On the Power of Non-spoofing Adversaries

H.B. Acharya[1] and Mohamed Gouda[1,2]

[1] Department of Computer Science
University of Texas at Austin
[2] National Science Foundation
{acharya,gouda}@cs.utexas.edu

Abstract. One of the fundamental concepts in network security is the *active adversary.* Such an adversary is defined, in the classic paper by Dolev and Yao, as an adversary that (in addition to eavesdropping passively), can "impersonate another user and ... alter or replay the message". Thus, the original definition of an active adversary includes the ability to spoof (lie about its identity). In this paper, we study the special case of active adversaries who are restricted from spoofing. As in the original study by Dolev and Yao, the motivation of our adversary is to break the confidentiality of the message being transmitted using a cascade protocol (a protocol in which neither sender nor receiver name stamps the messages they send). We prove a very surprising result: our weaker adversary, who is restricted from spoofing, is in fact exactly as powerful as the unrestricted Dolev-Yao adversary with respect to the goal of breaking confidentiality of cascade protocols.

1 Introduction

Public-key encryption is widely used to secure network communication. As Needham and Schroeder [12] as well as Dolev and Yao [6] point out, these systems can be attacked by "active" adversaries that can, in addition to listening passively,"*impersonate another user* and ... alter or replay the message".

In recent years, there has been substantial research on guaranteeing the authenticity of packet information. IPsec [10] and hop integrity [9] are two important examples of signing packets. Cryptographic signatures are computationally expensive, so other groups have developed other anti-spoofing safeguards. The most popular approach is packet filtering (notably ingress filtering [7], Martian address filtering [1], forwarding-table based filtering, route-based distributed filtering [13], and Source Address Validity Enforcement [11]). A similar approach, packet tracing, involves observing traffic at routers and reconstructing a packet's actual path [2]. Network intrusion detectors such as DECIDUOUS [4] can also be used to locate an adversary.

Given this wide range of tools against spoofing, it becomes reasonable to assume that in many cases the active adversary is no longer able to "impersonate another user". How much of the power of Dolev and Yao's active adversary is lost when it cannot spoof?

In this paper, we study the power of a non-spoofing adversary to break the security of cascade protocols, as studied in the original paper that defines active adversaries [6]. We obtain an extremely surprising result: if the aim of the adversary is to compromise confidentiality, then there is no difference whatsoever between the power of an adversary that can spoof and an adversary that cannot. If a protocol can protect the confidentiality of a message from a non-spoofing adversary, it will also resist an adversary that can spoof.

In this paper, we first provide a proof of the above statement for the simple case of two-step cascade protocols. Next, we prove the result that the security of a general k-step cascade protocol is equivalent to the security of a set of two-step cascade protocols. Formally, we demonstrate how to convert any given k-step cascade protocol P into a set of two-step cascade protocols $P_{\{\}}$, such that P can be broken by adversary X iff $P_{\{\}}$ can be broken by X. (We define a set of protocols to be broken by an adversary iff at least one of the protocols in the set is broken by the adversary.) Note that, iff P can be broken by a spoofing adversary A, $P_{\{\}}$ can be broken by A also. By definition, there exist one or more two-step cascade protocols in $P_{\{\}}$ that can be broken by A. But by our first result, as these are two-step cascade protocols, they are also broken by a non-spoofing adversary Z. In other words, iff P is broken by a spoofing adversary A, $P_{\{\}}$ is also broken by the non-spoofing adversary Z. Applying the equivalence of P and $P_{\{\}}$ again, we find that this means P is broken by Z. This concludes our proof.

The next section introduces our notational and conceptual conventions.

2 Users and Adversaries

In this paper, we have used the lambda-calculus convention of representing the application of a function to an argument, so $F(X)$ is written FX. However, we assume right-associativity: FGX represents $F(G(X))$ (unlike lambda calculus, where FGX means $H(X)$ where $H = F(G)$).

As FX means $F(X)$, in order to represent concatenation, we use the comma operator. "F concatenated with X" is written F, X.

A *user* is a process with a unique identifier such as I, J or K. The users are connected by a communication network, and can send messages to each other. The protocol followed by such messages, in order to ensure their confidentiality, is the focus of this paper.

