Password-Authenticated Key Exchange Based on RSA

Philip MacKenzie[1], Sarvar Patel[1], and Ram Swaminathan[2]

[1] Bell Laboratories, Lucent Technologies,
{philmac, sarvar}@lucent.com
[2] Hewlett-Packard Research Laboratories,
swaram@godel.hpl.hp.com

Abstract. There have been many proposals in recent years for password-authenticated key exchange protocols. Many of these have been shown to be insecure, and the only ones that seemed likely to be proven secure (against active adversaries who may attempt to perform off-line dictionary attacks against the password) were based on the *Diffie-Hellman* problem. In fact, some protocols based on Diffie-Hellman have been recently proven secure in the random-oracle model. We examine how to design a provably-secure password-authenticated key exchange protocol based on *RSA*. We first look at the OKE and protected-OKE protocols (both RSA-based) and show that they are insecure. Then we show how to modify the OKE protocol to obtain a password-authenticated key exchange protocol that can be proven secure (in the random oracle model). The resulting protocol is very practical; in fact the basic protocol requires about the same amount of computation as the Diffie-Hellman-based protocols or the well-known `ssh` protocol.

1 Introduction

Consider the following scenario: Alice and Bob share a short secret (say, a 4 digit PIN number or a 6 character password) that they wish to use to identify and authenticate each other over an insecure network (say, the Internet). They do not carry any other information with them. Of course, neither wants to reveal the secret to the other until the other has revealed his/her own knowledge of the secret. In fact, neither wants to reveal anything that could be used to verify the secret (such as a one-way function applied to the secret) since the secret can then be found by anyone using a dictionary attack (by simply iterating through the relatively small number of possible secrets, applying the one-way function to each of them, and comparing each result to the transmitted value). So how do Alice and Bob authenticate themselves? In general, Alice and Bob will want to not only authenticate themselves, but set up a secure channel between themselves. For this they need a cryptographically strong shared *session key*. So a variation of the question above would be: how do Alice and Bob bootstrap a short secret into a secure strong secret?

This problem, which we call *password-authenticated key exchange*, was first proposed in Bellovin and Merritt [BM92]. In that paper, the *Encrypted Key*

T. Okamoto (Ed.): ASIACRYPT 2000, LNCS 1976, pp. 599–613, 2000.

Exchange (EKE) protocol was proposed as a solution. The problem has since been studied extensively [BM93, GLNS93, Gon95, Jab96, Jab97, Luc97, STW95, Wu98], but only two recent papers [BPR00, BMP00] present protocols along with proofs of security, and in fact, many of the previously-proposed protocols have been shown to be insecure [Ble99, Pat97]. Both of the protocols that were proven secure were based on Diffie-Hellman. Specifically, [BPR00] developed a clean and elegant protocol based on EKE and proved its security based on Computational Diffie-Hellman (CDH), using the random oracle and ideal symmetric encryption function assumptions. The protocol in [BMP00] is similar, but with the proof of security based on Decisional Diffie-Hellman (DDH), using only the random oracle assumption.

1.1 Overview of Our Results

We study password-authenticated key exchange protocols based on RSA. We first look at the OKE (Open Key Exchange) and protected-OKE protocols of Lucks [Luc97], since they are the first ones that were based on RSA and were claimed to have proofs of security. We show that in fact they are insecure. Then we show how to modify the OKE protocol to obtain a protocol that we prove to be secure. This new protocol requires only 4 moves, and only one public-key operation (i.e., a modular exponentiation) per side (either encryption or decryption). Thus it is efficient enough to be used in practice, e.g., for securing remote user access to a server, and is roughly as efficient as the other Diffie-Hellman-based protocols or the ssh protocol, all of which require two exponentiations per side.[1]

In this scenario, it is actually useful for the server to store only some verification information for the password (such as a one-way function applied to the password) but not the password itself. This provides resilience to server compromise, meaning that an adversary that compromises the server and steals the password information is still not able to impersonate a user, unless the adversary actually performs a dictionary attack on the verification information. We show how to extend our protocol to provide some resilience to server compromise, but due to space limitations, we omit the full proof of security for this extended protocol.

The proposals presented in this paper have been presented in informal settings under the names SNAPI (Secure Network Authentication with Password Information) and SNAPI-X. To avoid confusion, we will continue to use those names here.

