Proc. of International Symposium on Information Theory and its Applications, pp. 467–470 (corrected)

(Mexico City, Oct. 14–16, 1998)

Protection of Authenticated Key-Agreement Protocol against a Denial-of-Service Attack

Kanta Matsuura and Hideki Imai
Institute of Industrial Science, the University of Tokyo
Roppongi 7-22-1, Minato-ku, Tokyo 106-8558, JAPAN
E-Mail: kanta@iis.u-tokyo.ac.jp

Abstract. Authenticated and secure key-agreement protocols without a trusted key-distribution center usually owe a lot to public-key primitives whose implementation includes modular exponentiation. Since modular exponentiation is computationally expensive, protocols should be carefully designed so that attackers will not be motivated to use Denial-of-Service (DoS) attacks. Considering this design direction, this paper first shows a basic protection strategy against DoS attacks based on public-key related computational cost. We then propose a three-pass authenticated Diffie-Hellman key-agreement protocol conforming to the strategy; DoS attacks impose expensive computation on the attackers themselves.

Key words: denial-of-service attack; modular exponentiation; Diffie-Hellman key-agreement; authentication; extra hash.

1 Introduction

In order to use cryptographic communication in open networks, how to establish session keys is a fundamental problem. The most well-known scheme is Diffie-Hellman key-agreement protocol [1]. It is also well-known that this protocol on its own is vulnerable to intruder-in-the-middle attack; an attacker who controls the path may intercept the protocol messages and masquerade as each of the users involved. Protection mechanisms against this attack has long been explored by providing entity authentication [2]–[4]. As a result, the protocol usually costs a lot in computation and/or in communication.

In a real network, attackers may use such a costly aspect for exhausting the resource of their targets; we should consider the issue of Denial-of-Service (DoS) attacks in designing key-agreement protocols. The following proposals are aimed for DoS-resistance of three-pass protocols.

2 Basic Strategy

Let us assume that the initiator I and the responder R can use the same public-key infrastructure where public and secret keys of an entity X are denoted by PK_X and SK_X, respectively. Then three-pass key-agreement protocols can be typically described as in Fig. 1, where we can find nine steps involved.

Our proposal aiming for DoS-resistance is as follows:

1. Do not use heavy computation such as modular exponentiation in Step ((4)).
2. Carry a random fresh material by Message ((5)) implicitly in a way that the reconstruction of the material requires heavy computation in Step ((6)) by the initiator with PK_R and/or SK_I. This computational load must be comparable to or higher than that of Step ((8)).
3. Use the random material for authentication.
4. Equip Message ((7)) with an acknowledgment material derived from the reconstructed random material.
5. Verify the acknowledgment material by inexpensive computation at the beginning of Step ((8)).

Once conformed, this mechanism discourages DoS attackers by "falling-together" nightmare; if the attacker and the responder or target have the same level of computational power, the attacker must exhaust his/her own resource in order to exhaust that of the target since a bogus acknowledgment material is detected by the light verification procedure before computationally-expensive parts of Step ((8)).

3 DoS-Resistant Protocol

Figure 2 illustrates our new protocol, where a large prime p and generator α of $\boldsymbol{Z}_p^*$ ($2 \leq \alpha \leq p-2$) are

Step	**(Initiator)**		**(Responder)**
((1))	Precomputation (as a regular or periodical background job, *etc.*).		Precomputation (as a regular or periodical background job, *etc.*).
((2))	Computation by using PK_R and/or SK_I.		
((3))	Request Message	$\Longrightarrow$	
((4))			Computation by using PK_I and/or SK_R. (Session-key computation may be included.)
((5))		$\Longleftarrow$	Reply Message
((6))	Computation by using PK_R and/or SK_I. (Session-key computation may be included.)		
((7))	Acknowledgment Message	$\Longrightarrow$	
((8))			Computation by using PK_I and/or SK_R.
((9))	Key establishment.		Key establishment.