The users of a protocol form a *public key system.*

1. Every user X has two functions:
   (a) The public key function B_X
   (b) The private key function R_X
   Both B_X and R_X map finite binary sequences (i.e. numbers) to finite binary sequences.
2. The pairs (X, B_X) are available to all users.
3. R_X is known only to X.
4. B_X and R_X satisfy the conditions

  – $\forall M, B_X R_X M = R_X B_X M = M$.
  – It is cryptographically hard to obtain M from $B_X M$ without access to R_X.

  Note that the second condition forces every user X to have distinct B_X as well as R_X. (Otherwise, let $B_I = B_J$. As J has access to all public keys, it knows this. Now J can deduce that $R_J B_I M = M$, so it can obtain M from $B_I M$.)

The only operations users can employ are their public and private key functions. Note that it is not possible to distinguish whether it is encryption or decryption which is being applied to an argument, unless the argument is known. For instance, suppose user I applies B_I to the argument M; this is an encryption. But if I applies B_I to $R_I M$, it is a decryption.

However, several layers of these operations can be applied. For instance, user I has access to the key R_I and all keys B_X (where X is any user), so from M it can generate $R_I B_J B_I M$. Such a combination of operations is called a *key sequence.*

Adjacent matching public and private key functions in a key sequence cancel each other out. So for instance, $R_I B_J R_J B_I M = R_I B_I M = M$. A key sequence with no such adjacent matching pairs is called a *simple* key sequence. Reducing a key sequence to a simple key sequence by recursively removing all such matching pairs is called *simplification.*

The *adversary* is a valid user in the protocol system, whose motive is to obtain the message M being communicated between two other users (say I and J). In this paper, we consider two models of adversary:

1. The "classic" Dolev-Yao adversary A.
2. The "non-spoofing" adversary Z.

The adversary A can perform the following actions:

1. A can obtain any message passing through the network.
2. As a legitimate user, A can send any message to any other user in the network. In particular, A can send messages to I with a fake source address stating the message is from J, and similarly, can send messages to J pretending to be I. Of course, it can also send messages that claim, honestly, that they are from A.
3. A has a private key function R_A, and access to the public key functions of all the other users in the system. It can apply any key sequence composed of these keys, to any message it obtains.

A has the following restrictions to its power:

– A cannot break the cryptography used. For instance, it cannot extract M from $B_I M$ without obtaining R_I.
– A cannot tamper with the public information. For instance, it cannot cause, say, I to think B_A is the value of B_J.

A can obtain messages, apply its own keys, and send messages to any user *with its own or a different source address*. Hence, as A can masquerade as another user (*spoof*), we refer to it as a *spoofing* adversary.

Adversary Z is identical to A, but has one more restriction on its power : it is a valid user of the system, and can obtain messages, apply its own keys, and send messages to any user, but such messages *bear the true source address*. Such an adversary cannot spoof, i.e. lie about its identity in a message; hence it is called a *non-spoofing* adversary.

In the next section, we describe the simple family of two-step cascade protocols; section 4 shows that A and Z are equivalent with respect to their power to break the security of these protocols.

3 Two-Step Cascade Protocols

In this section, we discuss a simple class of cascade protocols, consisting of only two steps : a *request* and a *reply*. The objective of such a protocol is to transmit a secret message (the plaintext) M between two users. This is a simplification of the general theory for k-step cascade protocols developed in [6].

A *two-step cascade protocol* is defined by two sets of key sequences a_{XY} and b_{XY}.

$$a_{XY} \in \{R_X, B_X, B_Y\}^*$$
$$b_{XY} \in \{R_Y, B_X, B_Y\}^*$$

where X and Y are any two users. In practice, X and Y are distinct (there is no reason why a user should send messages to itself).

The protocol has the property of being *uniform*: the key sequences have the same structure, irrespective of which users are trying to communicate. For instance, suppose $a_{IJ} = R_I B_J B_J$; then in this protocol, $a_{KL} = R_K B_L B_L$. Note that it is not necessary that $a_{XY} = a_{YX}$ or $b_{XY} = b_{YX}$.

The first class of messages in the protocol, the *request*, has the form

$$X \rightarrow Y : X, a_{XY}M, Y$$

The only part of the message that is encrypted is the plaintext. Source and destination are sent in the clear.

The second class of messages, the *reply*, has the form

$$X \leftarrow Y : Y, b_{XY}a_{XY}M, X$$

Note that b_{XY} is applied to the entire sequence $a_{XY}M$ and not to the original message M.

