the protocol are described through ASSERT statements as a predicate that represents the desired end states of all principals in the protocol. The satisfaction of this predicate by execution of the protocol means the goals given for the protocol are met.

The transformation of the goals into verification conditions is accomplished by an iterative examination of each statement in the specification. Weakest precondition reasoning proceeds by defining the preconditions that that guarantee that the target state would be reached after the last statement, then iteratively finds the guaranteeing precondition of the previous statement. In a sense, the analysis process starts at the end of the protocol and proceeds to the beginning. Specific transformations are defined for each operation in the CPAL language, so each statement modifies the predicate in some way. In total, these changes result in the creation of the verification

condition for the protocol. Upon completion, the resulting predicate is the verification condition for the protocol to successfully complete its goals.

To give the reader a better idea of the workings of the weakest precondition reasoning process, we will walk through a small example. Consider the ISO symmetric key two pass unilateral authentication protocol shown in Figure 1. This protocol utilizes a nonce value (Na) to provide authentication. In order for the protocol to work, the key used by B to encrypt Na and the key used by A to decrypt would need to be identical. The notation shown in Figure 1 is typical of the many pseudo-codes used for protocol description. The variety we use is often termed *standard notation*. Unfortunately standard notation is not conducive to productive analysis with weakest precondition reasoning. The CPAL specification in Figure 2 gives a more complete picture of what is actually happening, for example by showing the overt action of receiving data into the principal's address spaces.

A -> B: Na
B -> A: {Na, A}K

Figure 1. ISO Symmetric Key Two Pass Unilateral Authentication Protocol

The symbols used in CPAL describe the operations undertaken by the protocol participants. The => and <- operators describe the send and receive operations respectively. Decryption of an encrypted symbol x with a key K is written as d[e[x]K]K. The symbol := represents assignment, and the == symbol represents equivalence. Finally, the assert(x) statement is used to describe a goal of the protocol. In this example, A expects to receive from B the same nonce it sent to B. An assertion is required for logical analysis of the protocol to take place. That is, if you do not have a goal, you can never fail. Figure 3 shows the weakest precondition that will guarantee that the protocol meets its goals along with the intermediate weakest preconditions at each statement.

A: => B(na);
B: <-(nonce);
B: => A(e[<nonce, A>]K);
A: <-(msg);
A: (response, A) := d[msg]K;
A: assert(Na == response);

Figure 2. CPAL ISO Specification

As noted earlier, analysis by formal methods is complicated as the size of the protocols increase. One reason for this is that formal methods often base their

analysis on the protocol states. As the size of the protocol grows, state space explosion occurs, that is, the number of states increases much faster than the number of statements.

CPAL-ES does not suffer from state space explosion, per se. However, in weakest precondition reasoning, large, complex protocols tend to explode the size of the verification condition predicate. Most protocol actions (assignment, encryption, send message, etc.) result in no growth, or at most linear growth of the predicate. A branching statement, on the other hand, can result in doubling the predicate size. Thus, a series of branches within a protocol can cause exponential growth in

The handshake protocol contains the key exchange and authentication operations of interest to us. Several specifications of the TLS handshake protocol were created. First we created a CPAL specification true to the TLS standard yet free of details extraneous to the analysis of the protocol's security. To describe the specification used in the analysis, we will first describe the specification in general and then point out what was omitted and why.

As described in Section 2, the actions carried out in the handshake are determined primarily by the cipher-suite agreed upon by the participants. The cipher-suite specifies characteristics of the handshake and the ensuing session. Our specification combines these characteristic

Table 1. Sub-protocol Attributes

Sub-protocol Name	Key Exchange	Authentication	Public Key Use
Resume	None	Old Secret	No
Anonymous	DH	None	No
DHE_CA	DH ephemeral	Both	Signatures
DHE	DH ephemeral	Server Only	Signatures
DH_CA	DH fixed	Both	Signatures
DH	DH fixed	Server Only	Signatures
RSA_EXP_LS_CA	PK w/short enc. key	Both	Sigs w/ long key
RSA_EXP_LS	PK w/short enc. key	Sever Only	Sigs w/ long key
RSA_EXP_CA	PK w/short enc. key	Both	Sigs w/ short key
RSA_EXP	PK w/short enc. key	Sever Only	Sigs w/ short key
RSA_CA	PK w/long key	Both	Sigs w/ long key
RSA	PK w/long key	Sever Only	Sigs w/ long key
Sub-protocol Name	Key Exchange	Authentication	Public Key Use

predicate size.

