Secure and Minimal Protocols for Authenticated Key Distribution

Philippe Janson Gene Tsudik
IBM Zurich Research Laboratory
Säumerstrasse 4
CH-8803 Rüschlikon - Switzerland
{*pj,gts*}*@zurich.ibm.com*

November 5, 1993

DRAFT–please do not re-distribute...

Abstract

The problem of secure key distribution has been the subject of much attention in the recent years. This paper describes a novel method for authenticated key distribution in the distributed systems environment. In particular, a *braiding* technique for key distribution is introduced. The underlying protocols are extremely compact in both the number of messages and message sizes which facilitates their application at any layer (at lower layers, in particular) in the protocol hierarchy. Furthermore, the protocols are shown to be resistant to a wide range of interleaving attacks. All this is achieved with minimal computational requirements and without the necessity of using traditional encryption (a strong one-way function suffices.)

Keywords: Network Security, Network Protocols, Authentication, Key Distribution.

1 Introduction

Research in authentication protocols has been fairly active since the publication in the late 1970s of Needham and Schroeder's landmark paper [13]. In it, they proposed a family of protocols for two- and three-party authentication and key distribution. The distinguishing feature of these protocols was the use of encryption for authentication. The protocols were subsequently shown to contain some subtle bugs and weaknesses (e.g., [6]). Nonetheless, the protocols in [13] served as a basis for the well-known Kerberos network security service originally developed at MIT [15, 16]. Kerberos has been subject to considerable criticism both for the protocols it is based on as well as for their implementation (see, for example, [1].)

More recent research efforts started to develop (or provide tools needed for developing of) authentication protocols with some assurance of security. Most notably, formal tools for testing authentication protocols were developed, e.g., BAN logic [5] and its successor, GNY logic [8]. Instead of devising specific protocols, these methodologies provide a means for showing that appropriate beliefs are attained as a result of running an authentication protocol. They are, however, no panacea as there remain a number of issues (i.e., non-disclosure of secret information) that these methodologies do not address.

Other efforts, in particular, Bird et al. [2, 3], concentrated on constructing protocols that can be shown secure (in some restricted definition of the term.) The main result of Bird et al. [2, 3] is a two-party authentication protocol shown to be secure against an important class of attacks known as *interleaving* attacks. Such attacks are based on the adversary's ability to use either i) legitimate flows obtained from past executions of the protocol or ii) protocol flows elicited by the adversary from legitimate parties.

Both efforts stopped short of demonstrating provably secure two- and three-party key distribution protocols. In fact, the follow-up to [2, 3] is the family of three-party key distribution and authentication protocols [4] that has been subsequently realized in an actual network security service, KryptoKnight [12]. The underlying protocols, however, have not been shown secure; at least, not insofar as the authentication protocols in [2, 3].

The purpose of this paper is to demonstrate a basic, secure two-party key distribution (and authentication) protocol with properties very similar to the one in [2]. The protocol described below can in turn be used as a basic building block for constructing more elaborate, e.g., three-party or inter-domain, key distribution protocols.

2 Terminology

The following terminology is adopted throughout this paper:

- 2PAP/3PAP - two- and three-party authentication protocols, respectively.
- 2PKDP/3PKDP - two and three-party key distribution protocols, respectively.
- A, B, P, Q - full principal names (e.g., in X.500 format). All names are assumed to be distinct.
- S - full name of the authentication server; S is assumed to be universally trusted by all constituent principals.
- $E_K(X)$ - *encryption*[1] of plaintext block X under key K. Both K and X are each assumed to be not longer than the basic block size of the underlying encryption function, e.g., 64 bits in case of DES [10] or 512 bits in case of MD5 [14].
- $MAC_K(X)$ - DES-style message authentication code (MAC) computed in CBC mode over plaintext X with key K. However, this notation does not in any way imply the use of DES.
- K_{ab} - all symbols beginning with K are keys. The two-letter subscript identifies the two principals sharing that key (e.g., A and B share K_{ab}).

[1] We use the term *encryption* to refer to a single-block transformation of cleartext input. This includes traditional encryption function (such as DES) as well as strong one-way functions such as MD5 (when the key is used as a suffix and/or a prefix).

- N_{ab} - all symbols beginning with N are nonces, i.e., unpredictable, used-only-once random numbers. The first letter of the subscript (e.g., a in N_{ab}) refers to the party that originated the nonce while the second subscript letter identifies the target party.
- $A \Longrightarrow$ B X - this notation captures a single protocol flow; it denotes that A sent a message X to B.
- $A \stackrel{auth}{\longrightarrow} B$ - denotes one-way authentication of A to B.
- $A \stackrel{auth}{\longleftrightarrow} B$ - denotes mutual (two-way) authentication of A and B.
- Superscripts i, j and n are used to denote different protocol runs.

3 Two-party Key Distribution (2PKDP)

As mentioned in the introduction, both [2, 3], concluded with a design of a basic secure two-party authentication protocol (hereafter referred to as simply 2PAP) illustrated in figure 1.

