Password-authenticated key exchange using efficient MACs

Maurizio A. Strangio
Department of Computer Science, Systems and Production, University of Rome "Tor Vergata", Rome, ITALY
Email: strangio@disp.uniroma2.it

Abstract— **This paper is concerned with password-authenticated key agreement protocols. Designing such protocols represents an interesting challenge since there is no standard way of choosing a password that achieves an optimum trade-off between usability and security. Indeed, passwords belonging to a highly structured language are essentially equivalent to low entropy strings.**

A fundamental goal is that of obtaining secure and efficient protocols, with optimum computational complexity, round complexity and communication efficiency. These properties make them ideal candidates for mobile devices.

We present DH-BPAKE1 which is an improved version of the protocol presented in previous work (DH-BPAKE). The construction builds upon the encrypted key exchange protocol of Bellovin and Merritt augmented with a key confirmation round based on the use of efficient message authentication codes.

We discuss in detail the security properties of the two efficient message authentication schemes which form the basic building blocks of the protocol. In addition, we formally prove the security of protocol DH-BPAKE1 in a modified version of the model of Boyko *et al.*.

Index Terms— **key agreement protocols, password based authentication, message authentication scheme**

I. Introduction

In general, password-authenticated key agreement protocols are hard to design since an adversary eavesdropping on the network can first store the messages exchanged in a run of the protocol and later try to replay them off-line using many different candidate passwords chosen from a *dictionary*, until she encounters a correspondence with the set of transcripts that were captured. Some poorly designed protocols may be also vulnerable to *partition attacks* [17], [18], where entire subsets of the password space can be ruled out in a single attack. With a well designed protocol, we want to eliminate the possibility of mounting an off-line dictionary attack and constrain the adversary to actively participate in at least one run of the protocol (on-line) for each password guess.

With the proliferation of wireless networks (e.g. in public hot-spots, at home, at work) and mobile devices there will be a greater demand for secure wireless communications in the near future. When roaming users meet in a public area (e.g. airport) and need to synchronise the data on their mobile devices they may either use any available wireless service (e.g. WLAN) or a local wireless technology (e.g. Bluetooth—in this case, it would be easy for them to agree on a shared password on-the-fly). However, there is no standard way of choosing a password that achieves an optimum trade-off between usability and security. Indeed, passwords belonging to a highly structured (human) language (and therefore easy to remember) are equivalent to low entropy strings. Furthermore, open-air networks are inherently insecure since eavesdropping is "free" (e.g. war driving with open source software tools and a portable computer).

Optimizing resource usage is an important issue from both the perspective of the user and of the network provider. On one hand, users can benefit from the increased processing power and memory capacity of mobile devices (except for battery duration which is now becoming a major concern). On the other hand, although bandwidth is growing in wireless networks the number of available connections is still limited (implying expensive services for the customer). For these reasons, it is desirable to design secure key agreement protocols with optimum computational complexity, round complexity and communication efficiency.

In this paper we present protocol DH-BPAKE1 which is an improved version of the protocol (DH-BPAKE) developed in previous work [21]. Protocol DH-BPAKE1 is essentially based upon basic encrypted Diffie-Hellmann key exchange with an additional communication round that provides *explicit key confirmation* (whereby each party is convinced that its intended communication partner holds the same session key).

The main feature (and novelty) of the protocol derives from the use of efficient *message authentication codes* (MACs) to achieve key confirmation (e.g. see the compilers of [1], [4] for standard key confirmation techniques). One algorithm is conjectured in [5] as a computationally secure MAC under known message attacks, but only for randomly distributed messages. The most widely accepted notion of security for a MAC requires resistance to selective forgery against adaptively chosen message attacks. The other scheme is based on the family of strong 2-universal hash functions constructed in [16]. We prove that the resulting scheme is $1/q$*-secure after a single tag*, a notion that corresponds to perfectly secure MACs [11].

Due to its symmetry the protocol can achieve an optimal round complexity in any communication network

This paper is based on "An Optimal Round Two-Party Password-Authenticated Key Agreement Protocol," by M.A.Strangio, which appeared in the Proceedings of the 1st IEEE Int'l Conference on Availability, Reliability and Security, Vienna, AUSTRIA, April 2006. © 2006 IEEE.

© 2006 ACADEMY PUBLISHER

that support simultaneous message transmission between principals (in a single round of the protocol). Fortunately such networks exist; for example, a wireless network is inherently full-duplex and therefore offers the possibility of realistic simultaneous communication (provided the communicating devices have two radios).

The rest of this paper is organized as follows. In section II we compare the major features of protocol DH-BPAKE1 with several similar protocols. Section III introduces some background material and provides a detailed description of the protocol. An in depth security analysis is presented in Section IV. The final section contains the concluding remarks and future work.

II. Related work

There is a large body of work in the literature concerning password-authenticated key agreement protocols. One interesting line of research was opened by Bellovin and Merritt [2] which presented a password-authenticated key exchange protocol based on the encryption of Diffie-Hellmann public keys. The papers of [1], [3], [6], [7], [9], [10], [12], [13], [20] significantly continue and extend the work of Bellovin and Merritt.

Table I summarizes the properties of several password-authenticated key agreement protocols that provide some form of key confirmation. The protocols are equivalent in terms of the number of communication rounds, as shown in column one. However, protocol DH-BPAKE1 requires only two rounds when run in an full-duplex network. Column two counts the number of exponentiations that each principal must perform in order to run the protocol. Column three shows that protocol DH-PBAKE1 requires the least number of nonces to be generated. Column four gives evidence of the cryptographic primitives used as the basic building blocks in all the protocols with SENC standing for "symmetric encryption", AENC for "asymmetric encryption" and HASH for "hash function". Finally, column five reveals whether there exists a formal security proof developed for each of the protocols.

III. Protocol Specification

A. Preliminaries

If X is a finite set then $x \stackrel{R}{\leftarrow} X$ denotes the sampling of an element uniformly at random from X. If α is neither an algorithm nor a set $x \leftarrow \alpha$ represents a simple assignment statement.

The symbol $\oplus$ denotes bitwise *exclusive or* on strings of equal length.

Let $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $\mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^\ell$ denote two cryptographic hash functions where ℓ is the security parameter.

