The Journal of
Mathematics and Computer Science

Available online at
http://www.TJMCS.com

The Journal of Mathematics and Computer Science Vol .3 No.4 (2011) 357 - 375

Using Identity-Based Secret Public Keys Cryptography for Heuristic Security Analyses in Grid Computing

Seyed Hossein Kamali[1], Maysam Hedayati[2], Reza Shakerian[3],Saber Ghasempour[4]

[1] Islamic Azad University of Qazvin Branch, Qazvin, Iran
[2] Islamic Azad University-Ghaemshahr Branch, Ghaemshahr, Iran
[3] Payame Noor University,PO BOX 19395-3697 ,Tehran , Iran
[4]Department of Mathematics, Payame Noor University,PO BOX 19395-3697 ,Tehran , Iran

Received: September 2011, Revised: November 2011
Online Publication: December 2011

Abstract

The majority of current security architectures for grid systems use public key infrastructure (PKI) to authenticate identities of grid members and to secure resource allocation to these members. Identity-based secret public keys have some attractive properties which seem to align well with the demands of grid computing. In this Paper, we proposed identity-based secret public keys. Our new identity-based approach allows secret public keys to be constructed in a very natural way using arbitrary random strings, eliminating the structure found in, for example, RSA or Diffie-Hellman keys. We examine identity-based secret public key protocols and give informal security analyses which show that they may well be secure against online password guessing and other attacks. More importantly, we present an identity-based secret public key version of the standard TLS protocol. Our new protocol allows passwords to be tied directly to the establishment of secure TLS channels.

Keywords: Grid Computing, ID-SPK, Three-Party, Two-Party ID-SPK, TLS Protocol, Key exchange.

1. Introduction

The term grid was first used in the mid-1990s to denote a distributed computing infrastructure for advanced science and engineering applications [1]. It implies a common and uniform means of providing computer programs with the required amount of computational resources, analogous to providing electrical power to household appliances. That said the concept of sharing distributed resources is not new. We refer to *grid computing* as a type of distributed computing technology, while a *computational grid* indicates the physical infrastructure that supports grid computing. Note that we also use a more general term *grid* to represent both grid technology and infrastructure.

The use of *secret public keys* in password-based authentication protocols was first proposed by Gong *et al.* [2] in 1993. As implied by its name, a secret public key is a standard public key which can be generated by a user or a server, and is known only to them but is kept secret from third parties. A secret public key within a password-based protocol, when encrypted with a user's password, should serve as an unverifiable text1. This may significantly increase the difficulty of password guessing even if it is a poorly chosen password as an attacker has no way to verify if he has made the correct guess. The secret public key can then be used by the user for encrypting protocol messages. However, it may not be easy to achieve unverifiability of text by simply performing naive symmetric encryption on public keys of standard types such as RSA or Diffie-Hellman. This was overlooked in [2] and other variants of secret public key protocols in [3], but later found to be the main culprit in various attacks on the protocols. These include undetectable on-line password guessing attacks of Ding and Horster [4] and number theoretic attacks due to Patel [5]. It is worth noting that the attacks discovered in [4] may not work against a secret public key protocol which uses a secure public key encryption scheme such as RSA-OAEP [6]. Nevertheless, Patel's attacks seem to be one of the crucial factors that caused diversion of interest away from using secret public keys in password-based protocols. The concept of secret public keys, therefore, was thought to be obsolescent. For example, in more recent work on password-based protocols that requires servers' public keys2, it is assumed that the public keys are fixed and known to all users. Independent of the previous work on secret public key protocols, Steiner *et al.* [7] proposed a method for integrating password-based key exchange protocols with the TLS protocol. Simple password authentication through a secure TLS channel is widely used in email and e-commerce applications between a user and a web server. It can be achieved by first establishing secrecy and integrity protected channel between the user's client (e.g. a web browser) and a remote server.

Subsequently, the server authenticates the user by verifying the submitted user name and password. Note that users within a grid environment make use of exactly the same technique when accessing the My Proxy server to request their proxy credentials. In such a set-up where the user enters his password only after the secure channel is established, the authentication of the user (through his password) is not directly tied to the secure channel. This provided the motivation for the password-based TLS protocol proposed in [7]. However, the proposal of Steiner *et al.* requires significant alterations to the structure of the TLS handshake protocol, an undesirable property which limits the acceptability of their proposal. The aims of this chapter are twofold: (i) revisit the notion of secret public keys and uncover some unexplored potential benefits of using identity-based secret public keys, through IBC, in password-based protocols; and (ii) show how identity-based secret public keys can support the use of passwords in the TLS protocol in a more natural and less disruptive way than was proposed in [7].

In our quest to revive the notion, we introduce some new properties for secret public keys. In the IBC setting, we show that an identity-based secret public key can offer more flexibility in terms of key distribution. For example, an identity-based secret public key can be computed by a user on-the-fly without needing his authentication server to transport the key to him. More importantly, a random string can be used as the identifier for constructing a secret public key. This technique can offer a clean and natural way of eliminating any predictable structure in the secret public key. Through this, the number theoretic attacks that plague existing secret public key protocols can easily be prevented. Since both public and private keys in the IBC setting are kept secret, we also propose the notion of *secret signatures* which seem to provide data confidentiality in addition to their original cryptographic use, i.e. authentication and non-repudiation. This appears to provide additional properties in conventional secret public key protocols and in password-based authentication protocols in general. The TLS protocol is becoming increasingly ubiquitous for web-based applications that require secure authentication and key establishment. Our identity-based approach to the concept of secret public keys may well be of significant practical value when it is integrated with the standard TLS handshake protocol. We design a TLS compatible identity-based secret public key protocol which requires no structural or message flow modification but only minimal changes or extensions to the message contents.

2. Related Work

Extensive work on password-based key exchange protocols (which rely on user passwords only) has already been carried out. See for example [8, 9, 10, 11, 15], which all originate from [21, 22]. In order to circumvent off-line password guessing attacks, Bellare et al. [12, 13] proposed the use of a mask generation function E(0) as an instantiation of the encryption primitive for encrypting a Diffie-Hellman component, rather than using a standard block (or stream) cipher. For instance, a user with his password PW can encrypt a Diffie-Hellman component g^x by calculating $g^x \cdot H(PW)$, where H is a hash function mapping onto the Diffie-Hellman group and which is modeled as a random oracle in security proofs. Thus the result of the encryption is a group element. This special encryption primitive, which needs to be carefully implemented, is crucial in preventing any leakage of information about the password when an attacker mounts a guessing attack. To decrypt and recover g^x, one can simply divide the cipher text by H (PW). All recent work, such as [8, 9,10], utilizes this encryption primitive for their password-based key exchange protocols. Our proposal using identity-based techniques can be seen as a novel alternative to these current protocols. In addition, the identity-based techniques can be integrated naturally with the TLS handshake protocol, which seems to be difficult to achieve using current Diffie-Hellman encrypted key exchange techniques without more radical modification of the TLS handshake.