$A \Longrightarrow$ B	A, N_{ab}
$B \Longrightarrow$ A	$AUTH_{K_{ab}}(N_{ab}, N_{ba}, B), N_{ba}$
$A \Longrightarrow$ B	$ACK_{K_{ab}}(N_{ab}, N_{ba}, A)$

Figure 1: Secure Two-Party Authentication Protocol (2PAP)

$AUTH_{K_{ab}}$ in 2PAP denotes an authentication expression based on the shared key K_{ab}. It is always generated by the responding party (B in our case). This expression is computed over three inputs: two nonces generated by each party and the name of the message originator. One example of $AUTH$ is:[2]

$$E(B \bigoplus E(N_{ba} \bigoplus E(N_{ab}))) \text{ or, equivalently, } MAC(N_{ab}, N_{ba}, B)$$

Similarly, $ACK_{K_{ab}}$ is the authentication expression that the initiating party sends in order to complete two-way authentication. It is computed over the same inputs except for the message originator's name (A in this case.) An example of an ACK is:

$$E(N_{ba} \bigoplus E(N_{ab})) \text{ or, equivalently, } MAC(N_{ab}, N_{ba})$$

The purpose of this paper is to pick up where previous efforts left off, i.e., to design a key distribution protocol (KDP) with properties inherited from the more basic two-party authentication protocol.

The design methodology adopted herein is the *reduction* of proposed protocols to 2PAP. In other words, if a protocol can be shown to be equivalent to 2PAP then we argue that it inherits the basic properties of 2PAP, namely, resistance to interleaving attacks. Of course, protocol goals other than authentication cannot be inherited from 2PAP. In particular, a secure two-party key distribution protocol must meet an additional requirement of non-disclosure of the key being distributed.

[2] All encryption is performed with K_{ab}.

We begin our design with some initial assumptions about KDPs (both two- and three-party):

1. **A key is always generated by one party.**
2. **A key being distributed must be a random, unpredictable quantity. If A and B engage in a KDP and A generates a key, then neither B nor any other party can predict this key.**
3. **Keys are never re-used, i.e., a key generated and distributed once, will not be used again.**

These assumptions lead to an important observation that a *freshly-generated* key is very similar to a nonce. In fact, the only difference between a two-party KDP (2PKDP) and 2PAP is that in 2PAP nonces are communicated in the clear. If, on the other hand, a new key was somehow communicated in secret, a two-party KDP can be designed to resemble 2PAP.

Consider, for example, the following protocol: The protocol in figure 2 depicts a secure 2PKDP. It is

$A \Longrightarrow$ B	A, N_{ab}
$B \Longrightarrow$ A	$\overbrace{AUTH_{K_{ab}}(N_{ab}, K_{ba}, B)}^{\dot{N}_{ba}},\ \overbrace{E_{K_{ab}}(\dot{N}_{ba})}^{MASK} \bigoplus K_{ba}$
$A \Longrightarrow$ B	$ACK_{K_{ab}}(N_{ab}, K_{ba}, A)$

Figure 2: Secure Two-Party Key Distribution Protocol (2PKDP)

assumed that A and B share a strong key, K_{ab} prior to protocol execution. The nonce generated by B, K_{ba}, is also the new key being distributed.

The protocol execution starts when the initiating party, A, generates a random challenge (nonce) to its peer B. B proceeds as follows:

1. Generates a random, unpredictable key K_{ba}.
2. Computes the $AUTH$ and $MASK$ expressions as shown in Figure 2.
3. Hides the new key by XOR-ing it with the $MASK$ and sends a message to A.

Upon receiving the message, A does the following:

1. Recomputes the masking expression:
   $MASK = E_{K_{ab}}(\dot{N}_{ba})$
2. Extract a candidate value:
   $\tilde{K}_{ba} = [E_{K_{ab}}(\dot{N}_{ba})] \bigoplus K_{ba} \bigoplus MASK$

3. Compute the 2PAP-like expression:
   $AUTH_{K_{ab}}(N_{ab}, \tilde{K}_{ba}, B)$ and compare with its counterpart in the message. A match results in successful, authenticated (and secret) distribution of the new key, K_{ba}.

Although it resembles in many respects the two-party authentication protocol (2PAP), the distinctive feature of 2PKDP is the *braided* structure in the second message. This structure is important to achieve both authentication (in the sense of 2PAP) and, at the same time, key distribution; K_{ba} being the new key. Moreover, the *braided* structure makes it possible to construct a minimal-length protocol message. This is because it is impossible to distribute an unpredictable random quantity (in secret) and simultaneously authenticate the quantity being distributed with a message of length less than two data units.

