Analysis of the DCS.v2 authentication protocol

Richard Newman, Lisa Dyson and Oswaldo Sabina

Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611-6120 USA

ABSTRACT

Developers of the UF Distributed Conferencing System, version 2 (DCS.v2) have proposed a protocol that involves the distribution of a session key by a trusted server to a principal with whom it will communicate. The protocol differs from standard key exchange protocols in that it attempts to address the issue of the delivery of a private key as well as the desired session key in a secure fashion. Because of the complexity of asymmetric keys, it is necessary for human principals to store their private keys until needed. The DCS.v2 protocol assumes that principals encrypt their private keys and store them on a network. When they seek to communicate with other principals, they must request that the appropriate encrypted key be issued to them by the server which will in turn be used to decrypt the session key. This paper presents the protocol and analyses it using BAN logic.

Keywords: authentication, key exchange, cryptographic protocol, roaming, encryption, groupware, interdomain, interenterprise, DCS.v2

1. INTRODUCTION

In order to preserve important security properties (such as confidentiality, integrity and availability) of computer and information systems, it is necessary to establish the identity of principals (users, hosts, and processes) reliably, that is, to authenticate them. Distributed systems pose greater challenges, since hosts communicate using messages, which must both establish their origin and maintain confidentiality. When users in a distributed system may themselves roam over several independently administered domains, this poses even more challenges to the security of the system. In distributed systems, cryptography is the method of choice to preserve confidentiality and integrity, and to establish authentication of principals as well as message origin.

1.1. Cryptography basics

Cryptography is the science of transforming messages into forms that are unreadable except by those who possess the appropriate decryption key. There are two basic types of cryptosystems, shared key and public key. The are often referred to as symmetric key and asymmetric key, respectively. Several good texts exist on cryptography[1–5] and here we only mention the characteristics of cryptosystems needed to make this paper understandable. Symmetric key cryptosystems make use of secret keys shared between certain principals (users or host machines). In asymmetric key cryptosystems, principals have a key that is made public and a private key that is known only to themselves. The public key may be used by anyone to encrypt a message that only the holder of the corresponding private key can decrypt. In many asymmetric cryptosystems, the public key may also be used to decrypt a message encrypted with a principal's private key, thereby establishing that that principal performed the encryption on the message (i.e., signed it). Cryptography is generally used to preserve confidentiality and to establish identity (authentication).

Symmetric key cryptosystems are more efficient than asymmetric key cryptosystems and are thus more desirable when principals want to engage in secure communication over an insecure medium such as a network. A primary difficulty in this is that principals must agree on a common secret key which will be used to ensure confidentiality and authentication. This must be done in a secure fashion so that eavesdroppers are unable to obtain any information, in particular, they must not be able to discover the secret key.

Security within the asymmetric environment is based on the fact that the derivation of a private key from a given public key is computationally infeasible.[6] In this system, key exchange is not necessary for communication

Other author information: (Send correspondence to R. Newman)
E-mail: nemo@cise.ufl.edu

and principals do not have to trust one another unconditionally. The primary difficulty in these systems is reliable association of a principal with its public key (certification). Assuming that a principal can be associated reliably with its public key, because a principal's private key is known only to himself, the authenticity and confidentiality of messages are certain.

In order for a cryptosystem to resist brute force attacks, the key space must be sufficiently large that search is infeasible. This translates to key length. Typically keys are too random and too long for the average human principal to remember. To overcome this, in practical systems principals will often encrypt their keys with a pass-phrase that is easy for them to remember and store the encrypted version of their key until it is needed. This technique works for both private keys and for secret keys, but is typically used for storage of private keys, with secret keys associated with a particular session or storage object encrypted with the principal's public key and therefore accessible once the private key has been recovered.

This paper presents and analyzes a protocol designed for the DCS.v2 system that involves a principal C who wishes to communicate with a trusted server S. S possesses the encrypted version of C's private key and generates the symmetric session key that the two principals will use for the duration of their secure communications. This paper focuses on the proposed authentication and key exchange protocol in which a principal retrieves his private key from the server, both establish authentication with each other, and they establish a session key.

