Authenticated Multi-Party Key Agreement

Mike Just[1] and Serge Vaudenay[2]

[1] School of Computer Science, Carleton University, Ottawa, ON, Canada, K1S 5B6, e-mail: just@scs.carleton.ca

[2] Ecole Normale Supérieure–DMI, 45, rue d'Ulm, 75230 Paris Cedex 05, France, e-mail: Serge.Vaudenay@ens.fr

Abstract. We examine key agreement protocols providing (i) key authentication (ii) key confirmation and (iii) forward secrecy. Attacks are presented against previous two-party key agreement schemes and we subsequently present a protocol providing the properties listed above.
A generalization of the Burmester-Desmedt (BD) model (Eurocrypt '94) for multi-party key agreement is given, allowing a transformation of any two-party key agreement protocol into a multi-party protocol. A multi-party scheme (based on the general model and a specific 2-party scheme) is presented that reduces the number of rounds required for key computation compared to the specific BD scheme. It is also shown how the specific BD scheme fails to provide key authentication.

Key Words: key agreement, authentication, confirmation, forward secrecy.

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 [16] 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 [6]. There have been many attempts to provide authentic key agreement based on DH [7, 11, 12, 13, 20] In a separate direction, several attempts have been made to extend DH to a multi-party protocol [10, 17, 18], the most efficient being the result of Burmester and Desmedt [5].

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[3].

[3] We require a trusted center for creating public-key certificates for each user. However,

Users interact via an exchange of messages to obtain a common key.

Section 2 presents several definitions and building blocks that are used in the construction of our key agreement protocols. Section 3 demonstrates attacks to previous two-party protocols and presents the new key agreement protocol. Section 4 discusses the multi-party model, the specific Burmester/Desmedt protocol, as well as our own, and examines attacks against each.

1.1 Definitions and Notations

Let m be a prime and $\alpha \in \mathbb{Z}_m{}^*$ an element with order q, where q is a prime such that $q|m-1$ and computing discrete logarithms in the group generated by α is difficult (see recommended parameters given in [19]). 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 $s_U \in_R$[4] $\mathbb{Z}_q{}^*$. 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 key *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 h_K a Message Authentication Code (MAC), i.e. [15]. Furthermore, we assume that this MAC (of a hash function) behaves as a random oracle in the sense that its output reveals no meaningful information about its input. See [14] for details.

1.2 Summary of Results

We begin by examining a Diffie-Hellman based 2-pass key agreement protocol that has appeared in several variations in the literature. Two minor (repairable) attacks against this scheme are presented as well as two more serious attacks given that the attacker has some extra information available to him. It is also shown how the property of (perfect) *forward secrecy* as defined in [7] (as well as Section 2) has been mistakenly attributed to this protocol.

Subsequently we present a Diffie-Hellman based 3-pass protocol (Protocol IIA) which provides for (i) key authentication, (ii) key confirmation and (iii) forward secrecy (see Section 2 for definitions). The protocol is based on a general framework that is evident in several other key agreement schemes found in the literature. We examine the security of our protocol against some passive and active attacks.

We extend our two-party results by generalizing the specific multi-party protocol of Burmester and Desmedt [5] to obtain a multi-party key agreement model. Using our specific two-party protocol and this model, we are able to obtain a multi-party protocol (Protocol MIIA) which reduces the amount of communication required between participants (as compared to the scheme of [5]). It is

this can be completed off-line, and the center is not required to maintain the secrecy of any information for any users.

[4] We denote an element x chosen randomly and independently from a set S by $x \in_R S$.

also shown how the scheme of Burmester and Desmedt [5] fails to provide key authentication. Attacks against Protocol MIIA are also examined.

2 Fundamentals

In this paper, we build from 1-pass key transfer (KT) protocols to multiple pass key agreement (KA) protocols. Where a KT protocol involves contributions from only 1 user, 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.