4 Informal Analysis

4.1 Authentication

The first step in demonstrating the security of the proposed 2PKDP, is to show equivalence between 2PKDP and 2PAP. In other words, we need to support our claim that the proposed protocol provides a means for two-party authentication with security equal to that of 2PAP. We begin by observing that the only difference between the two protocols is the so-called "nonce" field of the second message. In 2PAP, the nonce field is simply N_{ba} while in 2PKDP it is a more complex expression:

$$E_{K_{ab}}(\dot{N}_{ba}) \bigoplus K_{ba}$$

The purpose of this expression is to *conceal* the nonce, i.e., K_{ba} which is subsequently used as a key. (We consider the question of key confidentiality later.) In order to demonstrate the equivalence of respective nonce fields, three issues must be addressed:

1. Is K_{ba} a nonce?
   If K_{ba} is not a true nonce, then the equivalence does not hold. However, as noted earlier, nothing distinguishes a key from a nonce other than the fact that a key is kept secret. In particular, the rules and the procedures of generating keys and nonces are quite the same.

2. Is $\dot{N}_{ba}$ a nonce?
   Since K_{ba} is a true nonce and $\dot{N}_{ba}$ is a result of a strong one-way function computed over (among other variables) K_{ba}, we conclude that $\dot{N}_{ba}$ is also a nonce.

3. Is there a one-to-one relationship between $E_{K_{ab}}(\dot{N}_{ba}) \bigoplus K_{ba}$ and K_{ba}?
   Or, equivalently, does there exist a distinct value $\ddot{N}_{ba}$ such that:
   $$E_{K_{ab}}(\dot{N}_{ba}) \;=\; E_{K_{ab}}(\ddot{N}_{ba})$$
   Under the assumption that $E_{K_{ab}}$ is a strong encryption function such as DES [10], this is clearly impossible (since, otherwise, decryption would not be one-to-one.) If, on the other hand, E is a strong one-way hash function (e.g., MD5 [14]) which produces a fixed-size digest of its input, different input values can be hashed into an identical digest. The issue is then the difficulty of computing $\ddot{N}_{ba}$.

4.2 Key Distribution

We now consider key distribution which is the added feature of the proposed 2PKDP. Strong authentication is not sufficient for achieving secure key distribution thus the results of the previous section are not applicable in this context.

The first issue is the definition of secure key distribution. For the purpose of key distribution, a protocol is considered secure if:

- **Non-disclosure:** the key being distributed cannot be obtained by the adversary (of course, assuming no assistance from either A or B.)
- **Independence:** knowledge of one key cannot be used to obtain other keys. Here we assume that the key being distributed will not be subsequently used as an encryption key in another 2PKDP. (The new key may, for example, be used for computing integrity checks on data messages exchanged between A and B.)[3]
- **Integrity:** the key being distributed cannot be modified by the adversary.

4.2.1 Non-disclosure

Assuming no collusion with a legitimate party, disclosure of a key is possible only if the adversary is somehow able to obtain the $MASK$ value, i.e., $E_{K_{ab}}(\dot{N}_{ba})$. However, this is impossible since each $MASK$ is, in effect, a result of a one-way transformation of $\dot{N}_{ba}$. This implies that the adversary must either: 1) possess the encryption key, K_{ab}, or, 2) somehow be able to elicit the desired $MASK$ value from one of the legitimate parties. The former is assumed to be a non-issue while the latter deserves a closer look.

The adversary can try to obtain the $MASK$ value by *interrogating* B and pretending to be A. However, no matter what value the adversary "feeds" to B, B always generates a *fresh* key in message 2. (Refer back to figure 2.) It is the *freshness* of the new key that makes it impossible for the adversary to obtain any useful information. In fact, it is quite irrelevant what the adversary tries to "feed" to B since

$$\dot{N}_{ba} = AUTH_{K_{ab}}(N_{ab}, K_{ba}, B)$$

is strongly dependent on the freshly-generated K_{ba}.

Alternatively, the adversary can try to elicit information from A. This is significantly harder because A responds with message 3 only if message 2 is authentic. Therefore, the adversary must be able to generate expressions of the type in message 2 which is impossible since both cryptographic operations in message 2 are computed over A's random challenge, N_{ab}.

4.2.2 Independence

We now suppose that the adversary has somehow managed to discover a key K^i_{ba} from one of the runs of 2PKDP (refer to figure 3). Furthermore, he has done so without any explicit cooperation with either A or B. The most important additional piece of information that the adversary can thereby obtain is: $E_{K_{ab}}(\dot{N}^i_{ba}) = MASK^i$.

Now, since $MASK^i$ is the encryption of $\dot{N}^i_{ba}$ under the long-term key, K_{ab}, the adversary can discover a [plaintext,ciphertext] block-pair which he can later use for trying to break K_{ab}. This is, however, a danger commensurate with the presumed breach of security which resulted in the adversary's discovery of K^i_{ba}.

The remaining question is whether or not the adversary can use the newly-acquired information to attack the protocol by:

[3] Refreshing long-term keys (i.e., those used as encryption keys in 2PKDP) is another matter which requires different protocols; see, for example, [9].