1.2. Basic Assumptions and Requirements

This analysis will ignore the details of encryption schemes while focusing on the properties of the proposed protocol. We will assume that "perfect encryption," where the use of a decryption key is the only way to obtain information from an encrypted message, holds.[7] In addition, we will assume that principals have local clocks whose time agrees sufficiently for them to determine freshness of messages and agree on session durations. Participating principals (C and S) are assumed to be honest, that is, they will not violate the rules of the protocol which would subvert its goals. Of course, an adversary X may behave in any manner. Finally, we assume that time is monotonically increasing, so that causality is preserved, e.g., a message cannot be received unless it was sent earlier, and a message M that depends on the prior receipt of another message M' cannot be sent until M' has been received.

1.3. The Logic Used

The logic used in this analysis is primarily based on that specified by Burrows, Abadi and Needham, BAN logic.[8] The relevant rules and axioms will be cited as they are used in the proof. We refer you to the original report or its successors for further details concerning the rules and axioms of this logic.

In one respect we extend the expressiveness of the logic in a way not normally needed. Usually, only the assertion that K_A is the public key for some principal A is needed, with the corresponding private key known to whatever principals have it at the start of the protocol run. In the case of this protocol, the private key must be distributed explicitly, so we include a way to assert that K_A^{-1} is A's private key, corresponding to K_A. The use of this will be seen in the analysis section.

Some symbols used in BAN logic are worth introducing here:

- $A \triangleleft X$ means that principal A sees formula X,
- $A \mid\Rightarrow X$ means that principal A has jurisdiction over formula X,
- $A \mid\equiv X$ means that principal A believes formula X,
- $A \mid\sim X$ means that principal A said formula X at some time,
- $\#(X)$ means that formula X is fresh (i.e., is not a replay),
- $A \longrightarrow B : X$ means that A sent B message X,
- $\{X\}_K$ indicates that formula X is encrypted with key K (if K is a private key, then this is signing),
- $A \overset{K}{\longleftrightarrow} B$ indicates that K is a good session key for use between A and B,
- $\overset{K_A}{\longmapsto}\ A$ states that K_A is A's public key, and

- $A \overset{K_A^{-1}}{\longmapsto}$ states that K_A^{-1} is A's private key.

The last symbol, used to assert a private key, is an addition to those used in BAN logic, but is needed here.

2. UF DISTRIBUTED CONFERENCING SYSTEM, VERSION TWO

The University of Florida Distributed Conferencing System, version two (DCS.v2) is intended to supply the infrastructure for distributed collaboration across multiple administrative domains. It requires strong authentication in order to provide basic security, and further must handle the problem of users that roam, that is, users may access the system from any point at which their is a system interface, regardless of the administrative domain. This implies that the users will not always have access to stored secret information, as they could at their home domain, so the system must provide some other secure means to allow them access to their secret information and to authenticate themselves.

There are two types of principals in DCS.v2: server processes and user or client processes. Each type of process must be associated with an identifier for the site or the user with on whose behalf it operates. These identifiers (IDs) are managed by DCS.v2.

Relevant to this discussion, when an ID is created, a public key pair is also generated and the user or the site administrator creating the ID is authenticated by standard, human means (e.g., their student ID card is examined by the system administrator for their home site). A user enters a pass-phrase that is converted into a standard symmetric key, which is then used to encrypt the user's private key. This pass-phrase is expected to be kept secret by the user and the system so that only the user can recover the user's private key. Each site has its public key pair, generated when the site is created and added to the instance of DCS.v2. While it is not critical to this discussion, the site key is managed by secret sharing so that any one of several trusted administrators can start DCS.v2 servers.

