IEICE TRANS. FUNDAMENTALS, VOL.E90–A, NO.2 FEBRUARY 2007

PAPER

An Efficient and Leakage-Resilient RSA-Based Authenticated Key Exchange Protocol with Tight Security Reduction*

SeongHan SHIN[†a)], ***Nonmember***, **Kazukuni KOBARA**[†], ***Member***, ***and*** **Hideki IMAI**[†,††], ***Fellow***

SUMMARY Both mutual authentication and generation of session keys can be accomplished by an authenticated key exchange (AKE) protocol. Let us consider the following situation: (1) a client, who communicates with many different servers, remembers only one password and has *insecure* devices (e.g., mobile phones or PDAs) with very-restricted computing power and built-in memory capacity; (2) the counterpart servers have enormous computing power, but they are *not perfectly secure* against various attacks (e.g., virus or hackers); (3) neither PKI (Public Key Infrastructures) nor TRM (Tamper-Resistant Modules) is available. The main goal of this paper is to provide security against the leakage of stored secrets as well as to attain high efficiency on client's side. For those, we propose an efficient and leakage-resilient RSA-based AKE (RSA-AKE) protocol suitable for the above situation whose authenticity is based on password and another secret. In the extended model where an adversary is given access to the stored secret of client, we prove that its security of the RSA-AKE protocol is reduced *tightly* to the RSA one-wayness in the random oracle model. We also show that the RSA-AKE protocol guarantees several security properties (e.g., security of password, multiple sever scenario with only one password, perfect forward secrecy and anonymity). To our best knowledge, the RSA-AKE protocol is the most efficient, in terms of both computation costs of client and communication costs, over the previous AKE protocols of their kind (using password and RSA).

key words: *authenticated key exchange, passwords, on-line and off-line dictionary attacks, RSA, leakage of stored secrets, efficiency, perfect forward secrecy*

1. Introduction

Since the discovery of public-key cryptography by Diffie and Hellman [11], one of the most fundamental research topics is to design a practical and provably secure protocol for realizing secure channels. This kind of protocol is necessary because higher-level protocols are frequently developed and analyzed assuming the existence of secure channels between all parties. In the 2-party setting (e.g., a client and a server), this can be achieved by an authenticated key exchange (AKE) protocol at the end of which the two parties authenticate each other and share a common (and temporal) session key to be used for subsequent cryptographic algorithms.

Manuscript received May 19, 2006.
Manuscript revised September 8, 2006.
Final manuscript received October 30, 2006.

[†]The authors are with the Research Center for Information Security (RCIS), National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, 101-0021 Japan.
[††]The author is with the Chuo University, Tokyo, 112-8551 Japan.

*A preliminary version appeared in [33]. Some mistakes about security proof are corrected in this paper.

a) E-mail: seonghan.shin@aist.go.jp

DOI: 10.1093/ietfec/e90–a.2.474

Mutual authentication typically requires some information to be shared between the communicating parties in advance. The shared information may be the form of high-entropy cryptographic keys: either a secret key that can be used for symmetric-key encryption or message authentication code (e.g., [8], [30]), or public keys (exchanged by the parties, while the corresponding private keys are kept secret) which can be used for public-key encryption or digital signatures (e.g., [2], [12], [22], [30], [36]).

In practice, high-entropy keys may often and commonly be substituted by low-entropy human-memorable passwords chosen from a relatively small size of dictionary (e.g., alphanumerical passwords). Owing to the usability of passwords, password-based AKE protocols have been extensively investigated for a long time where a client remembers a short password and the corresponding server holds the password or its verification data that is used to verify the client's knowledge of the password. However, there exist two major attacks on passwords: on-line and off-line dictionary attacks. The on-line dictionary attack is a series of exhaustive searches for a secret performed on-line, so that an adversary can sieve out possible secret candidates one by one communicating with the target party. In contrast, the off-line dictionary attack is performed off-line massively in parallel where an adversary exhaustively enumerates all possible secret candidates, in an attempt to determine the correct one, by simply guessing a secret and verifying the guessed secret with recorded transcripts of a protocol. While on-line attacks are applicable to all of the password-based protocols equally, they can be prevented by letting a server take appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks can be performed off-line and independently of the server. This results in many password-based protocols insecure or broken (cf. the attacks shown in [1], [4], [25], [26], [37], [39]).

At first sight, it seems paradoxical and more difficult to design a secure AKE protocol for the password-based setting partly because that has to "bootstrap" from a weak shared secret to a strong one. For that, Bellovin and Merritt opened the door by showing that a combination of symmetric and asymmetric (public-key) cryptographic techniques can provide insufficient information for an adversary to verify a guessed password and thus defeats off-line dictionary attacks [4]. By asymmetric cryptographic techniques, we can roughly classify AKE protocols to two categories: authenticated key agreement (e.g., incorporating the Diffie-

Hellman protocol) and authenticated key transport (e.g., using RSA) ones. In the next section, we revisit the previous AKE protocols, using password and RSA, from a point of view of which kind of information is needed for authentication.

1.1 Previous Works

Bellovin and Merritt [4] discussed the problem of off-line dictionary attacks and first proposed a set of protocols for Encrypted Key Exchange (including the RSA-based one) which have formed the basis for what we call Password-Authenticated Key Exchange (PAKE) protocols[†]. In PAKE protocols, a client is required to remember his/her password only (without any device) whereas the counterpart server has its verification data that should be stored securely. In other words, if the stored secret (or, password verification data) of the server is leaked out, the password eventually can be retrieved through off-line dictionary attacks, simply by verifying password candidates one by one using the leaked secret [3]. In particular, RSA-based PAKE protocols should be carefully designed to verify whether a server's RSA public key (e, N) is correctly generated or not (i.e., $(e, \varphi(N)) = 1$, see Definition 4 in Sect. 2). This fosters so-called e-residue attacks, a special type of off-line dictionary attacks, as noted in [4]. Until now, three approaches have been taken to thwart e-residue attacks. The first is to use a challenge-response method where a client can verify e interactively with a server based on the fact that for odd integer N and e ($e \geq 3$) such that $\gcd(e, \varphi(N)) \neq 1$ any e-residue modulo N should have at least three e-th roots (e.g., [4], [10], [34], [39]). As a second approach, Zhang fully exploits an algebraic property to ensure that e is relatively prime to $\varphi(N)$ [38]. The third is that Mackenize et al., mandated e to be a prime larger than N in their SNAPI protocol [25]. These approaches may render RSA-based PAKE protocols impractical in terms of computation costs of client (see Table 3). Note that, if a generated RSA key pair is used t ($t \geq 2$) times, such protocols cannot preserve perfect forward secrecy.

In contrast to the PAKE protocols, Lomas et al., proposed AKE protocols, with heuristic discussion of resistance to off-line dictionary attacks, where a client remembers his/her password and holds a server's public key in advance whereas the corresponding server has password verification data and its private key both of which should be stored securely [16], [24]. This type of AKE protocols were further studied by Gong [13], Halevi and Krawczyk [17], [18] gave formal definitions and rigorous proofs of security in this setting, and extensions to the multi-user case were later presented by Boyarsky [5]. Very recently, Kolesnikov et al., pointed out a subtle flaw in the Halevi and Krawczyk's protocol and then generalized the model by introducing another shared secret (i.e., MAC key) in addition to password and (public, private) key pair [23]. However, the leakage of one of the stored secrets (i.e., either the verification data or the private key) may cause a serious problem enough to break its security of the AKE protocol. For example, the leakage of the verification data makes it possible for an adversary to retrieve the password through off-line dictionary attacks and thus to impersonate the client (except [23]). With the leaked private key, an adversary can easily get the password by decrypting ciphertexts from the client. Additionally, these protocols do not provide perfect forward secrecy (see Table 2) without incorporating the Diffie-Hellman protocol. The interesting point is that this type of AKE protocols no longer suffer from e-residue attacks, discussed above, at the expense of storing the server's public key beforehand.

Other AKE protocols based on PKI (Public Key Infrastructures) can be found in SSL/TLS [14], [20] and SSH [19] where a client remembers his/her password and holds a server's public key whereas the corresponding server has password verification data and its private key both of which should be stored securely. The main difference from the above AKE protocols lies in that the parties first establish a secure channel with the server's public key and then the password is transmitted through the channel. Note that, before running the actual protocol, the client must verify the server's certificate via CRL (Certificate Revocation Lists) or OCSP (Online Certificate Status Protocol) which entails extra computation and communication costs. As for the leakage of the stored secrets and e-residue attacks, the same discussions of the above paragraph can be done.

1.2 Motivation

The motivation of this paper is that all of the previous AKE protocols, based on password and RSA, do not provide (1) leakage-resilience of stored secrets (2) perfect forward secrecy and (3) efficiency *at the same time*.

The leakage of stored secrets is a more practical risk than breaking a well-studied cryptographic hard problem. For example, client's devices storing secrets inside can be lost or stolen due to the holder's carelessness and an adversary (or a bad server administrator) can intrude into the server by exploiting bugs or mis-configurations of the system so as to obtain the stored secrets for wrongdoing. Each case makes the corresponding protocols insecure (see Sect. 5.1 and Table 1). One may think of TRM (Tamper-Resistant Modules) as a solution to reduce possibility of the leakage. However, TRM cannot prevent the damage caused by the leakage and it is still hard to make perfect TRM with low costs.

Here one question arises: why the leakage of stored secrets is so important in the password-based AKE protocols? The answer is that the previous AKE protocols may suffer from the leakage of stored secrets that will result in a serious catastrophe in the following multiple server scenario. Let us think of an ordinary client who would have access to a lot of different servers (e.g., Web mail server, Internet shopping mall, Internet bank, FTP servers and so on) each of which provides an independent service and requests the client to register a unique password for authentication. If

[†]Such protocols are in standardization of IEEE P1363.2.

the client registered the same password to many servers, either an adversary or a dishonest server administrator who finds out the password with the leaked stored secrets from one server can impersonate the client to the other remaining servers! In the real world, it is doubtful that all of the clients really register distinct passwords corresponding to servers and remember them without confusing. In such a scenario, it seems rather reasonable for a client to remember only one (relatively short) secret since ordinary people cannot remember distinct long secrets. This motivates us to design a secure AKE protocol against a more powerful active adversary who additionally can get some stored secrets from client and server.

The other motivation comes from the fact that some AKE protocols (e.g., [10], [25], [38]) are inefficient but provide perfect forward secrecy while others (e.g., [17], [23]) are efficient without perfect forward secrecy (see Table 2 and 3). Of course, the Diffie-Hellman protocol can be incorporated into AKE protocols (e.g., SSL/TLS and SSH) for perfect forward secrecy. However, when it comes to lower-power computing devices, RSA-based AKE protocols would be preferable to the Diffie-Hellman based ones since for computing one modular exponentiation the latter requires an algorithm that has cubic running time in average in the bit-length of its inputs, on the other hand, in the former its running time is around quadratic. In particular, a small RSA encryption exponent (e.g., $e = 3$ or $2^{16} + 1$) drastically decrease its computation costs which can be a good candidate for improvements of efficiency. Note that efficiency is crucial in the password-based protocols, which are motivated by practical applications.

1.3 Overview of Our Contributions

Let us consider the following situation for unbalanced wireless networks where a client holds some *insecure* devices (e.g., mobile phones or PDAs) with very-restricted computing power and built-in memory capacity, on the other hand, the counterpart server has enormous computing power but is *not perfectly secure* against various attacks (e.g., virus or hackers). In addition, neither PKI nor TRM is available at all.

For the above situation, we propose an efficient and leakage-resilient RSA-based AKE (for short, RSA-AKE) protocol where the client remembers his/her password and stores another secret along with the server's RSA public key, and the corresponding server stores the verification data as well as its RSA private key (it seems similar to [23], but its construction is completely different!). In the extended model where an adversary is given access to the stored secret of client, we prove that its security of the RSA-AKE protocol is reduced *tightly* to the RSA one-wayness in the random oracle model.

