共通鍵暗号に基づいたサーバ・クライアントプロトコルにおける会員資格の一時貸与方式

松浦幹太
東京大学生産技術研究所

インターネットのようにオープンなネットワークでは、いかにして不正行為を抑止するかが重要な課題である。一般的な対策は、暗号・認証技術の利用である。この場合、プロトコル実行ログがトラブル解決のための証拠となり、不正行為の抑止につながる。しかし、社会的要請次第では、より能動的な抑止策が求められる可能性がある。そのような場合の解として、共通鍵暗号に基づいたサーバ・クライアントプロトコルにおける抜き打ち検査方式を提案する。同方式では、会員権をオンラインで一時的に監査官に貸与することができる。応用例として、電子貸金庫サービスと、インターネットセキュリティプロトコルにおける鍵配送について述べる。

One-Time Rental of Membership for Server-and Client Protocol with Private-Key Cipher

Kanta Matsuura
Institute of Industrial Science, the University of Tokyo, Tokyo 106-8558

In an open network, how to provide deterrents to malicious behaviors is an important issue. A common solution is given by cryptographic primitives; execution logs are stored and used when needs arise for trouble settlement. Depending on social situation, however, more active authorized procedures would be of great help. In preparation for such a situation, this paper introduces a framework of an inspection mechanism for server-and-client protocols based on a private-key cipher. The mechanism allows one-session rental of the membership. Example protocols for Electronic Safe-Deposit Box and IPsec key exchange are described.

One-Time Rental of Membership for Server-and Client Protocol with Private-Key Cipher

KANTA MATSUURA†

In an open network such as the Internet, how to provide deterrents to malicious behaviors is an important issue. A common solution is the use of cryptographic primitives. In this solution, execution logs are stored by each entity and used when needs arise for trouble settlement; authorities are involved mainly in the settlement phase. Depending on social situation, however, more active authorized procedures would be of great help. In preparation for such a situation, this paper introduces a framework of an inspection mechanism for server-and-client protocols which are based on a private-key cipher. The mechanism uses two "current" private keys per server-and-client pair. One of them is updated session by session so that the system can accept one-session rental of the membership; voluntary clients can lend their membership to the inspection authority without disclosing the current keys of the next and future sessions. Not devoted to a specific encryption algorithm, two examples are described: a digital-valuable storage system called "electronic safe-deposit box" and an Internet key-exchange protocol authenticated with pre-shared keys.

1. Introduction

In order to make a wide use of an open network such as the Internet, it is very important to have deterrents to malicious behaviors. A common technical strategy is the use of cryptographic primitives such as encryption/decryption, digital signature, and digital signcryption. Encryption schemes provide confidentiality while digital-signature schemes provide authenticity$^{1)}$. A relatively new primitive is digital signcryption, which fulfills both the functions of public-key encryption and digital signature, *i.e.*, confidentiality of message contents, unforgeability, and non-repudiation$^{2)}$. In a real application of these primitives, execution logs and related information are stored by each entity and retrieved when needs arise for trouble settlement or judgment; authorities are involved mainly in the settlement. Depending on social situation or inspection policies, however, more active authorized procedures would be of great help.

In our society, authorized activities by police or similar organizations prevent or discourage people from committing a crime; potential criminals fear the consequences of their crimes which might be detected by the police. Inspection without notice, for instance, contributes to these activities as a deterrent to illegal behaviors. In addition, inspection contributes to protection of consumers from inferior goods and defective articles. We can find similar scenarios when considering how to keep miscellaneous social or organizational conditions safe and sound. The on-line inspection mechanism described in Section 2 is based on an idea similar to this inspection effect in the off-line world. Subsequently, Sections 3 and 4 show application examples of this mechanism: (1) "electronic safe-deposit box" for digital valuables in a future network life and (2) an authenticated key-exchange protocol in IPsec. Finally, Section 5 gives concluding remarks.