The use of algorithms from a public key encryption scheme in a secret/symmetric key setting is not new. In 1978, Hellman and Pohlig [18] introduced the Pohlig-Hellman symmetric key cipher based on exponentiation. Two different keys are involved in the symmetric key cipher, namely, a secret encrypting key e for the sender and a secret decrypting key d for the receiver, where $e \neq d$. obviously, the communicating parties must agree in advance to share these two symmetric keys. In more recent work, Brincat [16] investigated how shorter RSA public/private key pairs can be used securely in the secret key world. This is slightly different from [15], as each user has his own secret public/private key pair in [9]. Another related concept is that of public key privacy from Bellare *et al.* [6]. The notion of indistinguishability of keys in public key privacy is an extension of the cipher text privacy concept: given a set of public keys and a ciphertext generated by using one of the keys, the adversary cannot tell which public key was used to generate the ciphertext. In this chapter, we will make use of identity-based (secret) public keys in the secret key setting. These public keys are known only to the senders and receivers, and thus indistinguishability of encryptions and keys somewhat similar to [4] can be achieved. Moreover, in such a setting, a signature can be made verifiable to only a specific recipient, hence the moniker *secret signature*. In many ways, the concepts of secret public key encryption and signatures seem to be closely related to the notion of signcryption with key privacy from Libert and Quisquater [19]. The proposal of [16] combined Zheng's work on signcryption [20] and the key privacy concept of [4]. Our concept of secret signatures is also related to the strongest security notion for undeniable and confirmer signatures called invisibility in [10]. Extensive work on password-based key exchange protocols (which rely on user passwords only) has already been carried out. See for example [8, 9, 10, 11, 15], which all originate from [21, 22]. In order to circumvent off-line password guessing attacks, Bellare *et al.* [12, 13] proposed the use of a mask generation function $E(0)$ as an instantiation of the encryption primitive for encrypting a Diffie-Hellman component, rather than using a standard block (or stream) cipher. For instance, a user with his password PW can encrypt a Diffie-Hellman component g^x by calculating $g^x \cdot H(PW)$, where H is a hash function mapping onto the Diffie-Hellman group and which is modeled as a random oracle in security proofs. Thus the result of the encryption is a group element. This special encryption primitive, which needs to be carefully implemented, is crucial in preventing any leakage of information about the password when an attacker mounts a guessing attack. To decrypt and recover g^x, one can simply divide the cipher text by H (PW). All recent work, such as [8, 9,10], utilizes this encryption primitive for their password-based key exchange protocols. Our proposal using identity-based techniques can be seen as a novel alternative to these current protocols. In addition, the identity-based techniques can be integrated naturally with the TLS handshake protocol, which seems to be difficult to achieve using current Diffie-Hellman encrypted key exchange techniques without more radical modification of the TLS handshake.

The use of algorithms from a public key encryption scheme in a secret/symmetric key setting is not new. In 1978, Hellman and Pohlig [18] introduced the Pohlig-Hellman
symmetric key cipher based on exponentiation. Two different keys are involved in the symmetric key cipher, namely, a secret encrypting key e for the sender and a secret decrypting key d for the receiver, where $e \neq d$. obviously, the communicating parties

must agree in advance to share these two symmetric keys. In more recent work, Brincat [16] investigated how shorter RSA public/private key pairs can be used securely in the secret key world. This is slightly different from [15], as each user has his own secret public/private key pair in [9]. Another related concept is that of public key privacy from Bellare *et al.* [6]. The notion of indistinguishability of keys in public key privacy is an extension of the cipher text privacy concept: given a set of public keys and a ciphertext generated by using one of the keys, the adversary cannot tell which public key was used to generate the ciphertext. In this chapter, we will make use of identity-based (secret) public keys in the secret key setting. These public keys are known only to the senders and receivers, and thus indistinguishability of encryptions and keys somewhat similar to [4] can be achieved. Moreover, in such a setting, a signature can be made verifiable to only a specific recipient, hence the moniker *secret signature*. In many ways, the concepts of secret public key encryption and signatures seem to be closely related to the notion of signcryption with key privacy from Libert and Quisquater [19]. The proposal of [16] combined Zheng's work on signcryption [20] and the key privacy concept of [4]. Our concept of secret signatures is also related to the strongest security notion for undeniable and confirmer signatures called invisibility in [10].

3. Secret Public Key Protocols and Attacks

In this section, we revisit the first secret public key protocol proposed in the literature [8]. We will explain what the problems are with the protocol. This will motivate our introduction of identity-based techniques to this area.

Notation: We use $\hat{P}K$ and $\hat{S}K$ to represent a secret public key (SPK henceforth) and its matching private key, respectively. These are no different from conventional Asymmetric keys except that they are *both* kept secret. *PW* denotes a password-derived symmetric key which is shared between a user and an authentication server. A nonce and a random number are represented by n and r, respectively. We use the notation $Enc_{PK}(.)$ to indicate asymmetric encryption using a secret public key *PK* and $\{.\}_K$ for symmetric encryption under a symmetric key *K.* In the three-party scenarios that we will discuss in this section, we use *A* and *B* to denote two communicating parties, while *S* denotes a trusted authentication server whose role is to distribute a copy of a randomly generated session key to both *A* and *B.* Other notations will be introduced as they are needed.

The GLNS SPK Protocol [2], Assuming *A* and *B* share their respective passwords with the authentication server *S*, the server can distribute fresh copies of public keys to *A* and *B* encrypted using their respective passwords as symmetric keys at the beginning of each protocol run. Each public key is only known between the server and the relevant participant. This seems to make traditional chosen plaintext attacks more difficult, as the encryption keys are not known to the attacker. The details of the SPK protocol of [9] are depicted in Protocol 1.

$$
\begin{array}{lll}
(1). & A \to S: & A,\ B \\
(2). & S \to A: & A,\ B,\ n_s,\ \{pK_{SA}\}PW_A,\ \{PK_{SB}\}PW_B \\
(3). & A \to B: & Enc_{PK_{SA}}(A,\ B, n_{A1}, n_{A2}, c_A,\ \{ns\}PW_A), \\
 & & n_S,\ r_A,\ \{PK_{SB}\}_{PW_B} \\
(4). & B \to S: & Enc_{PK_{SA}}(A,\ B, n_{A1}, n_{A2}, c_A, \{ns\}PW_A), \\
 & & Enc_{PK_{SB}}(B,\ A, n_{B1}, n_{B2}, c_B, \{ns\}PW_B) \\
(5). & S \to B: & \{n_{A1},\ K_{AB} \oplus n_{A2}\}PW_A,\ \{n_{B1},\ K_{AB} \oplus n_{B2}\}PW_B \\
(6). & B \to A: & \{n_{A1},\ K_{AB} \oplus n_{A2}\}PW_A,\ \{H(r_A),\ r_B\}\ K_{AB} \\
(7). & A \to B: & \{H(r_B)\}K_{AB}
\end{array}
$$

