Authenticated Diffie–Hellman key agreement protocol using a single cryptographic assumption

L. Harn, W.-J. Hsin and M. Mehta

Abstract: In modern communication systems, a popular way of providing authentication in an authenticated Diffie–Hellman key agreement protocol is to sign the result of a one-way hash function (such as MD5) of a Diffie–Hellman public key. The security of such a protocol is based on the weakest of all the cryptographic assumptions of the algorithms involved: Diffie–Hellman key distribution, digital signature and a one-way hash function. If a protocol can be constructed using one cryptographic assumption, it would be at least as secure as that with multiple assumptions. The authors propose three authenticated Diffie–Hellman key-agreement protocols, each of which is based on one cryptographic assumption. In particular, the first protocol is based on a discrete logarithm, the second on an elliptic curve and the third on RSA factoring. The main objective of the paper is to show that the security of a protocol should be assessed at the protocol level as a whole, rather than at the level of individual algorithms that are used to build the protocol.

1 Introduction

The security of a communication protocol is usually based on one or more assumptions. For example, a key agreement protocol is built on one or more cryptographic assumptions. A protocol with multiple independent assumptions with a logic OR relationship (OR-related) is like a house with multiple outside doors with different security mechanisms. The more doors a house has, the more ways a thief can break into the house, and the weakest security mechanism of all is the easiest to overcome. Similarly, the more independent OR-related assumptions a protocol has, the more ways an attacker can try to attack the protocol and the weakest of all is the easiest to try. In other words, when multiple independent OR-related assumptions are involved, the security of a protocol is typically reduced to that of the weakest one. Therefore, for two protocols A and B, where A is based on multiple independent OR-related cryptographic assumptions and B is based on one of the assumptions in A, B is at least as secure as A. In this paper, we limit our scope to the area of authenticated key agreement protocols and propose three authenticated Diffie–Hellman key agreement protocols, each based on exactly one cryptographic assumption. Our main objective is to show that when a protocol is built upon several algorithms, the security of the protocol should be assessed in its entirety, rather than at the level of individual algorithms.

© IEE, 2005

IEE Proceedings online no. 20041041

doi:10.1049/ip-com:20041041

Paper first received 10th March and in revised form 23rd July 2004

L. Harn and M. Mehta are with the School of Computing and Engineering, University of Missouri - Kansas City, 5100 Rockhill Road, Kansas City, MO 64110, USA

W.-J. Hsin is with the Information and Computer Science, Park University, Parkville, MO 64152, USA

E-mail: harnl@umkc.edu

Authenticated key agreement [Note 1] is a process of verifying the legitimacy of communicating parties and establishing common secrets among the communicating parties for subsequent use (such as data confidentiality and integrity). Authenticated key agreement is very important for virtually all secure communication systems such as e-commerce, wireless, wireline and Internet applications. An authenticated key agreement protocol in general is constructed using multiple cryptographic algorithms which are based on various cryptographic assumptions.

The most well known assumptions of public-key cryptographic algorithms are the computational problems of a discrete logarithm (DL) with complexity $O(e^{((\ln p)^{(1/3)}(\ln(\ln p))^{(2/3)})})$ [1], an elliptic curve (EC) with complexity $O(e^{(1.098+o(1))n^{1/3}(\ln n)^{2/3}})$, in $GF(2^n)$ finite fields [2], and factoring (RSA) with the same complexity as a DL [3]. On the other hand, the security of most well known conventional cryptographic algorithms, such as the one-way hash functions and block ciphers, is based on the complexity of analysing a simple iterated function of multiple rounds. These two types of cryptographic assumptions are completely different and most of them are even incompatible.

For example, consider the most commonly used authenticated Diffie–Hellman key-agreement protocols. In particular, the popular SSL and IPSec standards provide such an option. Since the Diffie–Hellman public-key distribution algorithm itself does not provide authentication, the authentication in an authenticated Diffie–Hellman key agreement protocol is usually provided by signing a Diffie–Hellman public key [4, 5]. Thus, it involves at least three cryptographic algorithms in building this protocol: the

Note 1: We use the term 'authenticated key agreement' to describe the general idea of key agreement with authentication. The term 'authentication and key agreement' (also known as AKA) has been used in 3GPP wireless networks to provide network access security. Our authenticated key agreement can be applied to secure two-party communication networks. Additionally, AKA in the 3GPP wireless domain is symmetric-key based whereas ours is public-key based.

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.

