Chaemeleon-Based Deniable Authenticated Key Agreement Protocol

Chunbo Ma[1,3], Jun Ao[2], and Jianhua Li[1]

[1] School of Information Security Engineering
Shanghai Jiao Tong University, Shanghai, 200030, P. R. China
machunbo@sjtu.edu.cn

[2] State Key Laboratory for Radar Signal Processing,
Xidian University, Xi'an, Shanxi, 710071, P. R. China
Junjunao1@263.net

[3] The State Key Laboratory of Information Security,
Beijing, 100049, P. R. China

Abstract: As a useful means of safeguarding privacy of communications, deniable authentication has received much attention. A Chameleon-based deniable authenticated key agreement protocol is presented in this paper. The protocol has following properties. Anyone of the two participants can't present a digital proof to convince a third party that a claimed agreement has really taken place. Once a forgery occurs, the original entity with some information can present a digital proof to disclose the forgery.

Keywords: Chameleon, Deniability, authentication, key agreement

1. Introduction

Key agreement is one of most important security mechanisms in the area of secure communications. Such protocols allow two entities to exchange information between them and establish a shared secret over an insecure open channel. Later, they can encrypt actual data using a fast symmetric cipher keyed by the shared secret. The first two-party key agreement is the Diffie-Hellman protocol given in their seminal paper [1]. Currently, how to design an efficient and secure key agreement protocol have received much attention.

Due to the lack of authentication, the original Diffie-Hellman protocol is vulnerable to "man-in-the-middle" and some other attacks. In order to solve this issue, many two-party authenticated key agreement protocols have been proposed[2][3][4][5]. Authenticated two-party key agreement allows two users to establish a common secret key and ensures nobody besides them can possibly learn the secret key.

However, in some applications [6], deniability is needed to prevent an authorized user from disclosing information it receives legitimately. For example, Alice and Bob have complemented an authenticated key agreement protocol and established a session key (shared secret). Later, Bob presents a digital proof to convince Carol that Alice once sent some message to him. In this process, Bob discloses Alice's some information without Alice's authorization and impinges upon Alice's privacy.

To solve above issue, Alice and Bob can use deniable authenticated key agreement protocol. Under such circumstances, Bob can't present proof to convince the third party Carol that there is a certain key agreement protocol occurred between them. The deniable

authenticated key agreement protocol is seldom investigated, though there is a lot of research on deniable authentication technology, such as [7] and followed by a series of papers including [8][9]. Raimondo et al. [10] recently extended the work of Dwork from deniable authentication to deniable agreement protocol, and proved the deniability features of SKEME [11] and SIGMA [12].

We can't prevent a deniable key agreement protocol from being forged even though we design it with some secure technologies. One of the reasons is that we have no a method to prove the protocol secure against any attacks including know and unknown. In other words, after completing the key agreement protocol, Bob can produce a forged protocol using the message once transmitted and announces that the protocol has executed between him and Alice. And then, Bob submits corresponding information to convince the third party Carol that the forged protocol is true. Alice's privacy may be impinged while Carol can't judge the protocol's reality. Hence, we need a mechanism to disclose a forgery in case of occurrence.

Chameleon Hash has some special properties, and can be used to design signature and some other cryptography mechanism. To the Chameleon-based signature, the recipient can't convince the third party of the identity of the signer, as the recipient has the ability to forge the signature. In the case of forgery occurrence, the original signer can disclose the forgery in non-interactive manner. The first to formally treat the Chaemelon Hash problem were Krawczyk [13] in 2000, followed by papers [14][15]. The properties of Chameleon Hash are very useful to our designing two-party deniable authenticated key agreement protocol.

Motivated by above statement, we design a two-party key agreement protocol with Chameleon Hash and signature. In our mechanism, neither Alice nor Bob can forge a key agreement protocol. In the event that Bob forges the protocol between Alice and him, Alice can get Bob's private key with the forged message, and consequently presents a digital proof to disclose the forgery.