The RSA-AKE protocol guarantees a higher level of security: (1) it can avoid even on-line dictionary attacks as long as the leakage of client's stored secret does not happen; (2) the respective leakage of stored secret(s) from client and servers doesn't reveal any information on the password; (3) it can be extended to the multiple sever scenario with only one password; and (4) it provides perfect forward secrecy and anonymity.

To our best knowledge, the RSA-AKE protocol is the most efficient, in terms of both computation costs of client and communication costs, when compared with AKE protocols based on password and RSA. Specifically, the client is required to compute one modular exponentiation with an exponent e ($e \geq 3$) and the remaining computation costs, if the pre-computation is allowed, are only one modular multiplication and some negligible operations. As for communication costs, the RSA-AKE protocol has at most three flows of communications and requires $(l + 2k)$-bits bandwidth where l and k are security parameters for RSA and hash functions, respectively.

1.3.1 Organization

This paper is organized as follows. In Sect. 2, we introduce the model and security definitions (including computational assumption). In Sect. 3, we propose an efficient and leakage-resilient RSA-based AKE (RSA-AKE) protocol, followed by its security proof in Sects. 4, Appendix A and Appendix B. Section 5 is devoted to comparison with the previous RSA-based AKE protocols in aspects of several security properties and efficiency. In Sect. 6, we give some possible applications of the RSA-AKE protocol. Finally, we conclude in Sect. 7.

2. The Model and Security Definitions

In this section we introduce an extended model building on [6], [10] and security definitions for the AKE security and perfect forward secrecy.

We denote by C and $\mathcal{S}$ two parties that participate in a key exchange protocol P. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of P where we denote C (resp., $\mathcal{S}$) instances by C^I (resp., $\mathcal{S}^J$), or by U in case of any instance. For the j-th session in our protocol, the party C remembers a low-entropy secret pw drawn from a small dictionary of password $\mathbb{D}_{\mathsf{Password}}$, whose cardinality is D, and holds another secret α_j on *insecure* devices along with the counterpart's RSA public key (e, N). On the other hand, the party $\mathcal{S}$ stores a verification data p_j and its RSA private key (d, N). Here we suppose a far more powerful adversary (rather than an active one, considered in [6], [8], [17], who has the entire control of the network) by giving additional access to the Leak oracle that simulates *insecure* devices of the party C and *imperfect* server $\mathcal{S}$. Let us show the capability of adversary $\mathcal{A}$ each query captures:

- Execute$(C^I, \mathcal{S}^J)$: This query models passive attacks, where the adversary gets access to honest executions of P between the instances C^I and $\mathcal{S}^J$ by eavesdropping.
- Send(U, m): This query models active attacks by hav-

ing $\mathcal{A}$ send a message to instance U. The adversary $\mathcal{A}$ gets back the response U generates in processing the message m according to the protocol P. A query $\mathsf{Send}(C^I, \mathtt{Start})$ initializes the key exchange protocol, and thus the adversary receives the initial flow the party C should send out to the party $\mathcal{S}$.

- $\mathsf{Reveal}(U)$: This query handles the misuse of a session key (e.g., use in a weak symmetric-key encryption) by any instance U. The query is only available to $\mathcal{A}$ if the instance actually holds a session key and the latter is released to $\mathcal{A}$.
- $\mathsf{Leak}(U)$: This query handles the leakage of "stored" secrets by any instance U. The adversary $\mathcal{A}$ gets back the secrets $(\alpha_j, (e, N)$ and $(d, N))$ where the former (resp., the latter) is released if the instance corresponds to C^I (resp., $\mathcal{S}^J$). The query is available to $\mathcal{A}$ since the stored secrets might be leaked out due to a bug of the system or physical limitations in the sense that they should be stored on insecure devices all the time.
- $\mathsf{Test}(U)$: This oracle is used to see whether or not the adversary can obtain some information on the challenge session key by giving a hint on the latter. The Test-query can be asked at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the instance U is "fresh" (see below). This query is answered as follows: one flips a private coin $b \in \{0, 1\}$ and forwards the corresponding session key SK ($\mathsf{Reveal}(U)$ would output) if $b = 1$, or a random value except the session key if $b = 0$.

For the security notion of perfect forward secrecy, one has to account for a new type of query, the Corrupt-query, which models the compromise of the involving parties by adversary $\mathcal{A}$.

- $\mathsf{Corrupt}(U)$: This oracle is used to see whether or not the disclosure of "long-term" secrets of the involving parties does compromise the semantic security of session keys from previous sessions (even though that compromises the authenticity and thus the security of new sessions). With Corrupt-query to instance U, the adversary $\mathcal{A}$ gets back the long-term secrets (pw and its relevant secret p_j) but does not get any internal data.

Definition 1: (Freshness) We say that an instance is *fresh* (or has a *fresh* session key) if the following conditions hold: (1) the instance has computed and accepted a session key; (2) no Corrupt-query has been asked by $\mathcal{A}$ before the session key is accepted; and (3) no Reveal-query has been asked to the party C nor its partner $\mathcal{S}$ in the instance.

The aim of the adversary is to break the privacy of the session key (a.k.a., semantic security) in the context of executing P. The AKE security is defined by the game $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$, in which the adversary $\mathcal{A}$ is provided with random coin tosses, some oracles and then is allowed to invoke any number of queries as described above, in any order. When playing this game, the ultimate goal of the adversary is to guess the bit b in Test-query by outputting this guess b'. We denote AKE advantage, by $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$, as the probability that $\mathcal{A}$ can correctly guess the value of b. We formally define the AKE security; this will be necessary for stating meaningful results about our protocol in Sect. 5.

Definition 2: (AKE Security) A protocol P is said to be AKE secure if, when adversary $\mathcal{A}$ (with access to Leak oracle) asks q_s queries to Send oracle and passwords are chosen from a dictionary of size D, the adversary's advantage $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A})$ in attacking the protocol P is bounded by

$$O(q_s/D) + \epsilon(\cdot), \tag{1}$$

for a negligible function $\epsilon(\cdot)$ in a security parameter. The first term represents the fact that the adversary can do no better than guess a password during each query to Send oracle.

Definition 3: (Perfect Forward Secrecy) Suppose an adversary $\mathcal{A}$ with the ability to make the Corrupt-query as well as the other queries described above. A protocol P is said to provide perfect forward secrecy if the adversary's advantage, denoted by $\mathsf{Adv}_P^{\mathsf{pfs-ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$, in attacking P is negligible in a security parameter.

2.1 Computational Assumption

Next we define the standard RSA function on which the underlying computational assumption holds.

Definition 4: (RSA Function) An RSA generator RSA-KeyGen with associated security parameter l is a randomized algorithm that takes no input and returns a pair $((e, N), (d, N))$ such that (1) p, q are distinct and odd primes with each being about $l/2$ bits long, (2) $N = pq$ where $2^{l-1} \leq N < 2^l$ and (3) $e, d \in \mathbb{Z}^\star_{\varphi(N)}$ are integers satisfying $ed \equiv 1 \bmod \varphi(N)$. We call N an RSA modulus, (e, N) the public key and (d, N) the private key. The RSA function is a family of functions $\mathsf{RSA}_{N,f} : \mathbb{Z}^\star_N \to \mathbb{Z}^\star_N$ defined by $\mathsf{RSA}_{N,f}(w) \equiv w^f \bmod N$ for all $w \in \mathbb{Z}^\star_N$. That is, the encryption function $\mathsf{RSA}_{N,e}$ is defined by $\mathsf{RSA}_{N,e}(x) = y \equiv x^e \bmod N$ and the decryption function $\mathsf{RSA}_{N,d}$ is $\mathsf{RSA}_{N,d}(y) = x \equiv y^d \bmod N$ both of which are permutations on $\mathbb{Z}^\star_N$ and inverses of each other.

The computational assumption of the RSA function is equivalent to one-wayness (non-invertibility) of RSA: given $(N, e, y = \mathsf{RSA}_{N,e}(x))$ it is hard to compute x. Formally,

Definition 5: (One-wayness of RSA) Suppose that the RSA function is defined by Definition 4 and an adversary $\mathcal{I}$ is given the RSA instance $(N, e, y = \mathsf{RSA}_{N,e}(x))$. The RSA function is said to be *one-way* if the success probability of $\mathcal{I}$, defined as

$$\mathsf{Succ}_{RSA}^{\mathsf{ow}}(\mathcal{I}) = \Pr[x = x' | x' \leftarrow \mathcal{I}(N, e, y)], \tag{2}$$

is negligible in the security parameter l.

3. Our Protocol

Before presenting an efficient and leakage-resilient RSA-based AKE (for short, RSA-AKE) protocol, we will start by giving a considered situation and some notations to be used.

3.1 Considered Situation

Consider the following situation where a client is communicating with many disparate servers[†]. In particular, we focus on unbalanced wireless networks where the client has easy-to-be-lost/stolen devices (e.g., mobile phones or PDAs) with very-restricted computing power but some memory capacity itself, on the other hand, each server has its database and enormous computing power enough to generate a pair of RSA keys and to perform the RSA decryption function. Here, we do not assume that each server is completely secure against possible attacks (e.g., virus, hackers or insider attacks). In addition, neither PKI nor TRM is available.

3.2 Notations

Let k and l denote the security parameters, where k can be thought of as the general security parameter for hash functions (say, 160 bits) and l ($l > k$) can be thought of as the security parameter for RSA (say, 1024 bits). Let D be a dictionary size of passwords (say, 36 bits for alphanumerical passwords with 6 characters). Let $\{0,1\}^\star$ denote the set of finite binary strings and $\{0,1\}^k$ the set of binary strings of length k. If A is a set, then $a \xleftarrow{R} A$ indicates the process of selecting a at random and uniformly over A. Let "$\|$" denote the concatenation of bit strings in $\{0,1\}^\star$.

Let us define secure hash functions. While $\mathcal{G} : \{0,1\}^\star \to \mathbb{Z}_N^\star \backslash \{1\}$ denotes a full-domain hash (FDH) function, the other hash functions are denoted $\mathcal{H}_j : \{0,1\}^\star \to \{0,1\}^k$ for $j = 1, 2, 3$ and 4. Here $\mathcal{G}$ and $\mathcal{H}_j$ are distinct random functions one another. Let C and $\mathcal{S}$ be the identities of client and server, respectively, with representing each ID $\in \{0,1\}^\star$ as well.

3.3 The RSA-AKE Protocol

Here we propose an efficient and leakage-resilient RSA-based AKE (RSA-AKE) protocol suitable for the above-mentioned situation. The rationale behind the RSA-AKE protocol is that (i) client's password and another secret are combined to be used for authentication; (ii) in order to provide perfect forward secrecy, each secret (stored on client's devices and maintained by server) is updated whenever client and server correctly run the protocol; and (iii) a core technique appeared in [4], [10], [34], [39] is used in order to resist against off-line dictionary attacks after the leakage of client's stored secret.

3.3.1 Initialization

Whenever client C needs to register a verification data to each different server $\mathcal{S}_i$ ($i \geq 1$), they perform the following initialization phase. At first, server $\mathcal{S}_i$ sends its RSA public key (e, N), which is generated from $\mathsf{RSAKeyGen}(1^l)$, to the client[††]. The latter picks a secret value α_{i1} randomly chosen in $\mathbb{Z}_N^\star$ and registers securely a verification data p_{i1} to server $\mathcal{S}_i$:

$$p_{i1} \equiv \alpha_{i1} + \alpha_0 \bmod N \tag{3}$$

and sets the term $\alpha_0 = pw$ where pw is the client's password[†††]. Since both α_{i1} and p_{i1} are in the set of the same length, each is a share of $(2,2)$-threshold secret sharing scheme for α_0 [29].

