

An Efficient and Secure Key Agreement

Chang-Hyi Lee¹, Jong-In Lim², and Jeong-Soo Kim³

¹ ³Samsung Advanced Institute of Technology,
{chang¹, integer³}@sait.samsung.co.kr
² Korea University
jilim@tiger.korea.ac.kr

Abstract. In this paper we propose a new and efficient protocol for authenticated key agreement based on Diffie-Hellman key agreement, which works in an arbitrary finite group. Our AK(Key Agreement)protocol saves some of computational cost, since it requires only two dominant computation factors(e.g., modulo exponentiation in RSA type system or integer multiplication with a point in elliptic curve cryptosystem) for each entity. Our protocol is not also efficient for two passing AK but also for three passing authenticated AK protocol with key confirmation(AKC).

1 Introduction

The secure *key agreement* is the most important factor for the secure communication. For this, two or more distributed entities need to share some keys in secret, called *session keys*, among them. To solve such an AK-problem(Key Agreement problem), several techniques based on Diffie-Hellman problem[3] have been proposed. But many of them have been turned out to be flawed[5, 6]. In recent, a number of desirable *attributes* of key agreement protocols have also been identified[7] and nowadays most protocols are analyzed with such attributes. In this paper we propose a two pass AK-protocol and analyze it by checking whether our protocol meets these attributes.

2 Attributes of AK-Protocols

Here we list up a number of desirable attributes of AK protocols which referred to [8, 7, 5].

1. *known key security.* Each run of key agreement protocol between two entities A and B should produce a unique secret key; such keys are called *session keys*. A protocol should still achieve its goal in the face of an adversary who has learned some other session keys.
2. *(perfect) forward secrecy.* If long-term private keys of one or more entities are compromised, the secrecy of previous session keys established by honest entities is not affected.
3. *key-compromise impersonation.* Suppose A 's long-term private key is disclosed. Then clearly an adversary that knows this value can now impersonate A , since it is precisely this value that identifies A . However, it may be desirable that this loss does not enable an adversary to impersonate other entities to A .
4. *unknown key-share.* Entity A can not be coerced into sharing a key with entity B without A 's knowledge, i.e., when A believes the key is shared with some entity $C \neq B$, and B (correctly) believes the key is shared with A .
5. *key control.* Neither entity should be able to force the session key to a preselected value.

In addition to these security attributes, it would be desirable for a protocol to have low computational cost and low communication overhead for its practical use.

2.1 Flaws in Some AK-protocols

In [8], the small subgroup(see [13]) attack and unknown key-share attack(see [14]) on MTI/C0 and MTI/A0 AK-protocols respectively proposed by Matsumoto, Takashima and Imai[10] in 1986 are presented as illustrations for the flaws in protocols. For other considerations of active attacks on AK protocols, refer to [9, 11]. The authors in [8] proposed a new authenticated key agreement protocol, MQV, which is resistant against such attacks and has some computational advantage with about 2.5 integer multiplications for each entity in its model for elliptic curve cryptosystem(where the integer multiplication to an elliptic curve point is the dominant factor in its computational complexity). However, in [12] B. Kaliski showed that the MQV protocol does not possess the unknown key-share attribute.

In the next section, we describe a new authenticated key agreement protocol which meets the several attributes listed in the above and which has low computational cost, just two integer multiplications(or two modulo exponentiations in modulo number system) for each entity.

3 Proposed AK-protocol, LLK

In this section we describe our two-pass AK protocol between two entities, Alice and Bob in the ECC version, and consider its security and efficiency. Hereafter, we assume the following notations(in the notation, the subscript index a or b indicates the owner of the value, Alice or Bob):

- G : the generating element(point) of ECC under consideration, $E(\mathbf{F}_q)$
- n : the order of G in $E(\mathbf{F}_q)$
- h : the cofactor of n , i.e., $h = \#E(\mathbf{F}_q)/n$
- x_a : Alice's private key
- Y_a : Alice's public key(= $x_a G$)
- k_a : Alice's ephemeral key(random number in \mathbf{Z}_n)

3.1 Description of LLK protocol

Here we describe the AK protocol following the above notations and all the integer arithmetic work in \mathbf{Z}_n .

The protocol works in the following steps:

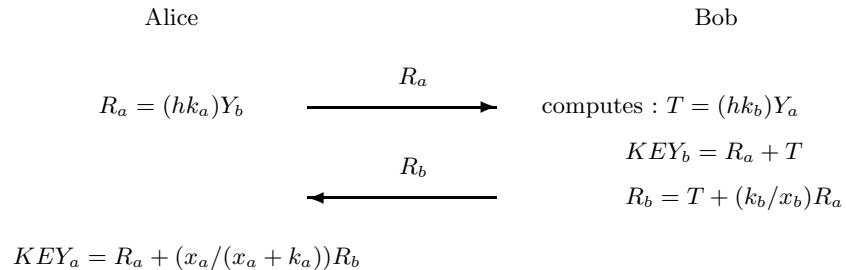


Fig. 1. Two-pass AK protocol, LLK

1. Alice generates a random integer $k_a \neq 0 \in \mathbf{Z}_n$, computes $R_a = (hk_a)Y_b$. If $R_a = \mathcal{O}$, she terminates the protocol run with failure. Otherwise she sends it to Bob.
2. Upon receiving R_a , Bob generates a random integer $k_b \neq 0 \in \mathbf{Z}_n$, computes the session key $KEY_b = R_a + hk_bY_a = h(k_ax_b + k_bx_a)G$. If the value hk_bY_a is \mathcal{O} , he terminates the protocol run with failure. Otherwise he computes $R_b = hk_bY_a + \frac{k_b}{x_b}R_a$, and sends it to Alice.
3. Upon receiving R_b , Alice computes the session key $KEY_a = R_a + \frac{x_a}{x_a+k_a}R_b = h(k_ax_b + k_bx_a)G$.
4. In each step 3 and 4, if K_a or K_b is \mathcal{O} , then the protocol run is terminated with failure.
5. After regular protocol running, Alice and Bob share the secret, $K = KEY_a = KEY_b$.

3.2 Security Consideration

Here we prove our protocol meets the following desirable attributes under the assumption that the discrete logarithm problem(DLP) is secure.

Known-Key Security: If, under the well set ECC, the two entities execute the regular protocol run, they clearly share their unique session key K as in the figure.

(Perfect) Forward Secrecy : As one sees in Fig.1, during the computation of the session key K for each entities, the random factors k_a and k_b still act on it. So an adversary who captured their private keys x_a or x_b should extract the ephemeral keys k_a or k_b from the informations R_a and R_b to know the previous or next session keys between them. However, this is the very discrete logarithm problem(DLP). Hence, under the assumption that DLP is secure, LLK meets the *forward secrecy*.

Key-compromise Impersonation : Suppose Alice's long-term private key, x_a , is disclosed. Now an adversary who knows this value can clearly impersonate Alice. Is it possible for the adversary impersonates Bob to Alice without knowing the Bob's long-term private key, x_b ? For the success of the impersonation, the adversary must know Alice's ephemeral key k_a at least. So, also in this case, the adversary should extract the value k_a from the Alice's ephemeral public value, $R_a = hk_aY_b$, to generate the same key, K , with Alice. This also comes to DLP.

Unknown Key-Share : Suppose an adversary E now try to make A believe that the session key is shared with B , while B believes that the session key is shared with E . To launch the unknown key-share attack, the adversary E should set his public key to be certified even though he does not know his correct private key. For this E makes it by utilizing the public values Y_a, Y_b and G (the Alice(A)'s public key, Bob(B)'s public key and generator point, G , of the ECC, respectively). Here we make some notations for simpler description. Let

$$f_\tau(X_1, X_2, \dots, X_r) := \sum_{i=1}^r \tau_i X_i,$$

where X_i 's are points on $E(\mathbf{F}_q)$ and $\tau = (\tau_1, \dots, \tau_r)$ be a stream of integers in \mathbf{Z}_n . Then, in general, E should set his public key Y_e as

$$Y_e = f_\tau(Y_a, Y_b, G) = \tau_1 Y_a + \tau_2 Y_b + \tau_3 G.$$

Now suppose E got the value Y_e certified as his public key and suppose the following generalized model for unknown key-share attack:

where $R_e = f_\rho(Y_a, Y_b, G, R_b)$ and $R_e^* = f_\sigma(Y_a, Y_b, G, R_a)$. For E to launch the unknown key-share attack successfully, he should force A and B to share the same secret $K = Key_a = Key_b$ through the protocol run. In practice, through the protocol run, A and B get their session keys Key_a and Key_b respectively as those in the following:

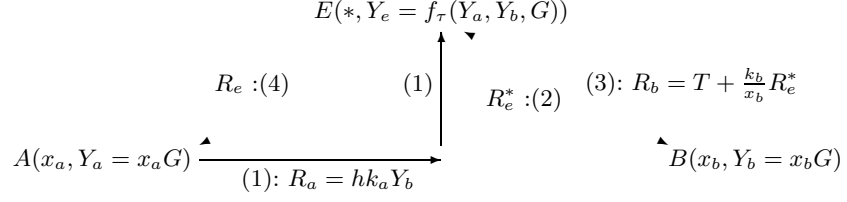


Fig. 2. Unknown key-share attack model

$$KEY_a = R_a + \frac{x_a}{x_a + k_a} R_b \quad (1)$$

$$KEY_b = (hk_b)Y_a + \frac{k_b}{x_b} R_e^* \quad (2)$$

Note here that E does not know x_a , x_b , k_a and k_b even though E can control the integer values τ_i , ρ_i and σ_i . E should force the equation $KEY_a = KEY_b$ to hold for many (in practice, for all) values of k_a and k_b . So we can consider the following equation as an identical one for the variables k_a and k_b .

$$R_a + \frac{x_a}{x_a + k_a} R_b = (hk_b)Y_a + \frac{k_b}{x_b} R_e^* \quad (3)$$

And we can change the equation 3) as the form $\alpha G = \mathcal{O}$, by unfolding the values R_a , Y_e , R_e and R_e^* with respect to G . Then we cannot solve this equation for τ_i , ρ_i and σ_i , since we don't have sufficient information on x_a , x_b , k_a and k_b . So the unknown key-share attack fails.