To prevent the predicate from becoming unmanageable in size during analysis, logical simplifications are performed automatically when the predicate grows too quickly. These theorem proving extensions needed to be added to perform analysis of TLS and will be discussed more thoroughly in the next section.

CPAL-ES has been used to model a number of security protocols. Combined with BAN logic the system has revealed some previously unknown flaws in a protocol developed by Kao and Chow [4]. Further information on CPAL-ES can be found in earlier work [15].

4. Specification of the TLS Protocol in CPAL

Our specification of the TLS protocol does not include all of the details given in the TLS standard. In particular, we concentrated on analyzing the handshake protocol.

actions into separate, non-overlapping sub-protocols within our protocol. Each of the sub-protocol choices included in our specification represents the actions of a cipher-suite of the TLS protocol.

Our model allows the following cipher-suite choices: Diffie-Hellman or RSA key exchange, strong or export (weak) RSA key exchange, short or long RSA public key in server certificate, anonymous, server only, or server and client authentication, and fixed or ephemeral keys. The combination of these cipher-suite characteristics into the sub-protocols used in the model is shown in Table 1. The choice of new cipher-suite characteristics is mute if the protocol participants agree to restart an old session. Restart of a session is listed as the "Resume" sub-protocol in Table 1.

The different action threads specified by the sub-protocols are created through the use of the "if/else" conditional branches provided by CPAL. In our

Final Verification Condition:	$(A.K == B.K) \rightarrow (Na == \langle Na, A \rangle.1) \wedge$ $(A.K \neq B.K) \rightarrow (Na == d[e[\langle Na, A \rangle]B.K]A.K.1)$
A: => B(Na);	$(Na == d[e[\langle Na, A \rangle]B.K]A.K.1)$
B: <-(nonce);	$(Na == d[e[\langle Bq.nonce, A \rangle]B.K]A.K.1)$
B: => A(e[\langle nonce, A \rangle]K);	$(Na == d[e[\langle nonce, A \rangle]B.K]A.K.1)$
A: <-(msg);	$(Na == d[Aq.msg]A.K.1)$
A: (response,A) := d[msg]K;	$(Na == d[msg]A.K.1)$
A: assert(Na == response);	$(Na == response)$

Figure 3. Weakest Precondition Reasoning for ISO

specification, the cipher-suite value determines the various characteristics that define the sub-protocol. This allows a nested structure to be used to specify the control. The nested "if/else" structure uses six characteristics to describe the twelve separate sub-protocols. An advantage of this method is that it improves the readability of the specification by grouping similar sub-protocols within the same block.

The control structure must be used repeatedly throughout the protocol as the control switches between the participants in the protocol. This is the result of the restriction of the CPAL system to one thread of control, which precludes the spawning of threads for the protocol participants. Thus, although the choice of which sub-protocol to run is made early in the protocol and is not changed, the CPAL specification of TLS selects the sub-protocol with a control structure upon each switch between the actions of the participants.

To simplify our analysis, some generalizations, abstractions and deletions were made in the specification of the TLS handshake protocol. These changes removed elements of the protocol unimportant to analysis with CPAL-ES. It was deemed that operations identical in purpose and requiring the same actions could be treated as one operation. This reasoning allows the choice between MD5 and SHA hashing algorithms to be omitted, as well as choosing between the RSA and DSS signing algorithms. Elimination of these choices prevents the modeling of the interaction of these choices within sub-protocols.

The specification also does not comprehensively model certificates. Many certificate standards may be used with TLS, and to avoid testing each one, it was assumed for our analysis that they were properly designed and secure. This decision also allowed the details of the specification of the request for a certificate to be dropped. Some details regarding the agreement of session characteristics were also discarded. The vendor identification value was considered extraneous to our analysis. Our analysis also avoided the question of handling older versions of the

protocol such as SSL 2.0 and SSL 3.0. Finally of note is that the cipher-suite value will also describe some details relating to the record layer, such as the encryption standard to be used. However, because our analysis was limited to the handshake protocol, these details were ignored.