2. Inspection Mechanism

In preparation for a situation where on-line active authorized procedures are required, this section introduces an inspection mechanism for a class of security protocols: server-and-client protocols based on a private-key cipher.

2.1 Protocol Requirements

Let us consider a security protocol which uses a private-key cipher in important message exchange between a server (say, Alice) and a client (say, Bob). In order to employ our inspection mechanism,

(1) the protocol MUST provide two private keys per server-and-client pair at the same time, and the two keys SHOULD be ordered or have identifiers,

(2) the first message in the protocol MUST be generated by Bob and sent from Bob

† Institute of Industrial Science, the University of Tokyo, Tokyo 106-8558

to Alice,

(3) each session MUST use not both but only one of the two private keys as a session key, and

(4) the session key MUST be arbitrarily chosen by Bob.

In the following, we refer to the two shared keys as the "current keys" and assume that the server-and-client protocol is designed in a way that all the four requirements are satisfied.

In addition to these requirements, we have several recommendations:

(1) The server-and-client protocol SHOULD be successfully synchronized. This paper does not care about how to make the protocol execution robust against synchronization failure.

(2) The server-and-client protocol SHOULD be divided into authentication phase, message-exchange phase, and acknowledgment phase. This is for easier design and update of the protocol.

(3) A reliable time-stamping service SHOULD be available in the authentication phase and the acknowledgment phase of the protocol. This makes citizens less reluctant to accept the authority's request for cooperation.

(4) Although beyond the server-and-client protocol itself, it is REQUIRED that there exists a secure communication channel between an authorized entity (say, Pole) and Bob. This secure channel is most probably established by very costly but highly secure protocols.

2.2 Normal Procedure

Conforming to the requirements mentioned in the previous subsection, the client Bob arbitrarily chooses one of the current keys and initiates a session by using the chosen session key in normal communication. Let us represent the session key by k_1 and the other current key by k_2. We can identify k_1 or k_2 by using its allocation number.

Main process of the session, then, uses k_1 for authentication, message delivery/exchange, and acknowledgment. In the authentication, the allocation number or key identifier would contribute to efficiency, depending on the original protocol.

Finally, the server Alice generates a new key k_3, encrypts it with the non-session key k_2, and sends the result to Bob. Alice's digital signature is attached to it, if necessary. On receiving this, Bob decrypts the new key, and the shared current keys are updated from (k_1, k_2) to (k_3, k_2); the next session will be initiated by using k_3 or k_2. The used key, k_1, will never be used by these two entities afterwards. The order of the current keys can be changed, depending on the protocol implementation.

2.3 Inspection Procedure

This subsection introduces an inspection procedure where a trusted entity Pole makes a performance check on Alice, the server. The inspection procedure starts with a request message from Pole to Bob through a secure channel. If Bob accepts the request, he arbitrarily chooses one of the current keys and securely transfers it to Pole. The order or allocation information of the current keys is attached, if necessary. Let us represent the chosen key by k_1 and the other key by k_2.

Pole then masquerades as Bob; Pole starts a session by using k_1. Main process of the session delivers messages which are carefully designed for inspection activities by Pole.

At the end of the session, Alice generates a new key k_3, encrypts it with k_2, and sends the result to Pole. Finally, Pole forwards the encrypted new key to Bob, and Bob decrypts it. Thus the shared current keys are updated from (k_1, k_2) to (k_3, k_2). The next session will be initiated by using k_3 or k_2. Neither k_3 nor k_2 is disclosed to Pole. The order of the current keys can be changed, depending on the implementation.

3. Upper-Layer Example: Electronic Safe-Deposit Box

More and more common use of information-security systems may probably request us to keep larger number of different cryptographic keys in a secure manner. In addition, once encouraged by security technologies to live with an active use of digitized applications, we will have a larger and larger number of important digitized materials; not only private keys for encryption/authentication, but also digital personal belonging such as artistic or multi-media contents, family treasures, memorials, and digital certificates. They require long-term secure storage in an electronic form. In addition, execution logs themselves could also be important objects to be securely stored for future trouble settlement.