$A \Longrightarrow$ B	A, N_{ab}^i
$B \Longrightarrow$ A	$\overbrace{AUTH_{K_{ab}}(N_{ab}^i, K_{ba}^i, B)}^{\dot{N}_{ba}^i},\ \overbrace{E_{K_{ab}}(\dot{N}_{ba}^i)}^{MASK^i} \oplus K_{ba}^i$
$A \Longrightarrow$ B	$ACK_{K_{ab}}(N_{ab}^i, K_{ba}^i, A)$

Figure 3: Compromised Run of 2PKDP

1. discovering a key ($K_{ba}^j\ i \neq j$) from any other protocol run (either past or future)
2. successfully distributing K_{ba}^i to A in another protocol run while posing as B

As stated earlier, $MASK$ is a result of a one-way function of $\dot{N}_{ba}$. (Recall also that $\dot{N}_{ba}$ is, itself, a one-way function of $N_{ab},\ K_{ba}\ and\ B$.) This essentially precludes any possibility of type 1 attacks.

An attack of type 2 requires that the adversary be able to construct a message of the form ($i \neq j$):

$$\dot{N}_{ba}^j \;=\; AUTH_{K_{ab}}(N_{ab}^j, K_{ba}^j, B),\;\; E_{K_{ab}}(\dot{N}_{ba}^j) \bigoplus K_{ba}^j$$

where $K_{ba}^j = K_{ba}^i$.

This, in turn, is possible only if the adversary is able to produce a fresh value of $\dot{N}_{ba}^j$ which includes the computation of the $AUTH$ expression over a fresh nonce N_{ab}^j. The adversary cannot obtain $\dot{N}_{ba}^j$ via *oracle* attacks since the party he chooses to *interrogate* will reply with a message implicitly re-freshed with its own nonce.

4.2.3 Integrity

The integrity of the key being distributed is *not* the foremost security feature of a KDP. A pure key distribution protocol need not concern itself with the integrity of the key as long as the key cannot be modified to a *particular* value selected by the adversary. For example, a family of KDPs in [4] are vulnerable to key modification attacks since the key itself is not authenticated but only distributed. Nonetheless, this is not considered to be a protocol "bug" because the adversary cannot change the key to a known value; he can only *offset* a key by a known value.[4]

On the other hand, attacks of this sort are not feasible with the present 2PKDP since key modification requires simultaneous modification of the authentication expression: $AUTH_{K_{ab}}(N_{ab}, K_{ba}, B)$.

[4] The term *offset* is used to indicate that the adversary is able to "fool" the parties to accept $K_{ba} \bigoplus X$ instead of intended K_{ba} (where X is a value the the adversary picks.)

5 More Detailed Analysis

Having presented some intuitive arguments about the security of 2PKDP in the previous section, we now attempt to capture the security of the proposed protocol in a more formal manner.

Assuming that two parties A and B engage in 2PKDP periodically, we can number the protocol runs sequentially from 1 to n. Furthermore, we assume that the same pairwise key K_{ab} is used in all protocol runs, i.e., K_{ab} is a fairly long-term key. In order to demonstrate the security of 2PKDP we need to give the potential adversary the widest possible latitude which essentially implies that the adversary knows *everything* except:

- The long-term key K_{ab}
- The details of the nonce-generating function, i.e., the adversary cannot compute (or predict) the nonces that will be used in the future protocol runs.

Supposing that $(n-1)$ protocol runs have transpired between A and B, the adversary can be presumed to *know* the following:

- $AUTH^i$, $\forall i, 0 < i < n$
- ACK^i, $\forall i, 0 < i < n$
- $MASK^i$ and K^i, $\forall i, 0 < i < n$

The first two items the adversary can obtain by simply eavesdropping on all communication between A and B. The last two, $MASK^i$ and K^i can not be obtained directly but we assume that the adversary has somehow discovered these values.

Three attack scenarios can be identified:

1. Attack on A:
   Adversary intercepts an initial message from A to B and manages to concoct a plausible response (message 2).
2. Attack on B:
   Adversary initiates a protocol run with B by supplying a challenge of his own choosing, receives a valid reply (message 2) and terminates the protocol by constructing a valid acknowledgement (message 3).
3. Key Disclosure Attack:
   Adversary either: i) eavesdrops on the on-going 2PKDP run and discovers the new key K^n, or ii) initiates a protocol run with B and, upon B's reply in message 2, discovers K_n newly-generated by B.

In the remainder of this section we consider each of the three attacks and discuss what it takes for them to succeed.

