

# Authenticated Multi-Party Key Agreement

Mike Just<sup>\*‡</sup>

Serge Vaudenay<sup>†§</sup>

## Abstract

We examine multi-party key agreement protocols that provide (i) key authentication, (ii) key confirmation and (iii) forward secrecy. Several minor (repairable) attacks are presented against previous two-party key agreement schemes and a model for key agreement is presented that provably provides the properties listed above.

A generalization of the Burmester-Desmedt model (Eurocrypt '94) for multi-party key agreement is given, allowing a transformation of any two-party key agreement scheme into a multi-party scheme. Multi-party schemes (based on the general model and two specific 2-party schemes) are presented that reduce the number of rounds required for key computation compared to the specific Burmester-Desmedt scheme. It is also shown how the specific Burmester-Desmedt scheme fails to provide key authentication.

**1991 AMS Classification:** 94A60

**CR Categories:** D.4.6

**Key Words:** multi-party, key agreement, key authentication, key confirmation, forward secrecy.

**Carleton University, School of Computer Science:** SCS-TR-96-04

---

<sup>‡</sup>School of Computer Science, Carleton University, Ottawa, ON, Canada, K1S 5B6, e-mail: just@scs.carleton.ca

<sup>\*</sup>Research supported by NSERC graduate fellowship.

<sup>†</sup>Research completed while this author was visiting the School of Computer Science, Carleton University – partially supported by NSERC grant.

<sup>§</sup>Ecole Normale Supérieure-DMI, 45, rue d'Ulm, 75230 Paris Cedex 05, France, e-mail: Serge.Vaudenay@ens.fr

# 1 Introduction

Private-key cryptography is widely used in security networks. Though it assumes that parties who share the same secret key are both secure, and do not reveal their key, it is still more efficient than public-key cryptography for most applications. To allow several parties willing to communicate using private-key cryptography while avoiding any long-term common private keys, the parties need to first agree on the same *session key* following a *key establishment protocol*.

Key establishment protocols can be divided into two categories. A *key transfer* protocol is a key establishment protocol in which one party securely transfers a key to the other parties participating in the protocol. A *key agreement protocol* is a key establishment protocol in which the parties contribute information that jointly establishes a shared secret key. (See [20] for an overview.)

In the early origins of public-key cryptography, a two-party key agreement protocol due to Diffie and Hellman (DH) was proposed [10]. There have been many attempts to provide authentic key agreement based on DH [11, 14, 15, 16, 25] (There also exist many non-DH solutions to key establishment. See [5, 6, 3, 1, 23] for example.)

In a separate direction, several attempts have been made to extend DH to a multi-party protocol [13, 21, 22], the most efficient being the result of Burmester and Desmedt [8].

This paper deals with key agreement protocols based on DH that use public-key techniques. We do not require the aid of an on-line or trusted third party<sup>1</sup>. Users interact via an exchange of messages to obtain a common key.

Section 2 presents several important definitions and building blocks that will be used in the construction of our key agreement protocols. In Section 3 we introduce our two-party key agreement model, present two implementations of the model and examine their resistance to passive and active attacks. In Section 4 we discuss the multi-party model, several old and new implementations, and examine attacks against each.

## 1.1 Definitions and Notations

Let  $m$  be a prime and  $\alpha \in \mathbb{Z}_m^*$  an element with order  $q$  (an integer such that  $q|m-1$ ). All operations in this paper will take place in  $\mathbb{Z}_m$ , unless otherwise noted. We will be working in a network of  $n$  users,  $t$  of which participate in the key agreement protocol. Each user  $U$  has a long-term public key  $p_U = \alpha^{s_U}$  for a random secret-key  $s_U \in_R \mathbb{Z}_q^*$ . We use  $I_U$  to refer to information identifying user  $U$ , i.e. name. We assume that each user has a copy of every other public keys *a priori*, or equivalently that certification is used so that each public-key is identity-based. If this is not the case then  $I_U$  will also contain a certified copy of  $U$ 's public key. We denote by  $f$  a one-way hash function that takes arbitrarily sized input and produces a fixed size output, i.e. [18]. We denote by  $h_K$  a keyed one-way hash function, i.e. [2, 17, 19].

---

<sup>1</sup>We will require a trusted center for the purposes of creating public-key certificates for each user. However, this can be completed off-line, and the center is not required to maintain the secrecy of any information.

<sup>2</sup>For simplicity, we use the notation from [8] whereby if an element  $x$  is chosen randomly and independently from a set  $S$ , we say  $x \in_R S$ .

## 1.2 Summary of Results

We begin by examining several two-party key agreement protocols. Minor (reparable) attacks against these protocols are presented. It is also shown how the property of (perfect) *forward secrecy* as defined in [11] (as well as Section 2) has been misinterpreted (as well as absent) in some of these protocols. Subsequently, we combine techniques from several protocols in the literature, to obtain a model that provably provides for (i) key authentication, (ii) key confirmation and (iii) forward secrecy (see Section 2 for definitions). We present two schemes that fit this model and examine their resistance to passive and active attacks.

We extend our two-party results by generalizing the specific scheme of Burmester and Desmedt [8] to obtain a multi-party key agreement model. Using our specific two-party schemes and this model, we are able to obtain multi-party schemes which reduce the amount of communication required (as compared to the scheme of [8]). It is also shown how the scheme of Burmester and Desmedt [8] fails to provide key authentication. Passive and active attacks against the multi-party schemes are also examined.