2. Background

2.1 Preliminaries

Let G_1 be a cyclic multiplicative group generated by g, whose order is a prime q and G_2 be a cyclic multiplicative group of the same order q. Assume that the discrete logarithm in both G_1 and G_2 is intractable. A bilinear pairing is a map $e: G_1 \times G_1 \to G_2$ and satisfies the following properties:

1. *Bilinear:* $e(g^a, p^b) = e(g, p)^{ab}$. For all $g, p \in G_1$ and $a, b \in Z_q$, the equation holds.
2. *Non-degenerate:* There exists $p \in G_1$, if $e(g, p) = 1$, then $g = \mathrm{O}$.
3. *Computable:* For g, $p \in G_1$, there exists an efficient algorithm to compute $e(g, p)$.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. Pairings and other parameters should be selected in proactive for efficiency and security.

2.2 Chameleon Hash

Let G_1 and G_2 be two groups that support a bilinear map as defined in section 2.1. PKG random chooses $v \in Z_q^*$ as the private key of the system, and computes the matching public key $PK_{pub} = g^v$, and then random chooses $a \in Z_q^*$ and generates Alice's key pair ($SK_A = g^{v \cdot a}$, $PK_A = g^a$). Similarly, PKG generates Bob's key pair $(SK_B, PK_B) = (g^{v \cdot b}, g^b)$. Alice chooses $x_A \in Z_q^*$ and $R_A \in G_1$ uniformly at random, and generates Chameleon Hash.

$$T_A(PK_A, x_A, R_A) = e(R_A, g)e((PK_A)^{x_A}, P_{pub}).$$

The Chameleon Hash function has following properties.

a) Alice who has known (T_A, x_A, R_A) random chooses $x_A^{'} \in Z_q^*$ and computes $R_A^{'} = g^{a \cdot v \cdot (x_A - x_A^{'})} \cdot R_A$, which satisfies $T_A(PK_A, x_A, R_A) = T_A(PK_A, x_A^{'}, R_A^{'})$. We have

$$\begin{aligned} & T_A(PK_A, x_A^{'}, R_A^{'}) \\ &= e(g^{a \cdot v \cdot (x_A - x_A^{'})} \cdot R_A, g)e(g^{a \cdot x_A^{'}}, PK_{pub}) \\ &= e(g^{a \cdot (x_A - x_A^{'})}, PK_{pub})e(R_A, g)e(g^{a \cdot x_A^{'}}, PK_{pub}) \\ &= e(g^{a \cdot x_A}, PK_{pub})e(R_A, g) \\ &= T_A(PK_A, x_A, R_A). \end{aligned}$$

In the circumstances of having known (x_A, R_A), if Alice can compute another $(x_A^{'}, R_A^{'})$ that satisfies above relationship, we say that Alice can successfully forge (T_A, x_A, R_A).

b) To the given $T_A(PK_A, x_A, R_A) = T_A(PK_A, x_A^{'}, R_A^{'})$, anyone can compute and get Alice's private key as follows.

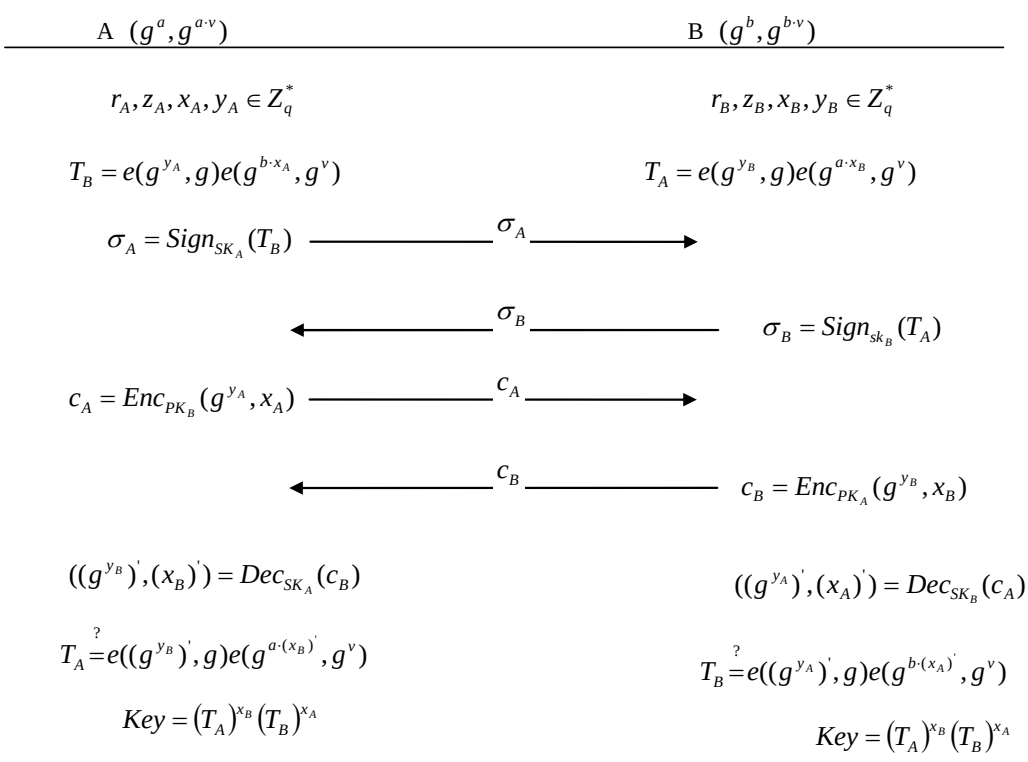
$$T_A(PK_A, x_A, R_A) = T_A(PK_A, x_A^{'}, R_A^{'})$$