5. Analysis of the TLS Protocol

Our main interest in evaluating the TLS protocol was to see if CPAL-ES could be used to analyze complex, Internet protocols that have multiple separate components. We wished to explore the ability of the CPAL-ES system to accurately model the interaction between the various TLS sub-protocols. Our effort was successful and we found a harmful interaction in TLS similar to one previously found in the SSL 3.0 protocol by Wagner and Schneier.

We used three different protocol specifications to perform the analysis. The first one accurately and comprehensively described the TLS handshake protocol. The other two specifications utilized specified intruder actions to examine the interactions between sub-protocols. We will describe analysis of these specifications and the results it provided in this section.

The first TLS specification was used to gain a greater understanding of the protocol's operation without interference from an intruder. Analysis with CPAL-ES reveals whether the specification satisfies its goals. The evaluation allows the CPAL specification to be tested for internal consistency. With the size of the complete CPAL specification, over five hundred lines, several mistakes were discovered in our representation during evaluation with CPAL-ES. As such, this evaluation verifies the operation of the protocol much like a program would be formally verified.

The specification and analysis also aid in the understanding of the protocol. Once the specification has been verified with CPAL-ES, it can then be confidently utilized as a reference for modeling attacks. Initially these attacks are limited in nature and do not subvert the

protocol. However, analysis of limited attacks can give information about possible weaknesses.

During CPAL-ES analysis, interactions between the different sub-protocols should be reflected in the verification condition. In properly designed protocol, these interactions should not interfere with any assertions in the protocol. Thus none of these interactions will be seen in the final verification condition. This is expected, as the participants should always be able to successfully complete the protocol when there is no outside interference. In practice, the predicates due to interactions between sub-protocols fall out due to the sub-protocol choice values agreed upon at the beginning of the protocol. These interactions are in the form of implications due to the conditional if structure. So when the conditional tests are false, the corresponding predicates drop out of the verification condition. In the following discussion, an unmatched protocol interaction represents a session where two principals proceed with a protocol even though they have not agreed on the same sub-protocol. Thus, whether or not an interaction between sub-protocols could be detected or produce an attack cannot be examined through analysis only of the basic specification. What we wish to produce is a verification condition that would allow us to examine whether the protocol participants will detect an interaction between unmatched protocols. Then we could determine under what conditions an unmatched protocol interaction would satisfy the protocol assertions.

We can force interactions to be retained within the final predicate or verification condition by replacing the protocol agreement cipher-suite value transmitted between the genuine participants. We accomplish this by inserting an intruder into the specification to intercept and insert a new value. In a productive attack, the substituted value would have a specific meaning; however by using an undefined value the statements corresponding to the mismatched protocols do not drop out as they usually would. With this interference, the predicates corresponding to the condition when two principals believe they are running the same protocols yet are not can be seen. At this point, the verification condition will retain the preconditions necessary for any of the sub-protocols interactions to satisfy the assertions. A simple example of how an intruder can confuse two principals is shown in Figure 4.

We want to retain these predicates in the verification condition because examination of the statements within the verification condition may reveal that another intruder action could allow subversion of the protocol. Additionally, if an interaction completely satisfied the relevant assertions, then there would be no corresponding

fragment left in the verification condition. Calculating the number of possible interactions and comparing it with the number of statements within the verification condition can detect this. In order to examine the conditions that satisfied the assertion, giving a variable in the assertion a new unique name will prevent the reduction of the assertion statement. An examination of the verification condition can then reveal the interactions that satisfy the assertion. We retain these interactions by interfering with the CS value received by at least one of the principals.

We used this strategy to see if the weaknesses found in the SSL 3.0 exist in TLS with the aid of CPAL-ES. The attack on SSL 3.0 is in part possible due to the lack of labeling of the Diffie-Hellman and RSA key exchange messages. Thus, the check for the key exchange used to verify the authenticity of the message cannot be used to recognize the substitution of one message for the other. In TLS it appears that the designer's tried to avoid this weakness by the elimination of RSA ephemeral key exchange in many of the sub-protocols. However, the designer's did retain ephemeral key exchange for the RSA_EXP cipher-suite. This left the possibility of the same type of vulnerability found before. The results of our evaluation revealed that indeed there remained a potential weakness in TLS held over from the SSL 3.0 protocol.

