

On the Security Effectiveness of Cryptographic Protocols

Rajashekar Kailar¹Virgil D. Gligor¹Li Gong²

Abstract

We introduce the notion of security effectiveness, illustrate its use in the context of cryptographic protocol analysis, and argue that it requires analysis of protocol property dependencies. We provide examples to show that, without dependency analysis, the use of some logics for cryptographic protocol analysis yields results that are inconsistent or unrealistic in practice. We identify several types of property dependencies whose use in protocol analysis methods can yield realistic analyses.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: General- *Security and Protection*, Distributed Systems; D.4.6 [Operating Systems]: Security and Protection - *authentication, cryptographic controls*; K.6.5 [Management of Computing and Information Systems]: Security and Protection - *authentication*; E.3 [Data]: Data Encryption.

Key Words and Phrases: Security effectiveness, cryptographic protocol, cryptographic assumption, logic, dependency, threat countermeasure.

1 Introduction

Cryptographic protocol analysis using logics has conclusively shown that subtle security vulnerabilities which may escape designers' attention can be detected using formal methods. In the last few years, several logics have been proposed and have been used in the analysis of protocols to show that the protocols can (not) achieve certain goals ([2], [10], [11]). The application of logics³ allows one to derive properties which imply that certain threats are infeasible in the system. By doing this, one attempts to show that a protocol is *effective* in countering a specified (set of) threat(s). For instance, protocol properties, such as *session-key freshness*, *jurisdiction* (i.e., established authority over a session key), and *session-key confidentiality*, are generally considered sufficient to establish the effectiveness of a cryptographic protocol with respect to threats of *unauthorized re-use of old session keys*, *unauthorized session-key generation*, and *unauthorized session-key disclosure*.

¹Electrical Engineering Department, University of Maryland, College Park, MD 20742.

²SRI International, Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park, CA 94025.

³Here, and in the rest of the paper, we will use the term *logics* and *protocol analysis methods* to refer primarily to the logics presented in references [2], [10] and [11], or other similar logics.

However, logics for the analysis of cryptographic protocols derive properties based on cryptographic assumptions about the underlying system. The results of such analyses are dependent not only on the validity of stated assumptions but also on that of often unstated ones. For example, extant logics for the analysis of cryptographic protocols do not take into consideration the dependency of message confidentiality on the lifetime of encryption keys while deriving properties of confidential message contents (e.g., session-key freshness). The analysis of this dependency is important in determining whether current session key, old session keys, and long-term keys (e.g., key-encrypting keys) are breakable (i.e., can be read by an active intruder).

Missing cryptographic assumptions in analyzing cryptographic protocols may yield inconsistent or unrealistic results. For example, the analysis and the logic itself may assume that a key is unbreakable during the key lifetime, yet the protocol itself may generate a very large number of known (or even chosen) plaintext-ciphertext block pairs, which can lead to the breaking of that key. Or, the analysis and the logic itself may assume that an authenticated source of time is available for an authentication protocol to establish message freshness. Missing cryptographic assumptions in analyzing cryptographic protocols is particularly ironic because the relationship between these protocols, logics, and cryptography is limited precisely to making assumptions about the properties of underlying cryptographic systems.

In this paper, we argue that analyzing the dependencies of the derived protocol properties on the underlying assumed properties is a necessary part of establishing security effectiveness. We provide examples to show that the use of logics for cryptographic protocol analysis without this dependency analysis may yield results that are inconsistent or unrealistic in practice. We present a *dependency graph* and illustrate how it can be used in protocol analysis.

This paper contains five sections. In section 2, we briefly introduce the notion of security effectiveness and illustrate its use; i.e., how a specific threat function can be rendered infeasible by the properties of system functions. In section 3, we provide examples to illustrate the consequence of disregarding dependencies among derived and assumed properties, on the results of protocol analysis. In section 4, we present a *dependency graph* that underlies much of the analyses and illustrate its use. Section 5 concludes this paper. Additional examples of protocol dependencies appear in [12].