Each user's public key along with additional information (name, timestamp, etc.) is signed with the home site's private key when the user ID is created. This is used as a certificate for the processes acting on behalf of the user to help authenticate themselves to other principals. Likewise, the user's private key is used to sign the home site's public key information so that the user's processes can authenticate the server later. When a user accesses a remote site, it is the remote site's server's responsibility to obtain a certificate for its public key signed by the user's home site's server. The mechanisms for this are not important for this paper, so we will show only the protocol used with the server's public key information signed by the user's private key.

These contortions are used to allow users who do not carry devices capable of storing their private key or the servers' public keys with them to roam. How these pieces of information are used to permit this will be given in the form of the DCS.v2 user authentication protocol in the next section.

2.1. The DCS.V2 User Authentication Protocol

The DCS.v2 protocol is a strong mutual authentication and key exchange protocol in which a server S stores a principal C's encrypted private key along with other information and sends it to him upon request. Both client and server then authenticate themselves to each other using the information they have or can derive, and they establish a session key for future use. All this is done in only three messages.

At the start of the protocol, the client has only two pieces of information: his name (C) and his secret passphrase from which a symmetric key (K_{pC}) used to preserve the confidentiality of his own private key (K_C^{-1}) and to authenticate the stored key to himself (only he knows the key K_{pC}, so only he could have produced the encrypted key data).

Each server has three pieces of information for each client for which the server is the home site, plus information for other servers to which it talks. Client information consists of two public key certificates (one for the client issued by the server, one for the server issued by the client) and one "private key certificate" for the client. Simplified versions of these are shown below.

$$C, K_C, S, \{N, C, K_C\}_{K_S^{-1}}$$
$$S, K_S, C, \{N_{pC}, S, K_S\}_{K_C^{-1}}$$
$$S', K_{S'}, S, \{N', S', K_{S'}\}_{K_S^{-1}}$$
$$C, \{N_{pC}, C, K_C^{-1}, K_C, S\}_{K_{pC}}$$

The first three of these are public key certificates (PKCs), which name the principal whose public key is certified, give that principal's public key, name the principal who vouches for that key (the authenticator), and then includes the principal, key, timestamp triple signed by the authenticator. Peculiar to this system, each user acts as an authenticator for their home server's public key (the second public key certificate above). The third one shows a PKC for a peer server S' that S can use to authenticate S' using standard peer public key authentication protocols. The final certificate is the "private key certificate" for C, which holds its name followed by a five-tuple consisting of a nonce N_{pC}, its name, its private key, K_C^{-1}, its public key, and the name of its home server, S, all encrypted by C's secret key, K_{pC}. Nonces N and N' represent timestamp and lifetime information for the servers. Both the secret key and the nonce N_{pC} are derived from C's secret passphrase, and S is explicitly mentioned in order for C to trust S later.

The protocol is specified as follows. Since only one client is involved, K_p is used as shorthand for K_{pC}. Likewise, N_p will stand for N_{pC}. For simplicity, S will send C three messages in response to its request, though in practice these would be combined into one message. Message numbers are provided to the left of each message.

$$
\begin{array}{rl}
M1 & C \longrightarrow S : C, N_c \\
M2a & S \longrightarrow C : \{N_p, C, K_C^{-1}, K_C, S\}_{K_p} \\
M2b & S \longrightarrow C : S, K_S, C, \{N_p, S, K_S\}_{K_C^{-1}} \\
M2c & S \longrightarrow C : \{\{N_C, N_S, K_{CS}\}_{K_S^{-1}}\}_{K_C} \\
M3 & C \longrightarrow S : \{N_S\}_{K_{CS}}
\end{array}
$$

Here N_c and N_s are randomly generated nonces for C and S respectively, and K_{CS} is the session key to be shared by C and S generated by S.