In preparation for such a situation, Matsuura *et al.*[3] discussed the importance of a deposit system for digitized valuables and introduced a concept of "electronic safe-deposit box (ESDB)". ESDB is a good application example of the proposed rental/inspection mechanism since the framework of ESDB mostly conforms to the protocol requirements described in 2.1. After an introduction of the backgrounds, this section shows an ESDB structure and describes two of the phases in the structure.

3.1 Backgrounds

Aiming at global use of digital communication and its applications, various information-security mechanisms are actively explored. The more such mechanisms get available, the more secret keys of different systems have to be stored by users in a secure manner. This might probably make individuals and organizations fear what happens if they lose their cryptographic keys. As a solution for this problem, Key Recovery Systems (KRSs) are currently of great interest and discussed from various viewpoints$^{4)-8)}$. However, our future network life would require more for storage, beyond the scope of KRSs.

Expected scenarios of the requirements can be described as follows. We may want to enjoy on-line shopping in a secure way. After choosing a security system, we register ourselves to the system and thus get very important digital information such as cryptographic keys, identities, certificates, and other related parameters. We may want to live an individual network life. Each family member might probably use different security parameters, or even different security systems. We may want to enjoy multimedia applications. When we obtain very favorite contents strictly private, we want to keep them in a secure way. We may want to make a contract with insurance companies in an electronic form for our conveniences. The electronic items related with the insurance contract have to be protected. Our storage devices would be flooded with digital valuables.

Thinking of the conventional life, a safe-deposit box, usually installed in a special room in a bank, has been one of the tools for secure storage of our valuables with the highest care. This system works in a way that valuables themselves are deposited and protected mainly by physical security mechanisms such as physical access-control. Defined as an information-security tool analogous to it, ESDB provides the following services:

Service 1: Long-term storage is available for any digitized electronic information.

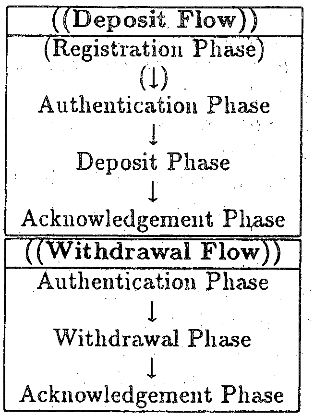

Fig. 1 Flows of an on-line ESDB. User registration should be securely completed in advance.

Service 2: Only the specified users can get access to and reconstruct the stored information of their own.

Service 3: The users can verify the stored information when reconstructing it.

Service 4: Additional secret materials imposes less burden or stress on the users.

Backup systems in general might probably generate additional secret materials to be kept by users. This would change user burden; it might be increased or decreased, depending on the system. Service 4 categorizes ESDB as a system which alleviates the burden.

3.2 Flow of On-Line ESDB

The ESDB is constructed in conformance with the requirements and recommendations described in 2.1. The message-exchange phase is Deposit Phase or Withdrawal Phase, as described in Fig. 1; they are the main routine of Deposit Flow or Withdrawal Flow, respectively. Before executing these flows, the customer or client Bob registers himself to the system; specifically, the bank or server Alice gives Bob two ordered secret keys of a private-key cipher. If necessary, they also negotiate the cryptographic functions to be used later. This registration is RECOMMENDED to be off-line or at least include an off-line procedure combined with physical human identification.

3.3 Authentication Phase

Since even execution time on the order of minutes is better than that in off-line backup systems, on-line ESDB may probably accept a relatively long duration and we can assume that a time-stamping service is available over the network. This subsection describes a protocol example of authentication phase based on time-stamps by using the following notation:

$TS(x)$: time-stamp generated by an entity x (x is either *Alice* or *Bob*).

$ID(x)$: ID of an entity x.

$E(k; M)$: message M in an encrypted form with a secret key k where a private-key cipher such as DES[9], SPEED[10], or MISTY[11] is used.

$SIG(x; M)$: message M digitally-signed with the secret key of an entity x.