## 2 Fundamentals

In this paper, we build up from simple 1-pass key transfer (KT) protocols to multiple pass key agreement (KA) protocols. Where a KT protocol involves contributions from only 1 user to the key, KA protocols involve mutual contributions to the final key. When a KA protocol involves more than 2 users, we refer to it as a multi-party key agreement (MPKA) protocol. If referring to properties that apply to both two-party and multi-party protocols, we simply refer to KA protocols.

**Definition 1** [11] *A key agreement protocol is successful if each of the parties accepts the identity of the other party, as well as the computed key.*

**Definition 2** *A key agreement protocol provides key authentication if only those parties specified to be engaging in the protocol, are able to compute the session key.*

Key authentication implies key confidentiality. For if only intended parties can compute the key, then unintended parties cannot compute the key.

**Definition 3** *A key agreement protocol provides key confirmation (direct authentication in [11]) if parties provide proof of possession of the session key.*

Key confirmation is usually achieved via encrypting or hashing a known quantity.

**Definition 4** *A key agreement protocol provides forward secrecy (perfect forward secrecy in [11]) if the loss of any long-term secret keying material does not allow the compromise of keys from previous sessions.*

(Since *perfect* usually makes reference to information theory, we avoid it here.) There are several meanings of *does not allow*. Here we use a practical ability to recover information. We note the compromise of long-term secret keys does not necessarily mean that they were obtained via an inversion of the long-term public key. Since users must store their secret keys for use in key computation, the secret keys may also be obtained through lack of

suitable physical security measures. This property may be attractive for the robustness of the security in most commercial applications where customers does not always protect their key sufficiently. As we shall see, this definition has been misinterpreted in the past.

Our protocols rely on the apparant difficulty of solving the well-known Diffie-Hellman (DH) problem [10], for appropriately chosen parameters [24]. The traditional DH problem is stated as follows. Given a generator  $\alpha$  for a group  $\mathbb{Z}_m^*$  and inputs  $y = \alpha^x$  and  $y' = \alpha^{x'}$ , compute (we omit reference to  $m$  for simplicity)

$$\text{DH}(\alpha; y, y') = \alpha^{xx'}$$

Likewise, for long-term public parameters  $p_A = \alpha^{s_A}$  and  $p_B = \alpha^{s_B}$ , we have  $\text{DH}(\alpha; y, p_A) = \alpha^{s_A x}$  and  $\text{DH}(\alpha; p_A, p_B) = \alpha^{s_A s_B}$ .<sup>3</sup>

The DH problem is the basis for the following two 1-pass KA (i.e. Key Transfer) protocols. The distinguishing feature for each (both here and in their use in more advanced protocols) is that Protocol IB requires *a priori* knowledge of public keys (or else an extra message pass) whereas Protocol IA does not. The first is based on ElGamal encryption [12]. It also can be considered to be a DH protocol with one fixed parameter.

### Protocol IA

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends  $\{y := \alpha^x, I := I_A\}$  to  $B$ .  $A$  computes the key  $K := p_B^x (= \alpha^{x s_B})$ .
2.  $B$  receives  $y$  and computes the key  $K := y^{s_B} (= \alpha^{x s_B})$ . □

Notice that the key computation for this protocol is  $\text{DH}(\alpha; y, p_B) = \alpha^{s_B x}$ . This second protocol is a simple variation on the first, with key computation  $\text{DH}(p_B; p_B^x, \alpha)$ .

### Protocol IB

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends  $\{y := p_B^x, I := I_A\}$  to  $B$ .  $A$  computes the key  $K := \alpha^x$ .
2.  $B$  receives  $y$  and computes the key  $K := y^{s_B^{-1}} (= \alpha^x)$ . □

## 3 Authenticated Key Agreement

In this section, we extend the key transfer Protocols IA and IB to provide for authenticated key agreement. The desirable properties being (i) key authentication, (ii) key confirmation and (iii) forward secrecy (see Section 2). (The provision of these properties are examined more closely in Section 3.3.) To achieve these goals, we incorporate techniques from [14, 11, 16, 15, 25].

---

<sup>3</sup>This is an abuse of notation. Since  $p_A$  and  $p_B$  are fixed parameters for each protocol run, their inclusion in the calculation should be distinguished from  $y$  and  $y'$  which are randomly chosen for each run. A result of this fact is that an ability to compute  $\text{DH}(\alpha; y, y')$  implies an ability to compute  $\text{DH}(\alpha; p_A, p_B)$  yet the reverse implication is not true. This point is not elaborated on here or in Sections 3.3 and 4.1 where equivalences to these terms are shown. This will be expanded on in the full version of the paper.

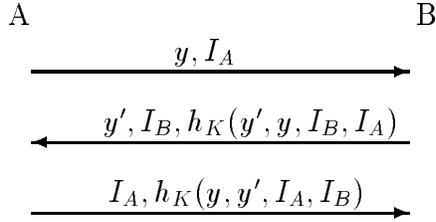


Figure 1: Generic Authenticated Key Agreement

Our model consists of a 2-pass (optionally 3-pass) authentic key agreement protocol.<sup>4</sup> The generic protocol is presented in Figure 3. The values  $y$  and  $y'$  are simply random tokens generated by each user (that will be used in the key computation). Key authentication (essentially key confidentiality) is obtained by the fact that the key computation is based on variations of the Diffie-Hellman problem. Certain variations also allow for the provision of forward secrecy (i.e. the variation in Protocol A0 does not provide forward secrecy, while the variation in Protocol IIA does). Unilateral key confirmation (mutual with 3-pass) is provided by the inclusion of hashed known value by  $B$  in step 2 (mutual with hashed known value by  $A$  in step 3).