1.2 Security Model and Definitions

What does it mean for a password-authenticated key exchange protocol to be secure? Informally, it means that the probability that an adversary can successfully authenticate itself is at most negligibly more than that of an adversary

[1] It is difficult to do more than rough comparisons, since modulus size and exponent size may vary among the different protocols.

who runs a trivial attack of simply iteratively guessing passwords and running the authentication protocol (i.e., attempting to login). In SNAPI, we specifically show that if the adversary can do non-negligibly better than this trivial attack, then one can break RSA [RSA78]. We use the random-oracle model [BR93a] for our proofs. While a protocol having a security proof in the random-oracle model is certainly less desirable than a protocol having a proof in the standard model (using standard cryptographic assumptions) [CGH98], it is certainly preferable over a protocol which lacks any proof. Other techniques proven secure in the random-oracle model include Optimal Asymmetric Encryption Padding [BR94] (used in PKCS #1 v. 2 [Not99]) and Provably Secure Signatures [BR96].

For our proofs we use the security model for password-authenticated key exchange from [BMP00], in which the adversary totally controls the network, a la [BR93b], and which is based on the multi-party simulatability paradigm as described in [Bea91, BCK98, Sho99]. In this paradigm, security is defined using an ideal system, which describes the service (of key exchange) that is to be provided, and a real system, which describes the world in which the protocol participants and adversaries work. The ideal system should be defined such that an "ideal world adversary" cannot (by definition) break the security. Then, intuitively, a proof of security would show that anything an adversary can do in the real system can also be done in the ideal system, and thus it would follow that the protocol is secure in the real system.

Although it is *not* a password-only protocol, we do point out that the (one-way) authentication protocol given in Halevi and Krawczyk [HK98] is the first password-based authentication protocol to be formally proven secure, with standard security assumptions. The proof methods in this paper are significantly influenced by their techniques. Boyarsky [Boy99] has recently discussed enhancements to the protocol of Halevi and Krawczyk to make it secure in the multi-user scenario.

We note that basic shared-secret authentication protocols (e.g., [BR93b]) are not secure when the parties share *short secrets*. However, there is a similarity between basic authentication and password-based authentication: both seem to be very difficult to get correct, and many protocols have been published for both, which have subsequently been broken. This is precisely the reason why we emphasize *provable security* in this paper.

2 Attack on the RSA Open Key Exchange Protocol

Interest in developing RSA-based password-authenticated key exchange protocols has been strong [Luc97, RCW98] ever since Bellovin and Merritt first described the RSA-EKE protocol [BM92], their RSA version of the Encrypted Key Exchange protocol. However, the use of RSA in password-only protocols has proven to be quite tricky. Many of the RSA-based password-authenticated key exchange protocols have been shown to be insecure [Ble99, Pat97]. A different approach from the EKE protocols was used by Lucks [Luc97] to propose an RSA-based protocol which has so far resisted attacks. In this section, we present

an efficient attack against this RSA-based protocol. In later sections, we modify the basic protocol in [Luc97] to obtain an RSA-based password-authenticated key exchange protocol that can be proven secure.

Lucks presented two protocols: the Open Key Exchange (OKE) protocol and the protected-OKE protocol, both described in terms of generic encryption functions with certain properties. When instantiated with RSA, the OKE protocol has a problem, as noted by Lucks, which can allow an attacker to recover the password. Hence, Lucks modified the basic OKE protocol to create the protected-OKE protocol that was supposed to be secure when the encryption function was instantiated with RSA. We present the basic steps of the RSA versions of the OKE and protected-OKE protocols along with our attack. Details of OKE and protected-OKE can be found in [Luc97]. We first describe OKE:

Step A Alice and Bob agree on a common secret π. Alice in advance generates an RSA key pair $((e, N), (d, N))$.

Step B Alice chooses a random m and sends (e, N) and m to Bob.

Step C Bob chooses a random μ and then a random a from $\mathbb{Z}_N^*$, computes $p = H(e|N|m|\mu|\pi)$ and $q = E(a) \diamond p$ and sends μ and q to Alice. H is a random function with range $\mathbb{Z}_N^*$, E is the RSA encryption function and $\diamond$ is the RSA multiplication operation.

Step D Alice computes p like Bob, and recovers a by performing an RSA decryption of $q \diamond p^{-1}$.

The remaining authentication and key generation steps are omitted because they are not needed for the attack. Lucks noted that there is a problem with this scheme because when an attacker can choose (e, N) such that function E may not be invertible, $E(a)$ would not be uniformly distributed. Hence some information about p will be leaked which would be useful in ruling out candidate values for p and π. This lead Lucks to propose protected OKE which changes step C to step C$'$ below:

Step C$'$ Bob chooses $a \in_R Z_N^*$, but instead of μ, Bob chooses 2 values μ_{-1}, $\mu_0 \in_R \mathbb{Z}_N^*$. Bob uses $\mu_i = E(\mu_{i-2} \diamond H'(\mu_{i-1}))$ to compute $\mu_1, \mu_2, ..., \mu_K$ where H' is another random function mapping to $\mathbb{Z}_N^*$. The value $p = H(e|n|m|\mu_{-1}|\mu_0|\pi)$ is computed, and $q = E(a) \diamond p$ is sent to Alice along with the last two values μ_{K-1} and μ_K.