A well-known example of such a protocol is due to Diffie and Hellman [5].

$$\begin{aligned} I \rightarrow J &: I, B_J R_I M, J \\ I \leftarrow J &: J, B_I R_J B_I R_J B_J R_I M, I \\ &= J, B_I R_J M, I \end{aligned}$$

We observe that $a_{XY} = B_Y R_X$ and $b_{XY} = B_X R_Y B_X R_Y$ in this protocol.

3.1 Conditions for Security

For a protocol to be secure, the adversary should not be able to extract the message plaintext M. (Note that, in this paper, we only consider attacks on confidentiality, and not on other measures of security such as freshness and integrity. For example, we do not care if A obtains messages by intercepting them instead of eavesdropping. We are also not concerned by attacks in which A generates and injects new messages into the system impersonating another user.)

We call the entire key sequence applied to the plaintext in a message the *guard* of the message. The adversary should not be able to remove the entire guard if the message is secure.

We now provide an example of a successful attack. Suppose I and J are using the Diffie-Hellman protocol shown in the previous section. A breaks the security in the following way :

1. A captures the first message, $B_J R_I M$.
2. A uses the protocol to send this message to J as a new "request" and receives the corresponding "reply" in return.

$$\begin{aligned} A \rightarrow J &: A, B_J R_I M, J \\ A \leftarrow J &: J, B_A R_J B_A R_J B_J R_I M, A \\ &= J, B_A R_J B_A R_I M, A \end{aligned}$$

3. A applies the key sequence $B_I R_A B_J R_A$ to the payload, obtaining M.

This attack proves that the Diffie-Hellman protocol which we presented in the previous section is clearly not secure.

However, an interesting feature of cascade protocols is that there exist clear necessary and sufficient conditions for the security of a cascade protocol. These conditions are derived in [6]; we will briefly discuss them below.

$\Sigma_A = R_A \cup \{\forall X : B_X\}$ is the library of keys user A has access to.

$\Pi_A = \{\forall X : b_{AX}\}$ is the library of key sequences user A can indirectly cause to be applied to a message. This is because, if A sends a message M_1 as a request to I, it receives $b_{AI} M_1$ as the reply.

Communication between users I and J using protocol P:

$$\begin{aligned} I \rightarrow J &: I, a_{IJ} M, J \\ I \leftarrow J &: J, b_{IJ} a_{IJ} M, I \end{aligned}$$

is secure from attack by adversary A if (and only if) there exists no sequence a'_{IJ} with both the following properties:

– a'_{IJ} is composed only of keys and key sequences which the adversary A can cause to be applied to a message, i.e.

$$a'_{IJ} \in (\Sigma_A \cup \Pi_A)^*$$

– The simplified key sequence $a'_{IJ} a_{IJ}$ is the empty key sequence, i.e. $a'_{IJ} a_{IJ} M = M$.

(Note that there is no need to separately state that the adversary should also not be able to remove $b_{IJ}a_{IJ}$, the guard of the reply message. This requirement is covered by the non-existence of a'_{IJ} defined above. The reason for this asymmetry is that $b_{IJ} \in \Pi$, i.e. the adversary can apply b_{IJ} to a message. To see how, note that the adversary can send $a_{IJ}M$ to J, pretending to be I, and obtain $b_{IJ}a_{IJ}M$.

Now, consider the case that there exists a key sequence c_{IJ} such that the adversary can apply c_{IJ}, and $c_{IJ}b_{IJ}a_{IJ}M = M$. But in this case, there clearly exists a sequence a'_{IJ}: $a'_{IJ} = c_{IJ}b_{IJ}$. Thus, the condition that the adversary A should not be able to remove the guard of the reply is subsumed by the condition that A should not be able to remove the guard of the request.)

Protocol P is secure from adversary A iff, for all possible choices of I, J, and A (given A is not I or J), communication between users I and J using P is secure from attack by A.

In the next section, we study the security of two-step cascade protocols from both adversaries A and Z defined in the previous section.

4 Security of Two-Step Cascade Protocols

The actions available to the spoofing adversary A are a proper superset of the actions available to a non-spoofing adversary Z, so A is clearly at least as powerful as Z. In this section, we prove the very interesting result that the converse also holds: Z is as powerful as A, with respect to the goal of attacking a two-step cascade protocol.

Theorem 1. *A non-spoofing adversary and a spoofing adversary are equivalent in power with respect to the goal of breaking the confidentiality of a two-step cascade protocol.*

Proof. We begin our proof by noting that the spoofing adversary A is at least as powerful as the non-spoofing adversary Z. If protocol P cannot be broken by A, then both A and Z are (equally) ineffective attacking it.