We applied this strategy to the demonstration that CPAL-ES could elicit this known flaw from SSL 3.0. For TLS, no authentication is done before a sub-protocol is chosen. This means that a man in the middle can subvert the sub-protocol selection process. If the principals do not recognize this later, it can lead to the compromise of the entire protocol. Following the strategy described in Figure 4 it is possible to elicit the conditions for satisfying

```
A: ->B(prot);
M: <-(mesg);
M: =>B(sub); // Introduces generic value.
B: <-(prot);
B: if (prot == 1) then reply := (X)
   else reply := (Y);
B: =>A(reply);
A: <-(test);
A: (prot == 1) then assert(test == X)
   else assert(test == Y);
```

Figure 4. Protocol Agreement Attack

all sub-protocol interactions. In demonstrating the ability to find the SSL flaw, we only activated the assertion for the client RSA_EXP protocol key exchange and used the intruder to all the server's sub-protocol value to be unset.

This produced conditions for the client RSA_EXP sub-protocol interacting with all possible server sub-protocols.

This process produced a verification condition containing conjuncts for each sub-protocol interactions that did not satisfy the client's test of the RSA_EXP parameters. A portion of this verification condition is shown below in Figure 5. We will explain this figure and discuss how it allows us to find harmful protocol interactions. The predicate is in conjunctive normal form. Each of the sub-protocol interactions form a separate conjunct composed of a list of conditions in disjunct form. No conjuncts from the server's RSA_EXP, DHE_CA

To illustrate how this predicate describes the protocol interactions, we explain the meaning of the fragment here. The CS value provided by the intruder, termed "I.iCS" in the Figure, prevents the conditionals below from eliminating one another. A statement of this type:

$(S.Anon == F.Auth(S.CS)) \text{ or } \sim(S.Anon == f.Auth(S.CS))$

would mask invalid protocol interactions in the final verification condition because this expression would be simplified to TRUE (since $\sim P \vee P \Rightarrow TRUE$). With one of the S.CS values replaced by I.iCS, the reduction will not occur. This follows the same strategy shown in Figure 4.

```

AND
(S.oldSession(C.SID) or
((S.Anon == f.Auth(I.iCS)) or
((S.DH == f.KEM(I.iCS)) or
(not ((S.RsaExp == f.KEM(I.iCS))) or // Client using RSA_EXP
(not ((S.SignOnly == f.PK(I.iCS))) or
((S.ClientCert == f.Cert(I.iCS)) or
((S.Anon == f.Auth(S.CS)) or
((S.DH == f.KEM(S.CS)) or
(not ((S.RsaExp == f.KEM(S.CS))) or // Server using RSA_EXP
(not ((S.SignOnly == f.PK(S.CS))) or
((S.ClientCert == f.Cert(S.CS)) or
(f.hash(<C.Nc,S.Ns,<S.mod,S.exp>>) ==
f.hash(<C.Nc,S.xNs,<S.mod,S.exp>>))))))))))
AND...AND
((S.Anon == f.Auth(I.iCS)) or
((S.DH == f.KEM(I.iCS)) or
(not ((S.RsaExp == f.KEM(I.iCS)) or // Client using RSA_EXP
(not ((S.SignOnly == f.PK(I.iCS)) or
((S.ClientCert == f.Cert(I.iCS)) or
((S.oldSession(C.SID)) or
((S.Anon == f.Auth(S.CS)) or
(not ((S.DH == f.KEM(S.CS)) or // Server using DH
(not ((S.DHE == f.DH(S.CS)) or
((S.ClientCert == f.Cert(S.CS)) or
(f.hash(<C.Nc,S.Ns,<S.P,S.G,f.dhy(<S.P,S.G,S.X>>)>>) ==
f.hash(<C.Nc,S.xNs,<S.P,S.G,f.dhy(<S.P,S.G,S.X>>)>>))))))))))
AND

```

Figure 5. Predicate Fragment of Sub-protocol Interactions

or DHE sub-protocols interaction with the client's RSA_EXP sub-protocol were present in the verification condition. Therefore, these conjuncts must have been true. Therefore, these sub-protocol interactions with RSA_EXP satisfied the testing of the key exchange parameters. To get a better idea of the structure of the missing conjuncts, we replaced one of the nonce values so that it would not match in the assertion. Two of these conjuncts are displayed in Figure 5.