2 Security Effectiveness

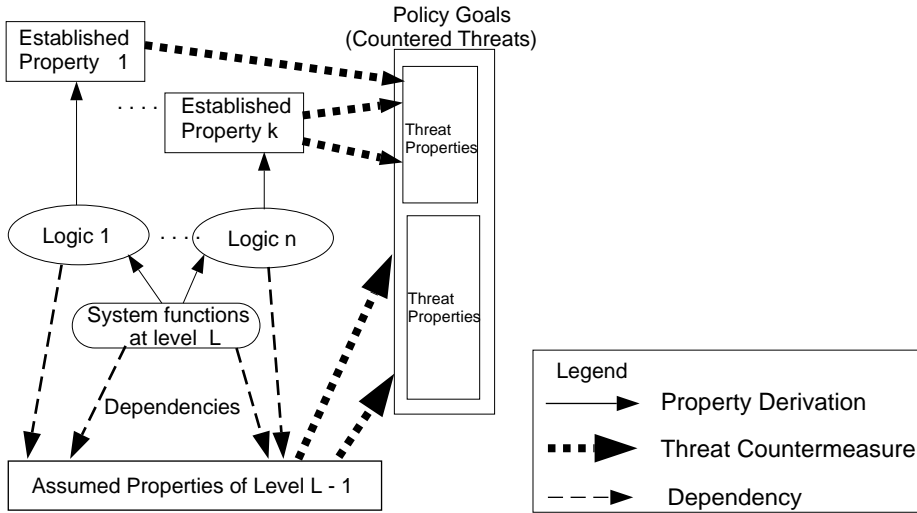
In figure 1, we illustrate the intuitive notion of *security effectiveness* of a system. The figure shows a set of reachable system states U . Threats are properties of intruder actions in a given environment. System properties are introduced to counter a specific set of threats. If P is a property that counters a threat T (i.e., $P \Rightarrow \neg T$), then the set of system states in which T does not hold contains the set of system states in which property P holds.

Intuitively, we say that a system security function is *effective* if its (correct) operation counters one or more identified threats. To reason about the effectiveness of a security function, it is useful to think of threats as threat functions that operate in the environment of system use, and to assume that the effects of these functions can be specified in the same way as that of the security functions; i.e., by specifying their properties. Then, *to demonstrate the effectiveness of a security function, we need only show that the properties of the system security functions and those of the threat functions cannot coexist.*

Properties of security functions have been derived in the past in various areas of computer system security. For example, functions implementing an information flow policy may have the property that “if information flows from variable x to variable y , the security level of y dominates that of x .” Functions exhibiting this property could counter the threat properties of “unauthorized downgrading of classified data” or “unauthorized flow of sensitive information,” which characterize functions implemented by Trojan Horses in unprivileged (malicious) application programs. Thus the effectiveness of the information-flow property can be established with respect to the threat of confidentiality violations by Trojan Horses.

2.1 Property Dependencies

Cryptographic protocols and the logics used to derive their properties typically make several assumptions about the system and about the operating environments. For example, the logics for the analysis of cryptographic protocols commonly assume that (1) message integrity is preserved, (2) clocks are synchronous, (3) timestamps are monotonic, (4) encryption keys are unbreakable, (5) message sender can distinguish his own message from those that are sent by others, (6) encryption keys are generated randomly, in a manner which does not allow their prediction from the knowledge of old keys, and so on. The results derived using logics, are, hence,



dependent on these assumptions. These assumptions, if justified, imply that certain threats cannot hold. For example, in cryptographic protocols which use long-term keys for encryption, the assumption about the strength (complexity) of the encryption mechanism implies that some threats of cryptanalysis cannot be successful over the lifetime of the long-term key. Hence, system states in which assumptions hold are a subset of those in which the *assumed threats* do not hold. That is,

$$\text{Assumed Properties} \rightarrow \neg \text{Threats}_{\text{ASSUMED}}.$$

Properties of the system, which are established by system analysis, imply that another set of threats do not hold. That is,

$$\text{Established Properties} \rightarrow \neg \text{Threats}_{\text{COUNTERED}}$$

The nature of the established and assumed properties differs only in the sense that the former is justified, whereas the latter may not necessarily be justified. Together, the established protocol properties and the assumed properties imply that a certain set of threats cannot materialize. Figure 2 shows a set of reachable system states, and the subset of states in which established and assumed properties hold. This subset of states is contained in the set of states in which the threats are negated by the assumed and established properties.

2.2 Effectiveness Analysis

Figure 3 illustrates the general approach to security effectiveness analysis. The figure shows three types of arcs - straight, dashed, and bold, denoting property derivation, property dependencies and threat countermeasures, respectively.

System function properties (represented as boxes labeled Established Property 1 through k) are

derived by analyzing the system functions at some level L (represented as the oval in the center of the figure) using logics (represented as ellipses labeled Logic 1 through n). The system functions, as well as the logics used to derive the properties, make assumptions about the protocols at level $L - 1$ or lower (represented as the rectangle at the bottom of the figure). The validity of derived results is checked by taking into account the property dependencies, and the assumptions about the environment of operation. The properties so derived are analyzed for their ability to invalidate a set of threats (represented as rectangles on the right). If the set of threats that are rendered infeasible by the system properties (both established and assumed) corresponds to those that are required to be countered by the system, then one can say that the system is *security effective*. In figure 3, we note that the set of threats that are rendered infeasible by assumed properties is disjoint from the set of threats that are countered by properties which are established using logics. This follows from the fact that if a property that counters a threat is established, then it need not be assumed.