$$e(R_A, g)e(g^{a \cdot x_A}, g^v) = e(R_A^{'}, g)e(g^{a \cdot x_A^{'}}, g^v)$$

$$R_A \cdot g^{a \cdot v \cdot x_A} = R_A^{'} \cdot g^{a \cdot v \cdot x_A^{'}}$$

$$g^{a \cdot v} = (R_A \cdot (R_A^{'})^{-1})^{(x_A^{'} - x_A)^{-1}}$$

3. Key Agreement Protocol



Let G_1 and G_2 be two groups that support a bilinear map as defined in section 2.1. The keys distribution is as defined in section 2.2. Assume that there exists a secure signature $Sign$ and IND-CCA2 encryption algorithm (Enc, Dec). The two entities Alice and Bob will execute the protocol as following steps.

1. Alice chooses $r_A, z_A, x_A, y_A \in Z_q^*$ uniformly at random, and computes $T_B = e(g^{y_A}, g)e(g^{b \cdot x_A}, g^v)$, and then signs T_B using secure signature algorithm $Sign$. Alice sends $\sigma_A = Sign_{SK_A}(T_B)$ to Bob. Similarly, Bob produces $\sigma_B = Sign_{sk_B}(T_A)$ and sends it to Alice.
2. Alice encrypts x_A and g^{y_A} using Bob's public key, and sends $c_A = Enc_{PK_B}(g^{y_A}, x_A)$ to Bob. Bob does the same things as Alice, and sends $c_B = Enc_{PK_A}(g^{y_B}, x_B)$ to Alice.
3. Alice gets $(x_B)^{'}$ and $(g^{y_B})^{'}$ by decrypting c_B, and then verifies the values as follows.

$$T_A \stackrel{?}{=} e((g^{y_B})^{'}, g)e(g^{a \cdot (x_B)^{'}}, g^v) \quad (1).$$

If above equation holds, Alice produces the session key $Key = (T_A)^{x_B}(T_B)^{x_A}$. Bob

does the same things as Alice, and verifies $(x_A)'$ and $(g^{y_A})'$ by the following equation.

$$T_B \stackrel{?}{=} e((g^{y_A})', g)e(g^{b\cdot(x_A)'}, g^v) \quad (2).$$

If above equation holds, Bob produces the session key $Key = (T_A)^{x_B}(T_B)^{x_A}$.

4. Security

a) Deniability

Neither Alice nor Bob can convince the third party Carol that a claimed key agreement protocol has really taken place. To Bob's signature $\sigma_A = Sign_{SK_A}(T_B)$, and the values x_A and g^{y_A}, Carol can distinguish whether the signature really comes from Alice, and verifies the equation $T_B \stackrel{?}{=} e(g^{y_A}, g)e(g^{b\cdot x_A}, g^v)$, but she can't judge whether the two values x_A and g^{y_A} come from Alice since Bob has the ability to forge the two values. Bob can random choose x_A' and forge $g^{y_A'} = g^{a\cdot v\cdot(x_A - x_A')} \cdot g^{y_A}$ as we have mentioned in the section 2.2. Obviously, x_A' and $g^{y_A'}$ satisfy the equation (2). Therefore, Carol can't tell whether there is a claimed key agreement between Alice and Bob, also she can't work out whether the session key $Key = (T_A)^{x_B}(T_B)^{x_A}$ is generated by them.