To prove that Z is also as powerful as A, we demonstrate that any protocol P that can be broken by A can also be broken by Z.

The necessary and sufficient conditions to ensure that P cannot be broken by A are given by Theorem 1 of Dolev and Yao [6]:

1. a_{IJ} contains either B_I or B_J.
2. If b_{IJ} has R_J then it also has B_J.

As A can break P, one of these conditions must be false. We now demonstrate that, if either of these two conditions does not hold, then Z can extract message M from the message $a_{IJ}M$, breaking protocol P.

1. Consider the case that a_{IJ} has neither B_I nor B_J.
   In this case, a_{IJ} is composed of R_I.
   As B_I is a public key, it is available to Z. Thus Z can remove any guard composed of R_I.
   Hence Z can obtain M from the message $a_{IJ}M$.

2. The second possible case where P is insecure occurs when b_{IJ} is composed of R_J and B_I only.
   We begin by observing two facts.
   First, for any G_x, the keys in every sequence G_x are constrained to be R_I, B_I, or B_J. In particular, there are three possible values for the left-most key in G_x.
   Second, b_{ZJ} is simply b_{IJ}, with B_Z substituted for B_I. In other words, b_{ZJ} is composed of R_J and B_Z only. But Z has R_Z and B_J. Thus, if any subsequence of b_{ZJ} (including b_{ZJ} itself) occurs as the outermost sequence of keys in a guard, Z can remove this sequence.
   We now present the attack Z can use to compromise protocol P.
   Note that a_{IJ} is a key sequence of finite length (say, of length n).
   We introduce the symbols $G_n, \ldots G_1, G_0$. G_x means the suffix of a_{IJ}, which has length x. Thus $G_n = a_{IJ}$, ... G_0 is the empty sequence.
   Furthermore, G_0 is a proper suffix of G_1, G_1 is a proper suffix of G_2, etc.
   We show that, given any G_xM, $0 \leq x \leq n$, adversary Z can always remove at least one of the left-most keys to obtain G_yM, $0 \leq y \leq x$.
   Let the $k+1$ right-most elements in b_{IJ} be $\ldots R_J B_I^k$. ($k \geq 0$. Note that, for the protocol to be insecure, it is guaranteed that there must be at least one R_J in the key sequence b_{IJ}.)
   (a) If the left-most element in G_xM is R_I, then Z can remove this R_I since Z has B_I.
   (b) If the left-most element in G_xM is B_J, then Z first applies the key sequence R_Z^k to G_xM. Next, Z (stating its true identity as Z) initiates protocol P with J:

$$Z \rightarrow J : Z, R_Z^k G_x M, J$$
$$Z \leftarrow J : J, b_{ZJ} R_Z^k G_x M, Z$$

   b_{ZJ} is the same as b_{IJ}, except that each occurrence of B_I is replaced by B_Z. In particular, its right-most $k+1$ elements are $R_J B_Z^k$. This sequence cancels out the $k+1$ leftmost elements in $R_Z^k G_x M$ (recall these elements were $R_Z^k B_J$).
   Adversary Z then removes all remaining elements of b_{ZJ} forming the outermost key sequence in the reduced $b_{ZJ} R_Z^k G_x M$. The resulting G_yM is shorter (at least one element shorter) than G_xM.
   (c) If the left-most element in G_xM is B_I, then by symmetry Z simply follows the same attack detailed in the item above. Z applies R_Z^k, then initiates protocol P with I, and so on. (Note that this algorithm removes B_K for any K that Z communicates with. In the previous item, we used $K = J$; here $K = I$. The working is identical.)

   This concludes our proof of the fact that Z can always remove at least the left-most element from the guard G_x of G_xM.
   But the length of the original guard G_n is finite. By well ordering, it is not possible to have an infinite chain of the form $G_nM, G_{n-1}M, G_{n-2}M \ldots$ without eventually reaching G_0M, i.e. M. Hence, at some point, Z will obtain M.

Hence, we conclude that Z is as powerful as A in compromising the confidentiality of two-step cascade protocols.

5 Security of k-Step Cascade Protocols

In this section, we generalize our study of two-step cascade protocols to k-step cascade protocols where $k \geq 2$. Such protocols consist of repeatedly sending messages back and forth between two users. (Note that we assume the traditional "ping-pong" model of cascade protocols, where only two users pass the message back and forth and apply layers of encryption and decryption.)