The correspondence between assumed/established properties and threats, however, is not necessarily one to one. Often, a threat property may be expressible as conjunctions and/or disjunctions of component threat properties. In order to counter the threat, the system must support properties which are at least as strong as the negation of the threat property. For example, a threat property may be of the form $T \equiv T_1 \vee T_2 \vee T_3$. If *Properties of system* $\Rightarrow (\neg T_1) \wedge (\neg T_2) \wedge (\neg T_3)$, then the system is said to be effective in resisting the threat T .

2.3 An Example

Consider the threat of plaintext disclosure by cryptanalysis; i.e., a message encrypted with key K becomes known to a principal who does not have key K . Suppose that principals in the system can find the plaintext message contents without knowing the decryption key in only one of the following ways, namely

- T_1 Obtain the plaintext message from some other principal (i.e., the plaintext message is leaked);
- T_2 Obtain the ciphertext of a message and,
 - T_2 obtain the key K from some principal in the session (i.e., the key is leaked), and use it to decrypt the ciphertext message;
 - T_2 derive the plaintext by conducting cryptographic analysis on some N_1 or more known plaintext and ciphertext block pairs without possessing or deriving the key; or
 - T_2 derive the key K from the ciphertext by conducting cryptographic analysis on some N_2 or more plaintext and ciphertext block pairs encrypted with the key, and then decrypt the ciphertext.

Threat T_2 typically can not be assumed away in open, distributed systems, since any principal with access to the network can obtain ciphertext messages. However, this need not affect the

effectiveness of the system, since other properties may render the possession of ciphertext useless in finding the plaintext. For example, if the system is known to support the properties P_1 , P_2 , P_3 and P_4 below, then threat T_2 may not pose a problem.

P_1 Principals maintain the secrecy of the key and of the message contents (i.e., the principals handling the encryption key and plaintext message do not use untrusted code).

P_2 The plaintext cannot be derived from the ciphertext without the key using the knowledge of fewer than N_1 plaintext-ciphertext block pairs.

P_3 The key K cannot be derived from the knowledge of less than N_2 plaintext and ciphertext block pairs.

P_4 The lifetime of key K and encryption throughput are limited such that the total number of plaintext and ciphertext block pairs generated using K is less than the minimum of N_1 and N_2 .

The threat can be denoted as

$$T \equiv [T_1 \vee (T_2 \wedge (T_{2a} \vee T_{2b} \vee T_{2c}))]$$

From the properties of the system, we can deduce that

$$P_1 \Rightarrow \neg(T_1)$$

$$P_1 \Rightarrow \neg(T_2)$$

$$(P_2 \wedge P_4) \Rightarrow \neg(T_2)$$

$$(P_3 \wedge P_4) \Rightarrow \neg(T_2) \text{ Hence,}$$

$$P_1 \wedge P_2 \wedge P_3 \wedge P_4 \Rightarrow$$

$$\neg(T_1) \wedge \neg(T_{2a}) \wedge \neg(T_{2b}) \wedge \neg(T_{2c})$$

$$\Rightarrow \neg(T)$$

3 Modeling Threats

To argue that properties of system functions counter properties of security threats, it is essential to model system functions and threats. The invariants (P) obtained by analyzing the system functions imply that specific threat properties (T) are infeasible (i.e., $P \Rightarrow \neg T$) if the functions are implemented correctly.

For example, the specification and verification of cryptographic protocols of [3] uses a state transition model whose semantics allow the specification of system function properties as well as properties of threats. In the formal policy model of [3], the notion of an *intruder process*

formally models external threats. Intruder actions (i.e., security threats) are modeled as state transforms which can be initiated by untrusted processes or intruder processes. The policy properties (P) are restrictions enforced on state transforms. If these restrictions are correctly enforced, the transforms that describe intruder actions are not permissible, and hence, the threat(s) due to these “illegal” state transforms (i.e., which result in unsafe states) are countered.

An example of implicit threat modeling can be found in [13], where the properties of message integrity intruder actions are axiomatized in relation to the design parameters of message integrity protection mechanisms. This axiomatization aids in comparing threat properties to those of the message integrity protection goals.

4 Examples of Protocol Dependencies

In earlier work [8], we noted that the logics for cryptographic protocols must ensure that whenever the required assumptions or policies are not satisfied by a protocol, the use of the logic’s inference rules should fail to achieve desired, but unsupported, conclusions. In this section, we provide examples of protocol analyses using a fairly well understood logic [2] to illustrate the consequence of disregarding property dependencies on specific cryptographic properties.

4.1 Kerberos IV

In the analysis of the Kerberos protocol using [2], the protocol is shown to achieve its goal, namely that of securely distributing session keys to the two parties A and B in the presence of a hostile environment. At the end of the protocol, both parties are said to obtain “beliefs” in the origin of the key, namely the authentication server, and on the *freshness* of the key, meaning that it has never been used prior to the session. However, the results derived are dependent on the lifetime constraints of the long-term keys, and in particular, can be false if the number of available plaintext and ciphertext pairs are not limited by limiting the lifetime of those keys. We reproduce the simplified protocol description used in the analysis [2].