Figure 1. Protocol 1: The GLNS SPK Protocol

As shown in Protocol 1, *S* generates two new sets of secret public/private key pairs *(PK_{SA}, $SK_{S\ A}$), ($PK_{S\ B}$, $SK_{S\ B}$)* and distributes the public components to *A* in encrypted form whenever *A* initiates the protocol run. Here, c_A and c_B are sufficiently large random numbers known as confounders. They serve no purpose other than to confound guessing attacks based on some verifiable texts. Also, *H* is assumed to be a well-designed hash function.

Patel's Attacks [5], assuming *$\{e\}_{PW}$,* there are various number theoretic attacks that would reveal the password *PW.* For example, the attacker could expect the decryption of *$\{e\}_{PW}$* under a guess *PW'_A* to be an odd integer; an even result would eliminate *PW_A* as a possible password. Thus, some countermeasures against these number theoretic attacks such as padding or randomization of the RSA exponent are inevitably required.

Patel [5] showed that even when module *N* are sent in clear, and *e* are randomized and padded, there is still a lethal off-line guessing attack. Protocol 2 illustrates Patel's RSA version of the SPK protocol. We only show the first 3 out of 2 protocol messages as this is sufficient to describe Patel's attack.

$$
\begin{array}{lll}
(1). & A \to S: & A,\ B \\
(2). & S \to A: & A,\ B,\ n_s,\ \{e_{SA}\}PW_A,\ N_A,\ \{e_{SB}\}PW_B, N_B \\
(3). & A \to B: & Enc_{PK_{SA}}(A,\ B, n_{A1}, n_{A2}, c_A, \{ns\}PW_A), \\
 & & n_S,\ r_A,\ \{e_{SB}\}\ PW \\
 & . & . \\
 & . & . \\
 & . & .
\end{array}
$$

Figure 2. Protocol 2: RSA SPK Protocol

An attacker can impersonate *S* and block *A's* communication with the real authentication server to mount the following attack.

1- When the attacker *E* detects *A* is sending message (1) to *S,* he blocks *S's* response from reaching *A. E* intercepts message (2) and replaces *N_A* with his own *N'_A* whose prime factors he knows. Also, since *E* does not know *PW_A,* he simply replaces *$\{e_{SA}\}_{PW_A}$* with a random string *R_A.*

2- *A* unwittingly decrypts *R_A* with her password-derived key *PW_A* and obtains *e'_{SA}* which *A* believes was generated by *S.* Subsequently in message (3), *A* forwards *Enc_{eSB} (A, B, ...) to B.*

3- E intercepts message (3) and can now perform off-line password guessing on R_A. For each possible PW_A, E decrypts R_A and retrieves a possible value for e'_{SA}. Since E knows the prime factors of N'_A, he has no problem com putting the decryption exponent d'_{SA} for each value of e'_{SA}. By decrypting $Enc_{ejSA}(A, B, ...)$ with d'_{SA} and checking if the plaintext is of the form $(A, B, ...)$, E can test if PW'_A is the correct password.
It was pointed out in [11] that the above attack on the RSA-based SPK protocol is unavoidable unless all protocol participants use an agreed-upon RSA modulus, or unless the protocol is radically modified.
Even supposing a discrete logarithm based SPK protocol was used, and the cipher-text (which contains a secret public key) transmitted to A was then of the form $\{g^x\}pw$, where g is a generator of a subgroup of Z; of prime order q and a; is a random integer, the password can still be discovered. If a naive encryption of elements in the subgroup is performed with a standard block (or stream) cipher, then there is an off-line password guessing attack. The attacker simply decrypts $\{g^x\}_{PW}$ with a guessed password and observes if the resulting plaintext is an element of the subgroup. If it was an incorrect guess, the likelihood that g^x is not an element of the subgroup is at least $(p - q)/p > 1/2$. This attack can only be prevented by ensuring that decryption of $\{g^x\}_{PW}$ with a guessed password PW' always results in an element of the subgroup. Furthermore, it is also essential that public parameters such as g, p and q have been agreed *a priori* among the users. More examples and discussion on this subject can be found in [10, 18].

4. New Properties from Identity-Based Secret Public Keys

Pre-distribution or fixing of some public/system parameters is common in password-based protocols. In the following sections, we assume that the system parameters for the Boneh-Franklin IBE and the Zhang-Susilo-Mu IBS schemes can be distributed by the server to all its users during the user registration phase using an out-of-band mechanism. This is important as failure to use an authentic set of system parameters would allow the attacker to inject his own chosen parameters. Also, during the registration phase between a user and the server, the user will pick a password *pwd* and send an image *PW* of the password to the server. Typically, one might set $PW = H_0(pwd)$. P, where H_0: $\{0,1\}^* \rightarrow Z_q^*, G_1$ is a group of prime order q used elsewhere in the protocol, and P generates G_1. Note then that the server only knows *PW* and not *pwd.* The actual password *pwd* still remains private to the user only. In some cases where *pwd* and *PW* are used together, stronger authentication can be provided in the sense that the user's authenticity can still be guaranteed even if the string *PW* stored in the server is revealed. This technique of using an image of the actual user-selected.
Password is common to many password-based protocols, for example [8, 11, 19, 22]. Here, we present and discuss some interesting properties of identity-based SPKs (ID-SPKs henceforth) which are new as compared to conventional SPKs based on RSA or Diffie-Hellman primitives. These properties can be obtained from using the Boneh-Franklin and Zhang-Susilo-Mu schemes, and they form the basis and motivation for the ID-SPK protocols that we will discuss in Section V.

