Modification of Internet Key Exchange Resistant against Denial-of-Service

Kanta Matsuura and Hideki Imai

Institute of Industrial Science, University of Tokyo,
Roppongi 7-22-1, Minato-ku, Tokyo 106-8558, JAPAN.
{kanta, imai}@iis.u-tokyo.ac.jp

Abstract:

The first phase of Internet Key Exchange (IKE) is an authenticated version of Diffie-Hellman (DH) key-agreement. Since the authentication is computationally expensive, computational burden caused by malicious requests may exhaust the CPU resource of the target. Attackers can also abuse inappropriate use of Cookies and exhaust the memory resource of the target.

In search of resistance against these Denial-of-Service (DoS) attacks, this paper modifies three-pass IKE Phase 1. The DoS-resistance is evaluated in terms of the computational cost and the memory cost caused by bogus requests.

Keywords:

Internet Key Exchange, Denial-of-Service, security, Cookie

1 Introduction

The Internet Key Exchange (IKE) [1] is based on a Diffie-Hellman key-agreement protocol [2]. In order to protect the protocol from man-in-the-middle attack[3], the first phase of IKE is authenticated with public-key primitives. Since the public-key primitives are computationally expensive, malicious entities can initiate a Denial-of-Service (DoS) attack; they may launch quite a large number of bogus requests to exhaust the computational resource of the target. The current IKE is vulnerable to memory exhaustion as well. This is because an anti-clogging token Cookie in IKE fails to meet the explicit requirements for DoS-protection [5]; the responder creates states after receiving the first message. Thus the memory resource of the target can be exhausted by a large number of malicious requests.

The purpose of this paper is to solve these DoS problems. Specifically, Section 2 introduces our basic strategies. Then, after reviewing conventional versions in Section 3, we propose a DoS-resistant version in Section 4. Security consideration of the proposed version is subsequently given in Section 5. Finally, Section 6 gives concluding remarks.

2 Basic Strategies

2.1 Against CPU Exhaustion

Three-pass key-agreement between an initiator I and a responder R is typically structured as follows.

((1)) Precomputation

((2)) Computation by I with PK_R and/or SK_I

((3)) Request Message from I to R

((4)) Computation by R with PK_I and/or SK_R

((5)) Reply Message from R to I

((6)) Computation by I with PK_R and/or SK_I

((7)) Acknowledgment Message from I to R

((8)) Computation by R with PK_I and/or SK_R

((9)) Key establishment

where public and secret keys of an entity X are represented by PK_X and SK_X, respectively.

Then the following "falling-together" strategy proposed in [6], [7] discourages DoS attackers; the attacker must pay computational cost comparable to that of the target.

1. Do not use heavy computation in Step ((4)).
2. Send a random fresh material by Reply Message ((5)) implicitly in a way that the reconstruction of the material requires heavy computation in Step ((6)). We will refer to this material as "RF material" in the following.

3. Use the RF material for authentication.

4. Carry an acknowledgment material derived from the reconstructed RF material by Acknowledgment Message ((7)).

5. Verify the acknowledgment material at the beginning of Step ((8)). This verification must be computationally less expensive than the reconstruction of RF material in Step ((6)).

2.2 Against Memory Exhaustion

We use two strategies against memory exhaustion:

(Stateless connection)
State materials are encrypted with a local secret key and sent back and forth between R and I [8].

(Appropriate use of Cookies)
R creates no additional state at Step ((4)), and the Cookie of R is not cached per initiator.

3 Aggressive Mode of IKE

3.1 Conventional Versions

Aggressive Mode has minimal number of message-passes in Phase 1 of IKE. Figure 1 describes two types of authentication in Aggressive Mode. We will refer to them as SIG-authentication and PKE-authentication, respectively.

Cookies CKY_I and CKY_R are set up in the header HDR. The date and time are added to the information hashed according to ISAKMP[4]. SA is an SA (Security Association) payload. SA is a set of policy and keys. The ISAKMP SA is the shared policy and keys used by the negotiating peers to protect their communication; the initiator I may provide multiple proposals regarding SA while the responder R must reply with only one. SAb is the entire body of the SA payload (minus generic header). KE is a key-exchange payload which carries keying materials such as DH public values $g^x \bmod p$ (generated by I) and $g^y \bmod p$ (generated by R) where p is a large prime and g is a primitive root modulo p. In the following, "mod p" will not always appear explicitly and a simplified representation like g^x will be used. N_I is the nonce payload of I while N_R is that of R. Likewise, ID_I and ID_R are their identity payloads, and SIG_I and SIG_R are their digitally-signed hash payloads.