Protocol Description

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{T, L, B, K_s, \{T, L, Kab, A\}_{K_{bs}}\}_{K_{as}}$
3. $A \rightarrow B : \{T, L, K_s, A\}_{K_{bs}}, \{A, T\}_{K_{ab}}$
4. $B \rightarrow A : \{T + 1\}_{K_{ab}}$

The notation $\{M\}_k$ denotes plaintext message M encrypted with key k . T is a timestamp that the server attaches to its messages. The analysis shows that at the end of the protocol run, both A and B obtain beliefs about the source that generated the key K_s , and that K_s is a key that

was not used prior to the start of the session [2]. This is represented as:

A believes $A \xleftrightarrow{K_{ab}} B$

B believes $A \xleftrightarrow{K_{ab}} B$.

The protocol, however, can be vulnerable to at least two types of attacks. For example,

1. If encryption is performed using DES [7] (in Cipher Block Chaining mode since the text exceeds one block), party A can use all 2^{56} values of keys K to encrypt known text $\{T, L, K, A\}$. Then A can compare each $\{T, L, K, A\}_K$ with $\{T, L, K, A\}_{K_{bs}}$. If they match, then A can conclude that $K = K_{bs}$.
2. Party A can obtain known plaintext-ciphertext block pairs by repeatedly sending requests to the authentication server S for a key to communicate with B . If A can obtain sufficient known plaintext-ciphertext block pairs, then A may exploit the knowledge of those pairs and the DES function to derive the key K_{bs} , since the only unknown component in the encryption is the key (the DES function itself is not a secret).

The first attack can be countered by using confounding text within the encrypted text, hence decreasing the probability of success in the known plaintext attack. However, the presence of confounding text cannot prevent the intruder from obtaining known plaintext and ciphertext block pairs in the second attack. This is true because the presence of a confounder in the plaintext only affects its neighboring ciphertext block in Cipher Block Chaining mode [17]. However, from the complexity of the encryption algorithm and the size of the block cipher (i.e., those of DES in this case), a limit may be derived on N , the number of known plaintext-ciphertext block pairs required to uniquely determine the encryption key. If the protocol does not limit the lifetime of the long-term key, K_{bs} , such that the total number of plaintext-ciphertext block pairs is less than N , then the results derived using logics are not justifiable, since A could uniquely derive K_{bs} , and send a key K' to B .

While it seems essential that the lifetimes of all long-term keys must be limited, the dependency of message secrecy on their non-derivability from the plaintext-ciphertext pairs may not exist in all protocols. For instance, the protocol proposed by Davis and Swick [4], and implemented as the ENC_TKT_IN_SKEY option of the Ticket Granting Server exchange in [14] eliminates this dependency by using limited lifetime session keys instead of the long-term keys, thereby reducing the amount of ciphertext that can be obtained using any one key (viz., [12]). However, that protocol still depends on the strength of the ticket-granting server's long-term key; i.e., K_{TGS} . The lifetime of this key can be limited appropriately.)

4.2 Needham-Schroeder Shared Key Protocol

The analysis of the Needham-Schroeder protocol [16] using the BAN logic [2] indicates that the protocol falls short of its goals, namely that of distributing a fresh session key K_{ab} to A and B , and in particular, causing the two parties to “believe” that the key is fresh and was created by the

right source (i.e., the authentication server). The simplified protocol description is given below:

Protocol Description

1. $A \rightarrow S : A, B, N$
2. $S \rightarrow A : \{N, B, K, \{K, A\}_{K_{bs}}\}_{K_{as}}$
3. $A \rightarrow B : \{K, A\}_{K_{bs}}$
4. $B \rightarrow A : \{N\}_{K_{ab}}$
5. $A \rightarrow B : \{N - 1\}_{K_{ab}}$

At the end of the protocol run, according to the analysis, party B does not have evidence to believe that K has been issued during the current session by the server, since the key is delivered to B in message 3, which has no component which B believes to be fresh. While this result is reasonable, it does not necessarily mean that the protocol is vulnerable to replay attacks, since B can still verify whether A knows K by the handshake message exchange (i.e., messages 4 and 5). Replay of message 3 from a previous session would be successful only if the party which replays the message has also compromised the expired session key $K_{(expired)}$ [5]. Replay of message 3 without the knowledge of $K_{(expired)}$ would only allow the intruder (say X) to obtain $\{N\}_{K_{ab}(expired)}$, but would not enable X to reply with $\{N - 1\}_{K_{ab}(expired)}$. The ciphertext $\{N\}_{K_{ab}(expired)}$ would not be of much use to X in deriving $K_{(expired)}$, since the corresponding plaintext nonce N is unknown to X . The protocol itself does not produce plaintext-ciphertext pairs for the session key. If suitable confounders would be added to plaintext and if the lifetime of the session keys would be limited such that the maximum number of known plaintext-ciphertext block pairs available becomes insufficient to derive an old session key by cryptanalysis during the (limited) lifetime of the long-term key K , this protocol would be less vulnerable to replay attacks. (Note that the Kerberos protocol, which was derived from that of Needham and Schroeder's, includes confounders and a mechanism to limit the lifetime of session keys.)