We say that a key agreement protocol is *successful* if each of the parties accepts the identity of the other party, and terminate with the same key. The protocol provides *key authentication* if the ability for computing the key implies knowledge of the secret corresponding to the identity of one expected participant. Key authentication implies key confidentiality. For if only intended parties can compute the key, then unintended parties cannot compute the key. *Key confirmation* (direct authentication in [7]) is provided if the protocol aborts unless participants demonstrate knowledge of the same shared session key. Note that in this context an encrypted exchange subsequent to the KA protocol "demonstrates knowledge" of the key. The distinction is that for key confirmation, knowledge of the key is demonstrated prior to the end of the KA protocol (and is usually achieved by encrypting or hashing a known quantity). A key agreement protocol provides *forward secrecy* (perfect forward secrecy in [7] and [9]) if the loss of any long-term secret keying material does not allow the compromise of keys from previously wire-tapped sessions. Since *perfect* usually makes reference to information theory, we avoid it here. 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.

Our goal throughout is for a dynamic set of users to securely compute a *session key* K for the purpose of participating in a secure communication session. Long-term public keys for each user serve to authenticate while short-term per-session tokens serve to add freshness to the KA protocol and hence to the computation of K.

2.1 Key Transfer Protocols

The traditional DH problem (upon which our protocols are based) can be stated as follows. Given α as defined in Section 1.1 and inputs $y = \alpha^x$ and $y' = \alpha^{x'}$, compute (we omit reference to m for simplicity) $\mathrm{DH}(\alpha; y, y') = \alpha^{xx'}$. Likewise, for long-term public parameters $p_A = \alpha^{s_A}$ and $p_B = \alpha^{s_B}$, we have $\mathrm{DH}(\alpha; y, p_A) =$

A		B
$x \in_R \mathbb{Z}_q, y = \alpha^x$ $K = {p_B}^x$	$\xrightarrow{\quad y, I = I_A \quad}$	$K = y^{s_B} (= \alpha^{x s_B})$
$x \in_R \mathbb{Z}_q, y = {p_B}^x$ $K = \alpha^x$	$\xrightarrow{\quad y, I = I_A \quad}$	$K = y^{{s_B}^{-1}} (= \alpha^x)$

Fig. 1. Protocol IA(top) and IB(bottom)

$\alpha^{s_A x}$ and $\mathrm{DH}(\alpha; p_A, p_B) = \alpha^{s_A s_B}$.[5] The DH problem is the basis for the two 1-pass KA (i.e. Key Transfer) protocols given in Figure 1. Protocol IA can be considered to be a DH protocol with one fixed parameter. Protocol IB is a simple variation on the first.

The key computation for Protocol IA is $\mathrm{DH}(\alpha; y, p_B)$ and $\mathrm{DH}(p_B; {p_B}^x, \alpha)$ for Protocol IB. Since each computation has one fixed parameter, these protocols are no harder than a DH computation (with two random parameters). Due to page limitations, protocols based on Protocol IB appear in Appendix A.

2.2 Framework for Key Authentication

The framework for our KA protocols follows similar work from [2, 7, 11, 12, 13, 20]. It consists of a 3-pass authentic key agreement protocol as shown in Figure 2. The values y and y' are random tokens generated by each user (that will be used in the key computation). The offsets "1:" and "2:" are included to prevent potential rebound attacks possible given the similarity of the inputs to the hash by both A and B. I and I' refer to the identities of the respective participants. The terms $< y >$ and $< y' >$ refer to pseudo-corroboration of the fact that the originating user actually constructed the term enclosed in the $<>$s. (By pseudo-corroboration we mean that it is not a true zero-knowledge proof of possession, nor is it as costly as one.) Particularly for our case, given that user A has constructed $y = \alpha^x$, A should also be able to produce $< y > = (y'p')^x$ for random y'. The difficulty of this task is considered in Theorem 2. (We note that such precautions have also been noted by Burmester [4].) As mentioned in Section 1.1, we assume that the output of h_K behaves as a random oracle in that it reveals no meaningful information about its input. The output of this hash serves to provide for key confirmation as well as the pseudo-corroboration described above.