Diffie–Hellman key-distribution algorithm, a digital signature and a one-way hash function. This protocol involves at least two independent cryptographic assumptions. For example, if the digital signature scheme, such as the digital signal algorithm (DSA), is DL-based, the above protocol (like the ones in [6, 7]) will have two cryptographic assumptions: one is the DL and the other is the hash function; if the digital signature scheme is RSA-based, the above protocol will have three cryptographic assumptions, namely, the DL, the RSA and the hash function. The security of the protocol would depend on the weakest cryptographic assumption involved. To further elaborate on this example, suppose that in this protocol, Diffie–Hellman key distribution is based on a 512-bit DL and the digital signature is based on a 1024-bit RSA. According to [3], RSA factoring has same complexity as a DL. Consequently, the 512-bit DL can be considered easier to break as compared to the 1024-bit RSA. An attacker can simply derive the session key by solving the DL problem from the exchanged key information and then use it to decipher the subsequent exchanged messages between the communicating parties without having to forge the signatures. Thus, in general, given M independent OR-related cryptographic assumptions, where M is a positive integer, a protocol with only one of the M assumptions is at least as secure as that with all M assumptions.

In the existing literature, to our knowledge, there is no single-assumption authenticated Diffie–Hellman key-agreement protocol [8]. The IEEE has standardised the authenticated key-distribution protocols in the P1363 standards [9]. Here, following the classification by the IEEE P1363 standards, we propose three authenticated Diffie–Hellman key-agreement protocols, each based on one cryptographic assumption.

In modern communication protocols, there are two types of authentication, namely, user authentication and shared-key authentication. User authentication is to authenticate a communicating user in real time. Shared-key authentication ensures that a shared key is known only to the legitimate users. A key-agreement protocol without either user authentication or shared-key authentication is not secure, leading to many kinds of attacks. Therefore we need both user authentication and shared-key authentication. In particular, we need to efficiently and securely integrate both user authentication and shared-key authentication into authenticated key-agreement protocols. We will discuss these two types of authentication and associated attacks in detail in Section 2.

In the literature, the term 'authenticated key-agreement' protocol can be somewhat misleading in terms of what type of authentication a protocol really provides. Many protocols provide shared-key authentication but not user authentication. For example, Harn and Lin [10, 11], Shim [12], Yen and Joye [13], and Wu *et al.* [14] provide shared-key authentication, but not user authentication. The 2-pass MQV protocol [15] provides shared-key authentication only, but the 3-pass MQV provides user authentication as well as shared-key authentication. In this paper, all our three proposed protocols provide both user authentication and shared-key authentication.

Specifically, the first of our one-assumption authenticated Diffie–Hellman key agreement protocols is based on a DL, the second on an EC, and the third on an RSA. Each of these three protocols can be described, based on a general framework, as a three-pass message transmission between two communicating parties. In addition to the above mentioned properties (i.e. user authentication, shared-key authentication, one cryptographic assumption), each of our proposed protocols can also prevent the attacks often discussed in literature.

The work in the literature closest to what we are proposing here is by Harn [16], and Harn and Lin [10]. Harn [16] proposed various digital signature schemes without using one-way functions to sign Diffie–Hellman public keys. Harn and Lin [10] proposed a protocol that utilises the digital signature schemes in [16], but the proposed protocol provides only shared-key authentication and is restricted to a DL assumption. In this paper, we base our work on [16] and propose three one-assumption protocols that can achieve both user authentication and shared-key authentication.

2 User authentication versus shared key authentication

User authentication determines the legitimacy of the intended parties in real time. For example, in a client-server application, a service provider needs to ensure the legitimacy of a user before providing services to the user. Similarly, a user needs to make sure that the service provider is genuine so that the user is willing to send its sensitive information (such as a credit card number) to the service provider.

Since communicating parties need a common key to encrypt and decrypt data, shared-key authentication makes sure that the shared common key is known only to the intended parties.

In a key-agreement protocol without user authentication, an attacker can misrepresent the identity of an innocent party, leading to attacks such as replay, resource exhaustion and unknown key-share. In the following, we discuss each of these attacks.

First, for the replay attack, consider the current Internet environment, where authentication is based on a trusted third party certificate authority (CA). As a certificate is public information, an attacker can get hold of an innocent user's public certificate, and along with previously recorded exchanged information between innocent parties, the attacker can impersonate the innocent user, resulting in a replay attack. Thus, in a protocol without user authentication, even with the use of a certificate, impersonation is possible.

Secondly, for the resource-exhaustion attack, consider client–server applications. Without user authentication, an illegitimate client can pretend to be an innocent user and replay key establishment requests repeatedly to exhaust the server's computing resource, leading to a denial-of-service attack. With user authentication, the server can reject the attacker early during establishment, reducing misuse of computing resources.