We consider the finite field $\mathbb{Z}_p$ where p is a large prime and $\mathcal{G}$ a q-order subgroup of $\mathbb{Z}_p$ with $q|p-1$. The element g is the generator of $\mathcal{G}$. We omit "mod q" from expressions when it is clear from context that we are working in $\mathcal{G}$. The public domain parameters are thus $\Phi = (\mathcal{G}, p, q, g)$.

The security of many cryptosystems is based on the conjectured unsolvability of the Discrete Logarithm (DLP) and Computational Diffie-Hellman (CDHP) problems in certain groups.

Assumption 1 (DLP): The group $\mathcal{G}$ satisfies the DL assumption if for all probabilistic polynomial time (PPT) algorithms $\mathcal{A}$ we have:

$$\Pr[x \stackrel{R}{\leftarrow} \mathbb{Z}_q^*; Y \leftarrow g^x : \mathcal{A}(\Phi, Y)=x] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of x).

Assumption 2 (CDHP): The group $\mathcal{G}$ satisfies the CDH assumption if for all PPT algorithms $\mathcal{A}$ we have:

$$\Pr[x, y \stackrel{R}{\leftarrow} \mathbb{Z}_q^*; Y \leftarrow g^x; Y \leftarrow g^y : \mathcal{A}(\Phi, X, Y)=g^{xy}] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of x, y).

In a concrete security analysis we consider the maximum probability over all adversaries $\mathcal{A}$ running in time bounded by t, in this case we say that $\mathcal{G}$ is (t, ϵ)-*secure* with respect to the DLP (resp. CDHP).

A MAC is a 3-tuple $\langle \mathit{key}(\cdot), \mathit{tag}_\kappa(\cdot), \mathit{ver}_\kappa(\cdot) \rangle$ of polynomial time algorithms where:

- $\mathit{key}(\cdot)$ is a randomized key generation algorithm that returns a key κ when the input is 1^ℓ;
- $\mathit{tag}_\kappa(m)$ is a (randomized) algorithm that accepts message $m \in \{0,1\}^*$, key κ in input and returns a tag τ;
- $\mathit{ver}_\kappa(\tau)$ is a (deterministic) tag verification algorithm that accepts (m, τ) in input and returns 1 iff $\mathit{tag}_\kappa(m) = \tau$ otherwise it returns 0.

We also require a correctness condition where for all m in the message space if $\mathit{tag}_\kappa(m) = \tau$ then $\mathit{ver}_\kappa(m, \tau)$ should output 1.

Consider the message authentication scheme $\mathrm{MAC}_{\langle u,v \rangle}$ defined on the field $\mathbb{Z}_q^*$, for a prime q, where the key κ is a pair of elements $\langle u, v \rangle$ in $\mathbb{Z}_q^*$, $\tau = \mathit{tag}_{\langle u,v \rangle}(m) = u + mv \mod q$ and $\mathit{ver}_{\langle u,v \rangle}(m, \tau) = (u + mv)^{-1} \cdot \tau \mod q$.

Also consider the message authentication scheme MAC_κ defined on the field $\mathbb{Z}_q^*$, for a prime q, where the key κ is an element of $\mathbb{Z}_q^*$, $\tau = \mathit{tag}_\kappa(m) = \mathrm{MSB}_k[(m+\kappa)^{-1} \mod q]$ and $\mathit{ver}_\kappa(m, \tau) = \mathrm{MSB}_k[(m+\kappa) \mod q]^{-1} \cdot \tau \mod q$. The function $\mathrm{MSB}_k[\cdot]$ returns the k most significant bits of its argument. We will use keys, denoted by π, that correspond to passwords $\mathtt{pw}$ that are efficiently mappable to elements of $\mathbb{Z}_q^*$ (e.g. $\pi = \mathcal{H}_1(\mathtt{pw})$).

B. Description of protocol DH-BPAKE1

The protocol is based on Diffie-Hellmann key exchange augmented with a key confirmation round. Key confirmation is achieved with the MACs described above. In a practical instantiation of the protocol the confirmation key should be immediately erased after its use. Note that communication between principals is asynchronous in each round but synchronous when considering one round respect to another.

© 2006 ACADEMY PUBLISHER

TABLE I.
COMPARISON WITH OTHER PASSWORD-AUTHENTICATED KEY AGREEMENT PROTOCOLS.

↓Protocol/Property→	*Flows*	*Exps*	*Nonces*	*Crypto Prim.*	*Sec. Proof*
DH-EKE [2]	5	2	2	SENC,HASH	No
SPEKE [9]	5	2	2	SENC,HASH	No
OKE [14]	4	3	2	AENC,SENC,HASH	Yes
GLNS [8]	5	4	2	AENC,SENC,HASH	No
PAK [6]	4	4	2	SENC,HASH	Yes
DH-BPAKE1	4(2)	2	1	SENC,HASH	Yes

The main actions of protocol DH-BPAKE1 using $\text{MAC}_{\langle u,v\rangle}$ are summarised as follows (refer to Figure 1 for the details):

1) Principal A (resp. B) selects a random r_A (resp. r_B) in $\mathbb{Z}_q^*$, computes the Diffie-Hellman ephemeral public key g^{r_A} (resp. g^{r_B}), encrypts it with the password π and sends the ciphertext x_A (resp. x_B) to B (resp. A);
2) If A (resp. B) does not receive any message from B (resp. A) (after sending its own) within a fixed time-slot, it aborts the protocol. Otherwise, A (resp. B) carries on by computing the key $\langle u_A, v_A\rangle$ (resp. $\langle u_B, v_B\rangle$), constructs the tag t_A (resp. t_B) that authenticates message x_A (resp. x_B) and sends it to B;
3) If A (resp. B) does not receive any message from B (resp. A) (after sending its own) within a fixed time-slot, it aborts the protocol. On receipt of t_B principal A accepts if the tag verifies otherwise it aborts;
4) The protocol successfully completes if both A and B accept holding the same session key $k_A = k_B$.