Then client C remembers his password pw *and* additionally stores the secret value α_{i1} and the RSA public key (e, N) on insecure devices (e.g., mobile devices or smart cards) which may happen to leak α_{i1} and (e, N). The server $\mathcal{S}_i$ also stores the verification data p_{i1} and its RSA private key (d, N) on its databases both of which may be leaked out. Finally, they set a counter j as 1. Note that this initialization is done only once. For the initialization phase, see Fig. 1.

3.3.2 j-th Protocol Execution

When client C wants to share an authenticated session key securely with server $\mathcal{S}_i$, they run the j-th ($j \geq 1$) execution of the RSA-AKE protocol as follows. At the start of the j-th protocol execution, client C and server $\mathcal{S}_i$ hold $\big(j, \alpha_{ij}, (e, N)\big)$ and $\big(j, p_{ij}, (d, N)\big)$, respectively, where $p_{ij} \equiv \alpha_{ij} + pw \bmod N$. The client C should recover the verification data p_{ij} by adding the secret value α_{ij} stored on devices with the password pw remembered in his mind. Then the client chooses a random value x from $\mathbb{Z}_N^\star$ and encrypts x under the RSA public key (e, N): $y \equiv x^e \bmod N$. Also, client C computes a full-domain hash $W \leftarrow \mathcal{G}(j, p_{ij})$ with the input of (j, p_{ij}) and calculates z using a mask generation function as follows: $z \equiv y \cdot W \bmod N$. The latter is sent to server $\mathcal{S}_i$ along with (C, j). If the received counter j is correct, the server divides z by a hash of the counter and its verification data p_{ij}, and then decrypts the resultant value under its RSA private key (d, N) so as to obtain x. Then, server $\mathcal{S}_i$ computes and sends its authenticator $V_{\mathcal{S}_i}$ to client C.

Upon receiving $(\mathcal{S}_i, V_{\mathcal{S}_i})$ from the server, client C computes his authenticator V_C and a session key SK_{ij}, as long as $\mathcal{H}_1(C\|\mathcal{S}_i\|j\|z\|p_{ij}\|x)$ is equal to $V_{\mathcal{S}_i}$, and sends V_C to server $\mathcal{S}_i$. If the authenticator V_C is valid, server $\mathcal{S}_i$ actually computes a session key SK_{ij} that will be used for their subsequent cryptographic algorithms (e.g., AES or HMAC).

[†]For the sake of simplicity, we assign the servers consecutive integer $i \geq 1$ where $\mathcal{S}_i$ can be regarded as the i-th server.

[††]It is obvious in the context that each server $\mathcal{S}_i$ generates its own RSA key pair (e_i, d_i, N_i). However, for the visual comfort we use (e, d, N) instead of (e_i, d_i, N_i) from here on.

[†††]The password pw is drawn from password space $\mathbb{D}_{\mathsf{Password}}$ according to a certain probability distribution.

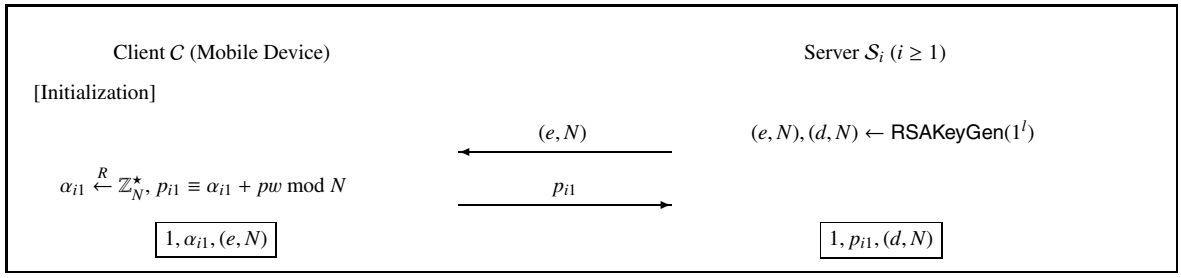


Fig. 1 The initialization of RSA-based AKE (RSA-AKE) protocol where the enclosed values in the rectangle represent stored secrets of client and server, respectively.

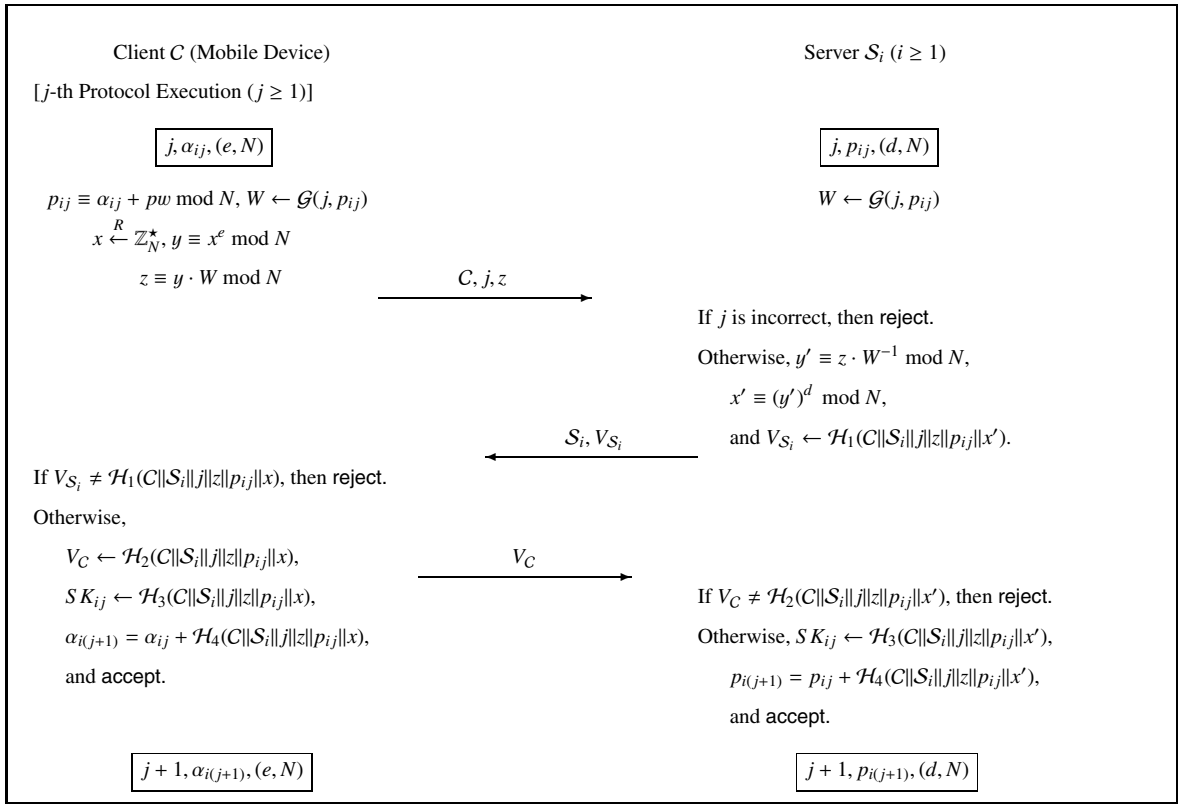


Fig. 2 The j-th protocol execution of RSA-AKE protocol where the enclosed values in the rectangle represent stored secrets of client and server, respectively.

At the end of the j-th protocol execution, client C refreshes the secret value α_{ij} to a new one $\alpha_{i(j+1)}$ without changing his password[†]. In the same way, server $\mathcal{S}_i$ also refreshes the verification data p_{ij} to a new one $p_{i(j+1)}$. Finally, client C stores $\left(j+1, \alpha_{i(j+1)}, (e, N)\right)$ on his devices and server $\mathcal{S}_i$ stores $\left(j+1, p_{i(j+1)}, (d, N)\right)$ on its databases for the next session. The whole protocol is illustrated in Fig. 2.

4. Security

In this section we show the RSA-AKE protocol of Fig. 2. is provably secure in the random oracle model [7][††], under the assumption that inverting an RSA instance is hard, by Definition 2.

4.1 Security Proof

In order to simplify the security proof, we omit the index i and only consider the first two flows of the j-th protocol execution (unilateral authentication of $\mathcal{S}$ to C). The latter is due to the well-known fact that the basic approach in the

[†]Notice that the frequent change of passwords might incur the risk of password to be exposed, simply because people tends to write it down on somewhere or needs considerable efforts to remember new passwords.

[††]Note that security in the random oracle model is only a heuristic: it does not imply security in the real world [9]. Nevertheless, the random oracle model is a useful tool for validating natural cryptographic constructions. Security proofs in this model prove security against adversaries that are confined to the random oracle world.

literature for adding authentication to an AKE protocol is to use the distributed Diffie-Hellman key or the shared secret to construct a simple "authenticator" for the other party [6], [10]. Therefore, the security proof with unilateral authentication can be extended to one with mutual authentication by simply adding the authenticator of C (the third flow) as in Fig. 2. However, this makes a proof more complicated.

Here we assert that the RSA-AKE protocol distributes semantically-secure session keys and provides unilateral authentication for the server $\mathcal{S}$.

Theorem 1: (AKE/UA Security) Let P be the RSA-AKE protocol of Fig. 2., where passwords are chosen from a dictionary of size D and the client's stored secret (i.e., $\alpha_{ij}, (e, N)$) is provided through the Leak-query. For any adversary $\mathcal{A}$ within a polynomial time t, with less than q_s active interactions with the parties (Send-queries), q_p passive eavesdroppings (Execute-queries), and asking q_g and q_h hash queries to $\mathcal{G}$ and any $\mathcal{H}_i$ respectively, $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq 4\varepsilon$ and $\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) \leq \varepsilon$, with ε upper-bounded by

$$\frac{3q_{\mathcal{S}}}{D} + 10\mathsf{Succ}^{\mathsf{ow}}_{RSA}\left(q_h^2, t + 2q_h^2\tau_{law}\right) + \frac{q_C}{2^{k_1}} + \frac{6q_{\mathcal{S}} + (q_C + q_p)^2 + (q_g + q_h)^2}{2^{l+1}}, \quad (4)$$

where q_C and $q_{\mathcal{S}}$ denote the number of C and $\mathcal{S}$ instances involved during the attack (each upper-bounded by $q_p + q_s$), k_1 is the output length of $\mathcal{H}_1$, l is the security parameter, and τ_{law} is the computational time needed for modular multiplication or modular division.

Informally speaking, an adversary who knows the client's stored secret cannot determine the correct password through off-line dictionary attacks since generating the valid authenticator after computing z falls into on-line dictionary attacks (which can be easily prevented and detected). The proof appears in Appendix A, whose technicality is based on [10] with some adjustments and changes, where the simulator has to "guess" which hash query to $\mathcal{G}$ will be used by the adversary to produce the correct bit b.

Theorem 2: (AKE/UA Security) Let P be the RSA-AKE protocol of Fig. 2., where the server's stored secret (i.e., (d, N)) is provided through the Leak-query. For any adversary $\mathcal{A}$, with less than q_s active interactions with the parties (Send-queries), q_p passive eavesdroppings (Execute-queries), and asking q_g and q_h hash queries to $\mathcal{G}$ and any $\mathcal{H}_i$ respectively, $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq 4\varepsilon$ and $\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) \leq \varepsilon$, with ε upper-bounded by

$$\frac{q_C}{2^{k_1}} + \frac{6(q_p + q_{\mathcal{S}}) + 3(q_C + q_p)^2 + 2(q_g + q_h)^2}{2^{l+1}}, \quad (5)$$

where q_C and $q_{\mathcal{S}}$ denote the number of C and $\mathcal{S}$ instances involved during the attack (each upper-bounded by $q_p + q_s$), k_1 is the output length of $\mathcal{H}_1$ and l is the security parameter.