Thirdly, Fig. 1, demonstrates an unknown key-share attack [17]. User A sends message m_1 to user B, however, attacker E intercepts the message and replaces it with message m_2. B believes that it is communicating with E as message m_2 contains the certificate of E. User B then sends message m_3 to E who then forwards the unmodified message m_3 to A. In this scenario, shared-key authentication is achieved between users A and B as only A and B know the shared secret. But, user authentication is not achieved because B thinks that it is communicating with E, not A. Thus, a lack of user authentication can lead to many kinds of attacks, even with the use of a CA signed by a trusted third party.

On the other hand, a communication protocol with only user authentication is not enough. A secure communication channel needs a shared key between communicating parties

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.

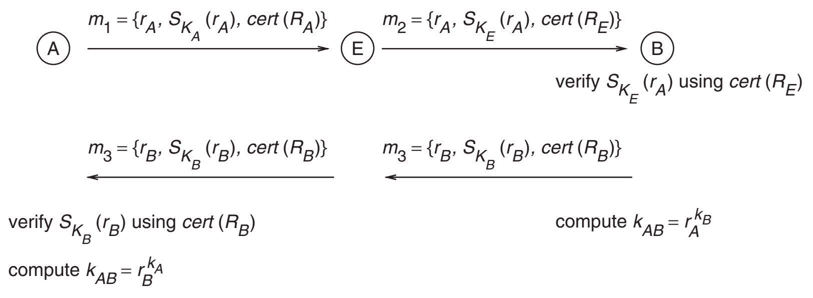
$m_1 = \{r_A, S_{K_A}(r_A), cert(R_A)\}$
A
E
$m_2 = \{r_A, S_{K_E}(r_A), cert(R_E)\}$
B
verify $S_{K_E}(r_A)$ using $cert(R_E)$
$m_3 = \{r_B, S_{K_B}(r_B), cert(R_B)\}$
$m_3 = \{r_B, S_{K_B}(r_B), cert(R_B)\}$
verify $S_{K_B}(r_B)$ using $cert(R_B)$
compute $k_{AB} = r_A^{k_B}$
compute $k_{AB} = r_B^{k_A}$

Fig. 1 *Messages transferred between users A and B in an unknown key-share attack where E is an attacker*
Notations:
k_i = a short-term private key of user i
k_{ij} = a common shared secret key between users i and j
r_i = a short-term public key of user i. In particular, $r_i = \alpha^{k_i}$
K_i = a long-term private key of user i
R_i = a long-term public key of user i
$cert(R_i)$ = a public key certificate of key R_i
$S_{K_i}(z)$ = a singature on key z signed using key K_i
m_n = message identifier

for purposes such as data confidentiality and integrity. Shared-key authentication ensures that the shared key is known only to the intended parties. Without shared-key authentication, after mutual user authentication, an attacker can hijack the communication channels.

Thus, a secure communication protocol needs both user authentication and shared-key authentication. In the next Section, we show how to efficiently and securely integrate both user authentication and shared-key authentication into our authenticated Diffie–Hellman key agreement protocols.

3 Authenticated key agreement protocols with one cryptographic assumption

In this Section, we provide three authenticated key agreement protocols, each integrating both user authentication and shared-key authentication into the Diffie–Hellman key-distribution algorithm. Each of the three protocols is based on one cryptographic assumption, i.e. a DL, an EC or an RSA.

3.1 General framework

The general framework of our three one-assumption protocols is a three-run message calculation passing between two communicating parties, as depicted in Fig. 2, where C_i^n means the computation performed by user i in the nth run, and T_{ij}^n means the set of information passed from user i to user j in the nth run.

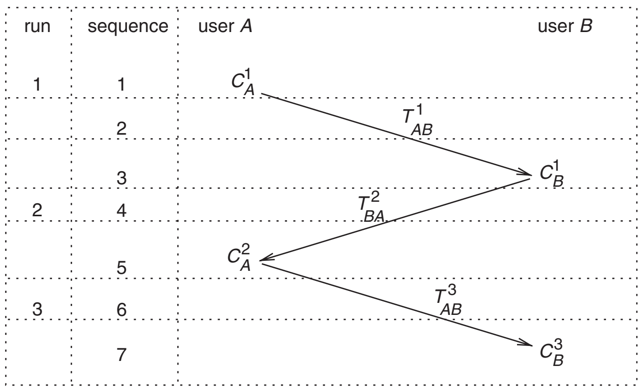

Fig. 2 *General framework: 3-pass message transmission*

Notations:

k	short-term private key
r	short-term public key
K	long-term private key
R	long-term public key

A key for a particular user is denoted with a single subscript, for example, k_A means user A's short-term private key.

A shared key selected by user i and sent to user j is denoted with two subscripts i and j, for example, k_{AB} is a short-term secret key that is selected by A and sent to B, and shared only between A and B.