The use of a logic in protocol analysis may inadvertently introduce assumptions about the underlying system. For example, the analysis of the Kerberos and Needham-Schroeder protocols above removes the first message from both protocols during the idealization step [2]. An assumption is made that a separate means of establishing the freshness of the second message exists [8]. For Kerberos, this assumption implies that (1) an *authenticated* source of time is available to all parties *prior* to running the authentication protocol; and (2) party A has a replay detection cache, and therefore has non-volatile memory, to verify the freshness of T . Assumption (1) is unrealistic in most systems, and (2) is unnecessarily restrictive (e.g., Kerberos is used with diskless workstations). For the Needham-Schroeder protocol, the assumption that the second message is fresh implies that party A remembers all the nonces sent in message 1 and not received in message 2, during the lifetime of the shared long-term key K . Again, this is both unrealistic and unnecessarily restrictive (e.g., use of non-volatile memory becomes necessary).

4.3 Andrew Secure RPC Handshake

In this example, we illustrate a protocol that allows an attacker to force the reuse of an expired session key, which is unknown to the attacker. The purpose of the attack is that of obtaining plaintext-ciphertext pairs, which can lead to the eventual discovery and replay of that key.

The simplified protocol description [2] for the Andrew Secure RPC handshake is shown below.

Protocol Description

1. $A \rightarrow B : A, \{N\}_{K_{ab}}$
2. $B \rightarrow A : \{N + 1, N\}_{K_{ab}}$
3. $A \rightarrow B : \{N + 1\}_{K_{ab}}$
4. $B \rightarrow A : \{K', N'\}_{K_{ab}}$

At the end of the protocol run, party A has no evidence to believe that K' has indeed originated from B , since A does not use any handshake messages to verify whether B actually knows K' or not. Hence, an intruder may replay message 4 consisting of some session key which is expired, and possibly make party A encrypt some message with $K'_{(expired)}$.

Unlike the Needham-Schroeder protocol, this protocol is vulnerable to replays irrespective of whether the expired key has been compromised or not. In addition, in the Needham-Schroeder protocol, an intruder could not obtain known plaintext-ciphertext block pairs for a session key beyond a preset limit, if this limit is specified and enforced within the underlying system (e.g., within the cryptographic facility [15]). In contrast, the Andrew Secure RPC Protocol permits an intruder to obtain known plaintext-ciphertext pairs even beyond the limits set by the underlying system. For example, the fourth message can be repeatedly inserted by party B possibly causing an unsuspecting party A to encrypt text with an expired session key K' . Thus, unlike the Needham-Schroeder protocol, this protocol may, in fact, aid the discovery and forced reuse of the expired session key K' by B .

Analyses using logics cannot distinguish between the properties of Needham-Schroeder and Andrew-Secure RPC protocols. The first protocol is shown to be weak because it does not result in the belief B believes $A \xleftrightarrow{K_{ab}} B$, and the second protocol is shown to be weak because it does not yield A believes $A \xleftrightarrow{K_{ab}} B$. These examples, and the Kerberos IV example, show that the analyses cannot take into account the dependency (or lack thereof) of the protocol properties on specific cryptographic properties (e.g., limited key lifetime) that may differ from protocol to protocol.

5 Dependency Graph

We have identified several property dependencies in cryptographic protocols. Figure 4 shows a graph which has properties (or assumptions) as its vertices and arcs showing dependencies. Cyclic dependencies are not illustrated because their elimination is reasonably well-understood [6]. The dependencies shown here are of two types - **(policy) goal-specific** and **protocol-specific**. The first type of dependencies (shown with straight arcs in Figure 4) appears in the analysis of all protocols that aim to establish specific (policy) goals (shown with dark ovals). The second type of dependencies (shown with dashed arcs) appears in the analysis of individual protocols, and take into account specific protocol attributes (e.g., types of encryption keys used, message freshness criteria, and so on). We briefly discuss both *goal-specific* and *protocol-specific* dependencies shown in the graph of Figure 4. In the dependency graph, the leaf nodes should be properties which can be related to actual design parameters (e.g., [13]).