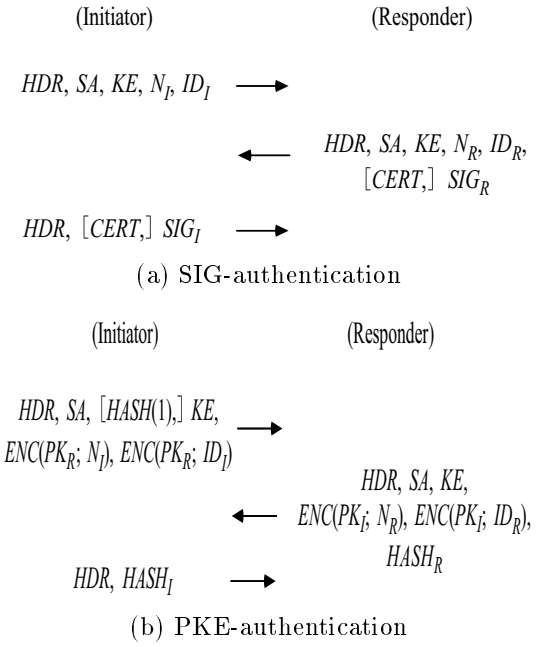

Figure 1: IKE's Phase 1 authenticated by the use of public-key primitives (Aggressive Mode). $ENC(PK_X;\ )$ indicates an encryption with the public key PK_X of an entity X.

The hash payloads are computed as

$$HASH_I = prf(SKEYID, g^x \| g^y \| CKY_I \| CKY_R \| SAb \| ID_I) \quad (1)$$

$$HASH_R = prf(SKEYID, g^x \| g^y \| CKY_R \| CKY_I \| SAb \| ID_R) \quad (2)$$

where $SKEYID$ is the output of a pseudo-random function; for PKE-authentication,

$$SKEYID = prf\,(hash\,(N_I \| N_R)\,, CKY_I \| CKY_R) \quad (3)$$

while

$$SKEYID = prf\,(N_I \| N_R, g^{xy}) \quad (4)$$

for SIG-authentication. $\|$ represents concatenation. $CERT$ is a certificate payload and the brackets $[*]$ indicate that the content of it $(*)$ is optional. $HASH(1)$ is a hash of the certificate which the initiator uses in encryption. The resultant keying materials are pseudo-randomly computed by using $SKEYID$ as:

$$SKEYID_d = prf(SKEYID, g^{xy} \| CKY_I \| CKY_R \| 0) \quad (5)$$

$$SKEYID_a = prf(SKEYID, SKEYID_d \| g^{xy} \| CKY_I \| CKY_R \| 1) \quad (6)$$

$$SKEYID_e = prf(SKEYID, SKEYID_a \| g^{xy} \| CKY_I \| CKY_R \| 2) \quad (7)$$

$SKEYID_e$ is used by the ISAKMP SA to protect its messages. $SKEYID_a$ is used by the ISAKMP SA to authenticate its messages. $SKEYID_d$ is used to derive keys for non-ISAKMP SAs.

PKE-authentication requires two public-key encryption and decryption operations of both the initiator and the responder. A modification of this PKE-authentication[1] still requires one public-key encryption and decryption operations.

3.2 CPU-Exhaustion DoS Attack

In SIG-authentication, R generates a digital signature *before* identifying I. When the signature generation is computationally expensive, the cost causes CPU-exhaustion. For example, RSA public-key primitives are recommended to be supported in IKE, and generation of RSA signatures costs much due to the deployment of a relatively larger exponent. By contrast, when the implementation deploys a signature scheme whose generation is relatively inexpensive but verification is expensive, the verification cost exhausts the CPU. This is because R must verify the signature even for a fake acknowledgment message.