Lucks reasoned that if E is not invertible then there are at least two choices for μ_{K-2}, and then for every choice of μ_{K-2} there are two choices for μ_{K-3} and so on. Thus we expect 2^K choices for μ_{-1}. For suitably large values of K, say 80, it would be infeasible for the adversary to evaluate all possible p; so the information leaked about p from $q = E(a) \diamond p$ is of no use to the adversary. Unfortunately, RSA protected OKE has a weakness and we present an example attack:

Step 1 The attacker picks e and N such that e is 3, N is a large prime, and $3|N - 1$ and then sends m and (e, N) to Bob.

Step 2 Unwittingly, Bob calculates $p = H(e|N|m|\mu_{-1}|\mu_0|\pi)$ and sends out q, μ_{K-1} and μ_K.

Step 3 We will now uniquely recover values up to μ_2 and μ_1 using some basic results from number theory by showing how to recover μ_{i-2} from μ_{i-1} and μ_i. We make a note that we can efficiently find dth roots if $d|N-1$ [BS96]. We thus decrypt $\mu_i = E(\mu_{i-2} \diamond H'(\mu_{i-1}))$ by solving for the three cubic roots of μ_i. Then we multiply each root with $(H'(\mu_{i-1}))^{-1}$ to get three possible solutions for μ_{i-2}. Of the three possible solutions, only one will be a cubic residue. We know apriori that the correct value for μ_{i-2} will be a cubic residue because μ_{i-2} was formed by encrypting (i.e., cubing): $\mu_{i-2} = E(\mu_{i-4} \diamond H'(\mu_{i-3}))$. Of the three possible values, we can identify the cubic residue because ${\mu_{i-2}}^{\frac{N-1}{3}} \equiv 1 \bmod N$. We continue to recover the rest of the μ_i values until μ_1.

Step 4 μ_0 and μ_{-1} are random values and thus cannot be uniquely recovered using Step 3. There will be three possible values for μ_0 and for each μ_0 value there will be three possible values for μ_{-1}. Hence there are nine possible (μ_0, μ_{-1}) pairs. We will now try to eliminate some candidate passwords from the list of possible password for Bob. If the password is guessed correctly and the (μ_0, μ_{-1}) pair is correct then solving for $E(a)$ from $q = E(a) \diamond p$ will result in an $E(a)$ which is a cubic residue. Conversely, if the solved $E(a)$ is not a cubic residue, assuming (μ_0, μ_{-1}) pair is correct, then we know the password guess is incorrect. We do not know the correct (μ_0, μ_{-1}) pair, however, if for all 9 pairs the 9 possible solutions for $E(a)$ turn out not to be cubic residues then we can eliminate this password guess.

This will happen with a significant probability and thus we can eliminate a significant portion of the possible passwords. The probability for a given password that a (μ_0, μ_{-1}) pair will be such that the result will be non-cubic residue is equivalent to a random number being a non-cubic residue which is $\frac{2}{3}$. The probability that all 9 (μ_0, μ_{-1}) pairs result in non-cubic residues is $(\frac{2}{3})^9$ which is about 2.5%.

Step 5 We repeat the above procedure (Step 1 - Step 4) eliminating a constant fraction of the remaining passwords in each run, until only one password remains.

It may be tempting to propose blocking this attack by checking for primality of N and rejecting the session if N is prime. Although we have described the example attack using a prime N to keep the presentation simple, we could have done the same steps using a composite $N = pq$ and using the chinese remainder theorem where necessary; we omit the details. The above attack can efficiently discover a user's password after a small number of sessions. One can also try to reduce the probability of the attack's success by requiring e to have only large factors. However, this may still allow some leakage and does not rule out other attacks. Ad hoc countermeasures are not very satisfactory in password-based protocols because every avenue of information leakage has to be blocked. Details matter.

3 Model

For our proofs, we use the model defined in [BMP00], which extends the formal notion of security for key exchange protocols from Shoup [Sho99] to password-authenticated key exchange. We assume the adversary totally controls the network, a la [BR93b].

Briefly, this model is defined using an ideal key exchange system, and a real system in which the protocol participants and adversaries work. The ideal system will be secure by definition, and the idea is to show that anything an adversary can do to our protocol in the real system can also be done in the ideal system, and thus it would follow that the protocol is secure in the real system.