Our model differs from [11] in that the embedding of the public keys directly in the key computation, removes the need for the signing of exchanged tokens. The use of a one-way keyed hash replaces the use of an encryption function (which is unnecessary since there is no decryption taking place – and relaxes the possibility of export restrictions).

### 3.1 Protocols Based on IA

Consider the following two party key agreement protocol between users  $A$  and  $B$ , given in [16]. (Similar protocols are given in [15, 25].)

#### Protocol A0 ([16])

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends  $\{y := \alpha^x, I := I_A\}$  to  $B$
2.  $B$  selects  $x' \in_R \mathbb{Z}_q$ , computes  $y' := \alpha^{x'}$  and  $K := p^{x'} y^{s_B}$  ( $= \alpha^{s_A x' + s_B x}$ ) where  $p$  is the public key extracted from  $I$ .  $B$  sends  $\{y', I' := I_B, z' := f(y', K, I, I')\}$  to  $A$
3.  $A$  computes  $K = (y')^{s_A} (p')^x$  where  $p'$  is extracted from  $I'$ , and continues only if  $z' = f(y', K, I, I')$ .  $\square$

Two minor attacks can be mounted against Protocol A0 in the absence of a proper implementation.

- **$E$  impersonates  $B$  to  $A$ .** In place of step 2,  $E$  sends  $\{\mathbf{y}' = \mathbf{0}, I' := I_B, z' := f(y', K, I, I')\}$  to  $A$ .  $A$  believes that  $B$  is the only party that is able to compute  $K$ . However, since  $K = 0$ , the key is easily obtained (by  $E$  or anyone else), hence a lack of key authentication.

---

<sup>4</sup>3 passes are required for the proof of Theorem 3 and for the multi-party schemes of Section 4.

- **$E$  impersonates  $A$  to  $A$ .** This more subtle (and less obvious) attack involves  $A$ 's initiation of a protocol with *some* user with a valid public key. Consider an Automatic Teller Machine ( $A$ ) that initiates a key agreement protocol with a customer ( $E$  – impersonating  $A$ ) and continues the transaction, so long as the customer is able to compute the corresponding key (i.e. if  $A$  believes the protocol provides key authentication). The following attack illustrates how an attacker ( $E$ ) can use the public key of  $A$  itself, to carry out an impersonation attack.

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends  $\{y := \alpha^x, I := I_A\}$  to  $E$ .
2.  $E$  selects  $\tilde{x} \in_R \mathbb{Z}_q$  and simulates the protocol as if  $x' = \tilde{x} - x$ . i.e.  $E$  computes  $y' := \alpha^{\tilde{x}}/y$ , as well as  $K := p_A^{\tilde{x}}$  and sends  $\{y', I' := I_A, z' := f(y', K, I, I')\}$ .
3. Using the received identifier  $I'$ ,  $A$  will use the public key  $p_A = \alpha^{s_A}$ , and compute the key  $K = (y')^{s_A} p_A^x = \alpha^{s_A \tilde{x} - s_A x} \alpha^{s_A x} = \alpha^{s_A \tilde{x}}$ .  $A$  also checks that  $z' = f(y', K, I, I')$ .

The result of the second attack being that  $A$  accepts  $E$  (impersonating  $A$ ) as a valid user, since  $E$  has successfully completed the protocol (i.e. is able to compute  $K$ ). Obvious solutions to both attacks are to implement the protocol so that trivial messages such as  $y$  (or  $y'$ ) = 0 or 1 are disallowed and that  $I \neq I'$ .

Also of note for Protocol A0 is the fact that it *does not provide* for forward secrecy (as claimed in [16]). Although recovery of a single long-term secret key does not allow recovery of previous session keys, recovery of both long-term secret keys  $s_A$  and  $s_B$  allows the computation of  $K = y^{s_B} (y')^{s_A}$  for all previous sessions involving  $A$  and  $B$ .

One way to solve this lack of forward secrecy is to alter Protocol A0, such that the key computation is now  $\alpha^{x x' + s_B s_A}$ . However, this presents other concerns which we do not elaborate on here. Instead, we present the following protocol which uses this idea to enhance Protocol A0 while incorporating the model described at the beginning of Section 3.

## Protocol IIA

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends to  $B$

$$\{y = \alpha^x, I := I_A\}.$$

2.  $B$  selects  $x' \in_R \mathbb{Z}_q$ , computes  $y' := \alpha^{x'}$  and  $K := y^{x' + s_B} p^{x'}$  ( $= \alpha^{x x' + x' s_A + x s_B}$ ) where  $p$  is the public key extracted from  $I$ .  $B$  sends to  $A$

$$\{y', I' := I_B, z' := h_K(y', y, I', I)\}.$$

3.  $A$  computes  $K = (y')^{x + s_A} (p')^x$  where  $p'$  is extracted from  $I'$ , and checks that  $z' = h_K(y', y, I', I)$ .
4. (optional)  $A$  computes and sends  $\{I, z := h_K(y, y', I, I')\}$  to  $B$ . □

Notice that knowledge of even both  $s_A$  and  $s_B$  does not allow computation of  $K$  by an eavesdropper. Passive and active attacks against this protocol are examined in Section 3.3.