Informally speaking, an adversary who knows the server's RSA private key cannot perform even on-line dictionary attacks since the authentication depends on the strong secret p_j as in [7], [30]. The proof appears in Appendix B.

It is of practical significance to note that in the real world applications the Leak-query is limited by the *physical* power of an adversary which are usually much less, whereas the Send and Execute-queries are directly relevant to interactions with the parties.

5. Comparison

In this section we compare the RSA-AKE protocol of Sect. 3.3 with the previous AKE protocols, described in Sect. 1.1, all of which are using password and RSA. As PAKE protocols, RSA-IPAKE [10], PEKEP and CEKEP [38], and SNAPI and SNAPI-X [25] are compared. Also, we compare with Mutual Authentication and Key Exchange (MAKE) and Mutual Authentication and Diffie-Hellman Key Exchange (MA-DHKE) in Sect. 3.4 of [17], and Construction 2 (KE-CKM) in Sect. 4 of [23]. As key-establishment parts of SSL/TLS and SSH, we focus on the password-based user authentication mode (SSL/TLS, SSH-1) and the public-key based user authentication mode with a password-protected private key (SSL/TLS, SSH-2). In the latter mode, client's private key is stored in an encrypted form by a symmetric-key encryption with password as its key. To fairly compare, we instantiate with the RSA function if a public-key encryption is not specified in the relevant previous works (e.g., [17], [23]). For simplifying its discussion, we omit additional computation and communication costs of SSL/TLS and SSH in order to verify the counterpart's certificate.

5.1 Security Properties

The RSA-AKE protocol may seem very similar to the KE-CKM protocol [23] in the sense that both don't require PKI, and a client remembers his password and holds not only another secret but also a server's RSA public key. However, the RSA encryption used in the KE-CKM protocol should be IND-CCA2 secure (e.g., RSA-OAEP); otherwise it is insecure because the server plays a role of decryption oracle in the protocol [15]. The other differences are explained later.

As for several security properties, we show the comparative results in Tables 1 and 2. For an easier comparison, the following three cases are considered.

- Case1: This is the case that an adversary gets the stored secrets from client C.
- Case2: This is the case that an adversary gets all of the stored secrets from server $\mathcal{S}$, except the RSA private key.
- Case3: This is the case that an adversary gets the RSA private key from server $\mathcal{S}$.

In the RSA-AKE protocol, Case1, Case2 and Case3 correspond to the leakages of $(\alpha_{ij}, (e, N))$, p_{ij} and (d, N), respectively.

From Table 1, one can see that the RSA-AKE protocol guarantees semantic security of session keys against Case1

Table 1 Comparison of RSA-based AKE protocols in a situation where no perfect TRM is available.

Protocols		Client's possessions			Semantic security of session key against		
		Password	Stored secret(s)*1	Public info.*2	Case1	Case2	Case3
PAKE	RSA-IPAKE [10]	√			secure	insecure	secure
	PEKEP [38]						
	CEKEP [38]						
	SNAPI [25]						
	SNAPI-X [25]	√		√			
MAKE [17]		√		√	secure	insecure	insecure
MA-DHKE [17]		√		√			
KE-CKM [23]		√	√	√			
SSL/TLS, SSH-1		√		√	secure	insecure	insecure
SSL/TLS, SSH-2		√	√	√	insecure	secure	insecure
RSA-AKE		√	√	√	secure*3	insecure	secure*4

*1: A secret value, a signing key, a decryption key and/or a symmetric key (for MAC).
*2: A CA's signature verification key, an (cached) encryption key or its fingerprint, public parameters for the Diffie-Hellman protocol.
*3: Theorem 1
*4: Theorem 2

Table 2 Comparison of RSA-based AKE protocols in a situation where no perfect TRM is available (con't).

Protocols		Security of password against		Extension	Perfect forward secrecy (Case1 ∨ Case2 ∨ Case3)	Anonymity
		Case1	Case2			
PAKE	RSA-IPAKE [10]	○	X (△*1)	impossible	PFS can be achieved only if server $\mathcal{S}$ changes its RSA key pair every time.	not provided
	PEKEP [38]					
	CEKEP [38]					
	SNAPI [25]					
	SNAPI-X [25]	○	△*1			
MAKE [17]		○	△*1	impossible	not achieved	provided
MA-DHKE [17]		○	△*1	impossible	achieved	provided
KE-CKM [23]		○	△*1	possible	not achieved	provided
SSL/TLS, SSH-1		○	X (△*1)	impossible	achieved	provided
SSL/TLS, SSH-2		△*1	○	possible	achieved	provided
RSA-AKE		○*2	○*2	possible	achieved*3	provided

*1: A client registers password verification data computed with a one-way hash function of the password, $f(pw)$, to the server instead of pw. Doing this somewhat slows down off-line dictionary attacks of an adversary.
*2: Theorem 3
*3: Theorem 4

and Case3, but not against both of them. The PAKE protocols provide the same security guarantee in the case that an adversary generates its RSA key pair or an RSA key pair generated by the honest party should be used only once as an internal state. In terms of semantic security against Case2, SSL/TLS, SSH-2 is the only survivor simply because the password is not used for client's authentication but for protecting the client's private key. As a result, if an AKE protocol is based on password for authentication we can not guarantee semantic security against Case2.

In terms of security of password against Case1 and Case2, we use three symbols in Table 2: ○ guarantees the security of password against both on-line and off-line dictionary attacks; △ guarantees the security of password against on-line, but not off-line attacks; and X guarantees the security of password against neither on-line nor off-line attacks. Now, we claim the following for the RSA-AKE protocol:

Theorem 3: (Security of Password) The password in the RSA-AKE protocol remains information-theoretically secure against off-line dictionary attacks even after either Case1 or Case2 happens.

Proof. The proof is straightforward. First, let us think of an adversary who obtained the stored secret α_{ij} of client C and is trying to deduce the password pw. Since α_{ij} is completely independent from pw, the adversary cannot get any information about the password.

$$H(pw) = H\left(pw|\alpha_{ij}\right) \tag{6}$$

where $H(X)$ denotes the (Shannon) entropy of X and $H(X|Y)$ denotes the conditional entropy of X conditioned on Y. Second, let us think of the security of password against an adversary who obtained the stored secret p_{ij} of server $\mathcal{S}_i$. The latter doesn't reveal any information about the password simply because p_{ij} is one share of $(2, 2)$-threshold secret sharing scheme. □

Contrary to the RSA-AKE protocol, the other AKE protocols don't have the security of password since their stored secrets in either the client or the server(s) contain enough information for an adversary to succeed in retrieving the relatively short password with off-line dictionary attacks.

As one of the security properties, the RSA-AKE protocol can be extended to the multiple server scenario with only one password (Extension in Table 2) but the number of stored secrets α_{ij} grows linearly to the number of servers. In a similar way, the KE-CKM protocol is extensible where a hashed form of password and salt is used.

Here we claim that the RSA-AKE protocol provides perfect forward secrecy for protecting previous communications (whereas the KE-CKM protocol does not) by the following theorem:

Theorem 4: (Perfect Forward Secrecy) Let P be the RSA-AKE protocol of Fig. 2. For any adversary $\mathcal{A}$, with less than q_s active interactions with the parties (Send-queries), q_p passive eavesdroppings (Execute-queries), and asking q_g and q_h hash queries to $\mathcal{G}$ and any $\mathcal{H}_i$ respectively, $\mathsf{Adv}_P^{\mathsf{pfs-ake}}(\mathcal{A}) \leq 4\varepsilon$, with ε upper-bounded by

$$\frac{q_C}{2^{k_1+1}} + \frac{6(q_p + q_S) + 3(q_C + q_p)^2 + 2(q_g + q_h)^2}{2^{l+1}}, \quad (7)$$

where q_C and q_S denote the number of C and S instances involved during the attack (each upper-bounded by $q_p + q_s$), k_1 is the output length of $\mathcal{H}_1$ and l is the security parameter.

This means that perfect forward secrecy can be achieved even if server S_i would use the same RSA key pair for many sessions (unlike PAKE protocols). The MA-DHKE, SSL/TLS, SSH-1 and SSL/TLS, SSH-2 protocols also provide perfect forward secrecy since the Diffie-Hellman protocol is used as well. The proof appears in Appendix C. More interesting, we get an almost same advantage in Appendix B and Appendix C since the information private to the simulator is the same, but each simulator behaves a little differently.

In terms of anonymity, the RSA-AKE protocol can use one-time ID that would change every time. The SSL/TLS, SSH-1 provides anonymity by sending an encrypted client's identity and password with the shared session key. The other RSA-based AKE protocols (except PAKE ones) send an encrypted client's identity with the server's public key.

5.2 Efficiency

Since password-based AKE protocols have been motivated by the very practical implementations and widely used even in wireless networks, we analyze computation costs of client, communication costs and the number of flows in the previous AKE and RSA-AKE protocols while comparing those in Table 3.

For simplicity, we denote by l (resp., k) the security parameter for the RSA function and the Diffie-Hellman protocol (resp., for the hash functions and temporal random values). The number of modular exponentiations is a major factor to evaluate efficiency of a cryptographic protocol because that is the most power-consuming operation. So we count the number of modular exponentiations as computation costs of client C. For brevity, we denote by RSA-Exp. (resp., DH-Exp.) the number of RSA modular exponentiations with an exponent e (resp., the number of Diffie-Hellman modular exponentiations with an exponent of 160-bits long). The figures in the parentheses are the remaining costs after pre-computation. In terms of communications costs, the length of identities is excluded and $|\cdot|$ indicates its bit-length. And in the number of flows, we consider mutual authentication of each protocol (if not described explicitly) by making each party send an authenticator that is computed

Table 3 Comparison of RSA-based AKE protocols as for efficiency.

Protocols		Computation costs of client C		Communication costs (bandwidth)	The number of flows
		DH-Exp.	RSA-Exp. with e		
P	RSA-IPAKE [10]		$m+1$ when $e \geq 3$, (m) *1	$(m+2)l + 3k$	6
	PEKEP [38]		$n+1$ when $e \geq 3$, (n) *2	$2l + 4k + \|e\|$	4
A	CEKEP [38]		$2n$ when $e \geq 3$, $(2n-1)$ *3 or 2, (2) *4	$3l + 6k + \|e\| + \|n\|$	6
K	SNAPI [25]		Primality test of large e and 1 *5, (Primality test of e)	$2l + 4k + \|e\|$	4
E	SNAPI-X [25]	2, (2)	Primality test of large e and 1 *5, (Primality test of e)	$3l + 4k + \|e\|$	4
MAKE [17]			1 when $e \geq 3$, (1)	$2l + 3k + \|e\|$	3
MA-DHKE [17]		2, (1)	1 when $e \geq 3$, (1)	$4l + 3k + \|e\|$	3
KE-CKM [23]			1 when $e \geq 3$, (1)	$2l + 4k + \|e\|$	3
SSL/TLS, SSH-1		2, (1)	1 when $e \geq 3$, (1)	$3l + \|E\|$	3
SSL/TLS, SSH-2		2, (1)	1 when $e \geq 3$ and 1 RSA-Exp. with d, (1 RSA-Exp. with d)	$3l$	2
RSA-AKE			1 when $e \geq 3$, (0)	$l + 2k$	3

*1: m is the system parameter.
*2: $n = \lfloor \log_e N \rfloor$
*3: $n = \left\lceil \log_e \omega^{-1} \right\rceil$ where $0 < \omega \leq 2^{-80}$
*4: 2 modular exponentiations with each having an exponent of $\left\lceil \log_e \omega^{-1} \right\rceil$ bits where $0 < \omega \leq 2^{-80}$.
*5: 1 modular exponentiation with an exponent e having the following explicit requirements on e and N. One is to set e to be a prime, in the range of $2^l + 1 \leq e < 2^{l+1}$, greater than N. The other is to set e to be a prime such that $e \geq \sqrt{N}$ and $(N \bmod e) \nmid N$.

with the shared secret.