We begin by defining the following protocol used by I and J to securely communicate the confidential message plaintext M.

$$\begin{aligned} I \rightarrow J &: I, g^1_{IJ}M, J \\ I \leftarrow J &: J, g^2_{IJ}g^1_{IJ}M, I \\ I \rightarrow J &: I, g^3_{IJ}g^2_{IJ}g^1_{IJ}M, J \\ &\dots \\ I \rightarrow J &: I, g^k_{IJ}g^{k-1}_{IJ}...g^1_{IJ}M, J \end{aligned}$$

The initial step consists of encrypting the plaintext with a key sequence (in this case g_1) and sending the result to another user. In each subsequent step, a key sequence is applied to the entire message received in the step before. The result of the operation is sent to the other user.

This subsequent step is repeated until the total number of messages reaches k. Note that, although in the example k is odd, in general k may also be even, in which case the final message is of the form

$$I \leftarrow J : J, g^k_{IJ}g^{k-1}_{IJ}...g^1_{IJ}M, I$$

For convenience, we assume for the remainder of this section that the two legitimate users communicating are always I and J. This allows us to use the clean notation $g_1, g_2 \dots$ as shorthand for the hard-to-read $g^1_{IJ}, g^2_{IJ} \dots$.

If we consider any two consecutive steps (say steps l and $l+1$) of the protocol, they must be of one of the two forms

$$\begin{aligned} I \rightarrow J &: I, g_l...g_1M, J \\ I \leftarrow J &: J, g_{l+1}g_l...g_1M, I \end{aligned}$$

or

$$\begin{aligned} I \leftarrow J &: J, g_l...g_1M, I \\ I \rightarrow J &: I, g_{l+1}g_l...g_1M, J \end{aligned}$$

The first form results when l is odd, and the second when l is even.

The first form is obviously a two-step cascade protocol in its own right, with $a_{IJ} = g_l ... g_1$ and $b_{IJ} = g_{l+1}$. For the second form, rewriting it as

$$J \rightarrow I : J, g_l ... g_1 M, I$$
$$J \leftarrow I : I, g_{l+1} g_l ... g_1 M, J$$

shows clearly that it is also a two-step cascade protocol with $a_{JI} = g_l ... g_1$ and $b_{JI} = g_{l+1}$. (Note that in this case, J issues the request and I the reply; thus, we use a_{JI} instead of a_{IJ} and b_{JI} instead of b_{IJ}.)

From these observations, we can conclude the following lemma :

Lemma 1. *Any two consecutive steps of a k-step cascade protocol constitute a valid two-step cascade protocol.*

The set of all two-step cascade protocols "contained" in a k-step cascade protocol P is called the *decomposition* of P. There are $k - 1$ such two-step cascade protocols, consisting of steps 1 and 2, 2 and 3 ... $k - 1$ and k.

The question naturally arises as to how the security of a k-step cascade protocol is related to the security of the elements of its decomposition. We now show the very interesting result that, if we define a set of protocols to be secure iff all the member protocols in the set are secure, then the security of a k-step cascade protocol P and the security of its decomposition $P_{\{\}}$ are equivalent.

Theorem 2. *A k-step cascade protocol P is secure iff every two-step cascade protocol in its decomposition is secure.*

Proof. We need to prove both directions - if and only if. To prove security, we will again employ Theorem 1 of Dolev and Yao [6], which states that a cascade protocol is secure iff it satisfies the following two conditions:

1. The first key sequence applied to the message plaintext (i.e. g_1) must, in its simplified form, contain either B_I or B_J.
2. All key sequences that are subsequently applied must, in their simplified forms, contain B_I if they contain R_I and B_J if they contain R_J.

The decomposition of protocol P is the set

$$P_{\{\}} = \{P_1, P_2, \ldots P_{k-1}\}$$

where P_l is the two-step cascade protocol formed by steps l and $l + 1$ of P.

To prove the "if" direction, suppose P is insecure even though $P_{\{\}}$ is secure. As P is insecure, it must violate at least one of the two conditions given above.