3.1 Ideal System

We assume there is a set of (honest) *users*, indexed $i = 1, 2, \ldots$. Each user i may have several *instances* $j = 1, 2, \ldots$. Then (i, j) refers to a given *user instance*. A user instance (i, j) is told the identity of its partner, i.e., the user it is supposed to connect to (or receive a connection from). An instance is also told its *role* in the session, i.e., whether it is going to *open* itself for connection, or whether it is going to *connect* to another instance.

There is also an *adversary* that may perform certain operations, and a *ring master* that handles these operations by generating certain random variables and enforcing certain global consistency constraints. Some operations result in a record being placed in a *transcript*.

The ring master keeps track of session keys $\{K_{ij}\}$ that are set up among user instances (as will be explained below, the key of an instance is set when that instance starts a session). In addition, the ring master has access to a random bit string R of some agreed-upon length (this string is not revealed to the adversary). We will refer to R as *the environment*. The purpose of the environment is to model information shared by users in higher-level protocols.

We will denote a password shared between users A and B as $\pi[A, B]$.

The adversary may perform the following operations: (1) *initialize user* operation with a new user number i and a new identifier ID_i as parameters; (2) *set password* with a new user number i, a new identifier ID', and a password π as parameters (modeling the adversary creating his own account); (3) *initialize user instance* with parameters including a user instance (i, j), its role, and a user identifier denoting the partner with whom it wants to connect; (4) *terminate user instance* with a user instance (i, j) as a parameter; (5) *test instance password* with a user instance (i, j) and a password guess π as parameters (this query can only be asked once per instance and models the adversary guessing a password and attempting to authenticate herself); (6) *start session* with a user instance (i, j) as a parameter (modeling the user instance successfully connecting to its partner and establishing a random session key; (7) *application* with a function f as parameter, and returning the function f applied to the environment and any session keys that have been established (modeling leakage of session key information in a real protocol through the use of the key in, for example, encryptions

of messages); (8) *implementation*, with a comment as parameter (modeling real world queries that are not needed in the ideal world).

For an adversary $\mathcal{A}^*$, *IdealWorld*($\mathcal{A}^*$) is the random variable denoting the transcript of the adversary's operations.

For a detailed description of the syntax and semantics of the above operations, see [BMP00].

3.2 Real System

In the real system, users and user instances are denoted as in the ideal system. User instances are defined as state machines with implicit access to the user's ID, PID, and password (i.e., user instance (i, j) is given access to $\pi[ID_i, PID_{ij}]$). User instances also have access to private random inputs (i.e., they may be randomized). A user instance starts in some initial state, and may transform its state only when it receives a message. At that point it updates its state, generates a response message, and reports its status, either *continue*, *accept*, or *reject*, with the following meanings:

- *continue*: the user instance is prepared to receive another message.
- *accept*: the user instance (say (i, j)) is finished and has generated a session key K_{ij}.
- *reject*: the user instance is finished, but has not generated a session key.

The adversary may perform the following types of operations: (1) *initialize user* operation as in the ideal system; (2) *set password* operation as in the ideal system; (3) *initialize user instance* as in the ideal system; (4) *deliver message* with an input message m and a user instance (i, j) as parameters, and returning the message output from (i, j) upon receiving m; (5) *random oracle* with the random oracle index i and input value x as parameters, and returning the result of applying random oracle H_i to x; (6) *application* as in the ideal system.

For an adversary $\mathcal{A}$, *RealWorld*($\mathcal{A}$) denotes the transcript of the adversary's operations.

Again, details of these operations can be found in [BMP00].

3.3 Definition of Security

Our definition of security is the same as the one in [Sho99] for key exchange. It requires

1. **completeness**: for any real world adversary that faithfully delivers messages between two user instances with complimentary roles and identities, both user instances accept; and
2. **simulatability**: for every efficient real world adversary $\mathcal{A}$, there exists an efficient ideal world adversary $\mathcal{A}^*$ such that *RealWorld*($\mathcal{A}$) and *IdealWorld*($\mathcal{A}^*$) are computationally indistinguishable.

4 SNAPI

In this section we will start by presenting the definition of RSA and giving a standard and well-accepted version of the RSA security assumption.[2] Then we will present the SNAPI and SNAPI-X protocols.

First we give some preliminary definitions. Let k and ℓ denote our security parameters, where k is the "main" security parameter and can be thought of as a general security parameter for hash functions and secret keys (say 128 or 160 bits), and $\ell > k$ can be thought of as a security parameter for RSA or discrete-log-type public keys (say 1024 bits). Let $\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^n$ the set of binary strings of length n. Let "|" denote the concatenation of bit strings in $\{0,1\}^*$. A real-valued function $\epsilon(n)$ is *negligible* if for every $c > 0$, there exists a $n_c > 0$ such that $\epsilon(n) < 1/n^c$ for all $n > n_c$.