The interactions that we address are created by the flexibility engineered into the TLS protocol suite. In the two lists connected by the and statements in Figure 5, the initial equivalence comparisons are comprised of the values used to determine which sub-protocol each principal selects. The comparisons containing the S.CS value correspond to the conditionals that determine the sub-protocol used by the server. The comparisons containing the I.iCS value are connected with the client in

the same way. The comparison values, S.Anon, S.DH, S.DHE, S.ClientCert, S.RsaExp, and S.SignOnly, have the following meanings: anonymous session, Diffie-Hellman key exchange, ephemeral Diffie-Hellman keys, client certificates, export RSA key exchange, and strong RSA keys for signatures only.

The logical relation between the assertion and sub-protocol conditional comparisons links the asserted condition to one interaction of two sub-protocols. The first conjunction indicates that both principals are running sub-protocols with the following characteristics: new session, server authentication, no Diffie-Hellman key exchange, RSA export key exchange method, long public keys for signing, and no client certificate. For the second conjunction in Figure 5, we can ascertain that the client was running a sub-protocol with the same characteristics as the conjunct above indicates. The server was running a sub-protocol with the characteristics: new session, server authentication, Diffie-Hellman key exchange, and ephemeral keys without client authentication. Note that each of the disjuncts, except for the assertion, must be false for the assertion to affect the logical value of the predicate. Examination of the last comparisons in the disjunctions reveals whether the interaction of these two sub-protocols would produce an unwanted satisfaction of the client's test of the key exchange parameters. Both comparisons are equivalent except for the value that was renamed so these conjuncts would be retained.

The first of the two sub-protocol interactions in the fragment is caused by the appropriate run of the RSA_EXP sub-protocol by both principals. The second conjunct is the result of the interaction between the DHE and RSA_EXP sub-protocols (both noted in Table 1).

These results indicate that the check of the key exchange parameters does not recognize the substitution of the DHE protocol message for the RSA_EXP protocol message. The check compares the hash of the parameters sent with the decrypted signature of the parameters sent by the server. The server believes they are the correct parameters but the client should not accept them since they are for a different sub-protocol. However, the check is designed to detect tampering with the parameters in transmission, not that the server sent unexpected parameters. As Wagner and Scheier explained, the use of the Diffie-Hellman parameters will result in the creation of a weak RSA key. Upon breaking this key, the rest of the protocol may then be subverted.

This process of introducing an intruder to intercept a message partially automates the creation of predicates that give the conditions necessary for mismatched sub-protocols to satisfy conditions of the protocol. The benefit of this process is that it gives us a logical formula

for a sub-protocol interaction to satisfy a condition. In this case, the flaw was readily apparent, as no further modification was necessary for the test of the key exchange parameters to be compromised. When this is not the case, the logical formula may be used as a base to create further intruder messages. These formulae condense the actions and assertions of a protocol into the requirements for the assertions to be satisfied. This provides a new viewpoint with which to examine protocols. This process will not find all protocol flaws, no tool can, but it provides new avenue to analyze protocols through the interactions of their sub-protocols. Once the initial specification was complete, the procedure given above was simple to create and analyze. Other tests that completely analyze the protocol in this fashion would be equally easy to create. Thus without direction from the flaw found in SSL 3.0, this flaw could still have been discovered with a more comprehensive test. Instead of looking only at validation of the key-exchange parameters for the client's RSA_EXP sub-protocol, we could have expanded the output to include all the tests done by the client sub-protocols for key exchange. The tests can be initially limited to those related to key exchange since these must be passed for the protocol to go any further. Therefore any flaws due to sub-protocol interactions must affect these tests for the protocol to be compromised. If a flaw was found, or assumed, the effects of sub-protocol interactions upon later tests could also be determined. One fact to be aware of that makes this limitation possible is that, once a sub-protocol is selected, it cannot be changed later in the protocol since no further selection takes place.