Figure 1: Typical structure of three-pass authenticated key-agreement protocols. PK_I and PK_R are the public keys of the initiator and the responder, respectively. SK_I and SK_R are the corresponding secret keys of the initiator and the responder, respectively.

chosen and published in advance. $\in_R$ represents random picking of an element from the right-hand side set. q is a large prime factor of $p-1$. A public integer g with order q modulo p is chosen randomly from $[1, 2, \cdots, p-1]$. A hashed value H plays a role of "acknowledgment material" in our falling-together trick. This hash will be referred to as an *extra hash*, in the following. Timestamp TS_I can be replaced with fresh materials of another kind such as nonces or cookies. Several different versions can be constructed by different formation of w_R and w_I.

4 Discussion

4.1 Passive Attack

Security of the Diffie-Hellman (DH) and related key-agreement protocols are actively explored (for example, in [5]–[7]). Among them, let us note that the DH problem for a subgroup with a smooth index is polynomial-time equivalent to the DH problem for the original group [6]. Therefore, we assume the following problem is sufficiently difficult and that there exists no probabilistic polynomial-time algorithm which can solve it with non-negligible probability.

[***Problem1***]
For given $u_I(=g^{k_I} \bmod p)$ and $u_R(=g^{k_R} \bmod p)$, compute $K_{IR} = g^{k_I k_R}(=K_{RI})$.

Regarding the security of the proposed three-pass protocol against passive eavesdropping, we have the following theorem:

[***Theorem I***]

If an oracle C can compute $K_{RI}(=K_{IR})$ with non-negligible probability by eavesdropping on the communication between any pair of I and R with any hash function, then there exists a probabilistic polynomial-time algorithm A^C which can solve Problem 1 with non-negligible probability.

(Proof)

For a given instance (u_I, u_R) of Problem 1, A^C simulates the execution of the proposed protocol by I and R in the following way:

1. Get a timestamp TS_I.

2. A^C randomly picks e_R, w_R from $\boldsymbol{Z}_q$, computes $s_R = u_R^{w_R} PK_R^{e_R \cdot w_R} \bmod p$, and then computes the extra hash as $H = hash\,(s_R, u_R, u_I)$.

3. A^C randomly picks e_I, w_I from $\boldsymbol{Z}_q$, computes $s_I = u_I^{w_I} PK_I^{e_I \cdot w_I} \bmod p$.

Subsequently, the resulting materials u_I, ID_I, TS_I, u_R, e_R, w_R, ID_R, H, e_I, and w_I are forwarded from A^C to C. Finally, A^C can solve the DH problem by obtaining the outputs of C for these forwarded inputs. $\square$

The proposed protocol is thus secure against passive eavesdropping.

4.2 DoS Attack

We consider CPU-exhaustion DoS attack assuming that memory resources can maintain sufficient amount of precomputed materials. Specifically, the proposed protocol is compared with a reduced version which omits the extra hash.

When an attacker A wants his target to compute $\tilde{s}_I$ in Step ((8)), A himself must compute $\tilde{s}_R$ in Step ((6)) in the proposed protocol. By contrast, in the reduced version, A can do the same attack by sending totally bogus message in Step ((7)) without computing $\tilde{s}_R$ in Step ((6)). Since u_R is not known in advance, the difference between the cost of attack in the propose protocol and in the reduced version is equivalent to 2 modular exponentiations even with the help of fast simultaneous multiple exponentiation technique which is attributed by ElGamal[8] to Shamir and well summarized in [2]. This is because u_R is not known to the initiator in advance.

5 Conclusions

In search of DoS-resistant version of authenticated key-agreement protocols with public-key infrastructure, "falling-together" strategy was proposed. This discourages DoS attackers from exhausting CPU resource of the target; CPU-exhaustion DoS attack imposes expensive computation on the attackers themselves.

The strategy was subsequently applied to authenticated Diffie-Hellman key-agreement protocol. Generation of acknowledgment material commits the attacker to perform a fairly heavy computation that can be easily checked by the responder. Thus the attacker fears his own resource exhaustion.

Currently explored issue is resistance against another type of DoS attack such as storage exhaustion by the use of half connection.

References

[1] W. Diffie and M. Hellman. "New directions in cryptography". *IEEE Trans. Information Theory*, Vol. IT-22, No. 6, pp. 644–654, 1976.

[2] A. Menezes, P. van Oorschot, and S. Vanstone. "Handbook of Applied Cryptography". CRC Press, Florida, 1996.