With respect to computation costs in the RSA-AKE protocol, client C is required to compute one modular exponentiation with an exponent e ($e \geq 3$) and one modular multiplication. Of course, the choice of RSA key pair $((e, N), (d, N))$ is in general left to the implementations. However, in order to speed-up computation of $\mathsf{RSA}_{N,e}$, e should be chosen to be a small prime with a small number of 1's in its binary representation (e.g., $e = 3$ or $2^{16} + 1$). When $e = 3$ and 1024-bit RSA modulus is used, the total computation time (for modular operations) of client is 416.56 msec on a 16 MHz Palm V according to [35]. In particular, the remaining costs after pre-computation is only one modular multiplication and some negligible operations for modular additions and hash functions! On the other hand, both the MAKE and KE-CKM protocols can neither allow pre-computation, more importantly, nor provide perfect forward secrecy. With respect to communication costs, the RSA-AKE protocol requires a bandwidth of $(l + 2k)$-bits approximately. For the minimum security parameters recommended in practice ($|N| = 1024$ and $|H| = 160$), the bandwidth needed is 168 Bytes. In addition, the RSA-AKE protocol has at most 3 flows of communications.

6. Applications

As we mentioned in the previous sections, the RSA-AKE protocol doesn't rely on both PKI and TRM while maintaining much more security and efficiency compared to the relevant AKE protocols. Actually, the RSA-AKE protocol can be used in any authentication system; however, it is more suitable especially in ad-hoc and sensor networks since these networks do not assume availability of trusted and reliable third parties.

One of the possible applications may be personal networks where the (portable) devices belonging to a single user are interconnected wirelessly in a cluster fashion. The concept of personal networks is very user-centric and representative for the next generation networks. However, the present security mechanism is not considering at all what happens whenever a mobile node (device) is compromised, lost or stolen [21], [27], [28].

7. Conclusions

In this paper, we first revisited the previous AKE protocols (using password and RSA) from a viewpoint of which kind of information is needed for authentication and of how much or whether each protocol does satisfy security against leakage of stored secrets, efficiency and perfect forward secrecy.

Then we have proposed an efficient and leakage-resilient RSA-based AKE (RSA-AKE) protocol suitable for the following situation: (1) a client, who communicates with many different servers, remembers only one password and has insecure devices with very-restricted computing power and some built-in memory capacity; (2) the counterpart servers are not perfectly secure against various attacks (e.g, virus, hackers or inside attacks); (3) neither PKI nor TRM is available. Under the notion of AKE security, its security of the RSA-AKE protocol is reduced tightly to the RSA one-wayness in the random oracle model. We also showed that the RSA-AKE protocol guarantees several security properties (e.g., security of password, multiple sever scenario with only one password, perfect forward secrecy and anonymity) and is the most efficient in terms of both computation costs of client and communication costs when compared with the previous AKE protocols of their kind.

Though the use of insecure devices may seem a strong assumption, we stress that this assumption (i) is significantly applicable to the real world (think of a client who carries mobile devices with some memory capacity); and (ii) is crucial in order to provide a higher level of security as well as more efficiency. Having said this, carrying insecure devices for client seems a small price to pay for more strengthened security and efficiency in the RSA-AKE protocol.

Acknowledgements

We would like to appreciate the anonymous reviewers' constructive comments and advices on this paper. This research has been sponsored by the Ministry of Economy, Trade and Industry (METI), Japan, under the contract of "New Generation of Information Security R&D Program."

References

[1] F. Bao, "Security analysis of a password authenticated key exchange protocol," Proc. ISC 2003, LNCS 2851, pp.208–217, Springer-Verlag, 2003.
[2] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," Proc. 30th ACM Symposium on Theory of Computing (STOC), pp.419–428, ACM, 1998.
[3] M. Bellare, D. Jablon, H. Krawczyk, P. MacKenzie, P. Rogaway, R. Swaminathan, and T. Wu, "Proposal for P1363 study group on password-based authenticated-key-exchange methods," Submitted to IEEE P1363.2 Working Group, Feb. 2000.
[4] S.M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictioinary attacks," Proc. IEEE Symposium on Security and Privacy, pp.72–84, IEEE Computer Society, 1992.
[5] M.K. Boyarsky, "Public-key cryptography and password protocols: The multi-user case," Proc. 6th ACM Conference on Computer and Communications Security, pp.63–72, ACM, 1999.
[6] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Proc. EUROCRYPT 2000, LNCS 1807, pp.139–155, Springer-Verlag, 2000.
[7] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," Proc. ACM CCS '93, pp.62–73, 1993.
[8] M. Bellare and P. Rogaway, "Entity authentication and key distribution," Proc. CRYPTO '93, LNCS 773, pp.232–249, Springer-Verlag, 1993.
[9] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," Proc. 30th ACM Symposium on Theory of Computing (STOC), pp.209–218, ACM, 1998.
[10] D. Catalano, D. Pointcheval, and T. Pornin, "Trapdoor hard-to-invert group isomorphisms and their application to password-based authentication," J. Cryptol., 2006. The extended abstract appeared

at CRYPTO 2004. Available at http://www.di.ens.fr/~pointche/pub.php?reference=CaPoPo06
[11] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–654, 1976.
[12] W. Diffie, P. van Oorschot, and M. Wiener, "Authentication and authenticated key exchange," Proc. Designs, Codes, and Cryptography, pp.107–125, 1992.
[13] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," Proc. IEEE Computer Security Foundation Workshop, pp.24–29, 1995.
[14] A. Frier, P. Karlton, and P. Kocher, The SSL 3.0 Protocol, Netscape Communication Corp., 1996. Available at http://wp.netscape.com/eng/ssl3/
[15] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP is secure under the RSA assumption," Proc. CRYPTO 2001, LNCS 2139, pp.260–274. Springer-Verlag, 2001.
[16] L. Gong, T. Lomas, R. Needham, and J. Saltzer, "Protecting poorly-chosen secrets from guessing attacks," IEEE J. Sel. Areas Commun., vol.11, no.5, pp.648–656, 1993.
[17] S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," ACM Transactions on Information and System Security, vol.2, no.3, pp.230–268, Aug. 1999.
[18] IETF (Internet Engineering Task Force), EAP Password Authenticated Exchange, http://www.ietf.org/internet-drafts/draft-clancy-eap-pax-06.txt, Jan. 2006.
[19] IETF (Internet Engineering Task Force), Secure Shell (secsh) Charter, http://www.ietf.org/html.charters/secsh-charter.html
[20] IETF (Internet Engineering Task Force), Transport Layer Security (tls) Charter, http://www.ietf.org/html.charters/tls-charter.html
[21] D.M. Kyriazanos, J.W. Floroiu, M. Argyropoulos, C.Z. Patrikakis, M. Imine, and N.R. Prasad, "MAGNET personal network security model: Trust establishment, Policy Management and AAA Infrastructure," Proc. WWRF15, Dec. 2005.
[22] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols," Proc. CRYPTO 2003, LNCS 2729, pp.400–425, Springer-Verlag, 2003.
[23] V. Kolesnikov and C. Rackoff, "Key exchange using passwords and long keys," Proc. TCC 2006, LNCS 3876, pp.100–119, Springer Verlag, March 2006.
[24] T. Lomas, L. Gong, J. Saltzer, and R. Needham, "Reducing risks from poorly chosen keys," Proc. 12th ACM Symposium on Operating System Principles, vol.23, no.5, pp.14–18, ACM Operating Systems Review, 1989.
[25] P. MacKenzie, S. Patel, and R. Swaminathan, "Password-authenticated key exchange based on RSA," Proc. ASIACRYPT 2000, LNCS 1976, pp.599–613, Springer-Verlag, 2000. The full version is available at http://cm.bell-labs.com/who/philmac/bib.html
[26] S. Patel, "Number theoretic attacks on secure password schemes," Proc. IEEE Symposium on Security and Privacy, pp.236–247, IEEE Computer Society, 1997.
[27] N.R. Prasad, A. Markopoulos, S. Mirzadeh, and K. Masmoudi, "MAGNET secure service management," Oct. 2005.
[28] C. Politis, K. Nyberg, S. Mirzadeh, K. Masmoudi, H. Afifi, J. Floroiu, and N.R. Prasad, "Personal network security architecture," Proc. WPMC2005, pp.328–333, Sept. 2005.
[29] A. Shamir, "How to share a secret," Proc. Commun. ACM, vol.22, no.11, pp.612–613, 1979.
[30] V. Shoup, "On formal models for secure key exchange," IBM Research Report RZ 3121, 1999. Available at http://eprint.iacr.org/1999/012
[31] V. Shoup, "OAEP reconsidered," J. Cryptol., vol.15, no.4, pp.223–249, Sept. 2002.
[32] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, available at http://eprint.iacr.org/2004/332
[33] S.H. Shin, K. Kobara, and H. Imai, "Efficient and leakage-resilient authenticated key transport protocol based on RSA," Proc. ACNS2005, LNCS 3531, pp.269–284, Springer-Verlag, June 2005.
[34] D.S. Wong, A.H. Chan, and F. Zhu, "More efficient password authenticated key exchange based on RSA," Proc. INDOCRYPT 2003, LNCS 2904, pp.375–387, Springer-Verlag, 2003.
[35] D.S. Wong, H.H. Fuentes, and A.H. Chan, "The performance measurement of cryptographic primitives on palm devices," Proc. 17th Annual Computer Security Applications Conference, Dec. 2001.
[36] S.B. Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," Proc. IMA International Conference on Cryptography and Coding, LNCS 1355, pp.30–45, Dec. 1997.
[37] T. Wu, "A real-world analysis of kerberos password security," Proc. Internet Society Symposium on Network and Distributed System Security, pp.13–22, Feb. 1999.
[38] M. Zhang, "New approaches to password authenticated key exchange based on RSA," Proc. ASIACRYPT 2004, LNCS 3329, pp.230–244, Springer-Verlag, 2004. Cryptology ePrint Archive, Report 2004/033, available at http://eprint.iacr.org/2004/033
[39] F. Zhu, D.S. Wong, A.H. Chan, and R. Ye, "Password authenticated key exchange based on RSA for imbalanced wireless networks," Proc. ISC 2002, LNCS 2433, pp.150–161, Springer-Verlag, 2002.

Appendix A: Proof of Theorem 1

Proof. In this proof, we incrementally define a sequence of games (cf. [32]) starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_5$. We use Shoup's lemma [31] to bound the probability of each event in these games.

Game $\mathbf{G}_0$: This is the real protocol in the random oracle model. We are interested in the following two events:

- S_0 (for semantic security) which occurs if the adversary correctly guesses the bit b involved in the Test-query;
- A_0 (for $\mathcal{S}$-authentication) which occurs if an instance $\mathcal{C}^I$ accepts with no partner instance $\mathcal{S}^J$ with the same transcript $((\mathcal{C}, j, z), (\mathcal{S}, V_{\mathcal{S}}))$

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[\mathsf{S}_0] - 1,$$
$$\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) = \Pr[\mathsf{A}_0]. \qquad (\mathrm{A}\cdot 1)$$

In any game $\mathbf{G}_n$ below, we study the event A_n and the restricted event $\mathsf{SwA}_n = \mathsf{S}_n \wedge \neg\mathsf{A}_n$.