4.1. ID-SPK as Secret Identifier

In the conventional IBC setting, an identifier refers to some public information which represents a user and is known to all parties. Here, however, we work with secret identifiers, that is, identifiers only known to the user *A* (or *B)* and the server *S.* These can be obtained by binding a secret value such as a password to an identifier. Such an ID-SPK of the form *PK* = H*!* (User || password || policy) can be generated by both the user and the server on-the-fly. Here policy denotes constraints that can be included in the ID-SPK such as a date, nonces, or roles. In other words, the server does not need to distribute a fresh secret public key to its users, in contrast to [12, 21]. Here we assume the users have access to the server's fixed system parameters. For example, referring back to Protocol 1, when *A* initiates the protocol she could, in principle, skip messages (1) *&* (2) and transmit message (3) to *B* as follows:

$$(3).\ A \rightarrow B:\quad Enc_{PK_{AS}}(A,\ B, \ldots)$$

Figure 3. Message 3

Where $PK_{AS} = H(A||S||PW_A||$ "10102005") denotes a public key in the IBE scheme of [7]. Here "10102005" represents a date. A date with more granularity (e.g. concatenated with time) or a nonce may well be needed to ensure freshness *of PK as.* We remark that the Boneh-Franklin IBE scheme is IND-ID-CPA secure and thus the attacker cannot use a guessed password PW'_A to verify his guess by generating *Enc*$_{PK}$*(.)Snc*p> *(A,B,...)* and comparing it with the actual ciphertext produced by *A,* even if he knows all the plaintext components. We also assume that given the ciphertext that *A* produced, the attacker should learn nothing about the encryption key, i.e. PK_{AS}.
On the server side, the server can extract the matching private key for PK_{AS} using its master secret. Unless the attacker can break the IBE scheme or recover the master secret, the above ciphertext is resistant to password guessing attacks. This identity-based technique offers a form of non-interactive distribution of secret public keys from the server to its users.

4.2. Random String as ID-SPK

We have explained earlier in Section III that a naive encryption of an RSA exponent or a group element with a standard block cipher would lead to effective off-line password guessing attacks. Therefore, some form of padding or randomization of the keys is needed. In the IBC setting, we note that a random string with arbitrary length without any predictable structure can also be used as an identifier. The corresponding public key is usually derived from this identifier by hashing. Since now only a random string needs to be encrypted under the user password, the possibility of using a standard block cipher for the encryption is opened up. For example, in Protocol 1, the server can transport random strings ST_A and ST_B to *A* and *B*, respectively, in message (2) as follows:

$$\begin{aligned}
&(2).\quad S \to A: && A,\ B,\ n_s,\ \{ST_A\}PW_A,\ \{ST_B\}PW_B \\
&(3).\quad A \to B: && Enc_{PK_{SA}}(A,\ B, n_{A1}, n_{A2}, c_A), ns,\ r_A, \\
& && \{ST_B\}PW_B \\
&(4).\quad B \to S: && Enc_{PK_{SA}}(A,\ B, n_{A1}, n_{A2}, n_s), \\
& && Enc_{PK_{SB}}(B,\ A, n_{B1},\ n_{B2}, n_S)
\end{aligned}$$

Figure 4. Message 2

Since ST_A and ST_B are just random strings, they do not contain any predictable structure which could leak some information to the attacker as in the case of RSA or Diffie-Hellman keys. Subsequently users *A* and *B* can derive their ID-SPKs $PK_{SA}=H_1\ (A||S||ST_A)$ and $PK_{SB} = H_1\ (B||S||ST_B)$, respectively and respond to the server via messages (3) and (4). If the server can decrypt *B's* reply and recover n_s from both the ciphertexts produced with PK_{SA} and PK_{SB}, it can be assured that the users have received the correct random strings. Thus, *A* and *B* are authenticated to *S.* The use of random strings as identifiers is a key property from our identity-based approach which may give the concept of SPK protocols new life.

We remark that to prevent off-line attacks, ciphertexts obtained by encryption under the keys PK_{SA} and PK_{SB} must not leak useful information about ST_A and ST_B, respectively. This is not a traditional requirement of a public key encryption scheme (it is related to the public key privacy concept in [4]). Also note that since we use a probabilistic encryption scheme here, we have removed the use of confounder's c_A and c_B originally proposed in Protocol 1 in messages (3) and (4). Furthermore, users *A* and *B* no longer need to encrypt n_s with their respective passwords in their replies to *S,* in messages (3) and (4). This is because users *A* and *B* can demonstrate knowledge of their respective passwords by their ability to construct correct keys from ST_A and ST_B.

4.3. Secret Signatures

In what follows, we show some extended properties that an ID-SPK can offer as compared to a conventional SPK. Again, referring to Protocol 1, if in the protocol *A* (and *B)* selects and sends ST_A (and ST_B) to the server (rather than the server sending it to the user), we can, in principle, remove messages (1) *&* (2) and modify messages (3) - (5) as follows:

$$\begin{aligned}
&(3).\quad A \to B: && Enc_{PK_{AS1}}(A,\ B, ST_A), r_A \\
&(4).\quad B \to S: && Enc_{PK_{SA1}}(A,\ B, ST_A), r_A, \\
& && Enc_{PK_{SB1}}(A,\ B,\ ST_B),\ r_B \\
&(5).\quad S \to B: && Sig_{SK_{SA2}}(K_{AB}),\ Sig_{SK_{SB2}}(K_{AB})
\end{aligned}$$

Figure 5. Messaage (3) – (5)

Note that we have replaced nonces n_{A1}, n_{A2}, n_{B1} and n_m in Protocol 1 by random strings ST_A and ST_B. For ease of exposition, we concentrate on the interaction between *A* and *S.* In message (3), *A* encrypts a random string ST_A with an ID-SPK $PK_{SA1}= H_1(A||S||PW_A)$. It is obvious that a symmetric encryption of the form $\{A, B,..., ST_A\}_{PWa}$ cannot be used in message (3) because the identities of *A* and *B* are verifiable texts. The server responds with a signature generated with a private key associated with the public key $PK_{SA2}= H_1$

$(S||A||PW_A||ST_A)$. The reason for doing this will be clear when we look at the motivation for using *$Sig_{SK}(.)$,* a signature scheme with a private key *SK,* in message (5). As compared to the modification of Protocol 1 given in Section B, the server cannot reply to *A* with an encrypted message using an ID-SPK constructed from *$H_1(S||A||PW_A||ST_A)$.* This is mainly because in such an asymmetric model (where the user only knows an easy-to-remember password and the server has access to the secret public/private key pairs), only the server itself can extract the corresponding private key. This prompts the requirement to use a secret signature which not only provides non-repudiation of the signed message and message recovery, but also preserves message confidentiality. This last property is needed because the server wants only *A* and *B* to be able to verify the signatures and recover the signed messages. This, in turn, leads us to the use of an ID-SPK signature scheme with message recovery which can be adapted from [19]. So long as the verification keys used in the scheme of [19] are kept secret between the intended parties, our concept of secret signatures can be used. However, we remark that the IBS scheme with message recovery must be used carefully because the scheme provides message integrity. In other words, a simple off-line password guessing attack would be enabled if a secret signature was created based on a private key corresponding to *PK_{SA2}= H_1 $(A||S||PW_A)$.* For instance, the attacker could construct an ID-SPK *$PK'_{SA2} = Hi$ $(A||S||PW_A)$* using a guessed password *PW'_A* and then attempt to verify the signature. If he used the wrong password, the Verify algorithm would return an error message. Because of that, the identifier from which the verifying key is derived must contain a secret value chosen from a space much larger than the password space. We achieve this by including *ST_A* (or *ST_B)* in the identifier. It is also worth mentioning that a secret signature should not leak information about the signing key, the verifying key, or the plaintext that has been signed.