In narrative form, the client C sends the server S a request to open a secure connection, including a one-time nonce that is used both as a challenge and to bind the reply to this session. For simplicity's sake, we will assume that this connection is with its home server rather than a foreign one, but this case will be discussed further in the conclusions. The server then replies with the client's private key certificate, the servers PKC signed by the client (both of which are simply taken from the server's databases), and a newly generated quantity. This new quantity has the client's nonce (so the client knows it is in response to the current request), the server's nonce (as a challenge for the client to prove it can use the session key and to associate the last message with this session), and the session key generated by the server, all signed by the server and encrypted using the client's public key, which the server can access reliably from the client's PKC that it signed. Upon receipt of the private key certificate , the client can use its knowledge of the passphrase to generate K_p and N_p, which are then used to access and validate C's private key as well as establish its trust in S (that is, that it should trust any principal that authenticates itself as S). Using its private key, C then validates S's public key from its PKC, and is able with these two to read the nonces and session key from the last quantity $M2c$. The client verifies this message to belong to the current session by virtue of its nonce N_C, and that S is the source of the message from the validated public key for S that C holds. Given C's trust in S, it can then trust that the session key K_{CS} is good, so it uses it to respond to S's challenge nonce, N_S with the last message. Seeing this last message and its fresh nonce, S can believe that C is authentic, since C had to use its private key to discover both the nonce and the session key, and that C accepts the session key as good, since C is using it. Thus C and S authenticate each other, and establish a session key (i.e., both believe it is good and believe that the other believes it is good).

3. ANALYSIS OF THE PROTOCOL

The protocol will be analyzed using BAN logic, which requires that it be idealized and that the initial assumptions are formalized. Rules of the logic can then be applied to these to derive new beliefs. The final beliefs should include the following goal beliefs in order for the protocol to mutually authenticate C and S and for them to establish a session key, K_{CS}.

$$C \models C \overset{K_{CS}}{\longleftrightarrow} S$$
$$C \models S \models C \overset{K_{CS}}{\longleftrightarrow} S$$
$$S \models C \overset{K_{CS}}{\longleftrightarrow} S$$
$$S \models C \models C \overset{K_{CS}}{\longleftrightarrow} S$$

In other words, each principal believes the session key is good and that the other principal believes the same.

3.1. The Idealized Protocol

In order to analyze this protocol using the BAN logic, it must be put into idealized form. Messages are numbered to correspond to the actual protocol messages for ease of reference. Any information that is neither encrypted nor signed is omitted from the protocol, as any principal could have sent it. In particular, $M1$ is missing entirely, so the idealized protocol starts with S apparently sending C a message spontaneously. Further, explicit statements are made about the meaning of the information included in the messages; these will be discussed following the idealized protocol.

$$M2a' \qquad S \longrightarrow C : \{N_p,\ C \overset{K_C^{-1}}{\longmapsto}, \overset{K_C}{\longmapsto}\ C, S \Rightarrow C \overset{K}{\longleftrightarrow} S\}_{K_p}$$
$$M2b' \qquad S \longrightarrow C : \{N_p, \overset{K_S}{\longmapsto} S\}_{K_C^{-1}}$$
$$M2c' \qquad S \longrightarrow C : \{\{N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S\}_{K_S^{-1}}\}_{K_C}$$
$$M3' \qquad C \longrightarrow S : \{N_S, C \overset{K_{CS}}{\longleftrightarrow} S\}_{K_{CS}}$$

As mentioned earlier, the first message does not appear in the idealized protocol since it is not authenticable. Message $M2a'$ includes a nonce for C to use in validating the rest of the message, the statement that K_C^{-1} is its own private key, that K_C is its own public key, and the statement that S has jurisdiction over session keys between S and C. The next message is C's certificate for S, and asserts that K_S is S's public key. Again, the same nonce is included for use in validation by C. These two messages allow C to build its base of beliefs to the point that it can authenticate S and authenticate itself to S. Neither of these messages contain nonces generated during the current session, which ordinarily would present a problem for replay attacks, but that is not a problem here. Replay of these messages is in fact expected, and causes no harm, as they only have meaning for C and any other recipient would discard them as gibberish after attempting to decode them, as only C has the keys to decrypt them. Message $M2c'$ contains the nonces generated in this protocol run, N_C to bind it to C's request and establish freshness for C, N_S to bind C's reply to this message and to act as a challenge for C, by which C can authenticate itself to S. It also contains the assertion by S that K_{CS} is a good session key for C to use with S. It is signed by S so that C can authenticate S and the message. The final message, $M3'$, contains C's response to S's challenge, which allows S to authenticate C. It also contains the assertion that K_{CS} is a good session key for C and S, implicit in the actual protocol in C's use of K_{CS} in the reply. These are bound to the protocol run by the nonce, which also allows S to establish that C currently believes in the goodness of the session key.