$||$: concatenation.

It is assumed that a public-key infrastructure is available for digital signature.

Let us see the authentication protocol. The customer Bob first generates a random number r_1. He then initiates the authentication phase by using one of the current keys as a session key. The first message from Bob to Alice is

$$ID(Bob), ind, E(k_1; r_1||ID(Bob))$$

where k_1 represents the session key arbitrarily chosen by Bob and *ind* is the indicator which tells the recipient that k_1 is used in this session. On receiving this, Alice decrypts it with k_1 consistently derived from $ID(Bob)$ and *ind*. She then checks whether the last part of the output is identical to $ID(Bob)$ or not. If this trial is successful, she subsequently generates a random number r_2 and sends a reply message to Bob by using the session key k_1. The reply message is

$$E(k_1; r_1||r_2||TS(Alice)).$$

This reply message is decrypted by Bob. He then confirms that the random number r_1 is identical to his original one and that Alice's time-stamp $TS(Alice)$ is valid. If both successful, he accepts the reply and the random number r_2 as a valid one. In deposit/withdrawal phase, k_1 is used as the session key.

3.4 Acknowledgment Phase

After deposit or withdrawal phase, the flow goes on to acknowledgment phase which finishes with the following two messages. The first one is from Bob to Alice:

$$E(k_1; M||r_1||r_2||TS(Bob)||TS(Alice))$$

where M includes important parts of the information already exchanged in deposit/withdrawal phase. $TS(Bob)$ is generated at this acknowledgment phase. The second one is from Alice to Bob:

$$E(k_1; SIG(Alice; M||r_1||r_2||TS(Bob)|| \\ TS(Alice)||E(k_2; k_3, ind))),$$

which is generated if Alice accepts $TS(Bob)$ as

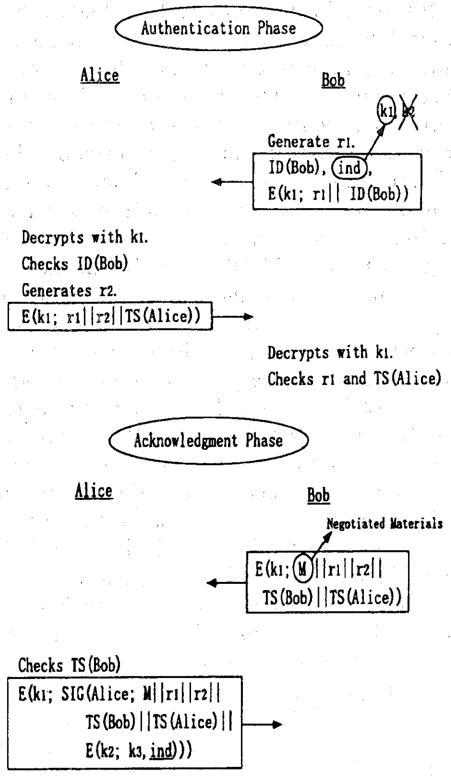

Fig. 2 The authentication phase and the acknowledgment phase of ESDB. As a result of the acknowledgment phase, current keys are updated from (k_1, k_2) to (k_2, k_3) or (k_3, k_2) where the order is indicated by *ind*.

a valid time-stamp and admits the whole session has been successfully executed. k_3 is a new key which is generated by Alice and *ind* indicates the order of the updated current keys: (k_2, k_3) or (k_3, k_2). In summary, the authentication phase and the acknowledgment phase are sketched in Fig. 2

3.5 Discussion

In order to protect users from attacks by a malicious entity, or at least as a deterrent to malicious acts, time-stamps are very useful; in security protocols such as key-distribution/agreement, for example, the protection or deterrent function is often supported by either time-stamps or fresh random numbers called nonces[12]. For this purpose, time-stamps SHOULD be somehow included in or attached to the messages at the first and the last stages of the inspection procedure, *i.e.*, when the session key is sent to the authority and when the new key encrypted with the non-session key is sent to the user. In the protocol described in the previous two subsections, both time-stamps and random numbers are available for future trouble settlement. This protection mechanism may probably encourage users to accept the cooperation requests from authorized organizations.