In PKE-authentication, R decrypts two public-key encrypted payloads *before* identifying I. Unfortunately, this decryption is also computationally expensive. Even in a revised PKE-authentication by using ephemeral keys [1], one heavy decryption is required before identification. In particular, an attacker can send a large number of fake requests, which work directly as DoS.

Our previous proposal in [9] improved the resistance against the CPU-exhaustion. The blocking-probability analysis in [9] suggested that the expected availability of honest responders might be acceptable, But it did not improve the resistance against memory exhaustion.

3.3 Memory-Exhaustion DoS Attack

As time-variant materials in Cookie generation, IKE uses a date and time. Although the exact size depends on implementation, this requires state in R's memory and leaves a "Cookie crumb" for every connection attempt. In addition to the Cookie-related state, R must keep DH-related state; since the reply message includes KE, R must cache the secret value y which is used to generate his/her DH public value g^y. Thus an attacker can exhaust the memory of the target by a huge amount of cache. This can occur both in SIG-authentication and in PKE-authentication.

4 DoS-Resistant Resolution

4.1 Protocols

We consider how to enhance the resistance of SIG-authentication against both of the DoS attacks. Specifically, we propose a modified hash payload in the acknowledgment message. This plays a role of the acknowledgment material; a reconstructed RF material is used in the computation of the modified hash. The RF material is originally included in the reply implicitly. Figure 2 illustrates the proposed protocol (revised SIG-authentication).

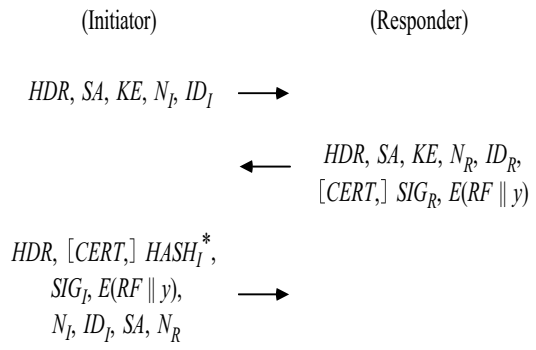

Figure 2: New Aggressive Mode resistant against DoS attacks (revised SIG-authentication) which can be implemented based on, for example, DSA or Schnorr's signature. $HASH_I^*$ is a modified hash payload signed by the initiator. $E()$ is a symmetric-key encryption function with a local secret key known only to the responder. RF represents a random fresh material R_r.

The request from I is the same as that in SIG-authentication. In the reply from R, HDR includes CKY_R in which R puts a hash of all the information which would otherwise create states: a private secret, SA, ID_I, N_I, N_R, CKY_I, y, and an RF material R_r. R_r depends on the signature scheme. Among them, connection-dependent secrets R_r and y are encrypted with a local secret key known only to R. This key can be used for different requests (*i.e.*, this symmetric encryption does

not create a state per initiator). The encrypted result is attached to the end of the reply. The current date and time are not included in CKY_R. The rest of the reply is the same as that in SIG-authentication. R must use a signature scheme with the following properties:

- Expensive computation in signature generation can be completed in advance independent of the initiator, *i.e.*, as a precomputation before receiving the request.
- The verification procedure includes reconstruction of an RF material R_r.

In the computation of digitally-signed hash payloads, *SKEYID* is replaced with a one-way hashed value $SKEYID' = hash(N_I \| N_R)$ which is random, fresh, but publicly-known. This does not change the security of the signature algorithm itself.

$SKEYID_e$, $SKEYID_a$, and $SKEYID_d$ are derived from the same *SKEYID* as in the conventional SIG-authentication. In the computation of the initiator's digitally-signed hash payload, the hash payload is replaced with a modified hash payload defined as

$$HASH_I^* = prf(SKEYID', g^x \| g^y \| CKY_I \| CKY_R \| R_r \| SAb \| ID_I).$$

The acknowledgment explicitly includes $HASH_I^*$, ID_I, N_I, and N_R. They are copies of what was exchanged in the first two messages. On receiving the acknowledgment message, R first decrypts $E(R_r \| y)$ and checks the hash in CKY_R. If successful, R verifies the modified hash to check whether it is really generated from the correct RF material. Then, if successful, R goes on to the signature verification. The signature scheme for SIG_I does not necessarily the same as that for SIG_R.