3.2. Initial Beliefs

In order to prove the protocol formally, we need to set the initial beliefs held by the principals. These are as follows.

$$C \qquad \models C \overset{K}{\longleftrightarrow} C \tag{1}$$
$$C \qquad \models \#(N_p) \tag{2}$$
$$C \qquad \models \#(N_C) \tag{3}$$

$$C \quad \models C \models (B \Rightarrow C \overset{K}{\longleftrightarrow} B) \tag{4}$$
$$C \quad \models C \models \overset{K}{\longmapsto} B \tag{5}$$
$$C \quad \models C \models C \overset{K}{\longmapsto} \tag{6}$$
$$S \quad \models \overset{K_S}{\longmapsto} S \tag{7}$$
$$S \quad \models \overset{K_C}{\longmapsto} C \tag{8}$$
$$C \quad \models \#(N_S) \tag{9}$$
$$S \quad \models C \overset{K_{CS}}{\longleftrightarrow} S \tag{10}$$
$$\tag{11}$$

3.3. Application of BAN Logic

Here, each message will spur the development of additional propositions and beliefs, according to the rules of the logic.

The first message of the actual protocol again does not appear in the idealized protocol. There is no confidentiality, nor is there authentication of origin. S assumes that this message came from C and responds accordingly, but has no way of verifying that C was the actual sender of the message. Although S does not know where the request originated, C will know whether or not the response to his query came from S because a confidential and authenticated message will contain the nonce, N_C. If the response does not contain the proper nonce, C will reject it. The first message does not change the state of beliefs of any principal, so we move to the second message.

$$C \ \triangleleft \ \{N_p,\ C \overset{K_C^{-1}}{\longmapsto}, \overset{K_C}{\longmapsto} C, S \models C \overset{K}{\longleftrightarrow} S\}_{K_p} \tag{12}$$

C sees the message $M2$ sent by S, actually created by C itself some time earlier. From C's possession of its passphrase-derived key, K_p, C may view the plaintext contents of this message and add to its beliefs.

$$C \quad \triangleleft N_p,\ C \overset{K_C^{-1}}{\longmapsto}, \overset{K_C}{\longmapsto} C, S \models C \overset{K}{\longleftrightarrow} S \tag{13}$$
$$C \quad \models C \mid\sim N_p,\ C \overset{K_C^{-1}}{\longmapsto}, \overset{K_C}{\longmapsto} C, S \models C \overset{K}{\longleftrightarrow} S \tag{14}$$
$$C \quad \models \#(N_p,\ C \overset{K_C^{-1}}{\longmapsto}, \overset{K_C}{\longmapsto} C, S \models C \overset{K}{\longleftrightarrow} S) \tag{15}$$
$$C \quad \models C \models N_p,\ C \overset{K_C^{-1}}{\longmapsto}, \overset{K_C}{\longmapsto} C, S \models C \overset{K}{\longleftrightarrow} S \tag{16}$$
$$C \quad \models C \models C \overset{K_C^{-1}}{\longmapsto} \tag{17}$$
$$C \quad \models C \models \overset{K_C}{\longmapsto} C \tag{18}$$
$$C \quad \models C \models S \models C \overset{K}{\longleftrightarrow} S \tag{19}$$
$$C \quad \models C \overset{K_C^{-1}}{\longmapsto} \tag{20}$$
$$C \quad \models \overset{K_C}{\longmapsto} C \tag{21}$$
$$C \quad \models S \models C \overset{K}{\longleftrightarrow} S \tag{22}$$