$A, B:$	π
$A:$	$r_A \overset{R}{\leftarrow} \mathbb{Z}_q^*$
	$x_A \leftarrow g^{r_A} \oplus \pi$
$B:$	$r_B \overset{R}{\leftarrow} \mathbb{Z}_q^*$
	$x_B \leftarrow g^{r_B} \oplus \pi$
$A \rightarrow B$:	x_A
$B \rightarrow A$:	x_B
$A:$	$s_A \leftarrow (x_B \oplus \pi)^{r_A}$
	$u_A \leftarrow \mathcal{H}_1(01, A, B, x_B \oplus \pi, s_A, \pi)$
	$v_A \leftarrow \mathcal{H}_1(10, A, B, x_B \oplus \pi, s_A, \pi)$
	$t_A \leftarrow (u_A + x_A \cdot v_A) \bmod q$
$B:$	$s_B \leftarrow (x_A \oplus \pi)^{r_B}$
	$u_B \leftarrow \mathcal{H}_1(01, A, B, x_A \oplus \pi, s_B, \pi)$
	$v_B \leftarrow \mathcal{H}_1(10, A, B, x_A \oplus \pi, s_B, \pi)$
	$t_B \leftarrow (u_B + x_B \cdot v_B) \bmod q$
$A \rightarrow B$:	t_A
$B \rightarrow A$:	t_B
$A:$	$u_A \leftarrow \mathcal{H}_1(01, A, B, g^{r_A}, s_A, \pi)$
	$v_A \leftarrow \mathcal{H}_1(10, A, B, g^{r_A}, s_A, \pi)$
	$t_B \cdot (u_A + x_A \cdot v_A)^{-1} \overset{?}{\equiv} 1 \bmod q$
	if `true` then `accept` else `reject`
	$k_A \leftarrow \mathcal{H}_2(A, B, x_A, x_B, s_A, \pi)$
$B:$	$u_B \leftarrow \mathcal{H}_1(01, A, B, g^{r_B}, s_B, \pi)$
	$v_B \leftarrow \mathcal{H}_1(10, A, B, g^{r_B}, s_B, \pi)$
	$t_A \cdot (u_B + x_B \cdot v_B)^{-1} \overset{?}{\equiv} 1 \bmod q$
	if `true` then `accept` else `reject`
	$k_B \leftarrow \mathcal{H}_2(A, B, x_A, x_B, s_B, \pi)$

Figure 1. Description of protocol DH-BPAKE1 using $\text{MAC}_{\langle u,v\rangle}$

On-line computation for each principal requires two exponentiations (e.g. for A: x_A, s_A), one modular inversion and two multiplications (in $\mathbb{Z}_q^*$) and five evaluations of the hash functions $\mathcal{H}_1, \mathcal{H}_2$. The computational complexity is reduced if pre-computation is possible since the exponentiations can be computed off-line.

The specification of protocol DH-BPAKE1 using MAC_π is described in Figure 2. For this protocol on-line computation for each principal again requires two exponentiations, one modular inversion and one multiplication (in $\mathbb{Z}_q^*$) and only three evaluations of the hash functions $\mathcal{H}_1, \mathcal{H}_2$.

IV. SECURITY ANALYSIS

A. Security of the MACs

In this section we examine with greater detail both the message authentication schemes introduced earlier since the security of protocol DH-BPAKE1 depends on the properties of these primitives. We start with the definition of a secure MAC.

Definition 1 (Secure MAC): A message authentication scheme is (t, ϵ)-secure if, for all adversaries $\mathcal{A}$ running in time bounded by t, the following holds:

$$\Pr\left[\begin{array}{l}\kappa \overset{R}{\leftarrow} \{0,1\}^\ell; \\ \langle m, \tau\rangle \leftarrow \mathcal{A}^{\mathit{tag}_\kappa(\cdot)}: \\ \mathit{ver}_\kappa(m,\tau) = 1 \wedge m \notin \mathcal{M}\end{array}\right] < \epsilon$$

where $\mathcal{M}$ is the set of messages that $\mathcal{A}$ submitted to its oracle $\mathit{tag}_\kappa(\cdot)$.

As far as perfect secrecy is concerned, it is impossible to achieve a zero-value for the above probability. This is due to the fact that for any MAC the adversary can always guess a random key $\kappa' \overset{R}{\leftarrow} \{0,1\}^\ell$ and output $\langle m, \mathit{tag}_{\kappa'}(m)\rangle$ without querying its oracle. It is straightforward to verify that the success probability of such an adversary is at least $1/2^\ell$. Furthermore, it is also possible for the adversary to guess a random string $\tau' \overset{R}{\leftarrow} \{0,1\}^n$ and output $\langle m, \tau'\rangle$ for any valid m in the message space; the probability that τ' is a valid tag is at least $1/2^n$. Summing this all up, we have that for any MAC with key length ℓ and tag length n, an adversary can always forge a valid message/tag pair with probability at least $\max\{1/2^\ell, 1/2^n\}$.

© 2006 ACADEMY PUBLISHER

$A, B:$	π
$A:$	$r_A \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$
	$x_A \leftarrow g^{r_A} \oplus \pi$
$B:$	$r_B \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$
	$x_B \leftarrow g^{r_B} \oplus \pi$
$A \rightarrow B:$	x_A
$B \rightarrow A:$	x_B
$A:$	$s_A \leftarrow (x_B \oplus \pi)^{r_A}$
	$u_A \leftarrow \mathcal{H}_1(A, B, x_B \oplus \pi, s_A, \pi)$
	$t_A \leftarrow (u_A + x_A)^{-1} \mod q$
$B:$	$s_B \leftarrow (x_A \oplus \pi)^{r_B}$
	$u_B \leftarrow \mathcal{H}_1(A, B, x_A \oplus \pi, s_B, \pi)$
	$t_B \leftarrow (u_B + x_B)^{-1} \mod q$
$A \rightarrow B:$	t_A
$B \rightarrow A:$	t_B
$A:$	$u_A \leftarrow \mathcal{H}_1(01, A, B, g^{r_A}, s_A, \pi)$
	$t_B \cdot (u_A + x_A) \stackrel{?}{\equiv} 1 \mod q$
	if `true` then `accept` else `reject`
	$k_A \leftarrow \mathcal{H}_2(A, B, x_A, x_B, s_A, \pi)$
$B:$	$u_B \leftarrow \mathcal{H}_1(01, A, B, g^{r_B}, s_B, \pi)$
	$t_A \cdot (u_B + x_B) \stackrel{?}{\equiv} 1 \mod q$
	if `true` then `accept` else `reject`
	$k_B \leftarrow \mathcal{H}_2(A, B, x_A, x_B, s_B, \pi)$

Figure 2. Description of protocol DH-BPAKE1 using MAC_π

However, since an unbounded adversary that can ask infinitely many queries to its oracle can easily forge a tag, a meaningful notion of security requires a limit on the number of queries. This leads to the following definition:

Definition 2 (Secure MAC): A message authentication scheme is *ϵ-secure after q tags* if, for all adversaries $\mathcal{A}$, keys κ and messages m the following holds:

$$\Pr[\mathit{tag}_\kappa(m) = \tau | \mathit{tag}_\kappa(m_1) = \tau_1, \ldots, \mathit{tag}_\kappa(m_q) = \tau_q] < \epsilon$$

where m_i are the messages that $\mathcal{A}$ submitted to its oracle $\mathit{tag}_\kappa(\cdot)$.