## 3.2 Protocols Based on IB

The protocols in this section require *a priori* knowledge of public keys (or an extra message pass). Protocol B0 was given in [16] (an adaptation of a scheme from [15]).

### Protocol B0 ([16])

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends  $\{y := p_B^x (= \alpha^{s_B x}), I := I_A\}$  to  $B$
2.  $B$  selects  $x' \in_R \mathbb{Z}_q$ , computes  $y' := p^{x'} (= \alpha^{s_A x'})$  where  $p$  is extracted from  $I$  and  $K := y^{s_B^{-1}} \alpha^{x'} (= \alpha^{x+x'})$ .  $B$  sends  $\{y', I' := I_B, z' := f(y', K, I, I')\}$  to  $A$
3.  $A$  computes  $K = (y')^{s_A^{-1}} \alpha^x$  and checks that  $I' = I_B$  and  $z' = f(y', K, I, I')$ .  $\square$

Similar attacks can be mounted against Protocol B0 in the absence of a proper implementation.

- **$E$  impersonates  $B$  to  $A$ .** In place of step 2,  $E$  sends  $\{y' = \mathbf{1}, I' := I_B, z' := f(y', K, I, I')\}$  to  $A$ .  $A$  believes that  $B$  is the only party that is able to compute  $K$ . However, since  $K = y (= \alpha^x)$ , the key is easily obtained, hence a lack of key authentication.
- **$E$  impersonates  $A$  to  $A$ .** Similar to the attack against Protocol A0,  $E$  can claim he is  $A$ , pick a random  $\tilde{x}$  and simulate the protocol as if  $x' = \tilde{x} - x$ . The verification that  $E$  can compute the values to be sent and the session key is left to the reader.

As before, obvious solutions to this attack are to implement the protocol so that trivial messages such as  $y$  (or  $y'$ ) = 0 or 1 are disallowed and that  $I \neq I'$ .

As in Protocol A0, Protocol B0 *does not provide* for forward secrecy (as claimed in [16]). Recovery of both long-term secret keys  $s_A$  and  $s_B$  allows the computation of  $K = \alpha^{x+x'}$  for all previous sessions involving  $A$  and  $B$ . Protocol B0 prevents the aforementioned attack and provides forward secrecy.<sup>5</sup> Its resistance to passive and active attacks is examined in Section 3.3.

### Protocol IIB

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends to  $B$ 

$$\{y := p_B^x (= \alpha^{x s_B}), I := I_A\}.$$
2.  $B$  selects  $x' \in_R \mathbb{Z}_q$ , computes  $y' := p^{x'} (= \alpha^{x' s_A})$  where  $p$  is extracted from  $I$  and  $K := y^{s_B^{-1} x'} (= \alpha^{x x'})$ .  $B$  sends to  $A$ 

$$\{y', I' := I_B, z' := h_K(y', y, I', I)\}.$$
3.  $A$  computes  $K = (y')^{s_A^{-1} x}$ , and checks that  $z' = h_K(y', y, I', I)$ .
4. (*optional*)  $A$  computes and sends  $\{I, z := h_K(y, y', I, I')\}$  to  $B$ .  $\square$

---

<sup>5</sup>The key computation is identical to schemes from [16, 15] but the protocol follows our model from the beginning of Section 3.

### 3.3 Passive and Active Attacks

In this section, we analyze the resistance of Protocols IIA and IIB to passive and active attacks. We prove the security of each protocol by showing its equivalence to variations of the DH problem. Hence, throughout this section, we assume that Diffie-Hellman computations (as described in Section 2) are infeasible without the proper, corresponding secret information. Also, we assume that the hash  $h_K$  behaves like a random oracle, in the sense that its output cannot be distinguished from random output.

**Theorem 1** *Protocols IIA and IIB are at least as secure against passive attack as Protocols IA and IB.*

**Proof (Outline)** Given a DH oracle, one can easily solve for the session key in protocol IIA using only a passive attack. This is done by computing

$$K = DH(\alpha; y, y')DH(\alpha; y, p_B)DH(\alpha; y', p_A).$$

We can also show that an oracle for Protocol IIA can be used to compute the output of a DH oracle. Given  $y = \alpha^x$ ,  $p_A = \alpha^{s_A}$  and  $p_B = \alpha^{s_B}$ , we choose a random  $x'$  and compute  $y' = \alpha^{x'}$ . Inputting these values to the IIA oracle gives us  $K = \alpha^{xx'+xs_B+x's_A}$ . Given our knowledge of  $x'$ , we can compute  $\alpha^{xs_B}$  (i.e. what would have been output by a DH oracle). (Similar arguments show that Protocol IIB is at least as hard as Protocol IB.) ■

**Theorem 2** *Protocols IIA and IIB provide forward secrecy.*

**Proof (Outline)** Consider Protocol IIA. We need to show that recovery of all long-term secret keys does not allow recovery of previous session keys. Assume the opposite is true. Then given  $s_A$  and  $s_B$  corresponding to the respective long-term public keys  $p_A$  and  $p_B$  allows recovery of the key  $K = \alpha^{xx'+xs_B+x's_A}$ . From this, we are able to compute  $\alpha^{xx'}$  for random  $y$  and  $y'$ ; a contradiction to the assumption that DH computations are computationally infeasible. (Similar arguments are used for Protocol IIB.) ■

The following Theorem shows that Protocols IIA and IIB are provably secure against impersonation. (Note that for this theorem, we require that the optional pass has been completed by  $A$ . In other words,  $A$ 's knowledge of the key has become explicit as opposed to implicit.) We say that  $E$  successfully impersonates  $A$  (resp.  $B$ ) if  $E$  is able to successfully complete the key agreement protocol with  $B$  (resp.  $A$ ).