This flurry of changes in C's beliefs is justified by C's receipt of the message $M2a$, and that it can decrypt the message using a key that it believes that only it holds. Seeing the nonce N_p, which C believes to be fresh, bound to the rest of the message by encryption causes C to accept that the message is fresh (this is how it is expressed in BAN logic - though this meaning is stretched here since the nonce is long-lived; this is justified by is use for only this purpose and with the two messages generated by C for this purpose). A fresh message said by a principal implies that the principal believes the statement contained in the message, which leads to the peculiar assertion that

C believes that C believes the statements. Believing (or seeing or saying) a compound statement implies believing (etc.) each component of the statement, a rule we will omit to mention explicitly from now on. Since C believes it has jurisdiction over private keys for itself, jurisdiction over public keys for anyone, as well as jurisdiction over whom it should trust for session keys, C can believe that K_C^{-1} is its private key, that K_C is its public key, and that S should be trusted to generate session keys. Jurisdiction in C's beliefs should be construed to mean that C trusts itself when it says something about these keys, not that it is capable of generating the keys.

Now C sees the second part of the second message, $M2b'$.

$$C \quad \triangleleft\{N_p, \overset{K_S}{\longmapsto} S\}_{K_C^{-1}} \tag{23}$$

$$C \quad \triangleleft N_p, \overset{K_S}{\longmapsto} S \tag{24}$$

$$C \quad \models C \mid\sim N_p, \overset{K_S}{\longmapsto} S \tag{25}$$

$$C \quad \models \#(N_p, \overset{K_S}{\longmapsto} S) \tag{26}$$

$$C \quad \models C \models N_p, \overset{K_S}{\longmapsto} S \tag{27}$$

$$C \quad \models C \models \overset{K_S}{\longmapsto} S \tag{28}$$

$$C \quad \models \overset{K_S}{\longmapsto} S \tag{29}$$

Here again, C sees the message, and having the key to decrypt it, sees the its contents. Believing that only it possesses its private key, C believes that C said the contents and by virtue of the nonce, believes the message to be fresh. This allows C to believe that it believes the message's assertion about S's public key, and because of the initial assumption that C has jurisdiction over public keys (in the sense that it will believe its own assertions about public key bindings), C accepts that K_S is S's public key.

These messages set the stage for interpretation of the third part of the second message, $M2c'$. Here is where S injects new information into this run of the protocol, and where the message must be protected against replay with a nonce.

$$C \quad \triangleleft\{\{N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S\}_{K_S^{-1}}\}_{K_C} \tag{30}$$

$$C \quad \triangleleft\{N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S\}_{K_S^{-1}} \tag{31}$$

$$C \quad \triangleleft N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S \tag{32}$$

$$C \quad \models S \mid\sim N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S \tag{33}$$

$$C \quad \models \#(N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S) \tag{34}$$

$$C \quad \models S \models N_C, N_S, C \overset{K_{CS}}{\longleftrightarrow} S \tag{35}$$

$$C \quad \models S \models C \overset{K_{CS}}{\longleftrightarrow} S \tag{36}$$

$$C \quad \models C \overset{K_{CS}}{\longleftrightarrow} S \tag{37}$$

Using its own private key, C can view the signed message contents, and using S's public key, C can view the message plaintext. Since the plaintext was signed by S, C believes S said it, and since the contents are bound to the fresh nonce N_C, C believes the message to be fresh and hence, that S believes the message's assertions. Since C certified S to itself as having jurisdiction over session keys, C accepts the session key S sent it.