This framework is by no means entirely new and is clearly evident in the works cited above. Whereas encryption and signatures are used in the respective

[5] This is an abuse of notation. Since p_A and p_B are fixed for each protocol run, their inclusion in the calculation should be distinguished from y and y' which are randomly chosen for each run. The result being that an ability to compute $\mathrm{DH}(\alpha; y, y')$ implies an ability to compute $\mathrm{DH}(\alpha; y, p_B)$ yet the reverse implication is still open.

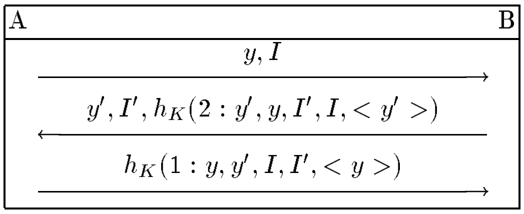

Fig. 2. Generic Authenticated Key Agreement

schemes of Krawczyk [11] and Diffie et al. [7] for authentication, we incorporate the public keys of each user directly into the key computation, as was done in [13, 12, 20]. Also, the use of a MAC for providing key confirmation replaces the use of an encryption function (which is unnecessary since there is no decryption taking place – and relaxes the possibility of export restrictions).

Though not as formalized as the work of [2] (which assumes only the existence of a pseudorandom function), the reliance on the DH problem by each of the remaining works cited above (including the current paper) allows for the provision of forward secrecy (a property not achieved in [2]). Such a property may be attractive for the robustness of the security in most commercial applications where customers does not always protect their secret long-term key sufficiently.

3 Authenticated Key Agreement

In this section, we extend Protocol IA to provide for authenticated key agreement. (Similarly, see Appendix A for extensions of Protocol IB.) 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 3.2.) Throughout the section, $p = p_A$ is the public key extracted from I, while $p' = p_B$ is extracted from I' (though the same notation follows if the public keys are *a priori* available).

3.1 Protocols Based on IA

Consider the two party key agreement protocol between users A and B from [13] given in Figure 3. (Similar protocols for which there was no key confirmation are given in [12, 20] and were attacked by Burmester [4].) Two minor attacks against Protocol A0 in the absence of a proper implementation are

- E IMPERSONATES B TO A. In place of B, E sends $\{y' = 0, I' = I_B, z' = h_K(y', I, I')\}$ 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.

A		B
$x \in_R \mathbb{Z}_q, y = \alpha^x$	$\xrightarrow{\quad y, I = I_A \quad}$	$x' \in_R \mathbb{Z}_q, y' = \alpha^{x'}$
		$K = p^{x'} y^{s_B}$
		$z' = h_K(y', I, I')$
$K = (y')^{s_A}(p')^x$	$\xleftarrow{\quad y', I' = I_B, z' \quad}$	
$z' \stackrel{?}{=} h_K(y', I, I')$		

Fig. 3. Protocol A0: $K = \alpha^{s_A x' + s_B x}$

- E IMPERSONATES A TO A. This more subtle attack succeeds so long as A does not verify that he is communicating with "himself". Suppose A is an automated system providing access to an encrypted session with a computer database. Access is granted to those users who successfully complete the protocol. After an initiation of the protocol by A, 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 $\{y', I' = I_A, z' = h_K(y', I, I')\}$ to A.

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 $I \neq I'$. The latter condition may be too restrictive. Possibly for maintenance purposes, some applications may want the option of having $I = I'$. However, the following more serious attacks motivate a solution that also appears to thwart the second attack described above.