Our previous proposal in [9] is shown in Fig. 3 for reference. The definition of the modified hash is the same as that in the version proposed in this paper.

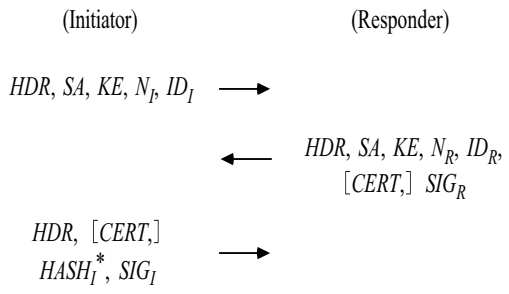

Figure 3: Previous version of modified Aggressive Mode resistant against CPU-exhaustion DoS attacks proposed in [9]. $HASH_I^*$ followed by SIG_I is a digitally-signed modified hash payload.

4.2 Example Algorithms

In the following, we show two specific examples of revised SIG-authentication. The responder's secret key $SK_R (\in_R [1, 2, \cdots, q-2])$ for SIG_R is related with his/her public key PK_R by

$$PK_R = \alpha^{SK_R} \bmod p \tag{8}$$

where p is a large prime and q is a large prime factor of $p-1$. α is a public integer with order q modulo p. $\in_R$ represents random picking of an element from the right-hand side set.

The first example is based on a shortened Digital Signature Standard (SDSS) [10]. As well as the original DSA (Digital Signature Algorithm) [11] or DSS (Digital Signature Standard) [12], the shortened DSS is unforgeable by adaptive attackers under the assumptions that discrete logarithm is hard and that the one-way hash function behaves like a random function [10], [13].

Precomputation by the responder:

$x_r \in_R [1, 2, \cdots, q-2]$, $R_r = \alpha^{x_r} \bmod p$.

Generation of the responder's signature:

$T_R = hash\,(R_r, HASH_R)$

$SIG_R = (s_1, s_2)$

$= (x_r / (T_R + SK_R) \bmod q,\ T_R)$

Verification of the responder's signature:

$\hat{R}_r = (\alpha^{s_2} \cdot PK_R)^{s_1} \bmod p$.

The initiator accepts the signature if and only if s_2 is equal to $hash\left(\hat{R}_r, HASH_R\right)$.

Computation of the modified hash:

$SKEYID' = hash(N_I \| N_R)$

$HASH_I^* = prf(SKEYID', g^x \| g^y \| CKY_I \| CKY_R \| \hat{R}_r \| SAb \| ID_I)$

Verification of the modified hash:

The responder accepts the modified hash if and only if $HASH_I^*$ is equal to $prf(SKEYID', g^x \| g^y \| CKY_I \| CKY_R \| R_r \| SAb \| ID_I)$.

The second example is based on Schnorr's signature scheme [14].

Precomputation by the responder:

$x_r \in_R [1, 2, \cdots, q-2]$, $R_r = \alpha^{x_r} \bmod p$

Generation of the responder's signature:

$T_R = hash(HASH_R \| R_r)$
$SIG_R = (s_1, s_2)$
$= (SK_R \cdot T_R + x_r \bmod q,\ T_R)$

Verification of the responder's signature:

$\hat{R}_r = \alpha^{s_1} PK_R^{-s_2} \bmod p$.
The initiator accepts the signature if and only if s_2 is equal to $hash(HASH_R \| \hat{R}_r)$.

Computation of the modified hash:

$SKEYID' = hash(N_I \| N_R)$
$HASH_I^* = prf(SKEYID',$
$g^x \| g^y \| CKY_I \| CKY_R \| \hat{R}_r \| SAb \| ID_I)$

Verification of the modified hash:

The responder accepts the modified hash if and only if $HASH_I^*$ is equal to $prf(SKEYID',$ $g^x \| g^y \| CKY_I \| CKY_R \| R_r \| SAb \| ID_I)$.

5 Discussion on Security

5.1 Assumption

In the example algorithms for the proposed protocol, security of secret keys relies on the hardness of discrete-logarithm (DL) problem. Let us consider whether this requires an additional assumption or not.