5.1 Attack on A

This type of an attack can succeed only if one of the following conditions holds:

1. $AUTH_n \;=\; AUTH_i$ where $0 < i < n$

2. $AUTH_n \;=\; ACK_i$ where $0 < i < n$

3. $AUTH_n \;=\; MASK_i$ where $0 < i < n$

Condition 1 means that $MAC(N_{ab}^n, K^n, B) = MAC(N_{ab}^i, K^i, B)$. By *unrolling* the MAC, we obtain:

$$E(B \oplus E(K_n \oplus E(N_{ab}^n))) = E(B \oplus E(K_i \oplus E(N_{ab}^i)))$$

which is equivalent to :

$$K_n \oplus E(N_{ab}^n) = K_i \oplus E(N_{ab}^i)$$

We have assumed that the adversary knows K_i; we can even go as far as allowing him to know $E(N_{ab}^i)$ for all previous protocol runs. However, since N_{ab}^n is a fresh hereto unused challenge generated by A, the adversary cannot know $E(N_{ab}^n)$. But, the adversary needs to know $E(N_{ab}^n)$ in order to set

$$K_n = E(N_{ab}^n) \oplus K_i \oplus E(N_{ab}^i)$$

and subsequently compute $MASK_n \oplus K_n = MASK_i \oplus K_n$. Thus, this attack is infeasible.

Condition 2 means that $MAC(N_{ab}^n, K^n, B) = MAC(N_{ab}^i, K^i)$. This implies:

$$B \oplus E(K_n \oplus E(N_{ab}^n)) = K_i \oplus E(N_{ab}^i)$$

which is equivalent to:

$$E(K_n \oplus E(N_{ab}^n)) = B \oplus K_i \oplus E(N_{ab}^i)$$

Again, to construct a plausible message, the adversary needs to set

$$K_n = E(N_{ab}^n) \oplus E^{-1}(B \oplus K_i \oplus E(N_{ab}^i))$$

which is impossible without knowing $E(N_{ab}^n)$.

Condition 3 means that $MAC(N_{ab}^n, K^n, B) = E(MAC(N_{ab}^i, K^i, B))$, or, equivalently:

$$B \oplus E(K_n \oplus E(N_{ab}^n)) = E(B \oplus E(K_i \oplus E(N_{ab}^i))$$

Again, this leads the adversary towards a *dead-end* since

$$K_n = E(N_{ab}^n) \oplus E^{-1}(B \oplus E(B \oplus E(K_i \oplus E(N_{ab}^i))$$

5.2 Attack on B

Under our assumptions this attack can succeed only if one of the following conditions holds:

1. $ACK_n \;=\; ACK_i$ where $0 < i < n$

2. $ACK_n \;=\; AUTH_i$ where $0 < i \leq n$

3. $ACK_n \;=\; MASK_i$ where $0 < i \leq n$

Condition 1 means that $MAC(N_{ab}^n, K^n, B) = MAC(N_{ab}^i, K^i)$, i.e.,

$$K_n \oplus E(N_{ab}^n) = K_i \oplus E(N_{ab}^i)$$

Here, since it is the adversary who initially selects N_{ab}^n it may be assumed that he also knows $E(N_{ab}^n)$. However, K_n is now fully under control of the legitimate party B; hence, the probability of:

$$K_n = E(N_{ab}^n) \oplus K_i \oplus E(N_{ab}^i)(\forall i < n)$$

is negligible.

Condition 2 means that $MAC(N_{ab}^n, K^n) = MAC(N_{ab}^i, K^i, B)$, i.e.,

$$K_n \oplus E(N_{ab}^n) = B \oplus E(K_i \oplus E(N_{ab}^i))$$

Similarly, due to the inherent "goodness" of K_n, we can ignore the chances of:

$$K_n = E(N_{ab}^n) \oplus B \oplus E(K_i \oplus E(N_{ab}^i))$$

Condition 3 is very similar to Condition 2 and terminates with an equation:

$$K_n = E(N_{ab}^n) \oplus E(B \oplus E(K_i \oplus E(N_{ab}^i)))$$

5.3 Key Disclosure

While the two previous attacks essentially re-stated the conditions guarding the security of 2PAP described in [2], the key disclosure attack is unique to 2PKDP. This type of attack entails the adversary discovering the new key K_n, which can only be accomplished by first discovering the mask protecting the key, $MASK_n$. The latter can be discovered only if one of the following conditions holds:

1. $MASK_n \ = \ ACK_i$ where $0 < i < n$
   This can occur only if Condition 3 in Section 5.2 holds.
2. $MASK_n \ = \ AUTH_i$ where $0 < i \leq n$
   This can occur only if Condition 3 in Section 5.1 holds.
3. $MASK_n \ = \ MASK_i$ where $0 < i < n$
   This can occur if:
   $E(MAC(N_{ab}^n, K^n, B)) = E(MAC(N_{ab}^i, K^i, B))$ or
   $MAC(N_{ab}^n, K^n, B) = MAC(N_{ab}^i, K^i, B)$ or
   $K_n \oplus E(N_{ab}^n) = K_i \oplus E(N_{ab}^i)$
   But, as we argued in Condition 1 of Section 5.2, this is impossible.

All three conditions are similar in that their verification requires knowledge of K_n since $MASK_n = E(MAC(N_{ab}^n, K^n, B))$. Since K_n is a fresh (hereto unused) quantity, it follows that:

1. K_n is unknown
   $\downarrow$
2. $E(K_n)$ is not computable
   $\downarrow$
3. $E(K_n \oplus E(N_{ab}^n))$ is not computable (even if $E(N_{ab}^n)$ is)
   $\downarrow$
4. $E(B \oplus E(K_n \oplus E(N_{ab}^n)))$ is not computable
   $\downarrow$
5. $E(E(B \oplus E(K_n \oplus E(N_{ab}^n))))$ is not computable

Therefore, key disclosure is not possible.