b) Unforgeability

If Bob forges a key agreement protocol after his agreement with Alice, Alice can compute Bob's private key in the case of having known (T_B, x_A, g^{y_A}). Due to T_B is the value that Alice sends to Bob, Alice can find corresponding $(T_B, x_A', g^{y_A'})$ in her recorder. So we have

$$e(g^{y_A}, g)e(g^{b\cdot x_A}, g^v) = e(g^{y_A'}, g)e(g^{b\cdot x_A'}, g^v)$$

$$g^{y_A} \cdot g^{b\cdot v\cdot x_A} = g^{y_A'} g^{b\cdot v\cdot x_A'}$$

$$g^{b\cdot v} = (g^{y_A} / g^{y_A'})^{(x_A' - x_A)^{-1}}$$

Therefore, Alice has the ability to get Bob's private key $g^{b\cdot v}$, and work out other two values

$g^{y_A^{"}}$ and $x_A^{"}$ that satisfy the requirement. She chooses $x_A^{"} \in Z_q^*$ uniformly at random and then computes $g^{y_A^{"}} = g^{b \cdot v \cdot (x_A - x_A^{"})} \cdot g^{y_A}$. We have

$$T_B = e(g^{y_A^{"}}, g)e(g^{b \cdot v \cdot x_A^{"}}, g^v)$$

5. Conclusion

In some communication scenarios, deniability is playing an important role in protecting privacy. A Chameleon-based deniable authenticated key agreement protocol is presented in this paper. In our mechanism, the two-party who participant the communication can't present digital proof to convince the third party that a claimed key agreement protocol is executed between them. If any participant forges a key agreement protocol and produces a session key, the original entity can work out the forger's private key and then discloses the forgery by giving other two values that satisfy the requirement. The key agreement protocol has such properties due to Chameleon hash function.

References

[1] W. Fiffie and M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, 6 (1976), 644C654.

[2] N. P. Smart. An Identity-based Authenticated Key Agreement Protocol based on the Weil Pairing. In Electronic Letters, 38, PP. 630-632, 2002.

[3] M. Scott. Authenticated ID-based Key Exchange and Remote Log-in with Insecure Token and PIN Number.

[4] L. Chen and C. Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings. Available at Http://Eprint.iacr.org/2002/184

[5] N. McCullagh and P. S. L. M. Barreto. A New Two-Party Identity-Based Authenticated Key Agreement. In proceedings of CT-RSA 2005, LNCS 3376, pp. 262-274, Springer-Verlag, 2005. Also available at http://eprint.iacr.org/2004/122.

[6] J. Katz, Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications, Advances in Cryptology-proc. of EUROCRYPT'03, LNCS 2656, Springer-Verlag, pp. 211-228, 2003.

[7] C. Dwork, M. Naor and A. Sahai, Concurrent Zero-Knowledge, proc. of 30th Symposium on Theory of Computing (STOC), ACM Press, pp. 409-418, 1998.

[8] M. Di Raimondo and R. Gennaro, New Approaches for Deniable Authentication, proc. of 12nd ACM Conference on Computer and Communications Security (CCS'05), ACM Press, pp. 112-121, 2005.

[9] R. Pass, On Deniability in the Common Reference String and Random Oracle Model, Advances in Cryptology-proc. of CRYPTO'03, LNCS 2729, Springer-Verlag, pp. 316-337, 2003.

[10] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable Authentication and Key Exchange. Available at http://eprint.iacr.org/2006/280

[11] H. Krawczyk, SKEME: a versatile secure key exchange mechanism for Internet, proc. of 1996 IEEE Symposium on Network and Distributed System Security (SNDSS'96), pp. 114-127.

[12] H. Krawczyk, SIGMA: The 'SiGn-and-Mac' Approach to Authenticated Diffie-Hellman and Its Use in the

IKE Protocols, Advances in Cryptology-proc. of CRYPTO'03, LNCS 2729, Springer-Verlag, pp. 400-425, 2003.

[13] H. Krawczyk and T. Rabin. Chameleon signatures. In Proc. of NDSS 2000, 2000: 132-154.

[14] G. Ateniese and B. de Medeiros. Identity-based chameleon hash and applications. http://eprint.iacr.org/2003/167.

[15] F. Zhang, R. Safavi-Naini, and W. Susilo. ID-based Chameleon hashes from bilinear pairings. http://eprint.iacr.org/2003/208.