The RSA encryption scheme is generally defined as follows: Let key generator GE define a family of RSA functions to be $(e, d, N) \leftarrow GE(1^\ell)$ such that $N = PQ$, where P and Q are prime numbers. Then, the public key is the pair (e, N) where $\gcd(e, \phi(N)) = 1$ and the order of the group $\phi(N) = (P-1) \cdot (Q-1)$. The encryption function $E : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ is defined by $E(x) \equiv x^e \bmod N$ and the decryption function $D : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ is $D(x) \equiv x^d \bmod N$, where the secret exponent d is chosen such that $ed \equiv 1 \bmod \phi(N)$.

The choice of P, Q, and e is generally left to the implementation, although it is recommended that P and Q be random large primes with about the same bit length (about $\ell/2$ for security parameter ℓ) [IEE98], and for efficiency e is often chosen to be a small prime and with a small number of ones in its binary representation, such as 3, 17, or 65537.

For the security of SNAPI, we make *explicit* requirements on the generation of P, Q, and e, which are well within the scope of the general RSA security recommendations. Specifically, we require that $GE(1^\ell)$ chooses two random primes P and Q from the range $\{2^{\ell/2-1}, \ldots, 2^{\ell/2}\}$ (for convenience, we assume ℓ is a multiple of 2). This implies that $2^{\ell-2} \leq N \leq 2^\ell$. We also require that e be a prime in the range $\{2^\ell + 1, \ldots, 2^{\ell+1}\}$. Note that this guarantees that $\gcd(e, \phi(N)) = 1$. For efficiency in our protocol, a standard value of e for a given security parameter ℓ could be chosen beforehand. This would eliminate the need for a primality test by Bob. (An alternative requirement on e would be that e is a prime, $e \geq \sqrt{N}$ and $(N \bmod e) \not| N$, since this can be checked in (probabilistic) polynomial time, and also implies that $\gcd(e, \phi(N)) = 1$ [Len84].)

Given these requirements on GE, we use the following assumption on RSA:

RSA Security Assumption: Let ℓ be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e, d, N) \leftarrow GE(1^\ell)$). For any probabilistic polynomial-time algorithm A, $\Pr[u^e \equiv w \bmod N : (e, d, N) \leftarrow GE(1^\ell); w \in_R \{0,1\}^\ell; u \leftarrow A(1^\ell, w, e, N)]$ is negligible.

[2] The security of the SNAPI protocol can actually be proven under a slightly more general security assumption. Details are omitted.

4.1 The Protocol

Before the SNAPI protocol starts, two players agree on a common password $\pi \in P$. Let A and B be the identities of the two players, with A playing the role of Alice, and B playing the role of Bob. From this point on, we will refer to A as Alice and B as Bob, except when we must explicitly use their identities.

We assume that Alice has chosen an RSA key pair $((e, N), (d, N))$. In general, Alice would most likely use the same key pair in many sessions, although for perfect forward secrecy, Alice would need to choose a new pair in each session. That is, if Adv discovers the decryption key then Adv can determine all session keys obtained in sessions using that key pair. Obviously, some tradeoffs of security versus efficiency could be performed. Alternatively, the two parties could obtain perfect forward secrecy by computing the session key with a Diffie-Hellman key exchange [DH76] using, for instance, the m and μ values. This, however, would require the Diffie-Hellman assumption for security, along with the RSA assumption. For simplicity, we will assume Alice uses the same encryption/decryption pair for each session, although if Adv impersonates Alice, Adv could use a different one.

Define hash functions $h, h', h'' : \{0,1\}^* \to \{0,1\}^k$ and $H : \{0,1\}^* \to \{0,1\}^\eta$ (where $\eta \geq \ell + k$). We will assume that h, h', h'', and H are independent random functions. Let $S_N = \{p : p \leq 2^\eta - (2^\eta \bmod N) \text{ and } \gcd(p, N) = 1\}$.

The protocol is shown in Figure 1. Alice and Bob exchange random values, and Alice also gives her public key to Bob. They both compute hashes of all of these values, plus the password. Then Bob encrypts a random value a, multiplies it by the hash, and sends it to Alice. Alice can divide the received value by the hash and decrypt the result (using her private key) to obtain a. This value a can be used as a "long" secret for authentication. The idea of why this works is that even if Bob computes hashes corresponding to other passwords, Bob cannot find another value a' whose encryption times the other hash would equal the value sent to Alice, since Bob does not have the private key.[3]

Theorem 1 *The SNAPI protocol is a secure password-authenticated key exchange protocol under the RSA assumption and the random oracle model.*

Proof in appendix.