We begin at the root of the dependency graph and proceed towards the leaves. The property *identity authentication using distributed session key or ticket* is dependent on

1. *Freshness of Session Key distributed*: Without this property, an old (possibly compromised) session key could be used for encryption, and the authenticity of encrypted messages cannot be ensured.
2. *Message sender recognition*: This is the property by virtue of which the sender of a message can be held accountable for the contents of the message. Recognizing the sender's identity is necessary in authenticating specific messages (and their contents), since not all parties are authorized to send all messages⁴.
3. *Possession of Session Key only by trusted parties*: If this property is not supported, the source of an encrypted message cannot be identified with certainty.

The property *Freshness of Distributed Session Key* is established using

1. *Nonce Verification Rule* [2], and
2. *Jurisdiction Rule* [2].

Hence, the *freshness of distributed session keys* is dependent on the validity of these two rules. In some protocols (e.g., [16]), freshness of session keys is also dependent on the fact that *old session keys are unbreakable*.

In protocols which fail to establish the freshness of session keys and do not allow session members to detect lack of key freshness using handshake messages (e.g., Andrew Secure RPC Protocol), the freshness of session key would depend on the lifetime restriction on the session key.

Possession of session keys only by trusted parties (i.e., confidentiality) can be ensured only if

1. the party which generates the session key has the authority to do so. Otherwise, the key may be leaked to intruders by the entity (process or machine) which generates the key.

⁴For instance, a message which has a secret key or ticket may originate only from a ticket granting server.

2. encrypted message contents are secret among session members. Since session keys are often encrypted with key encrypting keys, if secrecy of encrypted message contents cannot be ensured, the confidentiality of the session key cannot be established.

3. session members are honest (i.e., they have only trusted code). Without this property, confidentiality cannot be established.

In protocols which transmit plaintext messages, confidentiality is dependent on whether responses to plaintext messages are checked to see whether the plaintext was modified (e.g., [12]).

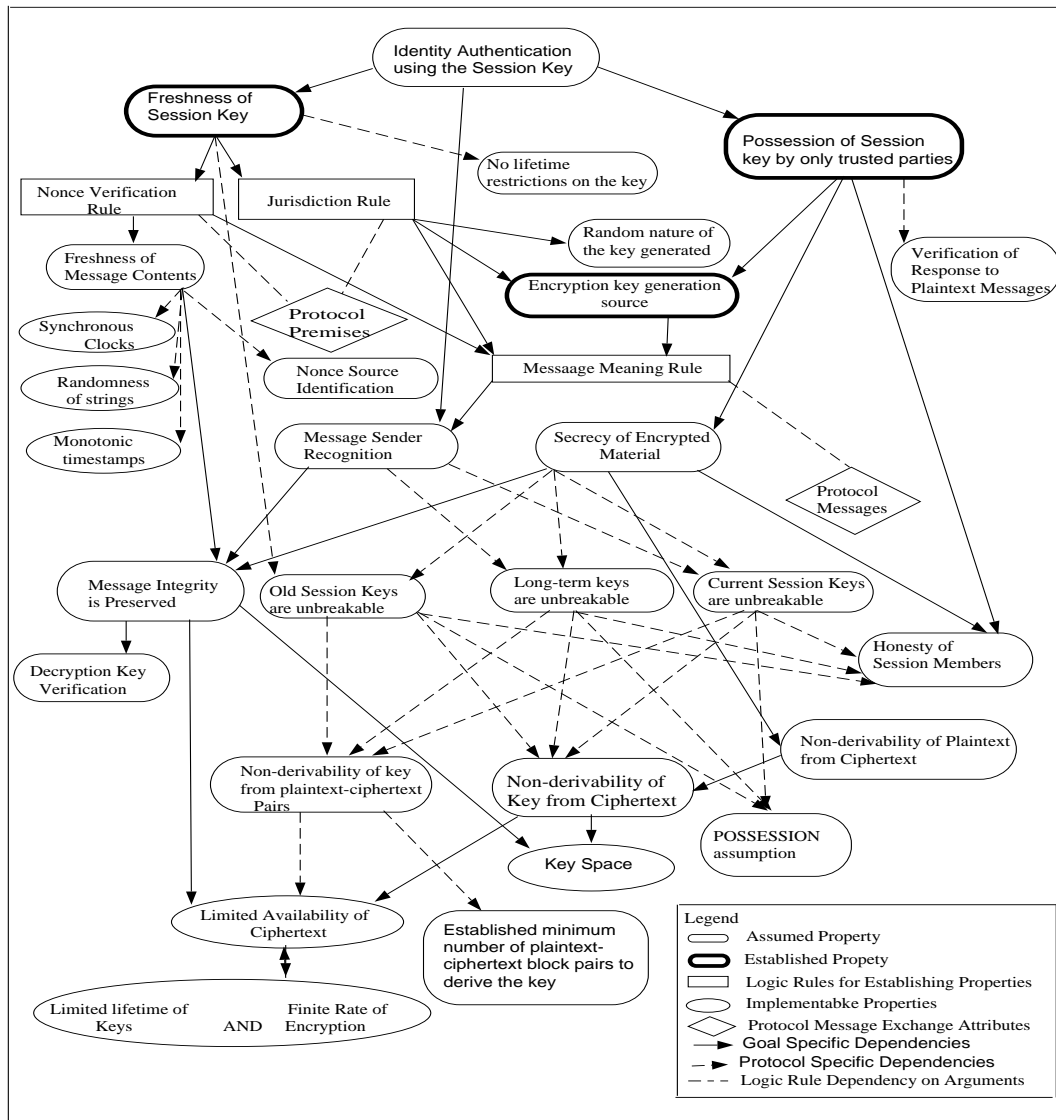
The validity of the *nonce verification rule* in analyzing protocol message exchanges depends on whether message freshness can be the basis for transforming beliefs of the form X **believes** Y **said** *Message* to X **believes** Y **believes** *Message*, if X **believes** that *Message* has some contents which are fresh. The validity of this rule is also dependent on the validity of the *message meaning rule* in establishing the statement X **believes** Y **said** *Message*.