1. Consider the case that P violates the condition 1 above. In this case, the security is broken in the first step of P. As $P_{\{\}}$ is known to be secure, P_1 is secure.
   g_1 is the guard of the message in the first step of the secure protocol P_1. Thus by condition 1, g_1 (in its simplified form) contains either B_I or B_J.
   g_1 is also the first key sequence applied to the message plaintext in P. Hence the first guard applied to M in P contains B_I or B_J, i.e. P obeys condition 1. Thus we obtain a contradiction.

2. Suppose the security of P is broken in step no. l, where $l > 1$. (The first step has been proved secure in the above item.) If it is broken in more than one step, we choose the smallest l. If P violates condition 1, then the first step is broken; as in this case we consider $l > 1$, it follows that P is insecure because it does not satisfy the second condition of the theorem.
   (a) If l is odd.
   In this case, the simplified form of g_l contains R_I and not B_I.
   The second step of P_{l-1} involves applying the key sequence g_l. We know that the simplified form of g_l contains R_I and not B_I. Hence we see that P_{l-1} violates condition 2 - in other words, P_{l-1} is insecure. But we started with the assumption that $P_{\{\}}$ is secure, which of course requires that the protocol P_{l-1} is secure. Hence we have a contradiction.
   (b) If l is even.
   This case is exactly analogous to the one above.
   The simplified form of g_l contains R_J and not B_J. In other words, P_{l-1} is insecure. But P_{l-1} is known to be secure, as $P_{\{\}}$ is secure. Thus, we obtain a contradiction.

As the assumption that P is insecure but its decomposition $P_{\{\}}$ is secure always leads to a contradiction, it must be impossible. Hence we conclude that, if $\{P_1, P_2, \ldots P_{k-1}\}$ is secure, P must also be secure.

To prove the "only if" direction, suppose P is secure even though $P_{\{\}} = \{P_1, P_2, \ldots P_{k-1}\}$ is insecure.

Let l be the smallest value such that P_l is insecure. There are two ways in which this can happen: violation of the first and of the second conditions of the theorem above.

1. If P_l violates the first condition.
   (a) Consider the case $l = 1$.
   P is known to be secure.
   g_1 is the guard of the first message in secure protocol P.
   Hence, by condition 1, g_1 (in its simplified form) contains either B_I or B_J.
   g_1 is also the guard of the first message in P_1. As g_1 in its simplified form contains either B_I or B_J, P_1 does not violate condition 1. This contradicts our initial assumption that P_1 violates the condition.
   (b) Consider $l > 1$.
   P_l is insecure because $g_l ... g_1$ contains neither B_I nor B_J.
   In this case, the guard $g_l ... g_1$ is composed of only R_I and R_J. In other words, an adversary can capture message $g_l ... g_1 M$ and completely remove the guard to obtain plaintext M, using only the keys B_I and B_J (which are public keys).
   This means that protocol P is also insecure as it has the message $g_l ... g_1 M$. But protocol P is known to be secure - a contradiction.
2. If P_l violates the second condition.
   This means that g_{l+1} contains R_I but not B_I, or R_J but not B_J.

(a) If l is even (so $l + 1$ is odd).
g_{l+1}, which is applied by I, contains R_I but not B_I.
As $l + 1 > 1$, g_{l+1} is not the first key sequence applied to M in protocol P. (g_1 is this first key sequence.)
Consequently, if g_{l+1} contains R_I but not B_I, protocol P violates condition 2 of Dolev and Yao, so it is insecure. But P is known to be secure - a contradiction.

(b) If l is odd.
This case is exactly analogous to the one presented above.
g_{l+1} contains R_J but not B_J.
g_{l+1} is not the first key sequence applied to M in protocol P.
P is thus proven insecure by the Dolev-Yao theorem. This contradicts our initial assumption that P is secure.

Therefore, the assumption that P is secure but its decomposition is insecure always leads to a contradiction and must be impossible. Hence, given that P is secure, $\{P_1, P_2, \ldots P_{k-1}\}$ is also secure.

As we have proved both the "if" and the "only if" directions, we conclude our proof.

It is now simple to prove our main result.

Theorem 3. *A non-spoofing adversary and a spoofing adversary are equivalent in power with respect to the goal of breaking the confidentiality of a two-step cascade protocol.*

Proof. A set of protocols is secure iff every protocol in the set is secure. By duality, we define a set of protocols to be broken by an adversary iff at least one of the protocols in the set is broken by the adversary.

By Theorem 2, if (and only if) P can be broken by an adversary A, so can its decomposition $P_{\{\}}$. Let us consider that this successful adversary is a spoofing adversary A.