Armed with the above notions we now prove the following theorem ([11]).

Theorem 1: The keyed message authentication scheme $\text{MAC}_{\langle u,v \rangle}$ is *$1/q$-secure after a single tag*. Concretely, $\Pr[\text{MAC}_{\langle u,v \rangle}(m') = \tau' | \text{MAC}_{\langle u,v \rangle}(m) = \tau, m' \neq m, \tau' \neq \tau] \leq 1/q$.

Proof: Consider arbitrary m, τ such that $\text{MAC}_{\langle u,v \rangle}(m) = \tau$; there are exactly q pairs $u, v \in \mathbb{Z}_q$ satisfying the equation $u + m \cdot v = \tau$ since for every choice of either u or v there is one and only one corresponding value of the other variable, therefore $\Pr[\text{MAC}_{\langle u,v \rangle}(m) = \tau] = q/q^2 = 1/q$.

Since $\Pr[\text{MAC}_{\langle u,v \rangle}(m) = \tau] \neq 0$, by definition of conditional probability we have

$$\Pr[\text{MAC}_{\langle u,v \rangle}(m') = \tau' | \text{MAC}_{\langle u,v \rangle}(m) = \tau] = \frac{\Pr[\text{MAC}_{\langle u,v \rangle}(m') = \tau' \wedge \text{MAC}_{\langle u,v \rangle}(m) = \tau]}{\Pr[\text{MAC}_{\langle u,v \rangle}(m) = \tau]}$$

Now consider $\langle m', \tau' \rangle \neq \langle m, \tau \rangle$, it is straightforward to verify that a single pair u, v exists such that $\text{MAC}_{\langle u,v \rangle}(m') = \tau' \wedge \text{MAC}_{\langle u,v \rangle}(m) = \tau$ or equivalently, that the system of two equations $u + m \cdot v = \tau \wedge u + m' \cdot v = \tau'$ has only one admissable solution. Therefore, $\Pr[\text{MAC}_{\langle u,v \rangle}(m') = \tau' \wedge \text{MAC}_\pi(m) = \tau] = 1/q^2$.

Using the above results we conclude that $\Pr[\text{MAC}_{\langle u,v \rangle}(m') = \tau' | \text{MAC}_{\langle u,v \rangle}(m) = \tau] = 1/q$. ■

The security of the MAC_π scheme is derived from the δ-Computational Modular Inversion Hidden Number Problem (δ-CMIHNP) [5] which enunciates that given n known random pairs $(m_i, \text{MSB}_k[(m_i + \pi)^{-1} \mod p])$ there is no efficient algorithm that outputs $(m, \text{MSB}_k[(m + \pi)^{-1} \text{ mod p}])$ with $m \neq m_i$ if $k < \delta \log_2 p$ ($\delta = 1/3$). In other words, if the most k significant bits of m are known, for many random m, then the problem of finding π is conjectured hard if $k < (\log_2 q)/3$. The next theorem immediately follows from the above results.

Theorem 2: The keyed message authentication scheme MAC_π is a $(t, \delta \log_2 p)$-secure under (randomly) known message attacks.

Although the above result is weaker than the standard notion of security for MACs, which requires resistance against an adversary that can adaptively sumbit queries to its oracle, it seems sufficient for the security of protocol DH-BPAKE1. Furthermore, it is an open issue whether such a strong notion is required for the security of the protocol (and in particular for the key confirmation mechanism).

B. Security of protocol DH-BPAKE1

The security of the protocol is analysed in the formal model of Boyko *et al* [6] which extends the model of Shoup [19] to the password-authenticated setting. We introduce additional modifications to account for the full-duplex communication model. In the password-authenticated setting principals share a secret (eventually agreed upon with some out-of-band mechanism); this is the standard "something you know" authentication paradigm. However, as already discussed, the main difficulty arises from the fact that users tends to choose low entropy passwords which are easier to guess with respect to cryptographic keys.

We consider *static corruptions*, meaning that adversaries make their decision as to whom to corrupt independently of the network traffic. As a consequence, the adversary has complete control over a set of corrupted users fixed in advance and these remain so in the game played by the adversary to break the protocol. Through out the paper we will only consider resource-bound adversaries. Furthermore, the adversary is given complete control over the network (this is a plausible hypothesis when considering the Internet or a public wireless network). Therefore, she can intercept, modify, delete, replay, etc all messages traversing the communication links.

The definition of security is essentially identical to the formulation suggested in [19]. We recall it here:

Definition 3 (Secure Key Agreement Protocol): A key agreement protocol Π is secure in the static corruption model if the following two conditions hold:

© 2006 ACADEMY PUBLISHER

a) for every efficient real world adversary $\mathcal{A}$, which faithfully delivers the protocol messages between compatible protocol instances both these instances accept holding the same session key;
b) for every efficient real world adversary $\mathcal{A}$, there exists an efficient ideal world adversary $\mathcal{A}^*$ such that the transcripts $tran_{\mathcal{A}}$ and $tran^*_{\mathcal{A}}$ are computationally indistinguishable.

This definition will become clear below as we precisely explain all the concepts involved (ideal and real worlds, transcripts, simulatability).

C. Security Model

Security is defined via *simulation* as follows. The *ideal world* adversary $\mathcal{A}^*$ can specify user instances that should be "connected" according to some predetermined rules; such instances obtain a common session key, but keys are otherwise uncorrelated and are concealed from the adversary. In the ideal world there is no TTP, nor certificates, signatures, ciphers or even protocol message flows, there is only the abstraction of a *service* a key agreement protocol affords to a higher layer protocol. There also is a *real world* adversary $\mathcal{A}$ which is given certain capabilities in the real network.

Random variables (transcripts) are generated by both the real ($tran_{\mathcal{A}}$) and ideal ($tran_{\mathcal{A}^*}$) world adversaries; these variables log the relevant events as they happen. Security via simulation is defined by requiring that for every real world adversary $\mathcal{A}$, there exists an ideal world adversary $\mathcal{A}^*$ such that $tran_{\mathcal{A}}, tran_{\mathcal{A}^*}$ are computationally indistinguishable.