5 SNAPI-X

We now present a protocol for password-only authentication that is "weakly" resilient to server compromise.[4]

Let g be a *generator* of a cyclic group Ω of size ω superpolynomial in k in which the Diffie-Hellman problem is hard. In the SNAPI-X protocol, we assume

[3] Naturally, *proving* that this is the case is much more difficult.

[4] By "weakly," we mean that our protocol can be proven secure assuming that once the adversary has corrupted the server it does not actually impersonate the server to a real client, perhaps because it is unable to do network address spoofing.

Alice (A) Step 1: Choose $m \in_R \{0,1\}^k$, and send $(A, m, (N, e))$ to Bob.

Bob (B) Step 2: If $m \notin \{0,1\}^k$, $N \notin [2^{\ell-2}, 2^{\ell}]$, $e \notin (2^{\ell}, 2^{\ell+1}]$, or e is not prime, then reject,
Else
1. Choose $\mu \in_R \{0,1\}^k$ and $a \in_R \mathbb{Z}_N^*$.
2. Compute $p = H(N|e|m|\mu|A|B|\pi)$.
3. If $p \notin S_N$ then set $q = a$,
   Else set $q \equiv pa^e \bmod N$.
4. Send (μ, q) to Alice.

Alice (A) Step 3: If $\mu \notin \{0,1\}^k$ or $\gcd(q, N) \neq 1$, then reject,
Else
1. Compute $p = H(N|e|m|\mu|A|B|\pi)$.
2. If $p \notin S_N$ then reject,
   Else, set $a \equiv (q/p)^d \bmod N$ and
   send $r = h(N|e|m|\mu|A|B|q|a)$ to Bob.

Bob (B) Step 4: If $p \notin S_N$ or $r \neq h(N|e|m|\mu|A|B|q|a)$, then reject,
Else, send $t = h'(N|e|m|\mu|A|B|q|a)$ to Alice,
set $K = h''(N|e|m|\mu|A|B|q|a)$, and accept.

Alice (A) Step 5: If $t \neq h'(N|e|m|\mu|A|B|q|a)$, then reject,
Else set $K = h''(N|e|m|\mu|A|B|q|a)$ and accept.

Fig. 1. SNAPI Protocol

there is an initialization in which a client with identity B (whom we will refer to as Bob) chooses a password $\pi \in P$, computes $x = H'(A|B|\pi)$ (where A is the identity of the server, whom we will refer to as Alice) and sends Alice $X = g^x$, which we call the *password verifier*. Alice generates an RSA key pair $((e, N), (d, N))$. After the initialization Bob only needs to remember π. As in SNAPI, we assume that Alice has chosen an RSA key pair $((e, N), (d, N))$.

Define hash functions $h, h', h'' : \{0,1\}^* \to \{0,1\}^k$ and $H, H' : \{0,1\}^* \to \{0,1\}^\eta$ (where $\eta \geq \ell + k$). We will assume that h, h', h'', H, and H' are independent random functions. Let $S_N = \{p : p \leq 2^\eta - (2^\eta \bmod N) \text{ and } \gcd(p, N) = 1\}$.

The protocol is shown in Figure 2. Alice and Bob exchange random values, and Alice also gives her public key to Bob. They both compute hashes of all of these values, plus the password verifier. Then Bob encrypts a random value a, multiplies it by the hash, and sends it to Alice. Alice can divide the received value by the hash and decrypt the result (using her private key) to obtain a. This value a can be used as a "long" secret for authentication. Also, to verify that Bob knows the password and not just the password verifier, a type of "Diffie-Hellman" exchange is used. Alice generates her Diffie-Hellman values randomly, and Bob uses the password verifier along with its discrete log as his value. The secret Diffie-Hellman value can thus be computed by both parties and included in the authentication value sent by Bob. Due to space restrictions, we omit the discussion of the security model and proof, and simply state our theorem.

Bob (B) Step 0: Send B to Alice.

Alice (A) Step 1: Retrieve X from password file for B.
Choose $m \in_R \{0,1\}^k$, and send $(A, m, (N, e))$ to Bob.

Bob (B) Step 2: If $m \notin \{0,1\}^k$, $N \notin [2^{\ell-2}, 2^\ell]$, $e \notin (2^\ell, 2^{\ell+1}]$,
or e is not prime, then reject,
Else
1. Set $x = H'(A|B|\pi)$.
2. Choose $\mu \in_R \{0,1\}^k$ and $a \in_R \mathbb{Z}_N^*$.
3. Compute $p = H(N|e|m|\mu|A|B|g^x)$.
4. If $p \notin S_N$ then set $q = a$, Else set $q \equiv pa^e \bmod N$.
5. Send (μ, q) to Alice.