This rule has an inherent dependency on the method used to establish the freshness of message contents. Freshness of entire messages can be inferred from the freshness of specific message contents only if message integrity is preserved. Hence, freshness of message contents is dependent on *message integrity*. Message freshness may be established using (and hence, is dependent on) one or more of

- Synchronous nature of clocks,
- Randomness of strings (nonces), or
- Monotonicity of timestamps.

Depending on the types of replay threats assumed, the establishment of message freshness may have additional property dependencies [9]. In addition, in systems which permit the simultaneous generation of the same nonce in two or more sessions, the freshness of message contents would depend on the fact that principals can identify the source which generates the nonce (e.g.,

an interleaved attack on the ISO protocol in [1]).



The validity of *jurisdiction rule* [2] in analyzing protocol messages is dependent on the

1. competence of the key generation source in generating random keys.
2. competence of the key generation source in preserving the secrecy of the key, and
3. the *message meaning rule*, since the former uses the latter to establish the message source.

Message meaning rule [2] can be applied for the analysis of protocol messages only if the sender of the message can be identified. Without message sender recognition, statements of the form

“*X said Message*” cannot be derived. Both *nonce verification rule* and *jurisdiction rule* are use *message meaning rule*, and hence, are dependent on it.

Message sender recognition is valid only if message integrity is preserved. Without message integrity, messages or parts thereof can be inserted or modified by intruders. In addition, depending on the types of keys that are used by the protocol for encryption, message sender recognition is also dependent on the fact that

- Long-term keys are unbreakable or
- Current session-keys are unbreakable.

Secrecy of encrypted material is a property which can hold only if the following properties are supported:

1. *Message integrity*: If message integrity protection mechanisms guarantee detection of message integrity compromises, then secrecy of messages which lack integrity may not be a concern. Unfortunately, this may not always be the case. For instance, in cipher-block chaining, message portions of known plaintext-ciphertext pairs can be spliced together to form new messages which pass checksum tests with high probability [17]. The contents of such spliced messages are not secret. Hence, secrecy is a function of whether integrity is preserved or not.
2. *Non-derivability of plaintext*: from the knowledge of ciphertext, without knowing the encryption key.
3. *Honesty of session members*: This property can be enforced by using only trusted software on trusted hardware platforms.

Further, based on the types of keys used for encrypting messages, *secrecy of encrypted material* is dependent on one or more of the following properties:

- Long-term keys are unbreakable.
- Session keys are unbreakable.
- Old session keys are unbreakable.

Message integrity depends on

1. *verification of decryption keys*: Without decryption key verification, one cannot distinguish between a message which has been decrypted with the wrong key, and a message whose integrity is not preserved.
2. *limited availability of plaintext-ciphertext pairs*, since these can be used to compose (splice) known ciphertext message portions to form new messages. The likelihood of success in this attack is a non-decreasing function of the number of available plaintext-ciphertext pairs.

3. *Size of encryption key space*, in protocols which use encryption for secrecy. If the encryption key space is small, an exhaustive search may have high rate of success, and may comprise a threat to the integrity of messages.

Unbreakability of encryption keys (i.e., current session keys, long-term keys or old session keys) is dependent on the following properties:

1. *Non-derivability of keys from ciphertext*: This property is dependent on the size of the encryption key space and the strength of the cryptosystem.
2. *Honesty of session members*, and
3. *Possession Assumption*: All other ways in which the key can be compromised are very unlikely to succeed (e.g., accidentally chancing upon the right key.)