**Theorem 3** *Impersonating either  $A$  or  $B$  in Protocols IIA and IIB is equivalent to solving an instance of the Diffie-Hellman problem.*

**Proof (Outline)** In Protocol IIA, if  $E$  successfully impersonates  $A$ , then  $E$  must have been able to compute the resultant key  $K$ , necessary for computation of the keyed hash in the last step of the protocol. However, from Theorem 1 we know that  $E$  is unable to compute  $K$ . Likewise, if  $E$  successfully impersonates  $B$  then  $E$  must have been able to compute the resultant key  $K$ , necessary for computation of the keyed hash in the second step of the protocol. From Theorem 1 we know that this is not possible. (Similar arguments are used for Protocol IIB.) ■

## 4 Multi-Party Key Agreement

We propose here a generic construction of a multi-party key agreement protocol MP from a two-party key agreement protocol P.<sup>6</sup> We assume that all users  $u_1, u_2, \dots, u_t$  are arranged on a ring and we will consider indices of  $u_i$  to be taken between 1 to  $t$  modulo  $t$ .

1. Each pair  $(u_i, u_{i+1})$  processes protocol P (full protocol version with the optional pass) to obtain a session key  $K_i$ .
2. Each  $u_i$  computes and broadcasts  $W_i \equiv \frac{K_i}{K_{i-1}}$ .
3. Upon receiving the broadcasts from other users,  $u_i$  computes the key

$$\begin{aligned} K &\equiv K_{i-1} {}^t W_i {}^{t-1} W_{i+1} {}^{t-2} \dots W_{i-2} \\ &\equiv K_1 K_2 \dots K_t. \end{aligned}$$

For specific implementations of this model we use Protocols IIA and IIB from Section 3 to obtain the respective multi-party protocols MIIA and MIIB.

Notice that for Protocol MIIA,  $u_i$  sends the same token (i.e.  $y_i = y'_i$ ) to both  $u_{i+1}$  and  $u_{i-1}$ . However, because of the use of the public key in the token computation, for Protocol MIIB,  $y_i \neq y'_{i-1}$ , where  $y_i$  is sent to  $u_{i+1}$  and  $y'_i$  is sent to  $u_{i-1}$ .

### Burmeister/Desmedt Scheme

In this section we show that the scheme of Burmeister and Desmedt [8] does not provide key authentication. Their scheme is a specific case of the model describe above with DH as protocol P. The authentication scheme that is used is described in the context of the attack presented below. The attack involves applying a different kind of impersonation attack that was used on several KA protocols in [16]. We note that our protocols are not susceptible to this attack as we build upon the protocols from [16].

We illustrate the attack with two users  $A$  and  $B$ . (The attack also works with more than 2 users as  $E$  would simply position himself between any two users.) An adversary  $E$  will attack the protocol by convincing  $B$  that he shares a key with  $E$ . This is done by  $E$ 's ability to authenticate the token sent from  $A$  to  $B$  as belonging to  $E$ . At the end of this protocol,  $A$  believes (and in fact does) share  $K$  with  $B$ . However,  $B$  believes (incorrectly) that he shares  $K$  with  $E$  (even though  $E$  is unable to compute the key). Hence, key authentication is not provided as the person with whom  $B$  believes he is sharing the key (namely  $E$ ), is not able to actually compute the key (as only  $A$  and  $B$  can compute the key).

From [8], each user  $i$  has a public key pair  $(\beta_i, \gamma_i)$  where  $\beta_i = \alpha^{v_i}$  and  $\gamma_i = \alpha^{w_i}$ . This version assumes that users' public keys are *a priori* available, and that  $q$  is prime.

### Attack on MDH

1.  $A$  selects  $x \in_R \mathbb{Z}_q$  and sends  $\{y := \alpha^x, I := I_A\}$  to  $B$ . At this point,  $E$  intercepts the communication and impersonates  $B$  to  $A$ . This is possible since at this point, there is no authentication of  $B$  to  $A$ .

---

<sup>6</sup>The construction is a generalization of the scheme from [8].

2.  $A$  authenticates  $y$  to  $E$  with a zero-knowledge interactive proof of knowledge of the discrete log of  $\beta_A^y \gamma_A$  (namely  $v_A y + w_A$ ) using methods described in [8].
3.  $E$  sends  $y$  to  $B$ , and using his public key pair  $(\beta_E, \gamma_E)$ , authenticates  $y$  to  $B$ .
4.  $B$  now sends  $\{y', I := I_B\}$  to  $E$ .  $E$  simply forwards this message to  $A$  and allows  $B$  to authenticate  $y'$  to  $A$ . This is possible since at this point, there is no authentication of  $A$  to  $B$ .
5.  $A$  and  $B$  complete the protocol by broadcasting  $W_A$  and  $W_B$  respectively. □

After this attack, messages that  $A$  sends to  $B$  will be interpreted by  $B$  as coming from  $E$ . One can imagine an attack where  $B$  is a bank and  $A$  and  $E$  are customers.