CPAL-ES allows the representation of the interaction between two sub-protocols. Within a protocol such as TLS these sub-protocols are selected from a collection of sub-protocols utilized by a principal. The modeling of this subversion is an important part of formally understanding attacks upon protocol suites. The CPAL environment contains sufficient functionality to verify the feasibility of these attacks. The intruder's interference with the sub-protocol agreement messages can be expressed and verified with the language of CPAL and CPAL-ES evaluation. Using this ability, an accurate and complete specification of the attack was made that verified its feasibility.

6. Extensions to CPAL-ES

The TLS protocol is much more complex than many of the older security protocols. TLS and other new protocols, such as IKE and SET, utilize a wider variety of features including operations such as hashing and key agreement. They also allow the principals to have

flexibility in the way the protocol proceeds. These choices complicate analysis. In order to handle these features we made extensions to CPAL-ES. These included changes to the semantic interpretation of several protocol actions. The analysis of choices within a protocol was already possible, however, testing of the features revealed the need for modifications to improve the performance of the system. This was handled by the addition of simplification operations to the analysis process to prevent excessive growth in the size of the predicate.

6.1 Semantic Interpretation of CPAL

Changes made to CPAL-ES improved its ability to handle security protocol features. The most significant change, although quite simple to implement, dealt with the way functions were identified. This change allowed functions used by different principals to be compared. Without this ability, the output of two identical functions upon identical sets of data could not be resolved as being equal. Thus, one could not test for the correctness or authenticity of messages by the use of hashing functions.

The main obstacle to allowing global functional equivalence was the lack of a complete representation for functions in CPAL-ES. In CPAL-ES, data can be accurately represented. However, functions cannot be recognized as being separate from its arguments. The identity of data and functions is determined by their name and the identity of their creator.

There are two ways of specifying that two principals in a CPAL protocol possess identical information. The first relies on the transmittance of information between principals using send and receive operations. The second relies upon assumption statements being made regarding data with different identities. However, CPAL-ES does previously did not allow one principal to execute another principal's function. This characteristic prevents the productive comparison of functions and function operations on data that has passed through the protocol being modified by different principals executing the same function, e.g. MD5. The result is that the proposition $(A.MD5(A.na) == B.MD5(A.na))$ will always be false, which did not accurately model the characteristics of the resulting data.

A completely accurate model of functions would allow them to be treated in the same manner as data that can be passed between principals or can be assumed to be the same during a protocol run. However, no security protocols that we are aware of require that functions be transmitted between principals, so we sought a more simple solution that would resolve this conflict with the protocol model.

In the solution adopted for our analysis, functions do not have their identity determined by the locality they are executed in. Any two functions with the same name and arguments but created in different address spaces could be found equivalent. This solution prevents two principals from using a secret function.

The only negative result of this solution is that CPAL-ES does not accurately model a protocol that depends upon a secret function to provide security. This restriction is not limiting since security theory rejects reliance upon the obfuscation of a function for security. This restriction in the new CPAL-ES representation of functions assumes that any participant in a protocol, including an intruder, can use all functions.

This function recognition addition to CPAL-ES extends the ability of CPAL to model the meaning of mathematical operation. Previously, the encryption and decryption operations were modeled to reflect their inverse natures (i.e. if $k == k$ then $d[e[X]k]k == X$). Using the function recognition extension allows more accurate modeling of message authentication with hashing functions such as MD5 and Diffie-Hellman operations found in TLS. However, an additional modification needed to be effected in order for Diffie-Hellman operations to be handled. This required a special simplification that recognized the asymmetric argument positions that allow key agreement. Functions named 'dhy' are tested to determine whether the arguments are ordered correctly as defined by the Diffie-Hellman protocol so that key agreement may be achieved. Here is a comparison of two Diffie-Hellman functions that would be found equivalent:

$$\begin{aligned} & \text{dhy}(p, \text{dhy}(p, g, \text{secretA}), \text{secretB}) == \\ & \text{dhy}(p, \text{dhy}(p, g, \text{secretB}), \text{secretA}) \end{aligned}$$

Without the special DH recognizer, this equation would be false, since the forms of the two equations are not identical. Again, this extension allows CPAL-ES to model the most important characteristic of the mathematics behind the Diffie-Hellman scheme.