IKE in itself is not devoted to any specific encryption/signature schemes. However, for keying, it is restricted to the use of DH public values. This means that the difficulty of DL problem is assumed in the security consideration of IKE since the DH problem is at most as hard as the DL problem [15]. Hence we can say that the proposed version (revised SIG-authentication) does not require any additional assumption regarding the security in that sense.

5.2 CPU Exhaustion

The computational cost is measured by the number of modular multiplications required for exponentiation. We classify attackers into two types:

(Type 1)

An attacker who launches fake requests with valid Cookies but does not pay heavy computational cost.

(Type 2)

An attacker who pays computational cost necessary for imposing modular exponentiation on the responder. In the proposed version (revised SIG-authentication), for example, the attacker must really verify the responder's signature in order to construct a correct modified hash.

Regarding the cost of RSA, we ignore the cost of encryption or signature verification as in [10]. This assumption allows better performance in the conventional schemes. Regarding the cost of modular exponentiation in RF-material generation, it is assumed that precomputation is available both for the attacker and for the responder. The computational cost is determined by the number of nontrivial modular multiplications and each multiplication costs in proportion to the size of the modulus (in bits).

In the case of an attacker of Type 1, the computational cost per request is summarized as follows.

Attacker's cost: 0

Responder's cost: PKE (RSA): $0.75|n| = 768$
PKE (ElGamal): $3|p| = 3072$
revised PKE (RSA): $0.375|n| = 384$
revised PKE (ElGamal): $1.5|p| = 1536$
SIG (RSA): $0.375|n| = 384$
SIG (ElGamal): $4.5|p| = 4608$
SIG (DSA): $3|q| = 480$
SIG (Schnorr): $3|q| = 480$
revised SIG (SDSS): 0
revised SIG (Schnorr): 0

where n is the RSA composite and $|n|$ indicates the size of it. Recommended sizes are, say, $|n| = |p| = 1024$ and $|q| = 160$. The numerical results are based on this assignment. In revised SIG-authentication, the responder does not pay expensive cost for the requests from attackers of Type 1.

In the case of an attacker of Type 2, the verification cost of the responder's signature is estimated by assuming the help of simultaneous multiple exponentiation (SME), which is attributed by ElGamal [16] to Shamir and well summarized in [3]. This assumption reduces the cost on the attacker's side. SME requires precomputation $\alpha PK_R \bmod p$. The attacker does not have to warry about memory cost for caching this precomputed value since it can be iteratively used for a lot of bogus requests to the same R. By contrast, the attacker's signature is assumed to be verified without the help of SME. This is because the responder does not want to pay precomputation cost for SME. It should be noted that the responder does not know whether the attacker is of Type 1 or of Type 2 in advance.

The computational cost per request is then summarized as follows.

Attacker's cost: PKE (RSA): 0
PKE (ElGamal): 0
revised PKE (RSA): 0
revised PKE (ElGamal): 0
SIG (RSA): 0
SIG (ElGamal): 0
SIG (DSA): 0
SIG (Schnorr): 0
revised SIG (SDSS): $1.75|q| = 280$
revised SIG (Schnorr): $1.75|q| = 280$

Responder's cost: PKE (RSA): $0.75|n| = 768$
PKE (ElGamal): $3|p| = 3072$
revised PKE (RSA): $0.375|n| = 384$
revised PKE (ElGamal): $1.5|p| = 1536$
SIG (RSA): $0.375|n| = 384$
SIG (ElGamal): $4.5|p| = 4608$
SIG (DSA): $3|q| = 480$
SIG (Schnorr): $3|q| = 480$
revised SIG (SDSS): $3|q| = 480$
revised SIG (Schnorr): $3|q| = 480$

The attacker of Type 2 does not have to pay computational cost in the conventional authentications, while revised SIG-authentication imposes computational cost which is $1.75|q|/3|q| = 58.3\%$ of that on the responder's side. This ratio is referred to as "Cost Ratio" in the following. If the attacker does not use SME due to the limitation of his/her memory resource, Cost Ratio becomes 100%. In either case, the attacker must exhaust the same order of computational resource if he/she wants to exhaust the responder's resource.

Table 1 summarizes the comparison of the resistance against CPU-exhaustion. When both costs are zero, Cost Ratio is NaN (Not a Number) which is considered to be better than finite values. When Cost Ratio is zero, the tie break is solved in a way that smaller responder's cost is better. In conclusion, the revised SIG-authentication proposed in this paper is the most resistant.