Game $\mathbf{G}_1$: In this game, we simulate the hash oracles ($\mathcal{G}, \mathcal{H}_1, \mathcal{H}_3$ and $\mathcal{H}_4$, but as well additional hash functions $\mathcal{H}_i' : \{0,1\}^\star \to \{0,1\}^{k_i}$ (for $i = 1, 3, 4$) that will appear in the Game $\mathbf{G}_3$) as usual by maintaining hash lists $\Lambda_{\mathcal{G}}$, $\Lambda_{\mathcal{H}_i}$ and $\Lambda_{\mathcal{H}_i'}$ (see Fig. A$\cdot$1). We also simulate all the instances, as the real parties would do, for the Send, Execute, Reveal, Leak and Test-queries (see Fig. A$\cdot$2). From this simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

Game $\mathbf{G}_2$: For an easier analysis in the following, we first forward any query to $\mathcal{H}_i$ to $\mathcal{G}$:

► **Rule** $\mathcal{H}^{(2)}$

The query q is parsed as $q = C\|\mathcal{S}\|j\|z\|p_j\|x$, then one queries $\mathcal{G}(j, p_j)$.

- For a hash-query $\mathcal{H}_i(q)$ (resp., $\mathcal{H}'_i(q)$), such that a record (i, q, r) appears in $\Lambda_{\mathcal{H}_i}$ (resp., $\Lambda_{\mathcal{H}'_i}$), the answer is r. Otherwise one chooses a random element $r \stackrel{R}{\leftarrow} \{0,1\}^{k_i}$, answers with it, and adds the record (i, q, r) to $\Lambda_{\mathcal{H}_i}$ (resp., $\Lambda_{\mathcal{H}'_i}$).
  - ► **Rule** $\mathcal{H}^{(1)}$
    Nothing to do. % To be defined later
- For a hash-query $\mathcal{G}(j, q)$, such that a record $(j, q, r, \star, \star)$ appears in $\Lambda_{\mathcal{G}}$, the answer is r. Otherwise the answer r is defined according to the following rule:
  - ► **Rule** $\mathcal{G}^{(1)}$
    Choose a random element $r \stackrel{R}{\leftarrow} \mathbb{Z}_N^\star$. The record $(j, q, r, \perp, \perp)$ is added to $\Lambda_{\mathcal{G}}$.

  Note: the fourth and fifth components of the elements of this list will be explained later.

Fig. A· 1 Simulation of the hash functions: $\mathcal{G}$ and $\mathcal{H}_i$ oracles.

Send-queries to C

We answer to the Send-queries to a C-instance as follows:

- A $\mathsf{Send}(C^I, \mathtt{Start})$-query is processed according to the following rules:
  - ► **Rule C1**$^{(1)}$
    Compute $p_j \equiv \alpha_j + pw \bmod N$.
  - ► **Rule C2**$^{(1)}$
    Generate $(x, y \equiv x^e \bmod N)$, and compute $W \leftarrow \mathcal{G}(j, p_j)$ and $z \equiv y \times W \bmod N$.

  Then the query is answered with (C, j, z), and the instance goes to an expecting state.
- If the instance C^I is in an expecting state, a query $\mathsf{Send}(C^I, (\mathcal{S}, V_{\mathcal{S}}))$ is processed by computing the alleged authenticator, the session key and the refreshed secret. We apply the following rules.
  - ► **Rule C3**$^{(1)}$
    Compute the expected authenticator and the session key
    $V'_{\mathcal{S}} \leftarrow \mathcal{H}_1(C\|\mathcal{S}\|j\|z\|p_j\|x)$, $SK_C \leftarrow \mathcal{H}_3(C\|\mathcal{S}\|j\|z\|p_j\|x)$.
  - ► **Rule C4**$^{(1)}$
    Compute the refreshed secret $\alpha_{j+1} = \alpha_j + \mathcal{H}_4(C\|\mathcal{S}\|j\|z\|p_j\|x)$.

  If $V'_{\mathcal{S}} = V_{\mathcal{S}}$, the instance accepts. In any case, it terminates.

Send-queries to $\mathcal{S}$

We answer to the Send-queries to a $\mathcal{S}$-instance as follows:

- A $\mathsf{Send}(\mathcal{S}^J, (C, j, z))$-query is first processed by checking that j is the correct counter, and in the case of correct j it is processed by computing the authenticator, the session key and the refreshed secret. We apply the following rules:
  - ► **Rule S1**$^{(1)}$
    Compute $W \leftarrow \mathcal{G}(j, p_j), y' \equiv z \times W^{-1} \bmod N$ and $x' \equiv (y')^d \bmod N$.
  - ► **Rule S2**$^{(1)}$
    Compute the authenticator and the session key
    $V_{\mathcal{S}} \leftarrow \mathcal{H}_1(C\|\mathcal{S}\|j\|z\|p_j\|x')$, $SK_{\mathcal{S}} \leftarrow \mathcal{H}_3(C\|\mathcal{S}\|j\|z\|p_j\|x')$.
  - ► **Rule S3**$^{(1)}$
    Compute the refreshed secret $p_{j+1} = p_j + \mathcal{H}_4(C\|\mathcal{S}\|j\|z\|p_j\|x')$.

  Finally the instance accepts, and the query is answered with $(\mathcal{S}, V_{\mathcal{S}})$.

Other queries

- An $\mathsf{Execute}(C^I, \mathcal{S}^J)$-query is processed using successively the above simulations of the Send-queries: $(C, j, z) \leftarrow \mathsf{Send}(C^I, \mathtt{Start})$, $(\mathcal{S}, V_{\mathcal{S}}) \leftarrow \mathsf{Send}(\mathcal{S}^J, (C, j, z))$, $\mathsf{Send}(C^I, (\mathcal{S}, V_{\mathcal{S}}))$, and then outputting the transcript $((C, j, z), (\mathcal{S}, V_{\mathcal{S}}))$.
- A $\mathsf{Reveal}(U)$-query returns the session key (SK_C or $SK_{\mathcal{S}}$) computed by the instance U (if the latter has accepted).
- A $\mathsf{Leak}(C^I)$-query returns the stored secret $(\alpha_j, (e, N))$ by the instance C^I.
- A $\mathsf{Test}(U)$-query first gets SK from $\mathsf{Reveal}(U)$, and flip a coin b. If $b = 1$, we return the value of the session key SK, otherwise we return a random value drawn from $\{0,1\}^{k_3}$.

Fig. A· 2 Simulation of the RSA-AKE protocol.

The number of queries to $\mathcal{G}$ thus becomes $q'_g \le q_g + q_h$. Furthermore we exclude games in which some events (Coll_2) are unlikely to happen:

- collision of the partial transcript (C, j, z): any adversary tries to find out at least one pair (j, z), coinciding with the challenge transcript, involving the honest party C, and then obtain the corresponding session key (i.e., the same as the challenge session key) using the Reveal-query;
- collision on the output of $\mathcal{G}$.

Both probabilities are bounded by the birthday paradox:

$$\Pr[\mathsf{Coll}_2] \le \frac{(q_C + q_p)^2 + q'^2_g}{2^{l+1}}. \quad (\mathrm{A}\cdot 2)$$

Game $\mathbf{G}_3$: In order to make the authenticators and the session keys unpredictable to any adversary, we compute them using the private oracles $\mathcal{H}'_1$ and $\mathcal{H}'_3$ (instead of $\mathcal{H}_1$ and $\mathcal{H}_3$), respectively, so that the values are completely independent from the random oracles as well as $\mathcal{G}$. We reach this aim by using the following rules:

► **Rule C3/S2**$^{(3)}$
Compute the authenticator
$V_S \leftarrow \mathcal{H}'_1(C\|\mathcal{S}\|j\|z)$.
Compute the session key
$SK_{C/S} \leftarrow \mathcal{H}'_3(C\|\mathcal{S}\|j\|z)$.

We also use the private oracle $\mathcal{H}'_4$ (instead of $\mathcal{H}_4$) so that the refreshed secrets are unpredictable to any adversary as well. We modify the simulation by using the following rules:

► **Rule C1**$^{(3)}$
Choose two random elements $(\beta_j, \gamma_j) \stackrel{R}{\leftarrow} \left(\mathbb{Z}_N^\star\right)^2$, and set $\alpha_j \leftarrow \beta_j$ and $p_j \leftarrow \gamma_j$.

► **Rule C4/S3**$^{(3)}$
Compute $\alpha_{j+1} = \alpha_j + \mathcal{H}'_4(C\|\mathcal{S}\|j\|z)$ and
$p_{j+1} = p_j + \mathcal{H}'_4(C\|\mathcal{S}\|j\|z)$.

Since the secret $x = x' = \mathsf{RSA}_{N,d}\left(z \times W^{-1}\right)$ depends on $(e, N), z$ and the parties (the common secret p_j), the games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable unless some specific hash queries are asked, denoted by event $\mathsf{AskH}_3 = \mathsf{AskH1}_3 \vee \mathsf{AskH3w1}_3 \vee \mathsf{AskH4w13}_3$, where $W = \mathcal{G}(j, p_j)$:

- $\mathsf{AskH1}_3$: $\left(C\|\mathcal{S}\|j\|z\|p_j\|\mathsf{RSA}_{N,d}\left(z \times W^{-1}\right)\right)$ has been queried by $\mathcal{A}$ to $\mathcal{H}_1$ for some transcript $((C, j, z), (\mathcal{S}, V_S))$;
- $\mathsf{AskH3w1}_3$: $\left(C\|\mathcal{S}\|j\|z\|p_j\|\mathsf{RSA}_{N,d}\left(z \times W^{-1}\right)\right)$ has been queried by $\mathcal{A}$ to $\mathcal{H}_3$ for some transcript $((C, j, z), (\mathcal{S}, V_S))$, where some party has accepted, but event $\mathsf{AskH1}_3$ did not happen;
- $\mathsf{AskH4w13}_3$: $\left(C\|\mathcal{S}\|j\|z\|p_j\|\mathsf{RSA}_{N,d}\left(z \times W^{-1}\right)\right)$ has been queried by $\mathcal{A}$ to $\mathcal{H}_4$ for some transcript $((C, j, z), (\mathcal{S}, V_S))$, where some party has accepted, but both $\mathsf{AskH1}_3$ and $\mathsf{AskH3w1}_3$ events did not happen.

After this modification, we do no longer need to know the value x nor to compute the value x' either. Thus we can simplify the corresponding rules:

► **Rule C2**$^{(3)}$
Generate a random pair $(\hat{x}, z = \mathsf{RSA}_{N,e}(\hat{x}))$.

► **Rule S1**$^{(3)}$
Do nothing.

Finally, one can notice that the actual password is not used any more. By the isomorphic property of $\mathsf{RSA}_{N,f}$ from $\mathbb{Z}_N^\star$ to $\mathbb{Z}_N^\star$, the new value z is perfectly indistinguishable from before, since there exists a unique pair (x, y),

$$x = \mathsf{RSA}_{N,d}\left(z \times W^{-1}\right) = \hat{x} \times \mathsf{RSA}_{N,d}\left(W^{-1}\right)$$
$$y = \mathsf{RSA}_{N,e}(x) \quad \text{such that } z = y \times W.$$

The authenticator is computed with a random oracle that is private to the simulator, then one can remark that it cannot be guessed by the adversary, better than at random for each attempt, unless the same partial transcript (j, z) appeared in another session with a real instance C^I. But such a case has already been excluded (in Game $\mathbf{G}_2$). Similarly, this can be applied to the session key.

$$\Pr[\mathsf{A}_3] \leq \frac{q_C}{2^{k_1}} \qquad \Pr[\mathsf{SwA}_3] = \frac{1}{2}. \tag{A·3}$$

Since collision of the partial transcript has been excluded, the event $\mathsf{AskH1}$ can be split in three disjoint sub-cases:

- $\mathsf{AskH1\text{-}Passive}_3$: the transcript $((C, j, z), (\mathcal{S}, V_S))$ comes from an execution between instances of C and $\mathcal{S}$ ($\mathsf{Execute}$-queries or forward of Send-queries, relay of part of them). In this case both (j, z) and the RSA key pair have been simulated where the latter has been provided through the Leak-query;
- $\mathsf{AskH1\text{-}WithS}_3$: the execution involved an instance of $\mathcal{S}$, but (j, z) has not been sent by any instance of C. This means that the RSA key pair has been simulated, but (j, z) has been produced by the adversary;
- $\mathsf{AskH1\text{-}WithC}_3$: the execution involved an instance of C, but an instance of $\mathcal{S}$ has not been involved during the attack. This means that both (j, z) and the RSA key pair have been simulated.