5. The ID-SPK Protocols

This section presents three-party and two-party ID-SPK protocols which can solve this structural issue in a clean and natural way. We assume that all the protocol participants have agreed on some system parameters for the ID-SPK encryption and signature schemes *a priori*.

Security Model. We sketch here our definition of the security for a password-based ID-SPK protocol, using an informal security model. In the model, there is an adversary *E,* who is allowed to watch regular runs of the protocol between a user, $U \in u$ where *u* is a set of protocol users, and a server *S. E* can actively communicate with the user and the server in replay, impersonation, and man-in-the-middle attacks. The adversary can prompt one of the parties to initiate new sessions. In each session, *E* can see all the messages sent between *U* and *S.* Furthermore, he can intercept the messages and modify or delete them. Also, *E* gets to see whether *S* accepts the authentication or not. In addition, we allow the adversary to establish as many “accounts” as he wishes with the server using his own chosen passwords. He can then run arbitrarily many authentication sessions using these accounts to obtain information for his attacks.

It is clear that if the user picks a password from his dictionary *V,* then the adversary that attempts *n* active impersonation attacks (or on-line guessing attacks) over *n* distinct sessions with the server can succeed with probability at least *n/|D|* by trying a different password from *D* in each attempt.

Definition 1 (*Informal*) *We say that the ID-SPK protocol is secure if all the following conditions are satisfied.*

- No useful information about a session key is revealed to the adversary during a successful protocol run and the exposure of past session keys does not leak any information about the current session key.
- The adversary cannot discover the correct user password after n active impersonation attempts with probability significantly higher than n/|D|.
- The protocol is resistant to off-line password guessing attacks.
- The protocol is resistant to undetectable on-line password guessing attacks.
- The exposure of the user's past session keys will not lead to the exposure of the user's password and vice versa.

The above informal security model and definition will be used instead in the following sections in describing three-party and two-party ID-SPK protocols.

5.1. The Three-Party ID-SPK Protocol

In [13], Gong further optimized the original SPK protocol in [9] by reducing the number of protocol messages to reduce the communication costs incurred by the protocol. We further modify Gong's optimized SPK protocol by building on the example given in Section C, as shown in Protocol 3.

(1). $A \rightarrow B: \; A, r_A, \; Enc_{PK_{A1}}(A, ST_A)$

(2). $B \rightarrow S: \; B, Enc_{PK_{B1}}(B, ST_B), A, r_A, Enc_{PK_{A1}}(A, ST_A)$

(3). $S \rightarrow B: \; Sig_{SK_{B2}}(K_{AB}), \; Sig_{SK_{A2}}(K_{AB})$

(4). $B \rightarrow A: \; Sig_{SK_{A2}}(K_{AB}), MAC_{K_{AB}}(r_A), \; r_B$

(5). $A \rightarrow B: \; MAC_{K_{AB}}(r_B)$

Figure 6. Protocol3: The Modified Gong SPK Protocol

In Protocol 3, users *A* and 5 select their respective random strings ST_A and ST_B and encrypt them with an ID-SPK. As before, $PK_{A1} = H_1(A||S||PW_A)$ and $PK_{B1} = H1(B||S||PW_B)$. The server recovers ST_A and ST_B, and computes private keys SK_A and SK_B matching the ID-SPKs constructed from the random strings, where $PK_{A2} = H_1(S||A||PW_A||ST_A)$ and $PK_{B2} = H_1(S||B||PW_B||ST_B)$. The private keys are then used to sign a session key. We assume that an IBS scheme with message recovery is used, so that the intended recipients are able to recover the session key these secret signatures also provide non-repudiation. Even though this is rarely a requirement in protocols for authentication and key establishment, it automatically provides the important data integrity and data origin authentication services [8]. Note that $MAC_{KAB}(r_A)$ and $MAC_{KAB}(r_B)$ in messages (4) and (5) are used by *A* and *B,* respectively, to prove to each other that they are indeed sharing the same session key. This provides key confirmation.

To improve the performance of Protocol 3, $Sig_{SK_{A2}}(K_{AB})$ and $Sig_{SK_{B2}}(K_{AB})$ in message (3) can be replaced with *{KAB} kA* and *{KAB} B* respectively, where *KA = F (STA)* and *KB = F (STB),* with *F* being a key derivation function. It is worth noting that *KA* and *KB* must not be derived from user passwords because this would allow the attacker to mount an off-line password guessing attack. The correctness of a candidate password could be verified by comparing $MAC_{K_{AB}}(r_A)$ from message (4) with $MAC_{K'_{AB}}(r_A)$, where *K'AB* is obtained using the guessed password.

Security Analysis. Protocol 3 shows that users *A* and *B* communicate with *S* using secret identifiers ID$_A$ = *A*//*S*//*PW*$_B$ and ID$_B$ = *B*//*S*//*PW*$_B$, respectively. These identifiers involve the users' passwords. Since *S* is the only party who has knowledge of *PW*$_A$ and *PW*$_B$ apart from *A* and *B,* the users should receive the same session key created by the server provided the correct private keys are used to transport the session key. If *A* and *B* can successfully recover *K*$_{AB}$ from their respective received secret signatures, they can be assured of the authenticity of the server.