6 Protocol Minimality

As mentioned in Section 1, our design goals include protocol simplicity and compactness in addition to security. Ideally, a simple and compact protocol would also be *minimal*.

Protocol minimality can be expressed in several dimensions:

- Number of Messages
- Size of each individual message
- Number of cryptographic operations

6.1 Number of Messages

With respect to the number of messages, 2PKDP is a minimal protocol. This is clearly evident from the protocol's challenge-based structure. It is certainly possible to design a timestamp-based authentication protocol that achieves the same task in two messages (commensurate with some well-known security consequences.) However, challenge-based protocols require at least three messages.

6.2 Message Sizes

We claim that the message sizes in 2PKDP are also minimal. To show this, we postulate that any authenticated key exchange protocol must involve the communication of the following information in a non-overlapping fashion:

- A challenge from A to B (N_{ab})
- A challenge from B to A (N_{ba})
- An authentication token from B to A ($AUTH$)
- An authentication token from A to B (ACK)

Thus, a minimum of four distinct *fields* (or quantities) must be communicated. In addition, the protocol must also communicate from B to A the new key. However, since we couple the new key with B's challenge N_{ba}, the protocol still makes do with four fields.

Since it is not possible to reduce the size of either message 1 or message 3 (both messages containing only a single data block), the issue of minimality concerns only the size of message 2. Message 2, being the only message emitted by B, has to carry both the authentication token and the challenge/key combination. Because both quantities must be random and unpredictable we can conclude that it is *impossible* to reduce the size of message 2 without compromising one of the key properties of the protocol.

In general, key distribution is attainable with only a single field in message 2. However, the underlying protocol **cannot** simultaneously guarantee the integrity of the new key. In other words, key distribution is possible while *authenticated* key distribution is not.

6.3 Cryptographic Operations

Minimal number of cryptographic operations is a more subtle issue. Since key distribution must be authenticated, two distinct authentication tokens are required: one to provide key integrity and one to confirm the receipt of the key. Assuming that the underlying authentication protocol (2PAP) is already minimal, 2PKDP involves only one additional cryptographic operation on each side: the computation of $E_{K_{ab}}(AUTH)$ by B, and the corresponding computation by A.

If this additional operation can be eliminated, 2PKDP cannot be considered minimal. As mentioned before, message 2 must carry two expressions: a key distribution token (which hides the key) and an authentication (or key integrity) token $AUTH$. The former *distributes* the new key while the latter authenticates the sender (B) and integrity of the new key. Clearly, both expressions cannot be the result of the same set of cryptographic operations. At the same time, both expressions must be cryptographically-based. The authentication token is meaningless without strong cryptographic features and the same is true of the key distribution token. In other words, reduction in the number of cryptographic operations leads either to compromised authentication, or compromised key distribution (key disclosure).

To illustrate the last point, we consider two protocols similar to 2PKDP, where the second flows are:

Example A (2nd flow): $B \Longrightarrow$ A $\quad \overbrace{AUTH_{K_{ab}}(N_{ab}, K_{ba}, B)}^{\dot{N}_{ba}},\ \overbrace{K_{ab}}^{MASK} \bigoplus K_{ba}$

Example B (2nd flow): $B \Longrightarrow$ A $\quad \overbrace{AUTH_{K_{ab}}(N_{ab}, K_{ba}, B)}^{\dot{N}_{ba}},\ \overbrace{MAC(N_a)}^{MASK} \bigoplus K_{ba}$

In the first example, the $MASK$ expression requires no extra cryptographic computations. However, discovery of only one session key, K_{ba}, immediately compromises the long-term secret K_{ab} (i.e., the non-interference property is violated.) Alternatively, we can consider the case where the $MASK$ expression is set to $MAC(N_a) = E(N_a)$. This is an attractive solution since N_a is a nonce and is, hence, never re-used. Also, $MAC(N_a)$ costs "nothing" to compute since it is an intermediate value in the computation of $AUTH$ expression. A $MASK$ value discovered by an intruder would not, in any way, compromise other sessions. On the other hand, the $MASK$ is not dependent on B's challenge. This is undesirable because an intruder who discovers just one plaintext-ciphertext pair $[X, E(X)]$ can continue discovering session keys by pretending to be A and always starting the protocol with X.

7 Three-party Key Distribution (3PKDP)

Having completed the demonstration of a secure 2PKDP we now turn to the more complicated issue of secure 3PKDP. The objective of 3PKDP is to distribute a fresh key to two parties who share no prior secrets. It is assumed that there exists a mutually-trusted third party typically referred to as the Authentication Server (we use the symbol S to denote it hereafter.) A secure 3PKDP must satisfy much the same conditions as a secure 2PKDP with one added requirement:

- Neither party must be able to alter the key being distributed.