For an elliptic curve, if a parameter is in bold letters, it means that it is not scalar but a vector with x and y coordinates.

3.2 Discrete logarithm assumption

Based on the general framework depicted in Fig. 2, Section 3.2.1 shows how an authenticated Diffie–Hellman key agreement based on a DL is achieved, and Section 3.2.2 provides security analysis of this scheme.

3.2.1 Algorithm: Table 1 lists the notations used by the algorithm, where an item ticked in the last column means that it is assumed to be available to the communicating parties before starting the key agreement process. Table 2 shows the computations performed by users A and B, and Table 3 shows the messages that are transferred between users A and B.

3.2.2 Security analysis: Our proposed protocol is one assumption and provides user authentication as well as shared-key authentication. In addition, it protects against replay attack, known key attack and unknown key-share attack. In this Section, we analyse the proposed protocol from all these aspects.

One-assumption property: Our DL-based authenticated Diffie–Hellman key-agreement protocol involves two algorithms: Diffie–Hellman key distribution and the modified Harn's digital signature scheme in [16]. Specifically, we add both shared secrets k_{AB} and k_{BA} into Harn's digital

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.

Table 1: DL notations

Property	Notation (i for user identity such as A or B)	Published information
Prime number	p	✓
Factor of $p-1$	q	✓
Generator with order q	α	✓
Short-term private key	k_i	
Short-term public key	$r_i = \alpha^{k_i} \bmod p$	
Long-term private key	K_i	
Long-term public key	$R_i = \alpha^{K_i} \bmod p$	✓
Public-key certificate	$cert(R_i)$	✓
Signature	s_i	✓

Table 2: DL computations

Run	Step	Computation	
1	1	C_A^1	A selects k_A
	2		A computes r_A
1	3	C_B^1	B selects k_B
	4		B computes r_B
	5		B computes $k_{AB} = (r_A)^{K_B} \bmod p$
	6		B computes $s_B = k_{AB}^{-1}(K_B - r_B k_B) \bmod q$
2	7	C_A^2	A verifies R_B by checking $cert(R_B)$
	8		A computes $k'_{AB} = (R_B)^{k_A} \bmod p$
	9		A verifies s_B and k'_{AB} by checking $R_B \overset{?}{=} (r_B)^{r_B}(\alpha)^{(s_B k'_{AB})} \bmod p$
	10		A computes $k_{BA} = (r_B)^{K_A} \bmod p$
	11		A computes $s_A = k_{BA}^{-1}(K_A - r_A k_A) \bmod q$
3	12	C_B^3	B verifies R_A by checking $cert(R_A)$
	13		B computes $k'_{BA} = (R_A)^{k_B} \bmod p$
	14		B verifies s_A and k'_{BA} by checking $R_A \overset{?}{=} (r_A)^{r_A}(\alpha)^{(s_A k'_{BA})} \bmod p$

Table 3: DL message passing

	Message
T_{AB}^1	$\{r_A\}$
T_{BA}^2	$\{r_B, s_B\}$
T_{AB}^3	$\{s_A\}$

signature equations in [16]. Harn's digital signature scheme is a modified ElGamal DL-based digital signature scheme [18]. Thus, its cryptographic assumption is based on a DL. In particular, Harn's digital signature scheme does not use a one-way hash function to sign the Diffie–Hellman public key. Since both Diffie–Hellman key distribution and modified Harn's digital signature algorithms are based on a DL, the security of our proposed protocol is entirely based on a DL assumption.

Shared-key authentication: Shared-key authentication ensures that the shared key is known only to the intended parties. Our protocol achieves shared-key authentication by using the Diffie–Hellman function. Specifically, after B receives r_A, B calculates $k_{AB} = (r_A)^{K_B} \bmod p$ in step 5 of Table 2. Since K_B is private to B, B can obtain k_{AB}. Similarly, after A receives R_B from B, A calculates $k_{AB} = (R_B)^{k_A} \bmod p$ in step 8 of Table 2. Since k_A is private to A, A can obtain k_{AB}. Therefore, k_{AB} is known only to users A and B. A similar analysis can be applied to k_{BA}. Thus our protocol provides shared-key authentication.