Having done this, C then formulates a reply message, encrypting the nonce S generated with the session key S provided it.

$$S \quad \triangleleft\{N_S, C \overset{K_{CS}}{\longleftrightarrow} S\}_{K_{CS}} \tag{38}$$

$$S \quad \triangleleft N_S, C \overset{K_{CS}}{\longleftrightarrow} S \tag{39}$$

$$S \quad \models C|\sim N_S, C \overset{K_{CS}}{\longleftrightarrow} S \tag{40}$$

$$S \quad \models \#(N_S, C \overset{K_{CS}}{\longleftrightarrow} S) \tag{41}$$

$$S \quad \models C|\equiv N_S, C \overset{K_{CS}}{\longleftrightarrow} S \tag{42}$$

$$S \quad \models C|\equiv C \overset{K_{CS}}{\longleftrightarrow} S \tag{43}$$

S possesses the session key, so S both sees the message contents and believes it to have been sent by C. Viewing the fresh nonce N_S, S believes the message to be fresh, hence that C believes its assertions. In particular, S believes that C has accepted the session key S chose.

With this, the four goals have been met. C's goal beliefs are met by 36 and 37, while S's goal beliefs are met by S's initial assumption 10 and by 43.

4. CONCLUSIONS

The DCS.v2 Protocol successfully uses cryptographic keys to maintain a confidential, mutually authenticated key exchange as well as to maintain message integrity. The resulting beliefs of the principals are that the session key is good and that the other principal holds the same belief. With the implementation of this protocol, principals will be able to engage in secure communication without facing the challenge of memorizing an asymmetric key or carrying physical devices prone to loss or damage.

The protocol introduces some notions rather strange for standard authentication protocols. To wit, one of the principals possesses neither its private key nor the trusted server's public key at the start. In fact, the client does not even know for sure that the server is trustworthy at first - this is only established after validation of the private key certificate. That the client only accepts the authority of the server to issue session keys after the protocol run has started is also unusual for these types of protocols, but not uncommon in practice (e.g., through chained authentication).

Furthermore, in order for the client to validate this certificate and the one it previously signed for the server's public key, it is necessary to include a timestamp or nonce that it can derive later, during a run of the protocol. This is contrary to the usual semantics of nonces (that they are fresh only if they were generated in the current run of the protocol). It is not expected for the client to remember the nonce used in the certificates, so it is derived from the one thing the client is expected to recall, its passphrase. Use of a nonce in this fashion is not so uncommon in practice - most certificates have some form of lifetime information or serial number associated with them for validation (and revocation) purposes.

It is worth noting that C may contact some other server rather than its home server at startup time. If a foreign server S' is contacted by a client C for initiation of a secure connection, then the client must be given the means to authenticate the foreign server. This consists of the PKS for S' signed by S, C's home server. For the proof to handle this level of indirection, C must also trust S not only to have authority over public keys for other principals, but to have the authority to confer authority upon other principals for the generation of good session keys (which S' will do). The protocol needs the PKC for S' signed by S in the second message, including a designation that S' is a trustworthy server. A few more steps are necessary in the proof to arrive at the same conclusions, but the bulk of it remains unchanged.

ACKNOWLEDGMENTS

The authors would like to express their appreciation to the U. S. Army for its partial support of this work through contract DASG-94-C-0076.

REFERENCES

1. B. Schneier, *Applied Cryptography, 2nd Edition*, John Wiley & Sons, New York, 1997.
2. D. R. Stinson, *Cryptography - Theory and Practice*, CRC Press, Boca Raton, 1995.
3. C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*, Prentice Hall, Englewood Cliffs, NJ, 1995.
4. S. Vanstone, *Handbook of Cryptography*, CRC Press, Boca Raton, 1998.
5. W. Stallings, *Network and Internetwork Security Principles and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1995.
6. J. van Tilburg, "Secret-key exchange with authentication," in *Computer Security and Industrial Cryptography*, *Springer-Verlag Lecture Notes in Computer Science* , 1991.
7. N. Heintze and J. D. Tygar, "A model for secure protocols and their compositions," *Proc. IEEE Computer Society Symposium on Research in Security and Privacy* , pp. 2–13, 1994.
8. M. Burrows, M. Abadi, and R. Needham, *Research Report 39: A Logic of Authentication (revised)*, Digital Systems Research Center, Palo Alto, CA, 1990.