- E IMPERSONATES B TO A. (GIVEN THAT E POSSESSES s_A.) It is obvious that s_A allows E to impersonate A to any user. However, suppose that A is an Automatic Teller Machine and the engineer E who initially performs the setup of A, is able to obtain s_A. After A initiates the protocol, E chooses $y' = \alpha^{\tilde{x}}/p_B$. Given s_A and $\tilde{x}$, E can easily compute K.
- E IMPERSONATES B TO A (OR A TO B). (GIVEN THAT E POSSESSES $\alpha^{s_A s_B}$.) Since s_A and s_B are long-term secrets this attack allows unlimited impersonations given only $\mathrm{DH}(\alpha; p_A, p_B)$. To impersonate A, E computes and sends $y = \alpha^{\tilde{x}}/p_A$ to B. Given $\tilde{x}$ and $\alpha^{s_A s_B}$, E can easily compute K. Similarly, E can impersonate B to A.

In each of these last two attacks, E does not know the discrete logarithm of its token, i.e. y or y', motivating the inclusion of a demonstration of knowledge of the construction of the token as discussed in Section 2 and included in Protocol IIA below. Also of note for Protocol A0 is the fact that it *does not provide* for forward secrecy (as claimed in [13]). Note that 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. In Figure 4 we present Protocol IIA which appears to prevent the aforementioned attacks and uses the framework from Figure 2.

A		B
$x \in_R \mathbb{Z}_q, y = \alpha^x$	$\xrightarrow{y, I = I_A}$	$x' \in_R \mathbb{Z}_q, y' = \alpha^{x'}$
		$w' = (yp)^{x'}, K = w' y^{s_B}$
		$z' = h_K(2 : y', y, I', I, w')$
$w' = (y')^{x+s_A}$	$\xleftarrow{y', I' = I_B, z'}$	
$w = (y'p')^x, K = w(y')^{s_A}$		
$z' \stackrel{?}{=} h_K(2 : y', y, I', I, w')$		
$z = h_K(1 : y, y', I, I', w)$		
	$\xrightarrow{z}$	$w = (y)^{x'+s_B}$
		$z \stackrel{?}{=} h_K(1 : y, y', I, I', w)$

Fig. 4. Protocol IIA: $K = \alpha^{xx'+x's_A+xs_B}$

3.2 Passive and Active Attacks

In this section, we analyze the resistance of Protocol IIA to passive and active attacks by demonstrating equivalence to variations of the DH problem. (Similar arguements can be made for Protocol IIB from Appendix A.) Hence, throughout this section, we assume that DH 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 it's output cannot be distinguished from random output.

A *passive attack* whose aim is key recovery for a given session involves eavesdropping on messages passed between participants for that session. The attack is successful if the session key can be recovered with a probabilistic polynomial time algorithm given as input, the eavesdropped message passes as well as any other publicly available parameters.

Theorem 1. *Protocol IIA is secure against passive attack and provides forward secrecy unless the Diffie-Hellman problem can be solved.*

We note that 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

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

Proof. (sketch) 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.

Since computing the final key with the extra knowledge of s_A and s_B is hard, it must be at least as hard without it. Protocol IIA is therefore secure against passive attacks. □

An *impersonation attack* involves an attacker who is given access to all publicly available information and attempts to successfully complete a protocol with B (resp. A) by impersonating A (resp. B). Recall that a key agreement protocol is successful if each of the parties accept the identity of the other, and terminate with the same key. Note that since Protocol IIA provides key confirmation, this assumes knowledge of the session key K.

Theorem 2. *Protocol IIA is secure against impersonation attack given that Protocol IA is secure.*

Proof. (sketch) If z' is accepted by A, it has necessarily been produced by someone who is able to compute both the pair $(y', (yp)^{x'})$ and the key K by using the (supposed) random y. From K and $(yp)^{x'}$, is it easy to compute y^{s_B}. Since y is fresh and s_B is supposed to be secret, z' has been forged by B unless IA is insecure.