Time-stamps are also useful for adjustment of non-malicious inspection activities; since the sessions executed as inspection procedures are charged by the ESDB server in the same way as normal sessions, the service charge must be adjusted between the user and the authority afterwards. As far as digital signatures are concerned, Bob's signature does not appear in the ESDB protocol. This is because the ESDB is designed to employ the membership-rental/inspection mechanism where users do not have to disclose secret information which will be used in the next and future sessions. The deposit and withdrawal phases SHOULD be designed without Bob's signature but in a way that Bob could not or is sufficiently discouraged to behave maliciously. Again, one useful material is a time-stamp.

One might argue that inspection is not necessary in ESDB systems since banks in general do not want to ruin their business reputation. If we want to build a new business field which accepts new small vendors as ESDB servers, however, specific mechanisms for discouraging such servers from malicious behaviors might be needed. In addition, the quality of service should be examined when needs arise for official qualification. The proposed inspection mechanism may play an important role in the fulfillment of such requirements. Finally, one of the most important open problems concerning ESDB is to find out what really measures the user burden and how to alleviate it.

4. Lower-Layer Example: Key Exchange in IPsec

The proposed rental/inspection mechanism for private-key based systems can be easily adapted to protocols authenticated with a keyed hash or pseudo-random function. This works not only in application protocols but also in lower-layer security protocols, even in the network layer. Since private-key based protocols are pretty much faster than those with public-key primitives, we are motivated to use our mechanism at the first step of the security protocols in IP.

Aimed for confident use of the Internet, widely-available security mechanisms are actively explored toward Internet Protocol version 6 (IPv6)[13]. Among them, dynamic key management is considered to play an important role; To ISAKMP (Internet Security Association and Key Management Protocol)[14], UDP (User Datagram Protocol) Port 500 has been assigned by the Internet Assigned Numbers Authority. Although ISAKMP does not assume a specific key exchange, the Internet Engineering Task Force (IETF) favors a resolution of ISAKMP with Oakley key-exchange[13],[15],[16].

4.1 ISAKMP/Oakley

In designing IPsec(IP security protocols) in conjunction with IPv6, several key-management mechanisms are considered. Among them, we consider ISAKMP/Oakley's Phase 1 where Main Mode or Aggressive Mode can be used; Main Mode is a six-pass key-exchange protocol while Aggressive Mode is of three-pass. Both modes allow four different authentication methods:

(1) authentication with digital signature,
(2) authentication with public key encryption,
(3) an efficient modification of (2) with the help of private-key cipher, and
(4) authentication with pre-shared key.

In 4.3, we show how to allow one-time rental of the membership in a system implemented with pre-shared key authentication.

The relationship between ISAKMP's Phase 1 and Phase 2 is very clear. In Phase 1, two servers/hosts agree on how to protect further negotiation between themselves. The established set of policies and parameters is called an ISAKMP Security Association (SA) and used as security properties in Phase 2. With the authentication of users or application level programs, Phase 2 establishes SAs for other security protocols which protect many message/data exchanges afterwards. When fast keying is necessary, a single Phase 1 negotiation can be used for more than one Phase 2 negotiation.

4.2 Pre-Shared Key Authentication

First, let us make a brief review of Phase 1 authenticated with pre-shared key. The notation used here is as follows:

HDR:	ISAKMP header
$HDR*$:	ISAKMP header followed by payloads encrypted with the newly established key
SA:	Security Association (SA) payload

KE: key-exchange payload containing Diffie-Hellman public values[17)]

N_x: nonce payload with respect to the entity x (x is either the initiator i or the responder r)