The authentication between pairs of users in [8] requires 1 round for the DH key token exchange,  $k$  rounds for the authentication of the tokens (for a security parameter  $k$ ) and 1 round for the broadcast of the  $W_i$ 's. This gives us a total of  $k+2$  rounds. Protocols MIIA and MIIB require 3 rounds for the processing of protocols IIA or IIB (including authentication of tokens) and 1 round for the broadcast of the  $W_i$ 's. This gives us a total of 4 rounds. Therefore, if more than 2 rounds are used for the authentication of the tokens in the Burmester-Desmedt scheme, our schemes are more efficient in terms of the number of rounds. Potential attacks to Protocols MIIA and MIIB are discussed in the following sections.

## 4.1 Passive Attacks

In this section, we show that the multi-party model specifically implemented with Protocols IIA and IIB (to produce MIIA and MIIB) are provably secure against a passive attacker. This is done by illustrating their equivalence to the respective schemes from Section 3 (using the same techniques as given in [8]).

**Theorem 4** *Given an even, polynomial number  $t$  of randomly chosen users, Protocols MIIA and MIIB are as secure against passive attacks as Protocols IIA and IIB respectively.*

**Proof (Outline)** Given an oracle to solve the IIA protocol, it is easy to solve for the key  $K$  in the scheme MIIA. This is done by computing the key (using the equivalent DH computation) as

$$\prod_{i=1}^t DH(\alpha; y_i, y_{i+1}) DH(\alpha; y_{i+1}, p_i) DH(\alpha; y_i, p_{i+1}).$$

Since this procedure is bounded by  $t$ , only a polynomial number of calls are required to be made to the IIA oracle.

Now consider the converse problem of being given  $p_1 = p_B = \alpha^{s_B}$ ,  $y_1 = y' = \alpha^{x'}$ ,  $p_t = p_A = \alpha^{s_A}$  and  $y_t = y = \alpha^x$ , solving for the key  $\alpha^{x_t x_1 + x_t s_1 + x_1 s_t}$  (i.e.  $\alpha^{x x' + x s_B + x' s_A}$  from Protocol IIA) by using an oracle that solves for the MIIA key. We must first prepare the remaining input to the MIIA oracle. We first compute for  $i = 2, \dots, t-1$

$$\begin{aligned} p_i &= p_{i-2} \alpha^{b_i} \\ y_i &= y_{i-2} \alpha^{c_i} \end{aligned}$$

using random  $b_i$  and  $c_i$ . This “randomizes” the virtual users as if we had  $s_i = s_{i-2} + b_i$  and  $x_i = x_{i-2} + c_i$  providing a good distribution. For  $i = 1, \dots, t-2$ , we can now compute

$$W_i = \left( \frac{p_{i+1}y_{i+1}}{p_{i-1}y_{i-1}} \right)^{x_i} \left( \frac{y_{i+1}}{y_{i-1}} \right)^{s_i} = (\alpha^{b_{i+1}} \alpha^{c_{i+1}})^{x_i} (\alpha^{c_{i+1}})^{x_i} = y_i^{b_{i+1}} (y_i p_i)^{c_{i+1}}$$

Since  $t$  is even, we also have that

$$\begin{aligned} p_t &\equiv p_2 g^{-b_2} \equiv p_4 g^{-b_2-b_4} \equiv \dots \equiv p_{t-2} g^{-b_2-b_4-\dots-b_{t-2}} \\ p_1 &\equiv p_3 g^{-b_3} \equiv p_5 g^{-b_3-b_5} \equiv \dots \equiv p_{t-1} g^{-b_3-b_5-\dots-b_{t-1}} \\ y_t &\equiv y_2 g^{-c_2} \equiv y_4 g^{-c_2-c_4} \equiv \dots \equiv y_{t-2} g^{-c_2-c_4-\dots-c_{t-2}} \\ y_1 &\equiv y_3 g^{-c_3} \equiv y_5 g^{-c_3-c_5} \equiv \dots \equiv y_{t-1} g^{-c_3-c_5-\dots-c_{t-1}}, \end{aligned}$$

allowing us to compute

$$\begin{aligned} W_{t-1} &\equiv \left( \frac{p_t y_t}{p_{t-2} y_{t-2}} \right)^{x_{t-1}} \left( \frac{y_t}{y_{t-2}} \right)^{s_{t-1}} \equiv (y_{t-1})^{-b_3-b_5-\dots-b_{t-1}} (y_{t-1} p_{t-1})^{-c_3-c_5-\dots-c_{t-1}} \\ W_t &\equiv \left( \frac{p_1 y_1}{p_{t-1} y_{t-1}} \right)^{x_t} \left( \frac{y_1}{y_{t-1}} \right)^{s_t} \equiv (y_t)^{-b_3-b_5-\dots-b_{t-1}} (y_t p_t)^{-c_3-c_5-\dots-c_{t-1}}. \end{aligned}$$

Inputting all the  $y_i$ ,  $p_i$  and  $W_i$  to the MIIA oracle, produces the output  $K$ . We have  $K_i = \alpha^{x_{i-1} x_i + x_i s_{i-1} + x_{i-1} s_i} y_i^{b_{i+1}} (y_i p_i)^{c_{i+1}}$ . From  $K$  and the  $W_i$ , we can obtain any  $K_i$ . More specifically, for  $u_1$  we solve for  $K_1$ , from which we obtain  $\alpha^{x_t x_1 + x_t s_1 + x_1 s_t}$ .