Similarly, if z is accepted by B, it has been produced by someone able to compute ${y'}^{s_A}$ from a fresh y'. □

Notice that for Theorem 2, there is an implicit assumption that Protocol IA does not reveal any partial information, i.e. each run of Protocol IA produces a key of the form $\mathrm{DH}(\alpha; y, p_B)$ for some random y and fixed p_B. Also note that this does not preclude more imaginative attacks. It simply states that one participant cannot successfully complete Protocol IIA by impersonating another, given only the publicly available parameters.

4 Authenticated 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.[6] We assume that all users $u_1, u_2, \ldots, u_t$ 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 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

$$K \equiv {K_{i-1}}^t {W_i}^{t-1} {W_{i+1}}^{t-2} \cdots W_{i-2} \equiv K_1 K_2 \cdots K_t.$$

Equivalently, we can use $W_i = K_i - K_{i-1}$ and $K = K_1 + \ldots + K_t$ (or even $W_i = K_i \oplus K_{i-1}$ and $K = K_1 \oplus \ldots \oplus K_t$), for example. Since addition is much cheaper than multiplication, such computations have an obvious practical

[6] The construction is a generalization of the scheme from [5].

benefit. Verification that all the following discussions hold for this modification is left to the reader.

For a specific implementation of this model we use Protocol IIA from Section 3 to obtain the respective multi-party protocol MIIA (likewise for Protocol IIB from Appendix A). 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}.

4.1 Attack to Burmester/Desmedt Scheme

In this section we demonstrate how the scheme of Burmester and Desmedt (BD) [5] does not provide key authentication. BD is a specific case of the model describe above with DH as protocol P and makes use of zero-knowledge techniques for authenticating each user. We make use of an attack first put forth in [13].

The adversary E positions himself between any two users A and B and convinces B that he shares a key with E (though E will be unable to compute K), yet B actually shares K with A. A believes (and in fact does) share K with B. 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). Subsequent to 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.

From [5], each user i has a public key pair (β_i, γ_i) where $\beta_i = \alpha^{v_i}$ and $\gamma_i = \alpha^{w_i}$. This version assumes that users' public keys are *a priori* available. The attack proceeds as follows. A selects $x \in_R \mathbb{Z}_q$ and sends $\{y := \alpha^x, I := I_A\}$ to B. E intercepts the communication so now 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 [5]. E sends y to B, and using his public key pair (β_E, γ_E), authenticates y to B. B sends $\{y', I := I_B\}$ to E. E simply forwards this message to A and allows B to authenticate y' to A. A and B complete the protocol by broadcasting W_A and W_B respectively.

The attack succeeds because of the lack of "binding" between the messages exchanged between A and B and lack of protection of the names of the intended recipients of the messages. These properties are identified in [1] as being important for obtaining a secure and authentic cryptographic protocol.

The authentication between pairs of users in [5] 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, giving a total of $k+2$ rounds. Protocol MIIA requires 3 rounds for the processing of Protocol IIA (including authentication of tokens) and 1 round for the broadcast of the W_i's, giving a total of 4 rounds. 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.

4.2 Passive Attacks

In this section, we show that the multi-party model specifically implemented with Protocol IIA (to produce MIIA) is provably secure against a passive attacker. (Similar arguements can be made for an implementation with Protocol IIB from Appendix A.) This is done by illustrating their equivalence to the respective schemes from Section 3 (using the same techniques as given in [5]).

Theorem 3. *Given an even, polynomial number t of randomly chosen users with long-term keys that are uniformly distributed, Protocol MIIA is as secure against passive attacks as Protocols IIA.*

Proof. We first note that breaking Protocol IIA obviously enables one to break Protocol MIIA (solve for each K_i followed by computation of their product).