User authentication: The foundation of our DL-based authenticated key agreement protocol is based on the Diffie–Hellman function for the key agreement and Harn's scheme [16] for the digital signature. However, Harn did not provide user authentication. In our protocol, we provide user authentication. Specifically, this is achieved by including the short-term shared secrets k_{AB} and k_{BA} in the signature-signing equations (see steps 6 and 11 in Table 2, respectively). For a detailed analysis of the provision of user authentication, we provide a security analysis from user A's point of view. A similar analysis can be applied to user B.

k_{AB} is a shared secret key selected by A and used by B to decrypt the messages from A. Similarly, k_{BA} is a shared secret key selected by B and used by A to decrypt the messages from B. For user authentication, from A's point of view, it is important to convince A that:

(i) the signature s_B is really generated by B, thus k_{BA} is selected by B;

(ii) k_{AB} has been received by B (i.e. the so-called 'key confirmation'); and

(iii) k_{BA} is fresh and not a replayed key

In the following, we explain how A is convinced by each of the three items.

Item (*i*): Where A needs to be convinced of s_B signed by B, in the second round of the protocol, A verifies s_B by checking the equation in step 9. After successful verification, A is convinced that s_B is signed by B. Convinced of the signature signer B, A can therefore be convinced that r_B is selected by B, as s_B is a signature of r_B in the signature equation in step 6. Furthermore, since A calculates k_{BA} based on r_B (see step 10), A is convinced that k_{BA} is selected by B.

Item (*ii*): A needs to be convinced that k_{AB} has been received by B. Since k_{AB} is included in the signature equation calculation in step 6, after successful verification in step 9, A is convinced that s_B is signed by B, therefore A is confirmed that k_{AB} has been received by B. This is also known as key confirmation.

Item (*iii*): A needs to be convinced that the k_{BA} is not a replayed key. In the original ElGamal signature scheme [18], a signature by B is represented by a pair of parameters (r_B, s_B) which are sent along with the original message to A for message verification. In our scheme, we treat r_B as the message itself and s_B serves as the single-parameter signature. In the first round, r_A, which is a computed number based on a random number and is sent by A to B, serves as a nonce. k_{AB} in step 5 is calculated based on the nonce r_A. In step 6, the signature s_B is calculated for message r_B by taking k_{AB}, therefore the nonce, into account. Thus, in the second round, if A successfully verifies the signature of the equation in step 6, A can be sure that r_B is

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.

not a replayed message, and hence k_{BA} (the equation in step 10) is not a replayed key.

A similar analysis of user authentication can be applied from user B's point of view. Thus, our protocol provides mutual user authentication between two communicating parties via the inclusion of k_{AB} and k_{BA} in the signature-signing equations.

As mentioned in Section 1, user authentication can prevent impersonation attacks such as replay attacks, unknown key-share and resource-exhaustion attacks. The preceding paragraph explaining how item 3 is achieved shows the prevention of replay attacks. For the unknown key-share attack, Kaliski [17] has shown that key confirmation can prevent this kind of attack. Since our protocol achieves item 2 which provides key confirmation, it prevents the unknown key-share attack. For the resource exhaustion attacks, consider a client-server model where user A is an illegitimate client and B is a server. With user authentication built into our protocol, server B can reject A's connection request after unsuccessful verification in the 3rd run of our protocol, thereby reducing misuse of B's computing resources.

Mutually independent shared keys: As with most standard protocols, such as SSL and IPSec, where different keys are used for different directions, our protocol has two short-term shared secrets k_{AB} and k_{BA}, one for each direction. Our proposed protocol is based on the signature equation scheme introduced by Harn [16]. However, Harn's scheme leads to a known key attack [19] in which an attacker can derive a key from an another known key between the communicating parties. Specifically, in [16], the signature equations for Diffie–Hellman key agreement without using a one-way hash function are

$$K_A = r_A k_A + s_A \bmod q \tag{1}$$

$$K_B = r_B k_B + s_B \bmod q \tag{2}$$

The security of this key agreement scheme can be referred to in [16]. In our current protocol, because of the known key attack, we cannot use these signature equations directly for key agreement. This is because using (1) and (2) will lead to mutual dependence of the shared keys. Specifically, by cross multiplying (1) and (2), we obtain

$$K_A r_B k_B + K_A s_B = K_B r_A k_A + K_B s_A \bmod q \tag{3}$$

Raising both sides to the power of α, one obtains

$$(k_{BA})^{r_B}(R_A)^{s_B} = (k_{AB})^{r_A}(R_B)^{s_A} \bmod p \tag{4}$$

Equation (4) shows that the two shared secret keys, k_{AB} and k_{BA}, are not independent of each other. Knowing one of these two keys will reveal the other, i.e. the so-called 'known key attack'.