(The actions for Protocol MIIB are similar.) ■

## 4.2 Information Revealed by the Protocol

We need to verify whether the  $W_i$ 's broadcast by each user reveal any information about the secret key  $s_i$ . Given the public key  $p_i$  of user  $u_i$ , we assign  $p_{i-1}$ ,  $p_{i+1}$  to dishonest users  $u_{i-1}$  and  $u_{i+1}$  and allow them to simultaneously execute a multi-party protocol to obtain  $y_i$  and  $W_i$  from  $u_i$ . We present an attack on protocol MIA to illustrate this issue, and show the security of protocols MIIA and MIIB.

### Attack against MIA

In MIA, using the real public keys of the dishonest users, we get

$$W_i = \frac{p_{i+1}^{x_i}}{y_{i-1}^{s_i}}$$

as well as  $y_i = \alpha^{x_i}$ . Colluding users can compute  $p_{i+1}^{x_i} = y_i^{s_{i+1}}$  obtaining  $y_{i-1}^{s_i}$ . Hence, this protocol (no matter if it aborts) can be followed to use  $u_i$  as an oracle to raise any chosen  $y_{i-1}$  to the secret key  $s_i$ . We can easily imagine how this allows recovery of a previous session key: in a previous session,  $\tilde{K}_{i-1}$  is  $\tilde{y}_{i-1}^{s_i}$ , and since  $\tilde{y}_{i-1}$  can be eavesdropped on the channel, one can ask  $u_i$  to raise it to  $s_i$  to obtain  $\tilde{K}_{i-1}$ , followed easily by computation of the previous session key  $\tilde{K}$ .

## Security of MIIA and MIIB against collusion attack

If they succeed Protocol P (either IIA or IIB), colluding users  $u_{i-1}$  and  $u_{i+1}$  obtain from  $u_i$

$$W_i = \frac{K_i}{K_{i-1}}$$

as well as  $y'_i$  and  $y_i$ . Since  $u_{i-1}$  ( $u_{i+1}$ ) has been able to produce  $z_{i-1}$  ( $z'_{i+1}$ ) and complete P before  $W_i$  is broadcast, he knew how to compute  $K_{i-1}$  ( $K_i$ ).<sup>7</sup> Therefore, no extra information is revealed by the broadcasting of the  $W_i$  (even to colluding users).

### 4.3 Active Attacks

For Protocols MIIA and MIIB, a traditional impersonation attack where user  $E$  successfully completes a protocol (including key computation in our case) with  $B$  (resp.  $A$ ) by impersonating  $A$  (resp.  $B$ ) would occur in step 1 from Section 4. (Recall that from the previous section an impersonation would not occur at step 2 since the  $W_i$  provide no extra information.) According to Theorem 3, this is not possible.

In all of the MP schemes we have investigated thus far, there has been an implicit assumption that if you were able to successfully complete a protocol with several users, then each of these users is honest. Relaxing this assumption introduces the following possible attacks (which are applicable to our schemes as well as the Burmester-Desmedt scheme).

Consider a multi-party protocol between users  $A$ ,  $B$ ,  $C$  and  $D$  who are oriented on a ring. If  $C$ 's left and right partners are  $B$  and  $D$  (i.e. the users with whom  $C$  will perform protocol P) then  $B$ ,  $C$  and  $D$  can collude by *shielding*  $C$ . By this we mean that  $B$  and  $D$  can construct their messages such that  $C$  can impersonate some  $Z$ . This is possible since no direct authentication is performed between users  $A$  and  $C$ . At the end of the protocol,  $A$  could be made believe that the protocol consists of users  $A$ ,  $B$ ,  $Z$  and  $D$ . Of course, this would not allow  $C$  (impersonating  $Z$ ) to compute the key on his own (see Theorem 1) but the key can be given to  $C$  by one of  $B$  or  $D$  ( $C$ 's colluding partners).

A solution to this attack is to include an additional step at the end of the protocol whereby each user  $i$  broadcasts the quantity

$$s_i(h_K(W_1, W_2, \dots, W_t))$$

which is  $i$ 's signature on the keyed hash of the  $W_i$ 's broadcast in step 2 of the multi-party protocol described in Section 4. Since our protocols are public-key based, the signature scheme can be easily implemented using ElGamal signatures [12] for example.

Note that the solution above works assuming that  $C$  is not able to falsify  $Z$ 's signature. A possible attack to this assumption occurs in [4]. Here, the authors present a so-called *Middleperson Attack*. Suppose we have Protocol 1, consisting of users  $A$ ,  $B$  and  $C$  and Protocol 2 consisting of users  $B$ ,  $C$  and  $Z$ . The attack involves  $C$  sitting in the middle of the two simultaneous protocols.  $C$  would impersonate  $Z$  in Protocol 1 and impersonate  $A$  in Protocol 2. Any challenges that  $C$  would be required to compute (as if they came from  $Z$ ), including possible signatures, in Protocol 1 would be obtained directly from  $Z$  in Protocol

---

<sup>7</sup>This is necessary under the assumption that knowledge of  $K_{i-1}$  (or  $K_i$ ) is necessary for computation of the keyed hash in Protocol P.

2 (and vice-versa for impersonating  $A$  in Protocol 2). At this point, users see Protocol I as consisting of users  $A$ ,  $B$  and  $Z$  and Protocol II of users  $B$ ,  $A$  and  $Z$ . Similar to the attack above,  $C$  would be unable to compute the key on his own as he is really only acting as a ‘wire’ between the two protocols, and passing along messages. Once again, he would require a collusion with  $B$  to obtain  $K$  (for the attack to be of any real use).