Game $\mathbf{G}_4$: In order to evaluate the above events, we introduce a random RSA instance (N, e, ρ), where ρ is a uniformly distributed random element in $\mathbb{Z}_N^\star$ (or equivalently, because of the isomorphism property, σ is randomly drawn from $\mathbb{Z}_N^\star$ and $\rho = \mathsf{RSA}_{N,e}(\sigma)$). We are looking for the value σ.
We introduce the instance (N, e, ρ) in the simulation of the oracle $\mathcal{G}$, using again the homomorphic property of RSA. Specifically, the simulation introduces values in the third and fourth elements of $\Lambda_{\mathcal{G}}$: the pre-image of the answer by $\mathsf{RSA}_{N,e}$, but does not use it. We modify the simulation as follows:

► **Rule** $\mathcal{G}^{(4)}$
Generate a random pair $(u, v = \mathsf{RSA}_{N,e}(u))$,
and with a random bit b compute $r \leftarrow v$,
if $b = 0$, and $r \leftarrow v \cdot \rho$, if $b = 1$. Then,
the record (j, q, r, u, b) is added to $\Lambda_{\mathcal{G}}$.

The probability remains unchanged because of the homomorphic property.

Game $\mathbf{G}_5$: It is now possible to evaluate the probability of the event AskH (or more precisely, the sub-cases). First, one can see that the password is never used during the simulation of $\mathbf{G}_4$. It does not need to be chosen in advance, but at the very end only. Then, an information-theoretic analysis can be done which simply uses cardinalities of some sets. This is crucial that the entire simulation is basically independent from the chosen password.
To this aim, we first exclude a few more games, wherein for some pair (j, z), involved in a communication between an instance $\mathcal{S}^J$ and either the adversary or an instance C^I, there exist two distinct values p_j, and

thus elements W, since $W = \mathcal{G}(j, p_j)$ such that both the tuples $(j, z, p_j, \mathsf{RSA}_{N,d}(z/W))$ are in $\Lambda_{\mathcal{H}}$ (which event is denoted CollH_5):

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_4]| \leq \Pr[\mathsf{CollH}_5]. \quad (A\cdot 4)$$

With the following lemma, the event CollH_5 can be upper-bounded.

Lemma 1: For any pair (j, z) involved in a communication with an instance $\mathcal{S}^J$, unless one can invert the RSA instance, there is at most one valid element W obtained from $\mathcal{G}$ such that $(j, z, p_j, \mathsf{RSA}_{N,d}(z/W))$ in $\Lambda_{\mathcal{H}}$:

$$\Pr[\mathsf{CollH}_5] \leq 2\mathsf{Succ}^{\mathsf{ow}}_{RSA}(q_h^2, t + 2q_h^2\tau_{law}). \quad (A\cdot 5)$$

Proof. We show the proof by contradiction. Here we assume that there exists (j, z) involved in a communication, and $W_0 = \mathsf{RSA}_{N,e}(u_0) \cdot b_0 \cdot \rho$ and $W_1 = \mathsf{RSA}_{N,e}(u_1) \cdot b_1 \cdot \rho$, both obtained from the $\mathcal{G}$ oracle, such that the tuples $(j, z, p_{ji}, \mathsf{RSA}_{N,d}(z/W_i))$ are in $\Lambda_{\mathcal{H}}$, for $i = 0, 1$. Then,

$$Z \stackrel{\text{def}}{=} \frac{\mathsf{RSA}_{N,d}(z/W_1)}{\mathsf{RSA}_{N,d}(z/W_0)} = \mathsf{RSA}_{N,d}(W_0/W_1). \quad (A\cdot 6)$$

With probability of $1/2$, the bits b_0 and b_1 involved in W_0 and W_1 are distinct. Without loss of generality, we may assume that $b_i = i$ for $i = 0, 1$:

$$Z = \mathsf{RSA}_{N,d}\left(\frac{\mathsf{RSA}_{N,e}(u_0)}{\mathsf{RSA}_{N,e}(u_1)\cdot\rho}\right) = \frac{u_0}{u_1 \cdot \mathsf{RSA}_{N,d}(\rho)} = \frac{u_0}{u_1 \cdot \sigma}. \quad (A\cdot 7)$$

As a consequence, $\sigma = u_0/(u_1 \cdot Z)$. By either guessing the two queries asked to the $\mathcal{H}_i$ or checking for each pair the validity of the computed σ, one concludes the proof. □

In order to conclude the proof, let us study separately the three sub-cases of AskH1, and then AskH3w1 and AskH4w13 (keeping in mind the absence of several kinds of collisions: for partial transcripts, for $\mathcal{G}$, and for p_j in $\mathcal{H}$-queries):

- AskH1-Passive: about the passive transcripts (in which both (j, z) and the RSA key pair have been simulated), one can state the following lemma:

  Lemma 2: For any pair (j, z) involved in a passive transcript, unless one can invert the RSA instance, there is no valid element W such that $(j, z, p_j, \mathsf{RSA}_{N,d}(z/W))$ in $\Lambda_{\mathcal{H}}$:

  $$\Pr[\mathsf{AskH1\text{-}Passive}_5] \leq 2\mathsf{Succ}^{\mathsf{ow}}_{RSA}(q_h, t + 2q_h\tau_{law}). \quad (A\cdot 8)$$

  Proof. Assume that there exist (j, z) involved in a passive transcript and $W = \mathsf{RSA}_{N,e}(u) \cdot b \cdot \rho$ such that the tuple $(j, z = \mathsf{RSA}_{N,e}(\hat{x}), p_j, Z \stackrel{\text{def}}{=} \mathsf{RSA}_{N,d}(z/W))$ is in $\Lambda_{\mathcal{H}}$. Then, as above, with probability of $1/2$, $b = 1$:

  $$Z = \mathsf{RSA}_{N,d}\left(\frac{\mathsf{RSA}_{N,e}(\hat{x})}{\mathsf{RSA}_{N,e}(u)\cdot\rho}\right) = \frac{\hat{x}}{u \cdot \mathsf{RSA}_{N,d}(\rho)} = \frac{\hat{x}}{u \cdot \sigma}. \quad (A\cdot 9)$$

  As a consequence, $\sigma = \hat{x}/(u \cdot Z)$. By either guessing the query asked to the $\mathcal{H}_i$ or checking the validity of σ for each candidate, one concludes the proof. □

- AskH1-WithS: the above Lemma 1, applied to games where the event CollH_5 did not happen (and without $\mathcal{G}$-collision), states that for each pair (j, z) involved in a transcript with an instance $\mathcal{S}^J$, there is at most one element p_j such that for $W = \mathcal{G}(j, p_j)$, the corresponding tuple is in $\Lambda_{\mathcal{H}}$. Thus, the probability for the adversary (who may have obtained α_j in the Leak-query, denoted by event Leak_5) over a random password is upper-bounded by:

  $$\Pr[\mathsf{AskH1\text{-}WithS}_5] \leq \frac{q_S}{2^l} + \frac{q_S}{D}. \quad (A\cdot 10)$$

- AskH1-WithC: this may correspond to an attack where the adversary tries to impersonate $\mathcal{S}$ to C (break unilateral authentication). But each authenticator sent by the adversary has been computed with at most one p_j. Thus, the above Lemma 2 also applies to this case:

  $$\Pr[\mathsf{AskH1\text{-}WichC}_5] \leq 2\mathsf{Succ}^{\mathsf{ow}}_{RSA}(q_h, t + 2q_h\tau_{law}). \quad (A\cdot 11)$$

About AskH3w1 (when the above three events did not happen), it means that only executions with an instance of $\mathcal{S}$ may lead to acceptance (and either C or the adversary). Exactly the same analysis as for AskH1-Passive and AskH1-WithS leads to

$$\Pr[\mathsf{AskH3w1}_5] \leq \frac{q_S}{2^l} + \frac{q_S}{D} + 2\mathsf{Succ}^{\mathsf{ow}}_{RSA}(q_h, t + 2q_h\tau_{law}). \quad (A\cdot 12)$$

About AskH4w13 (when both of the events AskH1 and AskH3w1 did not happen), the same analysis as for AskH3w1 leads to

$$\Pr[\mathsf{AskH4w13}_5] \leq \frac{q_S}{2^l} + \frac{q_S}{D} + 2\mathsf{Succ}^{\mathsf{ow}}_{RSA}(q_h, t + 2q_h\tau_{law}). \quad (A\cdot 13)$$

As a conclusion, we get an upper-bound for the probability of AskH_5 by combining all the cases:

$$\Pr[\mathsf{AskH}_5] \leq \frac{3q_S}{2^l} + \frac{3q_S}{D} + 8\mathsf{Succ}^{\mathsf{ow}}_{RSA}(q_h, t + 2q_h\tau_{law}). \quad (A\cdot 14)$$

Combining equations (A·2), (A·3), (A·5) and (A·14), one gets either

$$\Pr[\mathsf{A}_0] \le \frac{q_C}{2^{k_1}} + \Delta \qquad \Pr[\mathsf{SwA}_0] = \frac{1}{2} + \Delta, \quad (\text{A}\cdot 15)$$

where

$$\begin{aligned}\Delta \le\ & 2\mathsf{Succ}^{\mathsf{ow}}_{RSA}\left(q_h^2, t + 2q_h^2\tau_{law}\right) \\ & +8\mathsf{Succ}^{\mathsf{ow}}_{RSA}\left(q_h, t + 2q_h\tau_{law}\right) \\ & +\frac{3q_S}{D} + \frac{3q_S}{2^l} + \frac{(q_C+q_p)^2 + q_g'^2}{2^{l+1}} \\ \le\ & \frac{3q_S}{D} + \frac{6q_S + (q_C+q_p)^2 + q_g'^2}{2^{l+1}} \\ & +10\mathsf{Succ}^{\mathsf{ow}}_{RSA}\left(q_h^2, t + 2q_h^2\tau_{law}\right). \end{aligned} \quad (\text{A}\cdot 16)$$

One can get the result as desired by noting that $\Pr[\mathsf{S}_0] \le \Pr[\mathsf{SwA}_0] + \Pr[\mathsf{A}_0]$.

Appendix B: Proof of Theorem 2

Proof. In this proof, we incrementally define a sequence of games starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_4$. Here we describe differences only from Appendix A.

Game $\mathbf{G}_0$: This is the same as $\mathbf{G}_0$ of Appendix A.

Game $\mathbf{G}_1$: In this game, we modify the simulations of the hash oracle $\mathcal{G}$ and the Leak query in Figs. A· 1 and A· 2, respectively, as follows:

- For a hash-query $\mathcal{G}(j, q)$, such that a record (j, q, r) appears in $\Lambda_{\mathcal{G}}$, the answer is r. Otherwise one chooses a random element $r \xleftarrow{R} \mathbb{Z}_N^\star$, answers with it, and adds the record (j, q, r) to $\Lambda_{\mathcal{G}}$;
- A Leak($\mathcal{S}^J$)-query returns the stored secret (d, N) by the instance $\mathcal{S}^J$.

The remaining is the same as $\mathbf{G}_1$ of Appendix A. From this simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

Game $\mathbf{G}_2$: This is the same as $\mathbf{G}_2$ of Appendix A.

Game $\mathbf{G}_3$: This is the same as $\mathbf{G}_3$ of Appendix A except the sub-cases (AskH1-WithS and AskH1-WithC) of AskH1:

- AskH1-WithS_3: the execution involved an instance of $\mathcal{S}$, but (j, z) has not been sent by any instance of C. This means that the RSA key pair has been simulated and provided through the Leak-query, but (j, z) has been produced by the adversary;
- AskH1-WithC_3: the execution involved an instance of C, but an instance of $\mathcal{S}$ has not been involved during the attack. This means that both (j, z) and the RSA key pair have been simulated where the latter has been provided through the Leak-query.