In this paper, we propose to change (1) and (2) to the equations in steps 11 and 6, respectively. The equations in steps 11 and 6 are equivalent to

$$K_A = r_A k_A + s_A k_{BA} \bmod q \tag{5}$$

$$K_B = r_B k_B + s_B k_{AB} \bmod q \tag{6}$$

Similar to the above analysis, after cross multiplying of (5) and (6), we obtain

$$K_A r_B k_B + K_A s_B k_{AB} = K_B r_A k_A + K_B s_A k_{BA} \bmod q \tag{7}$$

Raising both sides to the power of α, one obtains

$$(k_{BA})^{r_B} R_A^{(s_B k_{AB})} = (k_{AB})^{r_A} R_B^{(s_A k_{BA})} \bmod p \tag{8}$$

In (8), notice that the shared secrets k_{AB} and k_{BA} are the exponents on both sides of the equation. According to Agnew *et al.* [20], given one of the shared secrets e.g. k_{AB}, finding the other is no easier than a discrete logarithm problem. In other words, knowing one key cannot lead to the other. Thus, k_{AB} and k_{BA} are independent of each other.

In summary, this Section shows that our proposed protocol is based on one cryptographic assumption. Furthermore, it demonstrates that our protocol provides both user authentication and shared-key authentication, and prevents known key attacks.

3.3 Elliptic-curve assumption

This protocol is a direct extension of the protocol based on the DL assumption in Section 3.2. Based on the general framework depicted in Fig. 2, Section 3.3.1 shows how an authenticated key agreement based on an elliptic curve is achieved, and Section 3.3.2 provides security analysis of this scheme.

3.3.1 Algorithm:

Table 4 lists the notations used by the algorithm, Table 5 shows the computations performed by users A and B, and Table 6 shows the messages that are transferred between users A and B.

Table 4: EC notations

Property	Notation (i for user identity such as A or B)	Published information
Elliptic curve	E	✓
Prime number	p	✓
Prime divisor of the number of points in E	q	✓
Curve point generating the subgroup of order q	$\boldsymbol{\alpha}$	✓
Short-term private key	k_i	
Short-term public key	$\boldsymbol{r}_i = k_i\boldsymbol{\alpha} = (x_{\boldsymbol{r}_i}, y_{\boldsymbol{r}_i})$	
Long-term private key	K_i	
Long-term public key	$\boldsymbol{R}_i = K_i\boldsymbol{\alpha} = (x_{\boldsymbol{R}_i}, y_{\boldsymbol{R}_i})$	✓
Public-key certificate	$cert(\boldsymbol{R}_i)$	✓
Signature	s_i	✓

Table 5: EC computations

	Computation
C_A^1	A selects k_A
	A computes $\boldsymbol{r}_A = (x_{\boldsymbol{r}_A}, y_{\boldsymbol{r}_A})$
C_B^1	B selects k_B
	B computes $\boldsymbol{r}_B = (x_{\boldsymbol{r}_B}, y_{\boldsymbol{r}_B})$
	B computes $\boldsymbol{k}_{AB} = K_B\boldsymbol{r}_A = (x_{\boldsymbol{k}_{AB}}, y_{\boldsymbol{k}_{AB}})$
	B computes $s_B = x_{\boldsymbol{k}_{AB}}^{-1}(K_B - x_{\boldsymbol{r}_B}k_B) \bmod q$
C_A^2	A verifies $\boldsymbol{R}_B$ by checking $cert(\boldsymbol{R}_B)$
	A computes $\boldsymbol{k}'_{AB} = k_A\boldsymbol{R}_B = (x'_{\boldsymbol{k}_{AB}}, y'_{\boldsymbol{k}_{AB}})$
	A verifies s_B and $\boldsymbol{k}'_{AB}$ by checking $\boldsymbol{r}_B \overset{?}{=} x_{\boldsymbol{r}_B}^{-1}(\boldsymbol{R}_B - x'_{\boldsymbol{k}_{AB}}s_B\boldsymbol{\alpha})$
	A computes $\boldsymbol{k}_{BA} = K_A\boldsymbol{r}_B = (x_{\boldsymbol{k}_{BA}}, y_{\boldsymbol{k}_{BA}})$
	A computes $s_A = x_{\boldsymbol{k}_{BA}}^{-1}(K_A - x_{\boldsymbol{r}_A}k_A) \bmod q$
C_B^3	B verifies $\boldsymbol{R}_A$ by checking $cert(\boldsymbol{R}_A)$
	B computes $\boldsymbol{k}'_{BA} = k_B\boldsymbol{R}_A = (x'_{\boldsymbol{k}_{BA}}, y'_{\boldsymbol{k}_{BA}})$
	B verifies s_A and $\boldsymbol{k}'_{BA}$ by checking $\boldsymbol{r}_A \overset{?}{=} x_{\boldsymbol{r}_A}^{-1}(\boldsymbol{R}_A - x'_{\boldsymbol{k}_{BA}}s_A\boldsymbol{\alpha})$

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.