As $P_{\{\}}$ is broken, we know that there must exist at least one member protocol $P_l \in P_{\{\}}$ such that P_l is vulnerable to A. (If there are multiple vulnerable members, we can choose any one at random).

But P_l is a two-step cascade protocol. By Theorem 1, P_l (and hence $P_{\{\}}$) is also broken by a non-spoofing adversary Z.

From Theorem 2, we conclude that P is also broken by Z.

As Z can break the confidentiality of protocol P in every case where A can, Z is at least as powerful as A. It is known that A is at least as powerful as Z. From these two statements, we conclude that A and Z have equivalent power. This concludes our proof.

6 Conclusion

This paper presents a novel result concerning the security of cascade protocols, as defined in the landmark paper of Dolev and Yao [6]. In the original definition of an "active" adversary, the adversary is defined as having two novel powers: impersonation, and altering or replaying messages. (This is in addition to the powers of a "passive" adversary, i.e., eavesdropping and applying its own keys to captured messages). As there now exist effective means to prevent spoofing, we studied the effect on an active adversary if it is made unable to spoof. The result was completely unexpected: if the aim of the adversary is to compromise confidentiality, then a guarantee of no spoofing does not reduce the power of the active adversary to compromise cascade protocols. As a component of our proof of this theorem, we also obtain the independently interesting theorem that any k-step cascade protocol can be decomposed into a set of two-step protocols, and this set is equivalent in security to the original protocol.

We believe this result to be interesting because, while it is directly applicable only to one particular class of protocols (namely cascade protocols), it allows us to carry on a discussion of the powers of different kinds of adversaries. Clearly, the scope for further research extends in two directions. In the first place, it would be interesting to explore the relative power of adversaries that are restricted from using one or more of the powers mentioned above. The other question raised by our research is how the protocol model could be strengthened. Starting with the simple model of cascade protocols, if we use more general protocol models, at what point does the ability to spoof, for example, become non-redundant? Thus, our adoption of a simple protocol model exposes several interesting lines for further inquiry. In contrast, previous work, which began by assuming stronger models such as namestamp protocols, yielded mostly negative results. For example, it has been proven that namestamp protocols have no simple test for security [3].

We propose, as an open problem, the generalization of this discussion (regarding the powers of different adversaries) to include more practical protocols. Stated as a question, "How can the protocol model be strengthened so that it remains possible to derive interesting results about the power of the adversary, but the family of protocols covered by the model becomes broad enough to include protocols which are in practical use?" In our immediate future work, we intend to explore one such stronger model: we are studying whether our results continue to hold when cascade protocols are generalized to allow more than two parties [8].

References

1. Baker, F.: Requirements for ip version 4 routers. RFC 1812 (1995)
2. Bellovin, S.: Icmp traceback messages. Internet Draft: draft-bellovin-itrace-00.txt (2000)
3. Book, R., Otto, F.: On the security of name-stamp protocols. In: Third Conference on Foundations of Software Technology and Theoretical Computer Science, vol. 39, pp. 319–325 (1985)
4. Chang, H., Narayan, R., Wu, S., Vetter, B., Wang, X., Brown, M., Yuill, J., Sargor, C., Jou, F., Gong, F.: Deciduous: decentralized source identification for network based intrusions. In: Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (1999)

5. Diffie, W., Hellman, M.: Multiuser cryptographic techniques. In: Proceedings of the AFIPS (1976)
6. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
7. Ferguson, P., Senie, D.: Network ingress filtering: Defeating denial of service attacks which employ source ip address spoofing. RFC 2827 (2000)
8. Goldreich, O.: On the security of cryptographic protocols and cryptosystems. D.Sc. Thesis, Technion. (1983)
9. Gouda, M.G., Elnozahy, E., Huang, C., McGuire, T.: Hop integrity in computer networks. In: Proceedings of the 8th IEEE International Conference on Network Protocols (2000)
10. Kent, S., Atkinson, R.: Security architecture for the internet protocol. RFC 2401 (1998)
11. Li, J., Mirkovic, J., Wang, M., Reiher, P., Zhang, L.: Save: Source address validity enforcement protocol. In: Proceedings of IEEE INFOCOM (2002)
12. Needham, R., Schroeder, M.: Using encryption for authentication in large networks of computers. Communications of ACM 2, 993–999 (1978)
13. Park, K., Lee, H.: On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In: Proceedings of IEEE INFOCOM (2001)