It is clear that requirement 1 of Definition 1 can be satisfied if the session key is randomly generated by the server. Moreover, the session key cannot be computed directly by the adversary *E.*

By observing a protocol run, *E* can gather information such as $Enc_{PK_{A1}}(A, ST_A)$, $Enc_{PK_{B1}}(B, ST_B)$, $Sig_{SK_{A2}}(K_{AB})$ and $Sig_{SK_{B2}}(K_{AB})$ .However, since we assume that the ID-SPK encryption scheme used in this protocol is IND-ID-CCA secure, *E* cannot gain any useful information about *ST*$_A$ and *ST*$_B$ from the encrypted random strings $Enc_{PK_{A1}}(A, ST_A)$ and $Enc_{PK_{B1}}(B, ST_B)$ without knowledge of the master secret held by the server. As for the session key transportation in the form of secret signatures from the server to the users, *E* can choose his own verification keys in an attempt to recover the session key. However, there seems to be no efficient way for *E* to predict the correct ID-SPK if the ID-SPK signature scheme used in the protocol offers appropriate security. In particular, we assume that *E* cannot distinguish a secret signature from a randomly generated string if the identifier is constructed using sufficient randomness. We also assume that the adversary cannot forge valid secret signatures, impersonating the server to users. Apart from that, it is very unlikely that *E* can impersonate a legitimate user by guessing the user's password. This is so since the adversary's impersonation attack would be detected immediately by the server if the user's chosen random string cannot be recovered successfully from message (2). Note that the number of impersonation attempts can be kept acceptably small by using mechanisms that can log and control the number of failed authentication attempts. A brute force attack on message (3) or (4) to deduce the session key can be easily thwarted by using random strings *ST*$_A$ and *ST*$_B$ with entropy significantly larger than the password space of *D.* Also, so long as *ST*$_A$ and *ST*$_B$ are fresh and randomly generated for each protocol run, *E* would not be able to mount a replay attack. It is thus conjectured that requirement 2 is satisfied.

When E uses a password $PW'_A \in v$ to mount an off-line password guessing attack on a recorded $Enc_{PK_{A1}}(A, ST_A)$, there is no way for the adversary to verify the correctness of $PK'_{A1} = H1(A||S||PW'_A)$ if the ID-SPK encryption is randomized and IND-ID-CPA secure. If E selects $PW'_A \in D$ and ST'_A at random, computes $PK'_{A2}= H_1(S||A||PW'_A ||ST_A)$ and then attempts to verify $Sig_{SK_{A2}}(K_{AB})$, his check will almost certainly fail since the entropy of ST_A is much larger than the entropy of PW_A. Thus this form of off-line guessing attack will not succeed and therefore, Protocol 3 also satisfies requirement 3.
If E has a valid account with S, he may possibly mount an insider attack by impersonating A to S, pretending to be want to establish a session key K_{AE} with himself. In the attack, E initiates the protocol by computing $Enc_{PK_{A1}}(A, ST'_A)$ with a guessed password PW'_A, and hence $PK'_{A1} = H_1 (A||S||PW'_A)$. However, once this message has reached S, the server should get an error message when decrypting $Enc_{PK_{A1}}(A, ST'_A)$ using the decryption key matching PK'_{A1}. Therefore it is clear that the protocol can detect on-line guessing attacks and thus requirement 4 is satisfied.
On certain rare occasions, E may have access to *A's* or *B's* machine and thus the past session keys shared between them are exposed. However, since E has no knowledge of the master secret of S and the matching private component of PK_{A2}, E still cannot determine PW_A even though he can mount a brute-force attack on PK_{A2}. On the other hand, if for some reason, E has the correct password for A, he may attempt to find the value of ST_A given *A's* password and the ciphertext $Enc_{PK_{A1}}(A, ST_A)$. Since the encryption scheme is IND-ID-CCA secure, E only has a negligible success probability to discover the correct STA. Also, since the value of the verification key for $Sig_{SK_{A2}}(K_{AB})$ depends on the secret value STA, E can only recover the session key with negligible probability and forward secrecy of the protocol is preserved. Hence, requirement 5 is also satisfied and we conclude that Protocol 3 is a secure ID-SPK assuming that the ID-SPK encryption and signature schemes are appropriately secure.
Nevertheless, it is worth noting that if the server's master secret is compromised, the adversary can deduce the users' passwords without much difficulty. For instance, for each candidate password PW'A, E can extract the private key matching the identifier ID'A = A||S||PW'A and use it to attempt to decrypt $Enc_{PK_{A1}}(A, ST_A)$ from message (1), and check if the decryption unveils A's identity. Hence, it is of the utmost importance that the server's master secret is kept private, for example by using a strong protective mechanism such as storing it in a tamper-resistant device.

5.2. The Two-Party ID-SPK Protocol

We now present a Diffie-Hellman type two-party ID-SPK protocol. Our protocol is adapted from [9, 11] which make use of the encrypted Diffie-Hellman ephemeral key exchange technique between two parties. We apply the identity-based techniques that we introduced in Section IV to obtain Protocol 4, as shown below.

$$(1).\quad A \to B: \quad A,\ Enc_{PK_{A1}}(aP)$$
$$(2).\quad S \to A: \quad S,\ Sig_{SK_{A2}}(xP)$$

Figure 7. Protocol 4: The Diffie-Hellman ID-SPK Protocol

In Protocol 4 the user randomly selects $a \in Z_q^*$ and computes aP, where $P \in G_1$ is part of the system parameters. A then encrypts the Diffie-Hellman component with PKA1= Hi(A||S||PWA) and sends message (1) to S. The server extracts the matching private key SKA1 with its master secret to recover aP. Subsequently, S picks a random number $x \in Z_q^*$ and calculates xP. The server then extracts another private key which is associated with PKA2 = Hi(S||A||PWA||aP), produces $Sig_{SK_{A2}}(xP)$, and transmits it to A. After receiving message (2), the user retrieves xP with PKA2. Both the user and the server calculate a session key as KAS = F(A||S||PWA||aP||xP||axP), where F is a key derivation function. Note that key confirmation can be provided by adding a third message from A to S, in which A provides a MAC computed on all the protocol messages using the session key (derived using a different key derivation function to F).

Security Analysis. As with Protocol 3, user A uses an ID-SPK, but in this case to transport a Diffie-Hellman ephemeral key aP to the server. It is worth noting that message (1) can be replayed but this is not an issue because the purpose of the protocol is to authenticate the session key. If the adversary E has captured message (1) and replays it, he will not gain any information about the session key unless he has access to a and to xP in message (2). Also, we note that since only S other than A has access to PWA, S is authenticated to A when A successfully recovers xP (recall that an ID-SPK signature scheme provides a message integrity check) using PKA2 = H1(S||A||PWA||aP).

Clearly, requirement 1 of Definition 1 can be satisfied if the ephemeral Diffie-Hellman components from *A* and *S* are randomly generated and information used to compute the session key including *a, aP, x, xP,* and PW_A cannot be computed directly by *E.*

E has access to $Enc_{PK_{A1}}(aP)$ and $Sig_{SK_{A2}}(xP)$ through watching a protocol run between A and S. However, since we assume that the ID-SPK encryption scheme used in this protocol is IND-ID-CCA secure, E cannot obtain any useful information about aP from $Enc_{PK_{A1}}(aP)$ without knowledge of the master secret held by the server. Also, we assume that the ID-SPK signature scheme used in the protocol produces secret signatures $Sig_{SK_{A2}}(xP)$ that are indistinguishable from random strings. Hence it is hard for *E* to deduce any information about the Diffie-Hellman component chosen by the server. Apart from that, as we have discussed when analyzing Protocol 3, it appears unlikely that *E* will successfully impersonate *A* in *n* attempts with probability significantly higher than *n/|D|* or mount a replay attack, provided *aP* and *xP* are fresh and their entropy is significantly higher than the entropy of *D.* Also, the use of an incorrect password in generating PK_{A1} can be easily detected by the server when the server uses the wrong matching private key to recover *aP.* It is thus conjectured that requirements 2, 3 and 4 are satisfied.