Alice (A) Step 3: If $\mu \notin \{0,1\}^k$ or $\gcd(q, N) \neq 1$, then reject, Else
1. Compute $p = H(N|e|m|\mu|A|B|X)$.
2. If $p \notin S_N$ then reject, Else,
   (a) Set $a \equiv (q/p)^d \bmod N$
   (b) Choose $\gamma \in \mathbb{Z}_\omega$.
   (c) Set $r = h(N|e|m|\mu|A|B|q|a)$ and $y = g^\gamma$.
   (d) Send (r, y) to Bob.

Bob (B) Step 4: If $p \notin S_N$ or $r \neq h(N|e|m|\mu|A|B|q|a)$, then reject,
Else,
1. Send $t = h'(N|e|m|\mu|A|B|q|a|y^x)$ to Alice.
2. Set $K = h''(N|e|m|\mu|A|B|q|a)$, and accept.

Alice (A) Step 5: If $t \neq h'(N|e|m|\mu|A|B|q|a|X^\gamma)$, then reject,
Else set $K = h''(N|e|m|\mu|A|B|q|a)$ and accept.

Fig. 2. SNAPI-X Protocol

Theorem 2 *The SNAPI-X protocol is a secure password-only authentication and key exchange protocol with weak resilience to server compromise, in the random oracle model under the RSA assumption and assuming the hardness of Decision Diffie-Hellman.*

Acknowledgements

We thank Daniel Bleichenbacher for helpful and stimulating discussions, and for showing us how our protocol can remain secure with a shorter RSA public exponent e (our alternative requirement for e in Section 4). We also thank Victor Boyko for his thorough reading of the paper and helpful comments.

References

[BCK98] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In STOC'98 [STO98], pages 419–428.

[Bea91] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[Ble99] D. Bleichenbacher, 1999. Personal Communication.

[BM92] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[BM93] S. M. Bellovin and M. Merritt. Augumented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In CCS'93 [CCS93], pages 244–250.

[BMP00] V. Boyko, P. MacKenzie, and S. Patel. Provably-secure password authentication and key exchange using Diffie-Hellman. In EUROCRYPT2000 [EUR00].

[Boy99] M. Boyarsky. Public-key cryptography and password protocols: The multi-user case. In CCS'99 [CCS99], pages 63–72.

[BPR00] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In EUROCRYPT2000 [EUR00].

[BR93a] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In CCS'93 [CCS93], pages 62–73.

[BR93b] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO '93*, *LNCS* vol. 773, pages 232–249. Springer-Verlag, August 1993.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT 94*, *LNCS* vol. 950, pages 92–111. Springer-Verlag, May 1994.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In *EUROCRYPT 96*, pages 399–416, 1996.

[BS96] E. Bach and J. Shallit. *Algorithmic Number Theory: Volume 1 Efficient Algorithms*. The MIT Press, Cambridge, Massachusetts, 1996.

[CCS93] *First ACM Conference on Computer and Communications Security*, 1993.

[CCS99] *Sixth ACM Conference on Computer and Communications Security*, 1999.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In STOC'98 [STO98], pages 209–218.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22(6):644–654, 1976.

[EUR00] *Advances in Cryptology—EUROCRYPT '2000*, *LNCS* vol. 1807. Springer-Verlag, 14–18 May 2000.

[GLNS93] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.

[Gon95] L. Gong. Optimal authentication protocols resistant to password guessing attacks. In *Proc. 8th IEEE Computer Security Foundations Workshop*, pages 24–29, 1995.

[HK98] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In *Proceedings of the Fifth Annual Conference on Computer and Communications Security*, pages 122–131, 1998.

[IEE98] IEEE P1363 Annex D/Editorial Contribution 1c: Standard specifications for public-key cryptography, June 1998.

[Jab] D. Jablon. Integrity sciences web site. http://www.IntegritySciences.com.

[Jab96] D. Jablon. Strong password-only authenticated key exchange. *ACM Computer Communication Review, ACM SIGCOMM*, 26(5):5–20, 1996.

[Jab97] D. Jablon. Extended password key exchange protocols immune to dictionary attack. In *WETICE'97 Workshop on Enterprise Security*, 1997.

[Len84] H. W. Lenstra. Divisors in residue classes. *Mathematics of Computation*, 42:331–340, 1984.

[Luc97] Stephan Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proc. Workshop on Security Protocols*, 1997.

[MPS] P. MacKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on rsa. full version.

[Not99] RSA Laboratories Technical Note. PKCS #1, version 2, RSA encryption standard. `http://www.rsa.com/rsalabs/pubs/PKCS/`, 1999.

[Pat97] S. Patel. Number theoretic attacks on secure password schemes. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 236–247, 1997.