More generally, we assume that one of the legitimate parties (A or B) may exhibit arbitrary *byzantine* behavior. Of course, not all types of such threats can be addressed within the protocol. For example, unauthorized disclosure of a session key (by A or B) cannot be prevented. However, a secure 3PKDP must be defended against attacks by a legitimate party. In fact, a legitimate party exhibiting byzantine behavior can be viewed as a *special case* of an adversary – one armed with additional knowledge.

7.1 The Protocol

We obtain a secure 3PKDP by simply combining two instances (runs) of 2PKDP and one instance of 2PAP. Two principals A and B each engage in a 2PKDP with the Authentication Server S. A and B are each assumed to share a long-term secret key with S, i.e., K_{as} and K_{bs}. After the distribution of the new key K_{ba}, A and B engage in a 2PAP in order to confirm each other's knowledge of K_{ba}. The resulting protocol is illustrated in Figure 4.

$$
\begin{array}{ll}
A \Longrightarrow \mathrm{S} & A, B, N_{as} \\
S \Longrightarrow \mathrm{A} & \overbrace{AUTH_{K_{as}}(N_{as}, K_{ba}, B)}^{\dot{N}_{sa}},\ \overbrace{E_{K_{as}}(\dot{N}_{sa})}^{MASK_a} \bigoplus K_{ba} \\
A \Longrightarrow \mathrm{S} & ACK_{K_{as}}(N_{as}, K_{ba}, A) \\
\hline
B \Longrightarrow \mathrm{S} & B, A, N_{bs} \\
S \Longrightarrow \mathrm{B} & \overbrace{AUTH_{K_{bs}}(N_{bs}, K_{ba}, A)}^{\dot{N}_{sb}},\ \overbrace{E_{K_{bs}}(\dot{N}_{sb})}^{MASK_b} \bigoplus K_{ba} \\
B \Longrightarrow \mathrm{S} & ACK_{K_{bs}}(N_{bs}, K_{ba}, B) \\
\hline
A \Longrightarrow \mathrm{B} & A, N_{ab} \\
B \Longrightarrow \mathrm{A} & AUTH_{K_{ba}}(N_{ab}, N_{ba}, B), N_{ba} \\
A \Longrightarrow \mathrm{B} & ACK_{K_{ba}}(N_{ab}, N_{ba}, A)
\end{array}
$$

Figure 4: Secure Three-Party Key Distribution Protocol (3PKDP)

7.2 Authentication

As with 2PKDP, the first issue we must address is the strength of 3PKDP as an *authentication protocol.* In other words, does the protocol achieve pairwise authentication of (A, S), (B, S) and, finally, (A, B).

Starting from the end of the protocol, it can be easily seen that, assuming secure distribution of K_{ba}, the authentication protocol between A and B is as secure as the underlying 2PAP (in fact, it is 2PAP.)

Viewed in isolation, 2PKDP between A and S is secure and so is 2PKDP between B and S. However, one important difference between 3PKDP and two unrelated runs of 2PKDP is that the nonce generated by S is the same in both runs of 2PKDP (i.e., K_{ba}). This is not a problem as long as it can be shown that the respective $AUTH$ expressions remain uniquely "tied" to the particular run of 3PKDP. We address this issue by explicitly including the name of the peer party in the calculation of every $AUTH$ expression, i.e., B is included in $AUTH_{K_{as}}$ and A – in $AUTH_{K_{bs}}$. While this argument is not particularly crisp, we note that one simple method for complete conformance with 2PKDP is to use $(K_{ba} \bigoplus A)$ instead of K_{ba} in $AUTH_{K_{as}}$ and, similarly, $(K_{ba} \bigoplus B)$ instead of K_{ba} in $AUTH_{K_{bs}}$. Then, we have the following results:

$$A \overset{auth}{\longleftrightarrow} S$$
$$B \overset{auth}{\longleftrightarrow} S$$

and, provided K_{ba} is distributed securely:

$$A \overset{auth}{\longleftrightarrow} B$$

7.3 Key Distribution

The *non-disclosure* and *independence* properties in 3PKDP are inherited directly from 2PKDP (as discussed in Section 3. The *integrity* of the new key is a different matter. The side-effect of the same key being distributed to both parties, is that an "insider" adversary, say B, can obtain the expression used to mask the key in the message from S to A.

$$MASK_a = E_{K_{as}}(\dot{N}_{sa}) = E_{K_{as}}(\ AUTH_{K_{as}}(N_{as}, K_{ba}, B)\ )$$

At the first glance, it appears that knowledge of this expression allows B to modify K_{ba} to any selected $\hat{K}_{ab}$ with impunity. Nonetheless, the $AUTH_{K_{as}}$ expression (enclosed in the same message) precludes this attack since B cannot alter $AUTH_{K_{as}}(N_{as}, K_{ba}, B)$ to be $AUTH_{K_{as}}(N_{as}, \hat{K}_{ab}, B)$.