It is possible that E may have access to A's machine and recover the past session keys used by A. In that case, despite the fact that E knows K'AS, he must be able to break the

key derivation function F in order to deduce A's password. On the other hand, if for some reason A's password is revealed to E, E may attempt to find the value of *aP* given *A's* password and the ciphertext $Enc_{PK_{A1}}(aP)$. Since the encryption scheme is IND-ID-CCA secure, *E* only has a negligible success probability to find the correct *aP.* Also, since the value of the verification key for $Sig_{SK_{A2}}(xP)$ depends on the secret value *aP, E* can only recover the session key with negligible probability. This is related to the forward secrecy of protocols discussed in [8, 11]. Therefore, requirement 5 is also satisfied. We note that in addition to having met this requirement, even if *E* knows *aP* and *xP*, he has to solve the intractable CDH problem in order to calculate *axP* and hence the session key. We conclude that Protocol 4 is a secure ID-SPK protocol.
As with the security of Protocol 3, it is essential to have the server's master secret adequately protected to ensure that the aforementioned security conditions hold.

6. Integrating ID-SPKs with TLS

Steiner *et al* [7], first proposed the integration of password-based Diffie-Hellman encrypted key exchange with the TLS handshake protocol (DH-EKE/TLS). Here we propose a server-authenticated, TLS-compatible ID-SPK protocol (ID-SPK/TLS) by building on the ideas developed in the previous sections.

$$\begin{array}{lll}
(1). & A \rightarrow S: & ClientHello = n_A,\ session\_id,\ cipher\_suite \\
(2). & S \rightarrow A: & ServerHello = n_S,\ session\_id\ \ cipher\_suite \\
 & & ServerKeyExchange = \{STA_A\}\ PW_A \\
 & & ServerHelloDone \\
(3). & A \rightarrow S: & ClientKeyEchange\ \ Enc_{PK_A}(pre\_master\_\mathrm{sec}ret) \\
 & & ClientFinished \\
(4). & S \rightarrow A: & ServerFinished
\end{array}$$

Figure 8. Protocol 5: The ID-SPK/TLS Protocol

The description of Protocol 5 and comparison with the standard TLS handshake protocol are as follows.
(1) As with the current TLS specification [11], Protocol 5 begins with A sending S a *ClientHello* message. The message contains a fresh nonce, a session identifier and a cipher specification. In the standard TLS protocol, *session_id* contains a value which identifies a previously established session between A and S some or all of whose security parameters wishes to re-use. Otherwise it is an empty field. In Protocol 5, we envisage that if there is no session identifier to be re-used or resumed, then *session_id* would carry the identity of A. Meanwhile, *cipher_suite* contains a cipher specification extended from TLS version 1.0 to handle the ID-SPK encryption and signature schemes. For example, TLS_ID_SPK_WITH_DES_CBC_SHA would define an ID-SPK-supported cipher specification that uses DES-CBC and SHA as the symmetric encryption algorithm and hash function, respectively.
(2) *S* responds with a *ServerHello* message which contains an independent nonce and a session identifier whose value depends on the client's input. For instance, if *session_id* = *A, S* will create a new session identifier. Otherwise, *S* will search its local cache to check if there is a session identifier which matches the value submitted by *A* and decide

whether or not to resume the session. *S* then generates a random string STA that will be used by *A* to communicate a pre master secret. String STA is encrypted using DES-CBC with A's password. As in Section IV-B, care must be taken to avoid introduction of redundancy that would enable off-line guessing attacks. The encrypted random string is sent to *A* in *ServerKeyExchange*. The *ServerHelloDone* message is then transmitted to indicate the end of step (2).

It is worth noting that if we replace {ST$_A$} PW$_B$ in the *ServerKeyExchange* message with $Sig_{SK_{A2}}(ST_A)$, the protocol will be insecure. For let the correct verification key for $Sig_{SK_{A2}}(ST_A)$ be PK$_A$ = H$_1$ (A||S||PW$_A$). At first glance, it may seem that the *ServerKeyExchange* message would achieve its intended purpose in Protocol 5, i.e. securely transporting a random string to *A*. However, the signature integrity check in an IBS scheme with message recovery can be used to mount a simple off-line password guessing attack, as described in Section IV-C.

(3) *A* uses her password to recover ST$_A$. *A* then selects a random pre-master secret and encrypts it using an ID-SPK encryption scheme with PK$_A$ =H$_1$ (A||S||PW$_A$||ST$_A$). It appears that the attacker could not learn any in formation about PW$_A$ by mounting an off-line password guessing attack on a recorded Enc_{PK_A} (*pre_master_secret*), since the ID-SPK encryption is assumed to be IND-ID-CPA secure (as discussed in the security analysis of Protocol 3 in Section V-A). For the same reason, the attacker only has a negligible success probability to discover the correct pre-master secret chosen by A even if PW$_A$. is revealed.

A transmits the encrypted pre-master key to S using the *ClientKeyExchange* message. Note that *A* still cannot authenticate S (and its system parameters) at this stage because A has only decrypted {ST$_A$.} PW$_A$. And used the recovered ST$_A$. to perform the ID-SPK encryption. However, *S* can authenticate *A* when *A* completes step (3) as the *ClientFinished* message contains a verification value:

PRF(master_secret,"clientfinished",h_messages_1,h_messages_2), where PRF is a pseudo-random function, and master_secret = PRF(pre_master_secret ,"master secret", nA,nS).

Here, h_messages_1 and h_messages_2 represent hash values of all handshake messages up to but not including this message, using different hash functions. If A has used the correct password in deriving the key PK$_A$. used to encrypt the pre-master secret and S has retrieved this value, then the verification value computed by S matches the value sent by A in *ClientFinished*, and thus A is authenticated.

(4) *S* calculates a verification value as above with "*server finished*" replacing "client finished" and transmits it to *A* in *ServerFinished*. Checks the verification value received from *S*. If it was correctly calculated, only now has *S* been authenticated by A, since only S could have derived the appropriate decryption key using the shared password PW$_A$. and its master secret to retrieve the correct pre-master secret. Subsequently, *master_secret* will be used to derive further keys for protecting application data between *A* and *S*.