[RCW98] M. Roe, B. Christianson, and D. Wheeler. Secure sessions from weak secrets. Technical report, Univ. of Cambridge and Univ. of Hertfordshire, 1998.

[RSA78] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signature and public key cryptosystems. *Comm. of the ACM*, 21:120–126, 1978.

[Sho99] V. Shoup. On formal models for secure key exchange. IBM Research Report RZ 3121, April 1999.

[STO98] *Thirtieth ACM Symposium on Theory of Computing*, May 1998.

[STW95] M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *ACM Operating System Review*, 29:22–30, 1995.

[Wu98] T. Wu. The secure remote password protocol. In *Proc. 1998 Internet Society Network and Distributed System Security Symposium*, pages 97–111, 1998.

[Wu99] T. Wu. A real world analysis of kerberos password security. In *1999 Internet Society Network and Distributed System Security Symposium*, 1999.

A Security of the SNAPI Protocol

The completeness requirement follows directly by inspection. Here we prove that the simulatability requirement holds. The basic technique is essentially that of Shoup [Sho99]. The idea is to create an ideal world adversary $\mathcal{A}^*$ by running the real world adversary $\mathcal{A}$ against a simulated real system, which is built on top of the underlying ideal system. In particular, $\mathcal{A}^*$ (i.e., the simulator combined with $\mathcal{A}$) will behave in the ideal world just like $\mathcal{A}$ behaves in the real world, except that idealized session keys will be used in the real world simulation instead of the actual session keys computed in the real system.

Thus our proof consists of constructing a simulator (that is built on top of an ideal system) for a real system so that the transcript of an adversary attacking the simulator is computationally indistinguishable from the transcript of an adversary attacking the real system. Due to space restrictions we are only able to sketch the simulation. Details may be found in the full version of the paper [MPS]. The difficult part of the simulation is to answer queries to user instances and random oracles that are consistent with the ideal world, but without a priori knowing the passwords.

First we deal with the random oracle queries. Note that the user IDs and nonces allow the simulator to know which conversations they correspond to.

The simulator always answers an H query with the encryption of a known value, which helps in the later simulation.

For an h query, the simulator is able to test whether this corresponds to a password test by encrypting the a value in the query, and then for each H query corresponding to the same conversation, multiplying the encryption of a by the result of that query and testing if the result equals the q value sent in the conversation. If for any H query this test is positive, the simulator must make a test instance password query to the ringmaster in the ideal world. Naturally we must show that the simulator never makes a test instance password query unless the adversary is actively involved in the conversation, and in that case the simulator makes at most one test instance password query. (We sketch the proofs of those below.) If there is no password being tested, or if there is a password being tested and it is incorrect, the simulator simply responds with a random bit string. Otherwise, the simulator responds with a bit string consistent with previous values of the protocol.

For h' and h'' queries, the simulator simply responds with random bit strings, and these will be indistinguishable (details omitted).

Now we deal with user instance queries. In general, they are handled as in the actual protocol, except that the q value is set to a random encryption (not multiplied by the result of an H query, since the password is not known to the simulator), and the authentication values r and t are generated randomly, except when a password test by the adversary is detected (by examing the random oracle queries). If necessary, the simulator makes a test instance password query to the ideal world ringmaster, and responds accordingly. If the simulator detects a matching conversation, i.e., an incoming authentication value was sent by a valid partner using the same nonces, then the simulator accepts the authentication value (even though it cannot actually check it since the simulator does not know the password).

To prove that the adversary never forces the simulator to make a test instance password query for a matching conversation, we assume that the adversary does and break the RSA assumption as follows. We take a challenge RSA key and ciphertext and guess the user A involved in the offending conversation. The simulator sends the challenge RSA key when simulating user A, and for any user instance in a conversation with A sends q equal to a random encryption multiplied by the challenge ciphertext. Then for any random oracle query that tests a password, one can compute the decryption of the ciphertext (using the fact that the output of the H oracle is a value whose decryption is known to the simulator).

To prove that the adversary never forces the simulator to make two test instance password queries for a non-matched conversation with an "Alice" user instance, we assume that the adversary does and break the RSA assumption as follows. We take a challenge RSA key and ciphertext and guess the user A involved in the offending conversation. The simulator sends the challenge RSA key when simulating user A, and for any H query involving user A, flips a coin to decide whether to set the output to the encryption of a known value, or the

encryption of a known value multiplied by the challenge ciphertext. If two h or h' queries are made along with two H queries such that two password tests must be performed, then these correspond to the same q value sent to user A and thus can be related by an equation which allows one to solve for the decryption of the challenge ciphertext, as long as exactly one of the H query outputs included the challenge ciphertext. This happens with probability $\frac{1}{2}$.