ID_x identification payload of the entity x

H_x: hash payload with respect to the entity x

CKY_x: x's cookie derived from the ISAKMP header

$\{X\}$: the body of payload X (*e.g.*, $\{SA\}$ is the entire body of the SA payload)

k: pre-shared key determined by some out-of-band mechanism

In ISAKMP/Oakley[18)], the contents of the hash payload are specific to the authentication method. In the case of authentication with pre-shared key, the hash payloads are computed as

$$H_i = prf(S, g^{v_i} \| g^{v_r} \| CKY_i \| CKY_r \| \{SA\} \| \{ID_i\}) \quad (1)$$

$$H_r = prf(S, g^{v_r} \| g^{v_i} \| CKY_r \| CKY_i \| \{SA\} \| \{ID_r\}) \quad (2)$$

where

$$S = prf(k, \{N_i\} \| \{N_r\}) \quad (3)$$

and g^{v_i}, g^{v_r} are the Diffie-Hellman public values of the initiator and responder, respectively. $prf(k, \cdot)$ is a pseudo-random function keyed with k, and $\|$ denotes concatenation.

Main Mode and Aggressive Mode are described in Figs. 3 and 4, respectively.

In Main Mode, the key can only be identified by the IP addresses of the participants since H_i must be computed before the initiator has processed ID_r. On receiving the encrypted ID and hash payloads from the initiator, the responder computes the expected value of the hash by using Eqs. (1) and (3), where

- the Diffie-Hellman public values are derived from the key-exchange payloads,
- the cookies are derived from the ISAKMP header,
- $\{SA\}$ is derived from the SA payload in accordance with the responder's selection in the second message,
- the nonces are derived from the nonce payloads, and finally
- the initiator's ID is derived from the encrypted message just received.

((Initiator))		((Responder))
HDR, SA	$\rightarrow$	
	$\leftarrow$	HDR, SA
HDR, KE, N_i	$\rightarrow$	
	$\leftarrow$	HDR, KE, N_r
HDR^*, ID_i, H_i	$\rightarrow$	
	$\leftarrow$	HDR^*, ID_r, H_r

Fig. 3 ISAKMP/Oakley's Phase 1 authenticated with pre-shared key (Main Mode).

((Initiator))		((Responder))
HDR, SA, KE, N_i, ID_i	$\rightarrow$	
	$\leftarrow$	HDR, SA, KE, N_r, ID_r, H_r
HDR, H_i	$\rightarrow$	

Fig. 4 ISAKMP/Oakley's Phase 1 authenticated with pre-shared key (Aggressive Mode).

Subsequently, the responder checks whether the hash value computed by him/herself is identical to the received hash value. If identical, the responder accepts the received information. Likewise, the initiator verifies the responder's hash on receiving the final message from the responder.

Aggressive Mode uses quite similar steps to compute the hash; the difference is that Aggressive Mode allows for a wider range of identifiers of the pre-shared secret. Aggressive Mode allows two parties to maintain multiple, different pre-shared keys and identify the correct one for a particular exchange.

4.3 Two Pre-Shared Keys

Conforming to the requirements of our mechanism, the first message of either Main Mode or Aggressive Mode is generated by a client (say, Bob); in other words, the initiator is a client. Likewise, the responder is a server (say, Alice) in our mechanism. The protocol MUST provide two ordered private keys (k_1, k_2) per server-and-client pair at the same time,

Either in Main Mode or in Aggressive Mode, a straightforward adaptation of our mechanism into Phase 1 is:

(1) provide two pre-shared private keys per initiator-and-responder pair at the same time
(2) use one of the two pre-shared keys in the construction of hash payloads
(3) accept the received hash if either of the pre-shared key gives the identical hash value
(4) send an encrypted key at the final stage of the session
(5) update the *current* keys

A more efficient adaptation is to use the key identifier; the computational cost of hash verification would be reduced in half with the help of the identifier. Aggressive Mode suits this modification in essence, since it allows the server and the client to maintain multiple, different pre-shared keys and identify the correct one for a particular exchange.