Table 6: EC message passing

	Message
T^1_{AB}	$\{\boldsymbol{r}_A\}$
T^2_{BA}	$\{\boldsymbol{r}_B, s_B\}$
T^3_{AB}	$\{s_A\}$

3.3.2 Security analysis: Since our EC-based authenticated key-agreement protocol is a direct extension of the protocol based on a DL assumption in Section 3.2, the security analysis in Section 3.2.2 applies to the EC-based authenticated key-agreement protocol as well.

3.4 RSA factoring assumption

This protocol is a direct extension of the protocol based on a DL assumption in Section 3.2. Based on the general framework depicted in Fig. 2, Section 3.4.1 shows how an authenticated key agreement based on RSA factoring is achieved, and Section 3.4.2 provides security analysis of this scheme.

3.4.1 Algorithm: Table 7 lists the notations used by the algorithm, Table 8 shows the computations performed by users A and B, and Table 9 shows the messages that are transferred between users A and B.

Table 7: RSA notations

Property	Notation (user j is user i's communication partner)	Published information
Long-term secret	p_i, q_i are two safe primes such that $p_i = 2p'_i + 1$, $q_i = 2q'_i + 1$	
Long-term generator	α_i with order $2p'_i q'_i$	✓
Long-term private key	$d_i \in [0, 2p'_i q'_i - 1]$	
Long-term public key	$n_i = p_i q_i$	✓
	e_i where $e_i\, d_i \bmod (2p'_i q'_i) = 1$	✓
	$R_i = \alpha_i^{d_i} \bmod n_i$	✓
Public-key certificate	$cert(n_i, R_i, e_i, \alpha_i)$	✓
Short-term private key	$k_i \in [0, 2p'_j q'_j - 1]$¶	
Short-term public key	$r_i = \alpha_j^{k_i} \bmod n_j$	
Signature	s_i	✓

¶ Because of Euler's totient function, k_i falls in the precise range of $[0, 2p'_j q'_j - 1]$. However, since user i cannot obtain user j's secrets p_j and q_j, user i can select $k_i \in [0, n_j - 1]$ instead. Even though this range is bigger than the precise range, a selection of k_i outside the range $[0, 2p'_j q'_j - 1]$ is modular congruent to some value within this range

3.4.2 Security analysis: This protocol involves both RSA factoring and DL cryptographic assumptions. Although, this protocol involves two cryptographic assumptions, their security relation is a logic AND relationship. To our knowledge, one possible way for an attacker to break this protocol is to first factor n_i into two large primes (i.e. p_i and q_i) and then solve the DL in order to find either the long-term or the short-term private key from its long-term or short-term public key, respectively. This is similar to a safe deposit box in a bank. To break a safe deposit box, one would have to break into the strong room, and then break

Table 8: RSA computations

	Computation
C^1_A	A selects k_A
	A computes $r_A = (\alpha_B)^{k_A} \bmod n_B$
C^1_B	B selects k_B
	B computes $r_B = (\alpha_A)^{k_B} \bmod n_A$
	B computes $k_{AB} = (r_A)^{d_B} \bmod n_B$
	B computes $s_B = (r_B)^{d_B} k_{AB} \bmod n_B$
C^2_A	A verifies (n_B, R_B, e_B) by checking $cert(n_B, R_B, e_B)$
	A computes $k'_{AB} = (R_B)^{k_A} \bmod n_B$
	A verifies s_B and k'_{AB} by checking $(r_B)(k'_{AB})^{e_B} \overset{?}{=} (s_B)^{e_B} \bmod n_B$
	A computes $k_{BA} = (r_B)^{d_A} \bmod n_A$
	A computes $s_A = (r_A)^{d_A} k_{BA} \bmod n_A$
C^3_B	B verifies (n_A, R_A, e_A) by checking $cert(n_A, R_A, e_A)$
	B computes $k'_{BA} = (R_A)^{k_B} \bmod n_A$
	B verifies s_A and k'_{BA} by checking $(r_A)(k'_{BA})^{e_A} \overset{?}{=} (s_A)^{e_A} \bmod n_A$

Table 9: RSA message passing

	Message
T^1_{AB}	$\{r_A\}$
T^2_{BA}	$\{r_B, s_B\}$
T^3_{AB}	$\{s_A\}$

the box. Because these two assumptions are logic AND-related (in particular, RSA factoring comes before the DL), RSA factoring has the same complexity as a DL [3], and the factoring of n_i has twice the number of bits as the DL in our protocol, the security of this protocol depends on the more secure of the two assumptions, which is RSA factoring.

Since our RSA-based authenticated key agreement protocol is also a direct extension of the protocol based on the DL assumption in Section 3.2, the security analysis in Section 3.2.2 applies to the RSA-based authenticated key agreement protocol as well.