Now, given $p_1 = p_B = \alpha^{s_B}$, $y_1 = y' = \alpha^{x'}$, $p_t = p_A = \alpha^{s_A}$ and $y_t = y = \alpha^x$, we want to solve for the key $\alpha^{x_t x_1 + x_t s_1 + x_1 s_t}$ (i.e. $\alpha^{xx' + xs_B + x's_A}$ 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 $i = 2, \ldots, t-1$, $p_i = p_{i-2}\alpha^{b_i}$ and $y_i = y_{i-2}\alpha^{c_i}$ 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 $i = 1, \ldots, t-2$, 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

$$\begin{aligned} p_t &\equiv p_2 g^{-b_2} \equiv p_4 g^{-b_2-b_4} \equiv \cdots \equiv p_{t-2} g^{-b_2-b_4-\cdots-b_{t-2}} \\ p_1 &\equiv p_3 g^{-b_3} \equiv p_5 g^{-b_3-b_5} \equiv \cdots \equiv p_{t-1} g^{-b_3-b_5-\cdots-b_{t-1}}, \end{aligned}$$

(and similarly for y_t and y_1) allowing us to compute

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

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 $\alpha^{x_t x_1 + x_t s_1 + x_1 s_t}$. $\square$

4.3 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 Protocol MIIA.

In MIA, using the real public keys of the dishonest users, we get $W_i = \frac{{p_{i+1}}^{x_i}}{{y_{i-1}}^{s_i}}$ and $y_i = \alpha^{x_i}$. 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 $\tilde{K}$.

If they complete Protocol MIIA (hence succeeded Protocol IIA), 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 IIA before W_i is broadcast, he knew how to compute K_{i-1} (K_i).[7] Thus, active attacks do not recover more information than passive ones from the W_i.

4.4 Active Attacks

For Protocol MIIA, 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 from the previous section, the W_i released during step 2 provide no extra information.) According to Theorem 2, this is unlikely.

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 Section 3.2) 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 $s_i(h_K(W_1, W_2, \ldots, W_t))$,

---

[7] 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 IIA.

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 [8] 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 [3]. 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 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 [3] was a hardware one rather than a crytographic 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

In this paper, we presented two new key agreement protocols, the two-party Protocol IIA and its multi-party counterpart, Protocol MIIA. Protocol IIA appears to improve upon several others for which long-term public keys are used in the key computation (and for which several attacks were given here). Their use in key computation is an alternative to the use of digital signatures.

Protocol MIIA is derived from our generalization of the multi-party protocol of Burmester and Desmedt (BD) [5]. A nice feature of the protocol is that it allows for authentication of participating users without requiring that each user authenticate every other user. Though one must be careful with such methods as evidenced by the shielding attack from Section 4.4.

Protocol MIIA differs from the scheme of BD in how participants are authenticated. Rather than using zero-knowledge techniques (which are susceptible to the attack from Section 4.1), we essentially use Diffie-Hellman computations in key confirmation. It seems likely that many of the other two-party key agreement protocols mentioned here can also provide for a multi-party protocol using the generalization from Section 4.

Acknowledgements. Thanks to Paul Van Oorschot for suggesting the current topic for research. Using a different group structure for the computation of the

W_i and K_i in Section 4 was suggested by Kazue Sako. Thanks to an anonymous referee for the fourth attack (where E possess $\alpha^{s_A s_B}$) given in Section 3.1. Thanks to Yvo Desmedt for pointing out the existence of [4].

Note. The first author is supported by an NSERC graduate fellowship. The second author is employed by the CNRS. This work was partially completed while the second author was visiting the School of Computer Science at Carleton University and supported by an NSERC grant.