Game $\mathbf{G}_4$: Actually, we don't need to introduce an RSA instance in the simulation of the oracle $\mathcal{G}$ since the adversary knows the RSA private key through the Leak-query.

We first exclude an event (denoted by CollH_4), wherein for some pair (j, z) there exist two distinct values p_j, and thus elements W, since $W = \mathcal{G}(j, p_j)$ such that both the tuples $(j, z, p_j, \mathsf{RSA}_{N,d}(z/W))$ are in $\Lambda_{\mathcal{H}}$. As a result, the probability of CollH_4 is bounded by the birthday paradox:

$$\Pr[\mathsf{CollH}_4] \le \frac{q_h^2}{2^{l+1}}. \quad (\text{A}\cdot 17)$$

This implies that for each pair (j, z) there is at most one element p_j such that, for $W = \mathcal{G}(j, p_j)$, the corresponding tuple is in $\Lambda_{\mathcal{H}}$.

In order to conclude the proof, let us evaluate separately the three sub-cases of AskH1, and then AskH3w1 and AskH4w13 (keeping in mind the absence of several kinds of collisions: for partial transcripts, for $\mathcal{G}$, and for p_j in $\mathcal{H}$-queries):

- AskH1-Passive: about the passive transcripts (in which both (j, z) and the RSA key pair have been simulated), the probability for the adversary over a random p_j is upper-bounded by:

$$\Pr[\mathsf{AskH1\text{-}Passive}_4] \le \frac{q_p}{2^l}. \quad (\text{A}\cdot 18)$$

- AskH1-WithS: this may correspond to an attack where the adversary tries to impersonate C to $\mathcal{S}$. But each z sent by the adversary has been computed with at most one p_j. Thus,

$$\Pr[\mathsf{AskH1\text{-}With}\mathsf{S}_4] \le \frac{q_S}{2^l}. \quad (\text{A}\cdot 19)$$

- AskH1-WithC: this may correspond to an attack where the adversary tries to impersonate $\mathcal{S}$ to C (break unilateral authentication). But each authenticator sent by the adversary has been computed with at most one p_j. Thus,

$$\Pr[\mathsf{AskH1\text{-}WichC}_4]\frac{q_C}{2^l}. \quad (\text{A}\cdot 20)$$

About AskH3w1 (when the above three events did not happen), it means that only executions with an instance of $\mathcal{S}$ may lead to acceptance (and either C or the adversary). Exactly the same analysis as for AskH1-Passive and AskH1-WithS leads to

$$\Pr[\mathsf{AskH3w1}_4] \le \frac{q_p + q_S}{2^l}. \quad (\text{A}\cdot 21)$$

About AskH4w13 (when both of the events AskH1 and AskH3w1 did not happen), the same analysis as for AskH3w1 leads to

$$\Pr[\mathsf{AskH4w13}_4] \le \frac{q_p + q_S}{2^l}. \quad (\text{A}\cdot 22)$$

As a conclusion, we get an upper-bound for the probability of AskH_4 by combining all the cases:

$$\Pr[\mathsf{AskH}_4] \leq \frac{3q_p + 3q_S + q_C}{2^l}. \quad (\text{A}\cdot 23)$$

Combining equations (A·2), (A·3), (A·17) and (A·23), one gets either

$$\Pr[\mathsf{A}_0] \leq \frac{q_C}{2^{k_1}} + \Delta \qquad \Pr[\mathsf{SwA}_0] = \frac{1}{2} + \Delta, \quad (\text{A}\cdot 24)$$

where

$$\Delta \leq \frac{3q_p + 3q_S + q_C}{2^l} + \frac{q_h^2 + (q_C + q_p)^2 + q_g'^2}{2^{l+1}}. \quad (\text{A}\cdot 25)$$

One can get the result as desired by noting that $\Pr[\mathsf{S}_0] \leq \Pr[\mathsf{SwA}_0] + \Pr[\mathsf{A}_0]$.

Appendix C: Proof of Theorem 4

Proof. In this proof, we incrementally define a sequence of games starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_4$. Here we describe differences only from Appendix A or Appendix B.

Game $\mathbf{G}_0$: This is the real protocol in the random oracle model. We are interested in the following event:

- S_0 (for semantic security) which occurs if the adversary correctly guesses the bit b involved in the Test-query;

$$\mathsf{Adv}_P^{\mathsf{pfs-ake}}(\mathcal{A}) = 2\Pr[\mathsf{S}_0] - 1. \quad (\text{A}\cdot 26)$$

Game $\mathbf{G}_1$: In this game, we modify the simulations of the hash oracle $\mathcal{G}$ and the Leak query in Figs. A·1 and A·2, respectively, as follows:

- For a hash-query $\mathcal{G}(j, q)$, such that a record (j, q, r) appears in $\Lambda_{\mathcal{G}}$, the answer is r. Otherwise one chooses a random element $r \xleftarrow{R} \mathbb{Z}_N^\star$, answers with it, and adds the record (j, q, r) to $\Lambda_{\mathcal{G}}$;
- A $\mathsf{Leak}(U)$-query returns the stored secret $(\alpha_{j+1}, (e, N))$ (resp., (d, N)) by the instance C^I (resp., $\mathcal{S}^J$).

We also simulate the $\mathsf{Corrupt}$ query and add it to Fig. A·2.

- A $\mathsf{Corrupt}(U)$-query returns the secret pw (resp., p_{j+1}) by the instance C^I (resp., $\mathcal{S}^J$).

The remaining is the same as $\mathbf{G}_1$ of Appendix A. From this simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

Game $\mathbf{G}_2$: This is the same as $\mathbf{G}_2$ of Appendix A.

Game $\mathbf{G}_3$: In order to make the authenticators and the session keys unpredictable to any adversary, we compute them using the private oracles $\mathcal{H}_1'$ and $\mathcal{H}_3'$ (instead of $\mathcal{H}_1$ and $\mathcal{H}_3$), respectively, so that the values are completely independent from the random oracles as well as $\mathcal{G}$. We reach this aim by using the following rules:

► **Rule C3/S2**$^{(3)}$
Compute the authenticator
$V_S \leftarrow \mathcal{H}_1'(C\|\mathcal{S}\|j\|z)$.
Compute the session key
$SK_{C/S} \leftarrow \mathcal{H}_3'(C\|\mathcal{S}\|j\|z)$.

We also use the private oracle $\mathcal{H}_4'$ (instead of $\mathcal{H}_4$) so that the refreshed secrets are unpredictable to any adversary as well. We modify the simulation by using the following rules:

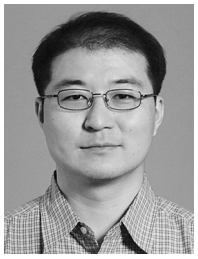
► **Rule C1**$^{(3)}$
Do nothing.

► **Rule C4/S3**$^{(3)}$
Compute $\alpha_{j+1} = \alpha_j + \mathcal{H}_4'(C\|\mathcal{S}\|j\|z)$ and
$p_{j+1} = p_j + \mathcal{H}_4'(C\|\mathcal{S}\|j\|z)$.

This modification makes the simulation consistent with the Leak and $\mathsf{Corrupt}$-queries. The remaining is the same as $\mathbf{G}_3$ of Appendix A except the sub-cases ($\mathsf{AskH1\text{-}WithS}$ and $\mathsf{AskH1\text{-}WithC}$) of $\mathsf{AskH1}$:

- $\mathsf{AskH1\text{-}WithS}_3$: the execution involved an instance of $\mathcal{S}$, but (j, z) has not been sent by any instance of C. This means that the RSA key pair has been simulated and provided through the Leak-query, but (j, z) has been produced by the adversary;
- $\mathsf{AskH1\text{-}WithC}_3$: the execution involved an instance of C, but an instance of $\mathcal{S}$ has not been involved during the attack. This means that both (j, z) and the RSA key pair have been simulated where the latter has been provided through the Leak-query.

Game $\mathbf{G}_4$: Since α_j and p_j are completely independent from α_{j+1} and p_{j+1}, the exact same analysis can be done as in $\mathbf{G}_4$ of Appendix B.



SeongHan Shin received the B.S. and M.S. degrees in computer science from Pukyong National University, Busan, Korea, in 2000 and 2002, respectively. In 2005, he received his Ph.D. degree in information and communication engineering, information science and technology from the University of Tokyo, Tokyo, Japan. From October 2005 to March 2006, he has joined the Institute of Industrial Science of the University of Tokyo as a post-doctoral researcher. From April 2006, he has been working for the Research Center for Information Security, National Institute of Industrial Science and Technology, Japan, as a researcher of the Research Team for Security Fundamentals. He received the CSS Student Paper Award and the IWS2005/WPMC'05 Best Student Paper Award in 2003 and 2005, respectively. His research interests include information security, cryptography and wireless security.

Kazukuni Kobara received his B.E. degree in electrical engineering and M.E. degree in computer science and system engineering from the Yamaguchi University in 1992, 1994, respectively. He also received his Ph.D. degree in engineering from the University of Tokyo in 2003. From 1994 to 2000 and 2000 to 2006 he was a technical associate and a research associate respectively for the Institute of Industrial Science of the University of Tokyo. In 2006, he joined the National Institute of Advanced Industrial Science and Technology (AIST) where he was the leader of the Research Team for Security Fundamentals in the Research Center for Information Security (RCIS). Currently he is a chief researcher at RCIS. His research interests include cryptography, information and network security. He received the SCIS Paper Award and the Vigentennial Award from ISEC group of IEICE in 1996 and 2003, respectively. He also received the Best Paper Award of WISA, the ISITA Paper Award for Young Researchers, the IEICE Best Paper Award (Inose Award), the WPMC Best Paper Award and the JSSM Best Paper Award in 2001, 2002, 2003, 2005 and 2006 respectively. He is a member of IACR. He served as a member of CRYPTREC (2000-present) and the vice chairperson of WLAN security committee (2003).

Hideki Imai was born in Shimane, Japan on May 31, 1943. He received the B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Tokyo in 1966, 1968, and 1971, respectively. From 1971 to 1992 he was on the faculty of Yokohama National University. From 1992 to 2006 he was a Professor at the Institute of Industrial Science, the University of Tokyo. In 2006 he was appointed as an Emeritus Professor of the University of Tokyo and a Professor of Chuo University. Concurrently he serves as the Director of Research Center for Information Security, National Institute of Advanced Industrial Science and Technology. His current research interests include information theory, coding theory, cryptography, and information security. From IEICE (the Institute of Electronics, Information and Communication Engineers), Dr. Imai received Best Book Awards in 1976 and 1991, Best Paper Awards in 1992, 2003 and 2004, Yonezawa Memorial Paper Award in 1992, Achievement Award in 1995, Inose Award in 2003, and Distinguished Achievement and Contributions Award in 2004. He also received Golden Jubilee Paper Award from the IEEE Information Theory Society in 1998, and Official Commendations from the Minster of Internal Affairs and Communications in June 2002 and from the Minister of Economy, Trade and Industry in October 2002. He was awarded Honor Doctor Degree by Soonchunhyang University, Korea in 1999 and Docteur Honoris Causa by the University of Toulon Var, France in 2002. He is also the recipient of the Ericsson Telecommunications Award 2005. Dr. Imai is a member of the Science Council of Japan. He was elected an IEEE Fellow in 1992. He has chaired many committees of scientific societies and organized a number of international conferences. He served as the President of the Society of Information Theory and its Applications in 1997, of the IEICE Engineering Sciences Society in 1998, and of the IEEE Information Theory Society in 2004. He is currently the Chair of CRYPTREC (Cryptography Techniques Research and Evaluation Committee of Japan).