As with Protocols 3 and 4, Protocol 5 also achieves forward secrecy. The exposure of the user's long-term credential, i.e. password, does not compromise the user's past session keys since the encryption scheme used is IND-ID-CPA secure. The master secret of the server must not be compromised in order for this condition to hold. Discussion. In [7], the authors claimed that five protocol flows are the best possible design to prevent dictionary attacks in the DH-EKE/TLS protocol. Also, swapping the order of *ClientFinished* and *ServerFinished* was necessary. Our Protocol 5 shows that no structural changes are required but only minimal alterations to the contents of the standard TLS protocol messages: (i) inclusion of the client's identity in *session_id*; and (ii) replacement of a signed temporary RSA or a Diffie-Hellman ephemeral key with an encrypted random string in *ServerKeyExchange*. These modifications only require small adjustments to data fields of the current TLS protocol specification. Hence, the advantages that changes (i) and (ii) could bring seem to outweigh any implementation issues that they may cause.

One limitation of our proposed protocol is the need for pre-distribution of the server's system parameters. Otherwise, a man-in-the-middle attack similar to Patel's attack on Protocol 2 is possible. In this attack, the attacker can impersonate *S* to A by inserting his own set of system parameters and substituting {ST_A.} PW_A. with a random string RA. A would then use her password to recover a value ST'A from RA. When A replies with his chosen pre-master secret encrypted under the identifier ID'_A= A||S||PW_A.|| ST'_A., for each candidate password PW'_A., the attacker now decrypts RA using PW'_A to obtain a value ST'_A., and then uses ST'A to derive a private key corresponding to the identifier I SD'_A.= A||S||PW'_A.||ST'_A. Subsequently, the server recovers a pre-master secret and computes a *ClientFinished* value. The guessed password can then be verified by comparing the *ClientFinished* value that *S* computed with the *ClientFinished* value that *A* sent to *S*. If they match, then the guessed password is a correct one.

7. Coclution

We studied the history of secret public key protocols and discussed some known problems with these protocols. We explored some interesting properties of identity-based cryptography which form the basis of our proposed identity-based secret public key protocols. These properties also allow us to convert a conventional identity-based encryption scheme and a standard identity-based signature scheme (with message recovery) into their secret public key equivalents.

We presented three-party and two-party identity-based secret public key protocols for key exchange. Our heuristic security analyses show that the protocols appear to be secure against off-line password guessing attacks and undetectable on-line password guessing attacks, and provide forward secrecy. Then we combined the new properties from identity-based secret public keys and the techniques used in constructing the identity-based secret public key protocols, and showed that secret public keys can support the use of passwords in the TLS handshake protocol in a very natural way.

References

[1] I. Foster and C. Kesselman. The grid in a nutshell. In J.Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski, editors, *Chapter 1 of Grid Resource Management: State of the Art and Future Trends*, pages 3-13, Boston, 2003. Kluwer Aca- demic.

[2] 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 Com- munications*, 11(5):648–656, 1993

[3] J.LUO, S.SHIEH AND J.SHEN, "Secure Authentication Protocols Resistant to Guessing Attacks", JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 22, 1125-1143 (2006)

[4] Y. Ding and P. Horster. Undetectable on-line password guessing attacks. ACM Operating Systems Review, 29(4):77{86, 1995.

[5] S. Patel. Number theoretic attacks on secure password schemes. In Proceedings of the 1997 IEEE Symposium on Security and Privacy, pages 236{247. IEEE Computer Society Press, 1997.

[6] M. Bellare and P. Rogaway. Optimal asymmetric encryption { how to encrypt with RSA. In A.D. Santis, editor, Advances in Cryptology - Proceedings of EUROCRYPT '94, pages 92{111. Springer-Verlag LNCS 950, 1995.

[7] M. Steiner, P. Buhler, T. Eirich, and M. Waidner. Secure password-based cipher suite for TLS. *ACM Transactions on Information and System Security*, 4(2):134{157, May 2001.

[8] M. Abdalla, O. Chevassut, and D. Pointcheval. One-time verifier-based en- crypted key exchange. In S. Vaudenay, editor, *Proceedings of the 8th Interna tional Workshop on Theory and Practice in Public Key Cryptography – PKC 2005*, pages 47{64. Springer-Verlag LNCS 3386, 2005.

[9] M. Abdalla, P. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In S. Vaudenay, editor, *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryp- tography - PKC 2005*, pages 65{84. Springer-Verlag LNCS 3386, 2005.

[10] M. Abdalla and D. Pointcheval. Simple password-based encrypted key ex- change protocols. In A. Menezes, editor, *Proceedings of the RSA Conference:Topics in Cryptology - the Cryptographers' Track (CT-RSA 2005)*, pages 191{208. Springer-Verlag LNCS 3376, 2005.

[11] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In C.S. Laih, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2003*, pages 452{473. Springer-Verlag LNCS 2894, 2003.

[12] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public- key encryption. In C. Boyd, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2001*, pages 566{582. Springer-Verlag LNCS 2248, 2001.

[13] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2000*, pages 139{155. Springer-Verlag LNCS 1807, 2000.

[14] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology - Proceedings of CRYPTO 2001*, pages 213{229. Springer-Verlag LNCS 2139, 2001.

[15] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establish- ment*. Springer-Verlag, Berlin, 2003.

[16] K. Brincat. On the use of RSA as a secret key cryptosystem. *Designs, Codes, and Cryptography*, 22(3):317{329, 2001.

[17] D. Chaum, E.v. Heijst, and B. Pfitzmann. Cryptographically strong unde- niable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Advances in Cryptology - Proceedings of CRYPTO'91*, pages 470{484. Springer-Verlag LNCS 576, 1992.

[18] M.E. Hellman and S.C. Pohlig. *Exponentiation Cryptographic Apparatus and Method*. U.S. Patent #4,424,414, 3 January 1984 (expired).

[19] B. Libert and J-J. Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In F. Bao, R.H. Deng, and J. Zhou, editors, *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2004*, pages 187{200. Springer-Verlag LNCS 2947,2004.

[20] F. Zhang, W. Susilo, and Y. Mu. Identity-based partial message recovery signatures (or how to shorten ID-based signatures). In A.S. Patrick and M. Yung, editors, *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, pages 45{56. Springer-Verlag LNCS 3570, 2005.

[21] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) ¿ cost(signature) + cost(encryption). In B.S. Kaliski Jr., editor, *Advances in Cryptology - Proceedings of CRYPTO'97*, pages 165{179. Springer-Verlag LNCS 1294, 1997.

[22] T. Dierks and C. Allen. The TLS protocol version 1.0. The Internet Engi- neering Task Force (IETF), RFC 2246, January 1999.