Because the ideal world adversary is essentially benign, the above notion of simulatability implies that a real world adversary is constrained to behave like the ideal world adversary and therefore the protocol is secure. Indeed, the fundamental intuition in the ideal world is that the key agreement is performed by means of a trusted third party (called *ring master*). The ring master generates all the session keys and delivers them to each user as if there existed a perfectly authenticated and secure communication channel between them. Hence, we expect a real world protocol to be secure if it can be efficiently simulated in the ideal world.

All the capabilities of the adversary in a real execution (deriving from the nature of the protocol or from the environment) are given to the adversary in the ideal world. There is a non negligible probability that an adversary can guess a password and succeed in spoofing a user. As a consequence, the model should explicitly include passwords [6]. Otherwise, it would be necessary to specify the probability of distinguishing real world from ideal world transcripts, given the number of impersonation attempts the real world adversary made (with each attempt corresponding to a guess of the password).

D. Security Model - Ideal World

We now turn to describe the ideal world. There is a set of honest users $\Upsilon \equiv \{U_i | i = 1, 2, \ldots\}$ that will be often referred to by the index "i". A user U_i may initiate multiple protocol runs (user instances) with one or more users. At any given time, the (eventually empty) set $\{I_{ij} | j = 1, 2, \ldots\}$ contains the protocol instances (indexed by j) currently running on U_i's machine. The ring master has access to a random bit string R (called the *environment*), unknown to the adversary, which models information shared by users in a higher level protocol. The ideal world adversary can specify any one of the following operations:

(initialize user,i,ID_i): assigns the identity ID_i to user U_i. ID_i must not have been already assigned to another user. In addition, a password (chosen from the dictionary $\mathcal{D}$) is assigned to $\mathtt{pw}[ID_i, ID_{i'}]$ by the ring master for each user $U_{i'}$. If $U_{i'}$ has already been initialised then set $\mathtt{pw}[ID_i, ID_{i'}] \leftarrow \mathtt{pw}[ID_{i'}, ID_i]$. We make no hypothesis on the distribution of the passwords. The simplest model would have a dictionary $\mathcal{D}$ and let all passwords be chosen uniformly at random from that set. This operation can be applied only to users that have not been yet initialised. It models the out-of-band communication required to setup passwords between users;

(set password,i,ID',$\mathtt{pw}'$): assigns the password $\mathtt{pw}'$, specified by the adversary, to $\mathtt{pw}[ID_i, ID']$. ID' must not have been already assigned to any user and, after this operation, it cannot be assigned to another user (in an initialise user operation). This operation allows the adversary to setup passwords between any user and herself;

(initialize user instance,i,j,PID_{ij}): accepts in input a user instance I_{ij} and the intended communicating partner's identity PID_{ij}. User U_i must have been previously initialised, but not instance I_{ij}. If PID_{ij} is not set to the identity of an initialised user, then we require that a set password operation has been previously performed for i and PID_{ij} (and hence there can be no future initialize user operation with PID_{ij} as the user id);

(test instance password,i,j,$\mathtt{pw}'$): returns whether $\mathtt{pw}' = \mathtt{pw}[ID_i, ID'_i]$ where $\mathtt{pw}'$ is provided by the adversary. This operation may be asked only once for an initialised instance (or on a terminated instance) and if no start session operation has been invoked for the instance. No records are written to $tran_{\mathcal{A}^*}$. The test instance password is supposed to capture active on-line and off-line dictionary attacks against the protocol. Indeed, the unfeasibility of an off-line dictionary attack is modeled by a (passive) adversary not being allowed to issue a test instance password operation on a successful key exchange between honest users. On the other hand, it is impossible to prevent an active adversary to guess the password and participate in a protocol run (e.g. with an honest user); this is modeled by allowing the adversary to ask one test instance password operation per protocol instance;

(abort session,i,j): aborts the execution of an initialised protocol instance I_{ij};

(start session,i,j,i',j',connect): specifies how a session key is generated for the two instances I_{ij} and $I_{i'j'}$. The

© 2006 ACADEMY PUBLISHER

ring master sets $K_{ij}, K_{i'j'}$ both equal to a random value from $\{0,1\}^{\ell_2}$ (where ℓ_2 is a security parameter). This operation is legal if (i) $I_{i'j'}, I_{ij}$ are initialised instances; (ii) $PID_{ij} = ID_{i'}$; (iii) $PID_{i'j'} = ID_i$ and (iv) no test instance password operation has been invoked of both instances;

(start session,i,j,compromise,key): specifies how a session key is generated for a compromised instance I_{ij}. The ring master sets $K_{ij} = key$. This operation is legal provided $PID_{i'j'}$ is not assigned to a user (it follows that this operation cannot be performed on "connected" protocol instances) or if there has been a set password operation involving U_i;

(application,f, $f(R, \{K_{ij}\})$): allows the adversary to obtain any information about the session keys $\{K_{ij}\}$ and the environment R. It models leakage of information on a session key in a real protocol (e.g. using the key to encrypt messages). The efficiently computable function f is specified by the adversary.

(implementation,$comment$): allows the adversary to write a comment to the random variable $tran_{\mathcal{A}^*}$. It is required to make ideal world views that are equivalent to real world views.

In the model of [19] a start session operation with a connect connection assignment requires the existence of a partner instance as the argument of a create connection assignment. However, in the real world an instance that was subject to a create connection assignment may never actually "connect" to its intended partner. This accounts for unilateral (entity) authentication only; needless to say, in peer-to-peer communications mutual authentication of parties is an essential requisite.

We no longer need to specify roles in our model nor does it make sense to distinguish between an initiator instance (the instance that sends the first message) and a responder instance, since the actions performed by each participant in the protocol are completely symmetric and message delivery is asynchronous.

Key authentication is explicit when two principals have established the exact identity of their partner by using specific authentication primitives (e.g. signatures, MACs); therefore it is the authenticated principals that will establish the session key. Conversely, key authentication is implicit when the claimed identities of both principals correspond to their real ones although principals are given no explicit evidence of this fact (but key authentication is implied by both agreeing on the same session key). As we have defined it, the start session operation with a connect connection assignment models both implicit and explicit key authentication.