In addition, in protocols which generate plaintext-ciphertext pairs (e.g., Needham-Schroeder Protocol generates plaintext-ciphertext pairs encrypted with long-term keys [16]), unbreakability of these keys is also dependent on non-derivability of keys from these pairs encrypted with the corresponding keys made available by the protocol message exchanges.

Non-derivability of plaintext from ciphertext is dependent on *non-derivability of the encryption key from ciphertext*. The latter property is dependent on the size of the encryption key. If the key size is small, this property cannot be supported, since an exhaustive search can be used to deduce the right key (Note that this attack need not be dependent on the availability of plaintext-ciphertext pairs. If the decryption algorithm is used on a ciphertext block with all possible keys, the ciphertext will decrypt properly only if the right key is used) with a high probability of success.

Non-derivability of encryption key from plaintext-ciphertext pairs made available by the protocol messages is dependent on the limited availability of such pairs, and on the established minimum number of plaintext-ciphertext block pairs that are required to derive the key (e.g., [18]). *Limited availability of ciphertext* is dependent on the fact that the key lifetime is limited, and the rate of encryption is limited.

6 Conclusions

To demonstrate the security effectiveness of cryptographic protocols, protocol properties must be specified in terms of the threats that need to be countered and the properties of lower level mechanisms that are assumed. The protocol properties derived using logics may then be validated by analyzing whether the underlying properties are supported. To perform this analysis, property dependencies must be identified. Ignoring these dependencies can lead to erroneous (false positive or false negative) conclusions. Once the properties established and assumed are all validated, then their threat resistance features can be analyzed.

In this paper, we illustrate the security effectiveness of cryptographic protocols with respect to some general cryptographic threats; e.g., the availability of plaintext-ciphertext pairs to an attacker. Deriving the dependency graph such as the one presented in this paper, is a necessary step in analyzing the security effectiveness of cryptographic protocols. The dependency graph may also be used to derive quantitative measures of security effectiveness based on the evaluation levels of the leaf properties.

Security effectiveness analysis is intended to complement existing analysis methods to make them useful in practice. The approach of evaluating the effectiveness of functions by property abstraction, dependency analysis, and goal analysis (in this case, threat resistance) is extensible to the analysis of other system functions.

Acknowledgements

The first two authors are grateful to Tom Tamburo and Marty Simmons of IBM Federal Systems Company for their continued support and encouragement.

References

- [1] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kuttan, R. Molva, and M. Jung. Systematic design of two-party authentication protocols. *IEEE Journal on Selected Areas of Communications*, 1993.
- [2] M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. Technical Report 39, DEC Systems Research Center, 1989.
- [3] P.-C. Cheng and V.D. Gligor. On the formal specification and verification of a multiparty session protocol. *IEEE Symposium on Research in Security and Privacy*, 1990.
- [4] D. Davis and R. Swick. Network security via private-key certificates. *ACM Operating Systems Review*, 24(4), October 1990.
- [5] D.E. Denning and G.M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 1981.
- [6] Federal criteria for information technology security. Vol. 1, Protection Profile Development, version 1.0, National Institute of Standards and Technology and National Security Agency, 1992.
- [7] Des modes of operation. Federal Information Processing Standard, National Bureau of Standards, 1980.
- [8] V.D. Gligor, R. Kailar, S.G. Stubblebine, and L. Gong. Logics for cryptographic protocols - virtues and limitations. *IEEE Computer Security Foundations Workshop*, 1991.

-
- [9] L. Gong. Variations on the themes of message freshness and replay. *IEEE Computer Security Foundations Workshop*, June 1993.
 - [10] L. Gong, R. Needham, and R. Yahalom. Reasoning about beliefs in cryptographic protocols. *IEEE Computer Society Symposium on Research in Security and Privacy*, 1990.
 - [11] R. Kailar and V.D. Gligor. On belief evolution in authentication protocols. *IEEE Computer Security Foundations Workshop*, 1991.
 - [12] R. Kailar, V.D. Gligor, and L. Gong. On the security effectiveness of cryptographic protocols. Technical Report 93066, Electrical Engineering Department, University of Maryland, College park, MD 20742, December 1993.
 - [13] R. Kailar, V.D. Gligor, and S.G. Stubblebine. Reasoning about message integrity. *IFIP Conference on Dependable Computing for Critical Applications*, January 1994.
 - [14] J. Kohl and B.C. Neuman. The kerberos network authentication service, draft (revision 5). MIT Project Athena.
 - [15] S.M. Matyas. Key handling with control vectors. *IBM Systems Journal*, 30(2), 1991.
 - [16] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), December 1978.
 - [17] S.G. Stubblebine and V.D. Gligor. On message integrity in cryptographic protocols. *IEEE Symposium on Research and Privacy*, 1992.
 - [18] M. Wiener. Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information Theory*, 36(3), May 1990.