5. Conclusions

Towards active authorized procedures in an open network, we introduced a one-time membership rental and an inspection mechanism for server-and-client protocols based on a private-key cipher. The mechanism makes one-session rental of the membership available for an authorized entity; voluntary clients or cooperators can lend their membership to the inspection authority without disclosing the keys which will be used in the next and future sessions.

As an upper-layer application example, an on-line digital-valuable storage system called ESDB was shown with a description of its authentication phase and acknowledgment phase. Among technical issues, the usage of time-stamping service is very important and will be investigated further. For a lower-layer application example, we extended the proposed mechanism into a protocol with a keyed pseudo-random function; authentication mechanism with two pre-shared keys are shown in conjunction with ISAKMP/Oakley's Phase 1.

Future work will include public-key based authentication.

References

1) Menezes, A., van Oorschot, P. and Vanstone, S.: *Handbook of Applied Cryptography*, CRC Press, Inc., Florida (1996).
2) Zheng, Y.: Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption), *Advances in Cryptology — Crypto'97*, Springer-Verlag, Lecture Notes in Computer Science 1294, pp. 165–179 (1997).
3) Matsuura, K., Zheng, Y. and Imai, H.: Electronic Safe-Deposit Box, *Abstracts of the 20th Symposium on Information Theory and Its Applications (SITA97)*, pp. 385–388 (1997)
4) Denning, D. E. and Branstad, D. K.: A Taxonomy for Key Escrow Encryption Systems, *Communications of the ACM*, Vol. 39, No. 3, pp. 34–40 (1996).
5) Walker, S. T., Lipner, S. B., Ellison, C. M. and Balenson, D. M.: Commercial Key Recovery, *Communications of the ACM*, Vol. 39, No. 3, pp. 41–47 (1996).
6) Ganesan, R.: How to Use Key Escrow, *Communications of the ACM*, Vol. 39, No. 3, p. 33 (1996).
7) Ganesan, R.: The Yaksha Security System, *Communications of the ACM*, Vol. 39, No. 3, pp. 55–60 (1996).
8) Abelson, H., Anderson, R., Bellovin, S. M., Benaloh, J., Blaze, M., Diffie, W., Gilmore, J., Neumann, P. G., Rivest, R. L., Schiller, J. I. and Schneier, B.: The Risk of Key Recovery, Key Escrow, and Trusted Third-Party Encryption, http://www.crypto.com/key_study (1997).
9) National Bureau of Standards. *Data Encryption Standard*, Federal Information Processing Standards Publication FIPS PUB 46, U. S. Department of Commerce (1977).
10) Zheng, Y.: The SPEED Cipher, *Proceedings of Financial Cryptography'97*, Springer-Verlag, Lecture Notes in Computer Science 1318 (1997).
11) Matsui, M.: New Encryption Algorithm MISTY, *Preproceedings of 4th International Workshop of Fast Software Encryption*, pp. 53–67 (1997).
12) Zheng, Y. and Imai, H.: "Compact and Unforgeable Key Establishment over an ATM Network". *Proc. of IEEE INFOCOM'98* (1998).
13) Atkinson, R. J.: Toward a More Secure Internet, *IEEE Computer*, Vol. 30, No. 1, pp. 57–61 (1997).
14) Maughan, D., Schertler, M., Schneider, M. and Turner, J.: Internet Security Association and Key Management Protocol (ISAKMP), *Internet draft*, draft-ietf-ipsec-isakmp-**.txt
15) Orman, H. K.: The OAKLEY Key Determination Protocol, *Internet draft*, draft-ietf-ipsec-oakley-**.txt
16) Harkins, D. and Carrel, D.: The Resolution of ISAKMP with Oakley, *Internet draft*, draft-ietf-ipsec-isakmp-oakley-**.txt
17) Diffie, W. and Hellman, M.: New Directions in Cryptography, *IEEE Trans. Information Theory*, Vol. IT-22, No. 6, pp. 644–654 (1976).
18) Harkins, D. and Carrel, D.: The resolution of ISAKMP with Oakley, *Internet draft*, draft-ietf-ipsec-isakmp-oakley-**.txt