The approach taken by [6] to account for implicit key authentication is to introduce *dangling* connection assignments. However, observe that (1) after sending a message, an instance can simply abort the protocol execution if it does not receive the expected reply message within a predefined "time-slot". We use timeouts explicitly in our model albeit they are usually implementation specific; (2) an instance that accepts and returns the session key to the higher level protocol that invoked it can simply leave the authentication phase to that protocol (e.g. the calling application could pass session ids resulting from an authentication protocol as arguments to the key agreement protocol). We believe the above conditions can lead to more efficient protocols, since all connection-related resources (e.g. memory) are immediately released. One also gets more secure protocols because not only they are less prone to denial of service attacks but if an instance aborts before accepting, there is no session key to expose.

E. Security Model - Real World

We now turn to describe the real world adversary. A protocol instance I_{ij} is a probabilistic state machine. When running, its state is updated as it receives a message, it then generates a response message and reports its status. The status is assigned any one of the following values: (a) accept, the instance has terminated and has output a session key K_{ij}; (b) reject, the instance abruptly terminates due to some error condition and does not output a session key; (c) continue, the instance is idle, waiting to receive an input message. The real world adversary can specify the following operations:

(initialize user,i,ID_i): assigns the identity ID_i to user U_i. ID_i must not have been already assigned to another user. In addition, a random value from the dictionary $\mathcal{D}$ is assigned to $\mathtt{pw}[ID_i, ID_{i'}]$ by the ring master (for each user i'). This operation can be applied only to users that have not been yet initialised;

(set password,i,ID',$\mathtt{pw}'$): assigns the password $\mathtt{pw}'$, specified by the adversary, to $\mathtt{pw}[ID_i, ID']$. ID' must not have been already assigned to different user and, after this operation, it cannot be assigned to another user (in an initialise user operation);

(initialize user instance,i,j,PID_{ij}): accepts in input a user instance I_{ij} and its intended communicating partner identity PID_{ij}. User U_i must have been previously initialised, but not I_{ij}. If PID_{ij} is not set to the identity of an initialised user, then we require that a *set password* operation has been previously performed for i and PID_{ij} (and hence there can be no future *initialize user* operation with PID_{ij} as the user ID);

(deliver message,i,j,Msg,$status_{ij}$): allows the adversary to deliver message Msg to an initialised user instance I_{ij} and returns $status_{ij}$;

(application,f, $f(R, \{K_{ij}\})$): allows the adversary to obtain any information about the session keys and the environment (R). It is analogous to the corresponding ideal world operation;

(initialize system,1^ℓ, R): as in the ideal world case this operation is added for convenience. It may be issued by the ring master only and allows initialization of the real world (system parameters). It takes in input the random bit string R (a.k.a. the *environment*) of some suitable length (which is not revealed to the adversary). The purpose of

© 2006 ACADEMY PUBLISHER

the environment is to model information shared by users in higher layer protocols.

An additional operation is given to the adversary:

(random oracle,$i, x, \mathcal{H}_i(x)$): the argument-response pairs $x, \mathcal{H}_i(x)$ are stored in a list $\mathcal{H}_i$-list; if x was never queried before the response is a random value, otherwise the response is the value in the list that corresponds to the given argument x. We call *indirect* queries those made to the random oracle by the logic of the protocol (and are therefore hidden to the real world adversary) while *direct* queries are those asked by the adversary. Note that application operations are not allowed access to random oracles, i.e., higher level protocols cannot access the random oracle used in the key agreement protocol.

We now prove the following theorem.

Theorem 3: Protocol DH-BPAKE1 is a secure password-authenticated key agreement protocol in the random oracle model assuming MAC_π is a secure message authentication scheme.

Proof: Condition *a*) of Definition 3 is straightforward to verify. It remains to prove the simulatability requirement (condition *b*) of Definition 3). The idea is to build a simulator $\mathcal{S}$ of the real world protocol so that the transcript resulting from the adversary $\mathcal{A}^*$ attacking the simulator is computationally indistinguishable from the transcript of an adversary $\mathcal{A}$ attacking the real world protocol. Recall that the model uses passwords explicitly generated by the ring master. This implies that in any security proof we need to construct a simulator that knows the actual value of a password. We describe the actions of the simulator for each possible operation of the adversary in the real world.

Let $\text{DH}(X, Y)$ denote the Diffie-Hellman secret g^{xy} computed from $X = g^x$ and $Y = g^y$. Let $\ell = |q|$ be the security parameter. We say that two initialised instances I_{ij} and $I_{i'j'}$ are *compatible* if $PID_{ij} = ID_{i'}$ and $PID_{i'j'} = ID_i$. Let $ID_i = i$, $ID_{i'} = i'$ and $\mathtt{pw}^* = \mathtt{pw}[i, i']$. Let $sid_i, sid_{i'}$ be the concatenation (in lexicographic order) of all messages exchanged by I_{ij} and $I_{i'j'}$ in a protocol run. If $sid_i = sid_{i'}$ we say that two compatible instances I_{ij} and $I_{i'j'}$ have had a *matching conversation*.

The simulator must try to detect guesses of the password made by the real world adversary and transform them into test instance password operations. The simulator "emulates" in the ideal world the operations issued by the real world adversary $\mathcal{A}$ as follows:

(deliver message,$i', j', x_i, status_{i'j'}$): There are the following subcases to consider:

(i) Message x_i was sent by an initialised and compatible instance I_{ij} (with $ID_i = i$, $PID_{ij} = i'$), therefore generate $x_{i'} \xleftarrow{R} \{0,1\}^\ell$ set $status_{i'j'} \leftarrow$ continue and perform operation (deliver message,$i, j, x_{i'}, status_{ij}$) before timeout expiration (the simulation would still be perfect even after timeout expiration since it is handled as in case (v));

(ii) Message x_i was sent by an initialised instance I_{ij} with $ID_i = i$ and PID_{ij} not assigned to a user: perform operation (start session,i', j',compromise,key) where key is the real session key extracted from instance $I_{i'j'}$;

(iii) Message x_i was sent by an initialised instance I_{ij} with $ID_i = i$, and a (set password, i, PID_{ij}, $\mathtt{pw}^*$) operation was called: perform operation (set password,$i, ID_{i'j'}, \mathtt{pw}^*$) where $\mathtt{pw}^*$ is the password extracted from the corresponding real world operation;

(iv) Message x_i was sent by an initialised instance I_{ij} but $ID_i \neq i$ or $PID_{ij} \neq i'$: perform operation (abort session,i', j') (this is exactly what happens in the real world).