Key Control : As the same argument in the above, the *key-control* is clearly impossible for the third party. The only possibility of *key-control* attack may be brought out by the participant of the protocol, B(Bob). But for the party, B , to make the party, A , generate the session key $K(= KEY_b)$ which is pre-selected value by B , for example B should solve the following equation(see the Fig. 1):

$$KEY_b = R_a + k_b(hY_a).$$

But this again falls into the problem of DLP.

3.3 Effectiveness of the AK-protocol, LLK

As one sees in the previous sections, for the key agreement by two-pass LLK, there are required only two dominant computations for each entity and the LLK meets the desirable attributes for secure AK protocol. Of course, the LLK has an one-pass delay through the protocol run which differs from the other two-pass AK protocols. But it is a minor factor for secure AK-protocols and moreover, in the case of three-pass AK protocol with key confirmation, LLK is more effective than others. Here we do not mention further about three-pass LLK with key confirmation. In the following, we use some abbreviations for convenience like as :

- UKA - Unknown Key-share Attack
- SSA - Small Subgroup Attack
- KRA - Key Recovery Attack

- MMK - Man in the Middle Attack

And the complexity is measured by the number of dominant computations(e.g. modulus exponentiation or integer multiplication with a point in ECC) for each party.

Table 1. Security and Complexity Comparison for various AK protocols

	LLK	EC-DH	EC-MQV [12]	MTI/A0 [14]	MTI/C0 [11]
Complexity	2	2	2.5	3	2
Attacks	?	KRA, MMA	UKA	UKA	SSA

4 Limitations and Intellectual Property Issue

Most of DH-based AK protocols are initiated by each communicator's sending their random ephemeral key to each other at the same time. In practical case, however, this is done by the session caller's sending his/her random ephemeral key and the receiver's responding with a random ephemeral key. In this point of view, our proposed AK protocol, LLK, can be considered as not to have more protocol passing delays than others do.

Currently a reasonable and non-discriminatory patent application of our AK protocol is being processed.

5 Conclusion

In this paper we proposed a new and efficient key agreement protocol, called LLK. It is secure in the sense that it meets some desirable attributes of secure AK protocol and it is effective in the sense of its computational cost. It requires only two dominant computations such as modulo exponentiation in modulo number system or integer multiplication to a generator point in an elliptic curve cryptosystem. This is more effective than other authenticated key agreement protocols which are believed secure.

References

1. ANSI X9.62, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, working draft, November 1997.
2. ElGamal, T., *A public key cryptosystem and a signature scheme based on discrete logarithm*, IEEE Trans. Inform. Theory, IT-31, no.4, pp469-472, July 1985.
3. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, IT-22, no.6, pp644-654, November 1976.
4. M. Bellare and P. Rogaway, *Entity Authentication and Key Distribution*, In *Advances in Cryptology CRYPTO'93*, pp.341-358, 1994.
5. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, chapter 12. CRC press, 1997.
6. M. Burmester, *On the risk of opening distributed keys*, In *Advances in Cryptology CRYPTO'94*, pp.308-317, 1994.
7. Simon Blake-Wilson, Don Johnson, and Alfred Menezes, *Key Agreement Protocols and Their Security Analysis*, Sixth IMA International Conference on Cryptography and Coding, Cirencester, England, December 1997.

8. Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas and Scott Vanstone, *An Efficient Protocol for Authenticated Key Agreement*, Available from <http://grouper.ieee.org/groups/1363/contrib.html>, 1988.
9. W. Diffie, P. van Oorschot and M. Wiener, *Authentication and authenticated key exchanges*, *Designs, Codes and Cryptography*, **2**(1992), pp.107-125.
10. T. Matsumoto, Y. Takashima and H. Imai, *On seeking smart public-key distribution systems*, *The Transactions of the IECE of Japan*, **E69**(1986), pp.99-106.
11. M. Just and S. Vaudenay, *Authenticated multi-part key agreement*, In *Advances in Cryptology ASIACRYPT'96*, *Lecture Notes in Computer Science*, **1294**, Springer-Verlag, 1996, pp.36-49.
12. B. Kaliski, Contribution to ANSI X9F1 and IEEE P1363 working groups, June 1998.
13. A. Menezes, M. Qu and S. Vanstone, *Key agreement and the need for authentication*, Presentation at PKS'95, Toronto, Canada, November 1995.
14. A. Menezes, M. Qu and S. Vanstone, *Some new key agreement protocols providing mutual implicit authentication*, Workshop on Selected Areas in Cryptography (SAC'95), pp.22-32, 1995.