References

1. M. Abadi, R. Needham, "Prudent Engineering Practice for Cryptographic Protocols", *DEC SRC Research Report* **125**, June 1, 1994.
2. M. Bellare, P. Rogaway, "Entity Authentication and Key Distribution", *Advances in Cryptology: Proceedings of CRYPTO '93*, Springer-Verlag, 1993, pp.232-249.
3. S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, J. Quisquater, "Secure Implementation of Identification Systems", *Journal of Cryptology*, Vol. 4, 1991, pp. 175-183.
4. M. Burmester, "On the Risk of Opening Distributed Keys", *Advances in Cryptology: Proceedings of Crypto '94*, Springer-Verlag, 1994, pp.308-317.
5. 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.
6. W. Diffie, M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22(6), November 1976, pp.644-654.
7. W. Diffie, P.C. van Oorschot, M.J. Wiener, "Authentication and Authenticated Key Exchanges", *Designs, Codes and Cryptography*, Vol. 2, 1992, pp. 107-125.
8. 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.
9. C. Günther, "An Identity-Based Key Exchange Protocol", *Advances in Cryptology: Proceedings of Eurocrypt '89*, Springer-Verlag, 1989, pp.29-37.
10. 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.
11. H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, Feb. 1996 (also presented at the *Crypto '95* rump session).
12. 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.
13. 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.
14. D. Pointcheval, J. Stern, "Security Proofs for Signature Schemes", *Advances in Cryptology: Proceedings of Eurocrypt '96*, Springer-Verlag, 1996, pp.387-398.
15. B. Preneel, *Cryptographic Hash Functions*, Kluwer Academic Publishers (to appear, 1996).
16. R. Rueppel, P. van Oorschot, "Modern Key Agreement Techniques", *Computer Communications Journal*, Vol. 17, July 1994, pp. 458-465.

17. 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.
18. M. Steiner, G. Tsudik, M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication", *3rd ACM Conference on Computer and Communications Security*, New Dehli, India, March 14-16, 1996.
19. P. van Oorschot, M. Wiener, "On Diffie-Hellman Key Agreement with Short Exponents", *Advances in Cryptology: Proceedings of Eurocrypt '96*, Springer-Verlag, 1996, pp.332-343.
20. Y. Yacobi, "A Key Distribution 'Paradox' ", *Advances in Cryptology: Proceedings of CRYPTO '90*, Springer-Verlag, 1990, pp.268-273.

A Protocols Based on IB

The protocols in Figure 5 require *a priori* knowledge of public keys (or an extra message pass). Protocol B0 was given in [13] (an adaptation of a scheme from [12]). Similar attacks to those presented in Section 3.1 can be mounted against Protocol B0. Protocol IIB prevents the attacks and provides forward secrecy. The key computation is identical to the two-pass protocols from [13, 12].

A		B
$x \in_R \mathbb{Z}_q, y = p_B{}^x$	$\xrightarrow{y, I = I_A}$	$x' \in_R \mathbb{Z}_q, y' = p^{x'}$
		$K = \alpha^{x'} y^{s_B{}^{-1}}$
		$z' = h_K(y', I, I')$
$K = (y')^{s_A{}^{-1}} \alpha^x$	$\xleftarrow{y', I' = I_B, z'}$	
$z' \stackrel{?}{=} h_K(y', I, I')$		
$x \in_R \mathbb{Z}_q, y = p_B{}^x$	$\xrightarrow{y, I = I_A}$	$x' \in_R \mathbb{Z}_q, y' = p^{x'}$
		$w' = \alpha^{x'}, K = y^{s_B{}^{-1} x'}$
		$z' = h_K(2 : y', y, I', I, w')$
$w' = (y')^{s_A{}^{-1}}$	$\xleftarrow{y', I' = I_B, z'}$	
$w = \alpha^x, K = (y')^{s_A{}^{-1} x}$		
$z' \stackrel{?}{=} h_K(2 : y', y, I', I, w')$		
$z = h_K(1 : y, y', I, I', w)$		
	$\xrightarrow{z}$	$w = (y)^{s_B{}^{-1}}$
		$z \stackrel{?}{=} h_K(1 : y, y', I, I', w)$

Fig. 5. Protocol B0(top): $K = \alpha^{x+x'}$; Protocol IIB(bottom): $K = \alpha^{xx'}$

This article was processed using the LaTeX macro package with LLNCS style