[3] S. Hirose and S. Yoshida. "An Authenticated Diffie-Hellman Key Agreement Protocol Secure against Active Attacks". In *Proc. of 1998 International Workshop on Practice and Theory in Public Key Cryptography (PKC'98)*, Lecture Notes in Computer Science 1431, Springer-Verlag, pp. 135–148, 1998.

[4] D. Harkins and D. Carrel. "The resolution of ISAKMP with Oakley". *Internet draft*, draft-ietf-ipsec-isakmp-oakley-**.txt

[5] U. M. Maurer. "Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms". In Y. Desmedt, editor, *Advances in Cryptology — CRYPTO'94*, Lecture Notes in Computer Science 839, Springer-Verlag, pp. 271–281, 1994.

[6] U. M. Maurer and S. Wolf. "Diffie-Hellman oracles ". In N. Koblitz, editor, *Advances in Cryptology — CRYPTO'96*, Lecture Notes in Computer Science 1109, Springer-Verlag, pp. 268–282, 1996.

[7] S. Blake-Wilson, D. Johnson, and A. Menezes. "Key agreement protocols and their security analysis". In *The Sixth IMA International Conference on Cryptography and Coding*, Cirencester, England, December 1997.

[8] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Trans. Information Theory*, Vol. IT-31, No. 4, pp. 469–472, 1985.

Step	(Initiator) Secret key: $SK_I \in_R [1, 2, \cdots, q-2]$ Public key: $PK_I = g^{SK_I} \bmod p$		(Responder) Secret key: $SK_R \in_R [1, 2, \cdots, q-2]$ Public key: $PK_R = g^{SK_R} \bmod p$
((1))	$k_I, r_I \in_R [1, 2, \cdots, q-2]$ $u_I = g^{k_I} \bmod p$ $s_I = g^{r_I} \bmod p$		$k_R, r_R \in_R [1, 2, \cdots, q-2]$ $u_R = g^{k_R} \bmod p$ $s_R = g^{r_R} \bmod p$
((2))	Get a timestamp TS_I.		
((3))	$\{u_I, ID_I, TS_I\}$	$\Rightarrow$	
((4))			Check if TS_I is acceptable. $e_R = hash\,(s_R, u_R, u_I, ID_R, ID_I, TS_I)$ $w_R = r_R / (k_R + e_R SK_R) \bmod q$
((5))		$\Leftarrow$	$\{u_R, e_R, w_R, ID_R\}$
((6))	$\tilde{s}_R = u_R^{w_R} PK_R^{e_R \cdot w_R} \bmod p$ Check $e_R = hash\,(\tilde{s}_R, u_R, u_I, ID_R, ID_I, TS_I)$? If successful, $H = hash\,(\tilde{s}_R, u_R, u_I)$ $K_{IR} = u_R^{k_I} \bmod p$ $HTS_I = hash(TS_I)$ $z_I = KH_{K_{IR}}(HTS_I)$ $e_I = hash\,(s_I, u_I, u_R, ID_I, ID_R, HTS_I, z_I)$ $w_I = r_I / (k_I + e_I SK_I) \bmod q$		
((7))	$\{H, e_I, w_I\}$	$\Rightarrow$	
((8))			Check $H = hash\,(s_R, u_R, u_I)$? If successful, $\tilde{s}_I = u_I^{w_I} PK_I^{e_I \cdot w_I} \bmod p$ $K_{RI} = u_I^{k_R} \bmod p$ $HTS_R = hash(TS_I)$. $z_R = KH_{K_{RI}}(HTS_R)$ Check $e_I = hash\,(\tilde{s}_I, u_I, u_R, ID_I, ID_R, HTS_R, z_R)$?
((9))	Accept K_{IR} as the session key.		Accept K_{RI} as the session key.

Figure 2: Proposed version of authenticated Diffie-Hellman key-agreement protocol. ID_I and ID_R are the IDs of the initiator and of the responder, respectively. TS_I is a timestamp on the initiator's request message. $\tilde{s}_I$ and $\tilde{s}_R$ indicate recovered values of s_I and of s_R, respectively. $hash(\cdot)$ represents a one-way hash function, and $KH_k(\cdot)$ represents a keyed hash function under a key k. H plays a role of "acknowledgment material" in our falling-together trick.