4 Summary and conclusion

This paper has proposed three single-assumption authenticated Diffie–Hellman key agreement protocols. Specifically, the first protocol is based on a discrete logarithm, the second on an elliptic curve, and the third on RSA factoring. All three protocols are described based on a general framework: a 3-pass message transmission. In the optimal three passes of message transmission between two communicating parties, not only are both user authentication and shared-key authentication achieved, but also two shared secret keys are established, one for each direction of a secure channel.

The purpose of this paper is to demonstrate that an authenticated Diffie–Hellman key agreement protocol based on one cryptographic assumption is at least as secure as OR-related multiple-assumption counterparts such as the DL-based Diffie–Hellman key distribution using an RSA

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.

signature on the MD5 hash value. We conclude that the security of a protocol should not be assessed at the level of the individual algorithms used to compose the protocol, rather it should be assessed at the overall protocol level.

5 References

1 Beth, T., Frisch, M., and Simmons, G.: 'Public-key cryptography: state of the art and future directions' (Springer-Verlag, New York, USA, 1991)

2 Menezes, A.: 'Elliptic curve public key cryptosystems'. Kluwer Int. Ser. Eng. Comput. Sci., Vol. 234 (Kluwer, 1993)

3 Lenstra, A., and Lenstra, H. Jr., (Eds.): 'The development of the number field sieve', *Lect. Notes Math.*, 1993, **1554**

4 Canetti, R., and Krawczyk, H.: 'Security analysis of IKE's signature-based key-exchange protocol', Presented at Crypto, 2002

5 Krawczyk, H.: 'SIGMA: the SIGn-and-MAc approach to authenticated Diffie–Hellman and its use in the IKE protocols'. http://www.ee.technion.ac.il/hugo/sigma.html. (A shorter version can be found in Proc. Crypto 2003, *Lect. Notes Comput. Sci.* **2729**)

6 Arazi, A.: 'Integrating a key cryptosystem into the digital signature standard', *Electron. Lett.*, 1993, **29**, (11), pp. 966–967

7 Harn, L., Mehta, M., and Hsin, W.-J.: 'Integrating Diffie–Hellman key exchange into the digital signature algorithm (DSA)', *IEEE Commun. Lett.*, 2004, **8**, (3)

8 Blake-Wilson, S., and Menezes, A.: 'Authenticated Diffie–Hellman key agreement protocols', *Lect. Notes Comput. Sci.*, 1999, **1556**

9 IEEE standard-p1363. 'Standard specifications for public-key cryptography'. http://grouper.ieee.org/groups/1363/

10 Harn, L., and Lin, H.-Y.: 'An authenticated key agreement without using one-way hash functions'. Proc. 8th Nat. Conf. on Information Security, Kaohsiung, Taiwan May 1998, pp. 155–160

11 Harn, L., and Lin, H.-Y.: 'Authenticated key agreement without using one-way hash function', *Electron. Lett.*, 2001, **37**, (10)

12 Shim, K.: 'Unknown key-share attack on authenticated multiple-key agreement protocol', *Electron. Lett.*, 2003, **39**, (1), pp. 38–39

13 Yen, S., and Joye, M.: 'Improved authenticated multiple-key agreement protocol', *Electron. Lett.*, 1998, **34**, (18), pp. 1738–1739

14 Wu, T., He, W., and Hsu, C.: 'Security of authenticated multiple-key agreement protocols', *Electron. Lett.*, 1999, **35**, (5), pp. 391–392

15 Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: 'An efficient protocol for authenticated key agreement'. Technical report CORR 98-08. Dept of C&O, University of Waterloo, Canada, March 1998

16 Harn, L.: 'Digital signatures for Diffie–Hellman public keys without using a one-way function', *Electron. Lett.*, 1997, **33**, (2), pp. 125–126

17 Kaliski, B.: 'An unknown key-share attack on the MQV key agreement protocol', *ACM Trans. Inf. Syst. Secur.*, 2001, **4**, (3), pp. 275–288

18 ElGamal, T.: 'A public key cryptosystem and a signature scheme based on discrete logarithms', *IEEE Trans. Inf. Theory*, 1985, **31**, pp. 469–472

19 Nyberg, K., and Rueppel, R.: 'Weaknesses in some recent key agreement protocols', *Electron. Lett.*, 1994, **30**, (1), pp. 26–27

20 Agnew, G., Mullin, R., and VanStone, S.: 'Improved digital signature scheme based on discrete exponentiation', *Electron. Lett.*, 1990, **26**, (14), pp. 1024–1025

Authorized licensed use limited to: University of Missouri. Downloaded on February 10, 2009 at 14:39 from IEEE Xplore. Restrictions apply.