The solution presented to the attack in [4] was a hardware one rather than a cryptographic one. Also note that the property of key authentication was never really violated since that principal attacker  $C$  was never able to compute the key on his own. Depending upon the application, the practicality of such attacks must be individually examined.

## 5 Conclusion

Starting with the Diffie-Hellman problem and two related Key Transfer protocols, we have constructed a model for both two-party and multi-party key agreement as well as implementations that provide for privacy of the computed key and authenticity of the participating users.

## References

- [1] A. Beigel, B. Chor, “Interaction in Key Distribution Schemes”, *Advances in Cryptology: Proceedings of CRYPTO '93*, Springer-Verlag, 1993, pp.444-455.
- [2] M. Bellare, R. Gu erin, P. Rogaway, “XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions”, *Advances in Cryptology: Proceedings of CRYPTO '95*, Springer-Verlag, 1995, pp.15-28.
- [3] M. Bellare, P. Rogaway, “Entity Authentication and Key Distribution”, *Advances in Cryptology: Proceedings of CRYPTO '93*, Springer-Verlag, 1993, pp.232-249.
- [4] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, J. Quisquater, “Secure Implementation of Identification Systems”, *Journal of Cryptology*, Vol. 4, 1991, pp. 175-183.
- [5] R. Blom, “An Optimal Class of Symmetric Key Generation Schemes”, *Advances in Cryptology: Proceedings of Eurocrypt '84*, Springer-Verlag, 1985, pp.335-338.
- [6] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, “Perfectly-Secure Key Distribution for Dynamic Conferences”, *Advances in Cryptology: Proceedings of CRYPTO '92*, Springer-Verlag, 1993, pp.471-486.
- [7] M. Burmester, “On the Risk of Opening Distributed Keys”, *Advances in Cryptology: Proceedings of Crypto '94*, Springer-Verlag, 1994, pp.308-317.
- [8] M. Burmester, Y. Desmedt, “A Secure and Efficient Conference Key Distribution System”, *Advances in Cryptology: Proceedings of Eurocrypt '94*, Springer-Verlag, 1995, pp.275-286.

- [9] Y. Desmedt, M. Burmester, “Towards Practical ‘Proven Secure’ Authenticated Key Distribution”, *1st ACM Conference on Computer and Communications Security*, Fairfax, VA, Nov. 3-5, 1993, pp. 228-231.
- [10] W. Diffie, M. Hellman, “New Directions in Cryptography”, *IEEE Transactions on Information Theory*, IT-22(6), November 1976, pp.644-654.
- [11] W. Diffie, P.C. van Oorschot, M.J. Wiener, “Authentication and Authenticated Key Exchanges”, *Designs, Codes and Cryptography*, Vol. 2, 1992, pp. 107-125.
- [12] T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, *IEEE Transactions on Information Theory*, Vol. 31, pp. 469-472, 1985.
- [13] I. Ingemarsson, D. Tang, C. Wong, “A Conference Key Distribution System”, *IEEE Transactions on Information Theory*, Vol. IT-28, No.5, Sept. 1982, pp.714-720.
- [14] H. Krawczyk, “SKEME: A Versatile Secure Key Exchange Mechanism for Internet”, to appear at the *Internet Society Symposium on Network and Distributed System Security*, Feb. 1996 (also presented at the *Crypto '95* rump session).
- [15] T. Matsumoto, Y. Takashima, H. Imai, “On Seeking Smart Public-Key Distribution Systems”, *The Transactions of the IECE of Japan*, Vol. E. 69, No. 2, February 1986, pp. 99-106.
- [16] A. Menezes, M. Qu, S. Vanstone, “Some New Key Agreement Protocols Providing Implicit Authentication”, presented at the *Workshop on Selected Areas in Cryptography (SAC '95)*, Carleton University, Ottawa, ON., pp. 22-32.
- [17] B. Preneel, P. van Oorschot, “MDx-MAC and Building Fast MACs from Hash Functions”, *Advances in Cryptology: Proceedings of CRYPTO '95*, Springer-Verlag, 1995, pp.1-14.
- [18] R. Rivest, “The MD5 message-digest algorithm”, *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992.
- [19] P. Rogaway, “Bucket Hashing and its Application to Fast Message Authentication”, *Advances in Cryptology: Proceedings of CRYPTO '95*, Springer-Verlag, 1995, pp.29-42.
- [20] R. Rueppel, P. van Oorschot, “Modern Key Agreement Techniques”, *Computer Communications Journal*, Vol. 17, July 1994, pp. 458-465.
- [21] D. Steer, L. Strawczynski, W. Diffie, M. Wiener, “A Secure Audio Teleconference System”, *Advances in Cryptology: Proceedings of CRYPTO '88*, Springer-Verlag, 1988, pp.520-528.
- [22] M. Steiner, G. Tsudik, M. Waidner, “Diffie-Hellman Key Distribution Extended to Group Communication”, to appear in *3rd ACM Conference on Computer and Communications Security*, New Dehli, India, March 14-16, 1996.

- [23] D. Stinson, “On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption”, preprint available as <http://bibd.unl.edu/stinson/broadcast.ps>.
- [24] P. van Oorschot, M. Wiener, “On Diffie-Hellman Key Agreement with Short Exponents”, to appear in *Advances in Cryptology: Proceedings of Eurocrypt '96*, Springer-Verlag.
- [25] Y. Yacobi, “A Key Distribution ‘Paradox’ ”, *Advances in Cryptology: Proceedings of CRYPTO '90*, Springer-Verlag, 1990, pp.268-273.