5.3 Memory Exhaustion

In the case of an attacker of Type 1, the attacker's memory cost is at most as large as Cookie state. By contrast, the responder's memory cost depends highly on the authentication method; at the worst case, the cost could be six or more times as large as Cookie state while the best case provides stateless feature. Specifically, the memory cost per request is summarized as follows.

Table 1: Comparison of Cost Ratio (=attacker's cost/responder's cost) in computation. In the current IKE, the best algorithm is chosen.

(a) Attacker of Type 1

Version (Algorithm)	Cost Ratio
Current IKE (RSA)	0
Previous proposal in [9]	NaN
Proposal in this paper	NaN

(b) Attacker of Type 2

Version (Algorithm)	Cost Ratio
Current IKE (RSA)	0
Previous proposal in [9]	58.3%
Proposal in this paper	58.3%

Attacker's cost: PKE (RSA): 0
PKE (ElGamal): 0
revised PKE (RSA): 0
revised PKE (ElGamal): 0
SIG (RSA): $|C|$=32
SIG (ElGamal): $|C|$=32
SIG (DSA): $|C|$=32
SIG (Schnorr): $|C|$=32
revised SIG (SDSS): $|C|$=32
revised SIG (Schnorr): $|C|$=32

Responder's cost: PKE (RSA): $|C| + |q|$=192
PKE (ElGamal): $|C| + |q|$=192
revised PKE (RSA): $|C| + |q|$=192
revised PKE (ElGamal): $|C| + |q|$=192
SIG (RSA): $|C| + |q|$=192
SIG (ElGamal): $|C| + |q|$=192
SIG (DSA): $|C| + |q|$=192
SIG (Schnorr): $|C| + |q|$=192
revised SIG (SDSS): 0
revised SIG (Schnorr): 0

where $|C|$ represents the size of a Cookie-related state which depends on the implementation but is expected to be 32 or 64. The numerical results listed above use the smaller one (32bit) for the attacker's convenience, and assume the same security level as in 5.2.

In the case of an attacker of Type 2, the memory cost per request is summarized in the same way as Type 1. This is due to the assumption on the use of exponentiation algorithms. Cost difference appears in our previous proposal, which results in the Cost Ratio later in Table 2

Attacker's cost: PKE (RSA): 0
PKE (ElGamal): 0
revised PKE (RSA): 0
revised PKE (ElGamal): 0
SIG (RSA): $|C|$=32
SIG (ElGamal): $|C|$=32
SIG (DSA): $|C|$=32
SIG (Schnorr): $|C|$=32
revised SIG (SDSS): $|C|$=32
revised SIG (Schnorr): $|C|$=32
Responder's cost: PKE (RSA): $|C| + |q|$=192
PKE (ElGamal): $|C| + |q|$=192
revised PKE (RSA): $|C| + |q|$=192
revised PKE (ElGamal): $|C| + |q|$=192
SIG (RSA): $|C| + |q|$=192
SIG (ElGamal): $|C| + |q|$=192
SIG (DSA): $|C| + |q|$=192
SIG (Schnorr): $|C| + |q|$=192
revised SIG (SDSS): 0
revised SIG (Schnorr): 0

where we assume that the responder's signature is verified with a help of SME and the memory cost (on the attacker's side) for SME is negligible. If the attacker does not use SME, the cost for revised SIG-authentication (either SDSS or Schnorr) can be reduced to $|CKY|$. But this increases his/her computational cost as mentioned in the previous subsection, and anyway the responder is stateless.

Table 2 summarizes the comparison of the resistance against memory-exhaustion. In the same way as in the evaluation of CPU-exhaustion, we define "Cost Ratio" as the ratio of the cost on the attacker's side to that on the responder's side. Therefore, larger Cost Ratio is better. We use the evaluation result in [9] for that version. In conclusion, the revised SIG-authentication proposed in this paper is the most resistant.

Table 2: Comparison of Cost Ratio (=attacker's cost/responder's cost) in memory. In the current IKE, the best authentication method is chosen.