On a related note, one advantage that B does gain as a result of 3PKDP is the knowledge of the cleartext-ciphertext pair of the form: $[\dot{N}_{sa}, E_{K_{as}}(\dot{N}_{sa})]$. This poses a threat insofar as B being able to discover K_{as} by means of a brute-force attack. Unfortunately, such dangers are unavoidable in three-party key distribution.

7.4 Protocol Optimized

The protocol in figure 4 was obtained by a straight-forward combination of 2PAP and 2PKDP. It requires a total of nine messages and three bi-directional communication flows ($A-B$, $A-S$, $B-S$). In order to reduce the number of messages and optimize the protocol some of the messages can be interleaved. Moreover, it is not necessary for both A and B to communicate with S directly. A more realistic scenario would be such that all communication with S is either through A or B. Figure 5 illustrates a *compacted* version of 3PKDP. The resultant protocol is minimal in the number of messages and in the number of "encryption" operations.

8 Summary

In conclusion, this paper addressed the issue of secure, authenticated key distribution in a network environment. Results of previous efforts in secure two-party authentication were used to construct secure and compact protocols for both two- and three-party key distribution. The protocols take advantage of a novel *braiding* technique which allows for key distribution coincident with authentication within minimal message space. The model protocols presented here can be subsequently used to construct (by message rearrangement) an entire family of three-party and inter-domain secure key distribution protocols.

$A \Longrightarrow$ B	A, B, N_{as}, N_{ab}
$B \Longrightarrow$ S	A, B, N_{as}, N_{bs}
$S \Longrightarrow$ B	$\overbrace{AUTH_{K_{as}}(N_{as}, K_{ba}, B)}^{\dot{N}_{sa}}, \overbrace{E_{K_{as}}(\dot{N}_{sa})}^{MASK_a} \bigoplus K_{ba}, \overbrace{AUTH_{K_{bs}}(N_{bs}, K_{ba}, A)}^{\dot{N}_{sb}}, \overbrace{E_{K_{bs}}(\dot{N}_{sb})}^{MASK_b} \bigoplus K_{ba}$
$B \Longrightarrow$ A	$AUTH_{K_{as}}(N_{as}, K_{ba}, B),\ E_{K_{as}}(\dot{N}_{sa}) \bigoplus K_{ba},\ AUTH_{K_{ba}}(N_{ab}, N_{ba}, B), N_{ba}$
$A \Longrightarrow$ B	$ACK_{K_{ba}}(N_{ab}, N_{ba}, A)$

Figure 5: A Compact Version of 3PKDP

References

[1] S.M. Bellovin, M. Merritt, *Limitations of the Kerberos Authentication System*, ACM SIGCOMM Computer Communication Review, October 1990.

[2] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, M. Yung, *Systematic Design of Two-Party Authentication Protocols*, Proceedings of Crypto'91, August 1991.

[3] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, M. Yung, *Systematic Design of a Family of Attack-Resistant Authentication Protocols*, IEEE JSAC Special Issue on Secure Communications, June 1993.

[4] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, M. Yung, *A Modular Family of Secure Protocols for Authentication and Key Distribution* Draft, in submission to IEEE/ACM Transactions on Networking, August 1992.

[5] M. Burrows, M. Abadi, and R. Needham, *A Logic of Authentication*, **Proceedings of ACM Symposium on Operating System Principles**, December 1989.

[6] D. Denning and G. Sacco, *Timestamps in Key Distribution Systems*, **Communications of the ACM**, August 1981.

[7] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, November 1976.

[8] L. Gong, R. Needham and R. Yahalom, *GNY logic fill in*, Proceedings of IEEE Symposium on Security and Privacy, May 1990.

[9] R. Hauser, P. Janson, R. Molva, G. Tsudik and E. Van Herreweghen) *Robust and Secure Password and Key Change Method*, in submission to 1994 IEEE Symposium on Security and Privacy, October 1994.

[10] National Bureau of Standards, *Federal Information Processing Standards*, National Bureau of Standards, Publication 46, 1977.

[11] T. Lomas, L. Gong, J. Saltzer, R. Needham, *Reducing Risks from Poorly Chosen Keys*, Proceedings of ACM Symposium on Operating System Principles, August 1989.

[12] R. Molva, G. Tsudik, E. Van Herreweghen, S. Zatti, KryptoKnight *Authentication and Key Distribution Service*, Proceedings of ESORICS 92, October 1992; Toulouse, France.

[13] R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, December 1978.

[14] R. Rivest, *The MD5 Message Digest Algorithm*, Internet DRAFT, July 1991.

[15] J. Steiner, *The Kerberos Network Authentication Service Overview*, MIT Project Athena RFC, Draft 1, April 1989.

[16] J. Steiner, C. Neuman, J. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Proceedings of USENIX Winter Conference, February 1988.

[17] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, February 1978.

[18] G. Tsudik, *Message Authentication with One-Way Hash Functions*, Proceedings of IEEE INFOCOM 1992. May 1992.