Another challenge to TLS analysis was posed by mismatched composite assignments produced by the interaction between protocol branches. CPAL-ES can recognize the assignment of a list of values to another list. In a protocol with no branches, all targets of an assignment must be at the same length as the destination, which will prevent a null assignment.

With a sequential protocol, mismatched list length would be caught as a syntax error at specification time. In a protocol with branches, this action could be inappropriate. The interaction between branches indicates that an error has occurred, so termination of analysis

would only hinder the analyst's ability to determine the root problems of the protocol's behavior in these instances. A message sent from a principal using one branch of the protocol to a principal using a different branch can result in the message sent having fewer elements than the number expected.

Ideally, we wanted to be aware of this event yet still allow the interaction between sub-protocols to be modeled. To reflect this occurrence, a unique, yet descriptive name is assigned to each destination element that lacks a target element in the composite assignment. This permits the result of a null assignment to be considered in the verification condition. Other effects on the protocol's operation are dependent upon the implementation and are outside the scope of this paper.

6.2 Predicate Simplification During Weakest Precondition Analysis

As we mentioned earlier, protocol actions routinely cause linear growth in the predicate size relative to the size of the elements in the protocol action during CPAL-ES analysis. Branches within the protocol implemented with "if/else" statements in CPAL are the exception to this rule.

In Figure 5 the CPAL branch statement is on the left while the right column shows the logical transformation that this statement would cause in the predicate P. Elements such as C and f(X) represent a conditional and a statement respectively. With respect to predicate growth the most important characteristic of the transformed predicate is that it contains two copies of the predicate P. Thus where P dominates, for n branches, the predicate should grow according to the function 2^n . We use recurring simplification performed during the analysis to ameliorate this exponential growth in predicate size.

CPAL branch stmt.	Predicate
if (C) then {f(X);}	$C \rightarrow wp(f(X), P) \wedge$
else {g(X);}	$\sim C \rightarrow wp(g(X), P)$

Figure 5

CPAL-ES already possessed some routines that simplified the predicate produced by the weakest precondition analysis of a protocol. However, these routines were not intended to perform a thorough reduction. Some new routines and strategies were required to modify and simplify the predicate.

The most important technique is to transform the binary predicate tree used for the weakest precondition analysis to a flattened tree utilizing sets for compound

statements with only one kind of binary connective. These flattened trees have equivalent logical meaning yet are no larger than the binary trees and allow a thorough examination for simplifications.

Before flattening the binary tree, the predicate was normalized to group the connectives appropriately for the flattening operation. Conjunctive normalization prepares the predicate for a flattening of disjunct structures, while a disjunctive normalization would do the opposite. These operations were expensive, so simplification calls were only invoked when a transformation caused a predicate to double in size. Testing showed that this works well with the predicates produced by the TLS protocol.

7. Conclusions and Future Work

Evaluating security protocols is very difficult, made more difficult by the complexities of modern, Internet protocols for ECommerce and general use. We have shown that CPAL-ES can effectively analyze complex security protocols that are beyond the capabilities of other protocol evaluation techniques based on other formal methods. We illustrated how CPAL-ES accomplishes this analysis and addressed the changes to CPAL-ES that additional protocol complexity required.

The complexities inherent in a protocol posed several problems. Overcoming these problems revealed much about our system and the TLS protocol. It first required us to recognize the limitations of CPAL-ES and the features that would allow us to overcome them. With these new features, we were able to create a comprehensive verified formal specification of the TLS handshake protocol. We also used CPAL-ES as a tool to examine the TLS protocol for harmful protocol interactions. This produced results verifying a weakness within the TLS protocol. In general, this system is very useful at representing protocols composed of sub-protocols where it is difficult for a human expert to keep track of all the effects of each protocol actions.