(a) Attacker of Type 1

Version (Authentication)	Cost Ratio
Current IKE (SIG)	16.7%
Previous proposal in [9]	100%
Proposal in this paper	∞

(b) Attacker of Type 2

Version (Authentication)	Cost Ratio
Current IKE (SIG)	16.7%
Previous proposal in [9]	6500%
Proposal in this paper	∞

6 Conclusions

We constructed a DoS-resistant resolution of Aggressive Mode in IKE's Phase 1 by using three strategies: (1) "falling-together" mechanism, (2) appropriate use of Cookies, and (3) stateless connections. As a result, attackers fear their own resource exhaustion both in CPU and in memory while honest responders can be stateless before starting expensive computation. Thus we obtain a deterrent to DoS attacks; if an adversary wants to exhaust the resource of a target, he/she must spend CPU resource comparable to that of the target and memory resource much larger than that of the target. This is better than our previous version in [9] where attackers fear their CPU exhaustion but honest responders also fear their memory exhaustion and thus a system analysis is required for estimating blocking probability of the responders.

One argument would be the second strategy is sufficient; since Cookies can restrict an attacker's address to a connection-dependent IP-reachable address, the attacker would fear of damaging his/her reputation. This does not work well, however, if the attacker uses stepping-stones in-between. Unfortunately, a lot of actual attacks occur often via stepping-stones. The new Aggressive Mode proposed in this paper solves this stepping-stone problem as well; once the stepping-stone is exhausted, no more malicious requests from it can attack the target. This does not directly damage the attacker but the target is protected.

Acknowledgments:
This work is partly supported by the 16th Research Promotion Fund (No. 28) from Casio Science Promotion Foundation.

References

[1] D. Harkins and D. Carrel: "The Internet Key Exchange (IKE)", rfc2409, November 1998.

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

[3] A. Menezes, P. van Oorschot, and S. Vanstone: *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, Florida, 1996.

[4] D. Maughan, M. Schertler, M. Schneider, and J. Turner: "Internet Security Association and Key Management Protocol (ISAKMP)", rfc2408, November 1998.

[5] P. Karn and W. Simpson: "Photuris: Session-Key Management Protocol", rfc2522, March 1999.

[6] K. Matsuura and H. Imai: "Protection of authenticated key-agreement protocol against a denial-of-service attack", Proc. 1998 International Symposium on Information Theory and Its Applications (ISITA'98), Mexico City, Mexico, pp. 466–470, October 1998.

[7] K. Matsuura and H. Imai: "Protection of Authenticated Key-Agreement Protocol against a Denial-of-Service Attack", Científica, Vol. 2, No. 11, pp. 15–19, September 1998.

[8] T. Aura and P. Nikander: "Stateless connections", in Information and Communications Security, ed. Y. Han, T. Okamoto, and S. Qing, pp. 87–97, Springer-Verlag, LNCS 1334, Berlin, November 1997.

[9] K. Matsuura and H. Imai: "Resolution of ISAKMP/Oakley key-agreement protocol resistant against Denial-of-Service attack", Pre-Proc. Internet Workshop'99 (IWS'99), Osaka, Japan, pp. 17–24, February 1999.

[10] Y. Zheng: "Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption)", in Advances in Cryptology — Crypto'97, pp. 165–179, Springer-Verlag, LNCS 1294, Berlin, August 1997.

[11] D. W. Kravitz: Digital signature algorithm, U. S. Patent # 5,231,668, July 1993.

[12] FIPS 186: Digital signature standard, Federal Information Processing Standards Publication FIPS PUB 186, U. S. Department of Commerce/N.I.S.T., National Technical Information Service, 1994

[13] D. Pointcheval and J. Stern: "Security proofs for signature schemes", in Advances in Cryptology — EUROCRYPT'96, ed. U. Maurer, pp. 387–398, Springer-Verlag, LNCS 1070, Berlin, 1996.

[14] C. P. Schnorr: "Efficient signature generation by smart cards", Journal of Cryptology, Vol. 4, pp. 161–174, 1991.

[15] U. M. Maurer and S. Wolf: "Diffie-Hellman oracles", in Advances in Cryptology — CRYPTO'96, ed. N. Koblitz, pp. 268–282, Springer-Verlag, LNCS 1109, Berlin, August 1996.

[16] 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.