(v) Message x_i is not received by $I_{i'j'}$ before timeout expiration: perform an abort operation and write (implementation,"$timeout_{i'j'}$") to $tran_{A^*}$ (this is exactly what happens in the real world);

(deliver message,$i, j, x_{i'}, status_{ij}$): this operation is analogous to the preceding case due to the symmetry of the protocol actions;

(deliver message,$i', j', t_i, status_{i'j'}$): t_i was not forged since it is the tag of a secure MAC; there are two subcases to consider:

(a) $I_{i'j'}$ has had a matching conversation with the compatible instance I_{ij}: messages x_i, t_i are sent by an initialised instance I_{ij} with $ID_i = i$, $PID_{ij} = i'$ and were respectively generated as $x_i \xleftarrow{R} \{0,1\}^\ell$ and $t_i \xleftarrow{R} \mathbb{Z}_q^*$. The value of t_i is computed as the result of the (indirect) query:

$$\mathcal{H}_1(i, i', x_{i'} \oplus \mathtt{pw}^*, \text{DH}(x_i \oplus \mathtt{pw}^*, x_{i'} \oplus \mathtt{pw}^*), \mathtt{pw}^*)$$

(to obtain the value u_i) and since $x_{i'}$ is a freshly chosen random string, t_i will be indistinguishable from the value in the real world. Since there was a matching conversation, $I_{i'j'}$ has sent messages $x_{i'}, t_{i'}$ to I_{ij} where $x_{i'} \xleftarrow{R} \{0,1\}^\ell$, $t_{i'} \xleftarrow{R} \mathbb{Z}_q^*$, therefore perform a (start session,i, j, i', j',connect) operation and set $status_{i'j'} \leftarrow$ accept. At this point the ring master extracts the session key (to obtain the value of $k_i = k_{i'}$) and assigns it to both K_{ij} and $K_{i'j'}$. Since all messages $x_i, t_i, x_{i'}, t_{i'}$ are freshly chosen random elements, it is unlikely that the value assigned to the session key was ever output by the random oracle before, therefore this value will be indistinguishable from the session key in the real world. The connect operation is legal since test password instance operations correspond to impersonation attempts of the real world adversary using the password $\mathtt{pw}^*$. Such attempts require direct queries of the random oracle $\mathcal{H}_1$ at the point:

$$\langle i, i', x_i, x_{i'}, \text{DH}(x_i \oplus \mathtt{pw}^*, x_{i'} \oplus \mathtt{pw}^*), \mathtt{pw}^* \rangle$$

(and the argument-response pairs are recorded in the respective random oracle lists). However, such queries can possibly be made only if I_{ij} and $I_{i'j'}$ are compatible

© 2006 ACADEMY PUBLISHER

instances that have had a matching conversation (since the messages $x_i, t_i, x_{i'}, t_{i'}$ are all freshly chosen random values), therefore it is unlikely that they have been stored in the list $\mathcal{L}_1$ with their respective responses (otherwise $I_{i'j'}$ would not have had a matching conversation with the compatible instance I_{ij});

(b) $I_{i'j'}$ has not had a matching conversation with the compatible instance I_{ij}: there are four subcases to consider:

(i) Message t_i was sent by an initialised instance I_{ij} with $ID_i = i$, PID_{ij} is not assigned to a user and instance $I_{i'j'}$ accepts (in the real world): perform (start session,i', j',compromise,key) where key is the real session key extracted from instance $I_{i'j'}$;

(ii) Message t_i was sent by an initialised instance I_{ij} with $ID_i = i$, and a (set password, i, PID_{ij}, $\mathtt{pw}^*$) operation has been issued (and instance $I_{i'j'}$ accepts): perform operation (set password,$i, ID_{i'j'}, \mathtt{pw}^*$) where $\mathtt{pw}^*$ is the password extracted from the corresponding real world operation;

(iii) Message t_i was sent by an initialised instance I_{ij} but $ID_i \neq i$ or $PID_{ij} \neq i'$: perform operation (abort session,i', j') (this is exactly what happens in the real world).

(iv) Message t_i was not received by $I_{i'j'}$ before timeout expiration: perform an abort operation and write (implementation,"$timeout_{i'j'}$") to $tran_{A^*}$ (this is exactly what happens in the real world);

(deliver message,$i, j, t_{i'}, status_{ij}$): this operation is analogous to the preceding case due to the symmetry of the protocol actions;

(initialize user instance,i, j, PID_{ij}): this operation is simply passed on to the ideal world;

(application,$f, f(R, \{K_{ij}\})$): this operation is simply passed on to the ideal world;

(set password,$i, ID', \mathtt{pw}'$) store $\mathtt{pw}'$ and answer as usual (the operation is passed on to the ideal world);

(random oracle,1,$\langle i, i', x_{i'} \oplus \mathtt{pw}^*, W, \mathtt{pw}^*\rangle$,$\mathcal{H}_1$): where $W = g^{xy}$ for some random x, y and the argument-response pairs are stored in the list $\mathcal{L}_1$. If the argument was never used before, the response is a random value, otherwise the response is the value in the list that corresponds to the given argument. This operation occurs in the real world transcript $tran_{\mathcal{A}^*}$ only when the two instances I_{ij} and $I_{i'j'}$ are compatible and have had a matching conversation. In this case the response is $k \xleftarrow{R} \{0,1\}^{\ell_2}$ (ℓ_2 is another security parameter polynomially related to ℓ). We argue that the adversary $\mathcal{A}$ has a negligible probability of asking the random oracle $\mathcal{H}_1$ a query in the point $\langle i, i', x_i, x_{i'}, g^{r_i r_{i'}}, \mathtt{pw}^*\rangle$ such that $\langle x_i \oplus \mathtt{pw}^*, x_{i'} \oplus \mathtt{pw}^*, g^{r_i r_{i'}}\rangle$ forms a DH tuple. Indeed, let ask-DH denote the event that such a query is made with non negligible probability $\Pr_{\mathcal{A}}$[ask-DH], then one can build an algorithm $\mathcal{F}$ that solves the CDHP. Algorithm $\mathcal{F}$ is given an instance $(X = g^x, Y = g^y) \in \mathcal{G}$ of the CDHP, and must determine Z=DH(X,Y). $\mathcal{F}$ will "embed" this instance in the conversation of two protocol instances $I_{ij}, I_{i'j'}$ that have had a matching conversation and have exchanged the messages $x_i, x_{i'}$. The algorithm runs as follows:

1) $\mathcal{F}$ is given $(X, Y) \in \mathcal{G}$. It chooses passwords for all parties (also playing the role of ring master);
2) $\mathcal{F}$ will run the simulator, answering operations as follows:
   - Whenever there are (deliver message,$i', j', x_i, status_{i'j'}$) and (deliver message,$i, j, x_{i'}, status_{ij}$) operations and I_{ij} and $I_{i'j'}$ are compatible instances, choose $u \xleftarrow{R} \mathbb{Z}_q^*$, $v \xleftarrow{R} \mathbb{Z}_q^*$ and generate $x_i, x_{i'}$ as $x_i = g^u X \oplus \mathtt{pw}^*, x_{i'} = g^v Y \oplus \mathtt{pw}^*$;
   - Whenever there is a query of the form $\mathcal{H}_1(i, i', U \oplus \mathtt{pw}^*, V \oplus \mathtt{pw}^*, W, \mathtt{pw}^*)$ answer with a random string if it was never asked before otherwise return the previous response. If the query is asked as the result of a matching conversation between two compatible instances I_{ij} and $I_{i'j'}$ with $ID_i = i, ID_{i'} = i'$ the DH tuple (U, V, W) is stored in the list $\mathcal{L}_1$;
   - All other operations are treated as usual (i.e. they are passed on to the simulator);
3) When the simulator ends, $\mathcal{F}$ chooses a random tuple from $\mathcal{L}_1$ (this list is not empty since $\Pr_{\mathcal{A}}$[ask-DH] is non negligible). It finds x, y such that $U = g^x X$ and $V = g^y Y$ and outputs $W/Y^u X^v \mathrm{DH}(U, V)$.

Now, the probability that $\mathcal{F}$ chooses the Diffie-Hellman tuple corresponding to event ask-DH is $1/n_h$ where n_h is an upper bound on the number of queries asked to $\mathcal{H}_1$. Therefore if event ask-DH occurs then $\mathcal{F}$ solves the CDH instance (X, Y) given in input since it outputs the correct value for DH(X,Y). This implies that $\Pr_{\mathcal{A}}$[ask-DH]$\leq n_h \epsilon$.

(random oracle,2,$\langle i, i', x_i, x_{i'}, t_i, t_{i'}, W, \mathtt{pw}^*\rangle$,$\mathcal{H}_2$): this case is treated similarly to the preceding one. ■

We also enunciate the following result.

Theorem 4: Protocol DH-BPAKE1 is a secure password-authenticated key agreement protocol in the random oracle model assuming $\mathrm{MAC}_{\langle u,v\rangle}$ is a secure message authentication scheme.

The proof of this theorem is analogous to the proof of Theorem 3.

V. Conclusions and Future Work

In this paper we have presented a provably secure password-authenticated key agreement protocol. Its main features are an optimal number of rounds and a reduced computational complexity. These properties make it ideal for mobile devices. Another attractive feature is that users need only share a password to establish a session key. As a result, two users can securely exchange data using, for example, a local wireless network technology (e.g. Bluetooth) in a public area (e.g. airport) by simply agreeing on a password on-the-fly. The subject of ongoing work is the development of a working prototype of the

© 2006 ACADEMY PUBLISHER

protocol for smartphones to verify the effectiveness of our results.

ACKNOWLEDGMENTS

The author is grateful to the anonymous reviewers for their detailed comments and helpful suggestions.

REFERENCES

[1] M. Bellare and P. Rogaway. The AuthA protocol for password-based authenticated key exchange. *Contribution to IEEE P1363*, 2000.

[2] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password based protocols secure against dictionary attacks. *In Proceedings IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[3] S. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. *In Proceedings of the 1st ACM on Computer and Communications Security*, pages 244–250, 1993.

[4] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. *In Proceedings of the 6th IMA Int.l Conf on Cryptography and Coding*, LNCS 1355:30–45, 1997.

[5] D. Boneh, S. Halevi, and N. Howgrave-Graham. The modular inversion hidden number problem. *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, LNCS 2248:36–51, 2001.

[6] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password authenticated key exchange using Diffie-Hellmann. *In Proceedings of EUROCRYPT 2000*, LNCS 1807:156–171, 2000.

[7] E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. *In Proc. 7th PKC'04*, LNCS 2947:145–158, 2004.

[8] L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[9] D. Jablon. Strong Password-only Authenticated Key Exchange. *SIGCOMM Computer Communication Review*, pages 26(5):5–26, 1996.

[10] D. Jablon. Extended Password-only Key Exchange immune to Dictionary Attacks. *In Proceedings of the WEWTICE'97 Workshop on Enterprise Security*, pages 248–255, 1997.

[11] J. Katz. Introduction to Cryptography - Course Lecture Notes. http://www.cs.umd.edu/∼katz/crypto/, 2004.

[12] K. Kobara and H. Imai. Pretty-simple password-authenticated key-exchange under standard assumptions. *IEICE Transactions*, pages 2229–2237, 2002.

[13] T. Kwon. Authentication and key agreement via memorable password. *In Proceedings of NDSS Symposium Conference*, 2001.

[14] C. E. Landwehr. Protecting unattended computers without software. *Proceedings of the 13th Annual Computer Security Application Conference*, pages 274–283, 1997.

[15] A. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[16] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, United Kingdom, 2005.

[17] S. Patel. Information leakage in encrypted key exchange. *In Proceedings of the DIMACS Workshop on Network Threats*, 1997.

[18] S. Patel. Number theoretic attacks on secure password schemes. *In Proceedings IEEE Symposium on Security and Privacy*, pages 236–247, 1997.

[19] V. Shoup. On Formal Models for Secure Key Exchange. Technical Report RZ 3120, IBM Research, 1999.

[20] M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *Operating Systems Review*, 29(3):22–30, 1995.

[21] M. Strangio. An Optimal Round Two-Party Password-Authenticated Key Agreement Protocol. *1st IEEE Proceedings of the ARES06*, 2006.

Maurizio Adriano Strangio recently received a Ph.D. from the University of "Tor Vergata", Rome. He also obtained his MS degree in computer science from the University of Bari, in 1989.

His current research interests include cryptography, computer forensics and security of distributed systems. He is also a member of the IEEE Computer Society and of the ACM.

© 2006 ACADEMY PUBLISHER