In our analysis of the TLS security protocol, we learned a great deal about TLS and the CPAL-ES system as well. In addition to these results, analysis of TLS made us consider further work. Previously, CPAL-ES has been used with BAN logic [2] to analyze the beliefs of the protocol participants. Unfortunately BAN lacks the functionality to handle the Diffie-Hellman key agreement present within TLS. An extension allowing the use of the SvO logic [12] developed by Syverson and Oorschot would allow key agreement operations to be modeled. Other logics developed to handle protocols that allow auctions, fair exchange, or electronic commerce could further extend the usefulness of the system. It should also

be noted that theorem proving tools such as the Prototype Verification System, (PVS), have been used to aid display and work with logical output. Further use of PVS could aid in the manipulation of verification condition into new forms to gain further insights. Analysis of protocols with more complexity than TLS, such as SET, should provide even more justification for the benefits of this tool.

The weakest precondition system is not tailored to any one protocol and our specification system is robust enough to handle complex cryptographic primitives such as hashes, certificates, signatures, symmetric encryption, public key encryption, etc. The tool has been successfully applied to over fifty simple protocols. Now it has been applied to a complex protocol suite. It should also be capable of handling other protocol suites

Another direction for further work would be to change the modeling of control within CPAL-ES. Protocols are run as two remote processes. With simple, sequentially oriented protocols, a model limited to one thread of control is more than adequate. With branching protocols, this restriction requires the seemingly redundant use of branch control for proper specification. A model that allowed separate principals to act as processes would eliminate the need for this repetition within the specification.

8. Bibliography

- [1] Martin Abadi and Roger Needham, "Prudent Engineering Practice for Cryptographic Protocols", *IEEE Transactions on Software Engineering*, 22(1):6-15, January 1996.
- [2] M. Burrows, M. Abadi, Roger Needham. "A Logic of Authentication". Research Report 39, Digital Systems Research Center, February 1989.
- [3] S. Brackin, "Automatically Detecting Most Vulnerabilities in Cryptographic Protocols", in The DARPA Information Survivability Conf and Exposition, Jan 2000, V.1, pp 222-36
- [4] Justin Childs, "Evaluating the TLS Family of Protocols with Weakest Precondition Reasoning", Technical Report. Florida State University Department of Computer Science. <http://www.cs.fsu.edu/research/reports>
- [5] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. A continually updated library of protocols analyzed in the literature, available at www.cs.york.ac.uk/~jac/.
- [6] T. Dierks and C. Allen, "The TLS protocol: Version 1.0. Request for Comments: 2246", available at <ftp://ftp.isi.edu/in-notes/rfc2246.txt>.
- [7] E. W. Dijkstra, *A Discipline of Programming*, Prentice Hall Series in Automatic Computation, Prentice-hall Inc. Englewood Cliffs, NJ, 1976.
- [8] J. Kelsey, B. Schneier, and D. Wagner, "Protocol interactions and the chosen protocol attack," in *Lecture Notes in Computer Science*, 1361. pp. 91-104 Heidelberg: Springer, Berlin, 1998
- [9] Catherine Meadows, "Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer," *IEEE Symposium on Security and Privacy*, 1999.
- [10] J. C. Mitchell, V. Shmatikov, and U. Stern, "Finite-State analysis of SSL 3.0 and related protocols," In *Workshop on Design and Formal Verification of Security Protocols* Sept. 1997. DIMACS.
- [11] Roger M. Needham, Michael D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", *Communications of the ACM*, December 1978 vol. 21 #12, pp. 993-999
- [12] L. C. Paulson, "Inductive Analysis of the Internet Protocol TLS," Technical Report 440, Cambridge University Computer Science Laboratory, 1998.
- [13] P. F. Syverson, "A Taxonomy of Replay Attacks," *Proceedings of the Computer Security Foundations Workshop VII*, Franconia NH, IEEE CS Press, Los Alamitos, 1994.
- [14] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 Protocol," In D. Tygar Ed., *USENIX Workshop on Electronic Commerce*, pages 29-40. USENIX Association, 1996.
- [15] Alec Yasinsac and William A. Wulf, "A Framework for A Cryptographic Protocol Evaluation Workbench," *Proceedings of the Fourth IEEE International High Assurance Systems Engineering Symposium (HASE99)*, Washington D.C., Nov. 1999