INSPECTION MECHANISM FOR SERVER-AND-CLIENT PROTOCOLS WITH PRIVATE-KEY CIPHER

By Kanta Matsuura and Hideki Imai
3rd Dept., IIS, Univ. of Tokyo, Roppongi 7-22-1, Minato-ku, Tokyo 106, JAPAN.
kanta@iis.u-tokyo.ac.jp, imai@iis.u-tokyo.ac.jp

ABSTRACT

In an open network, 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 or judgment; authorities are involved mainly in the settlement phase. Depending on system-design and security policies, 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 and thus 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. As an application example, a digital-valuable storage system called an “electronic safe-deposit box” is described and its protocol is shown in part.

1. Introduction

In order to make a wide use of an open network, 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. The readers can consult [Menezes, 96] about these schemes. 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 [Zheng, 97a]. 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 phase. Depending on system-design and security 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. 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, Section 3 shows an application example of this mechanism: electronic safe-deposit box [Matsuura, 97] for digital valuables in a future network life. Finally, Section 4 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 security protocols. This inspection mechanism considers server-and-client protocols which are based on a private-key cipher, since security services most probably have a private-key cipher scheme at their core.

In the following, the capitalized word “MUST” or the capitalized adjective “REQUIRED” means that implementation of the item is an absolute requirement of the specification while the capitalized word “SHOULD” or the capitalized adjective “RECOMMENDED” means that there might exist valid reasons in particular circumstances to not implement this item, but the full implications should be understood and the case should be carefully weighed before not implementing this at all or not implementing this in a conforming manner.

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 ordered private keys per server-and-client pair at the same time,

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

(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 the requirements (1)-(4) are satisfied.

2.2 Other Requirements and Recommendations

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. The protocol SHOULD be successfully synchronized and MUST be divided into authentication phase, message-exchange phase, and acknowledgment phase. In the authentication phase and the acknowledgment phase, a reliable time-stamping service SHOULD be available.

2.3 Normal Procedure

Conforming to the requirements, 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 k1 and the other current key by k2. Main process of the session, then, uses k1 for authentication, message delivery/exchange, and acknowledgment. Finally, Alice generates a new key k3, encrypts it with the non-session key k2, and sends the result to Bob. Alice's digital signature is attached if necessary. On receiving this, Bob decrypts the new key, and the shared current keys are updated from (k1, k2) to (k2, k3); the next session will be initiated by using k2 or k3. k1 will never be used by these two entities afterwards. The order of the current keys can be changed, depending on the protocol implementation.

2.4 Inspection Procedure

This subsection introduces an inspection procedure in which Pole performs a behavior check on Alice.

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 information of the current keys is attached if necessary. Let us represent the chosen key by k1 and the other key by k2. Pole then masquerades as Bob; Pole starts a session by using k1. 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 k3, encrypts it with k2, 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 (k1, k2) to (k2, k3). The next session

will be initiated by using k2 or k3, both of which are not disclosed to Pole. The order of the current keys can be changed, depending on the protocol implementation.

3. Electronic Safe-Deposit Box

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 contents; not only private keys for encryption/authentication, but also digital personal belonging such as artistic contents, family treasures, memorials, and digital certificates require long-term secure storage in an electronic form. In addition, execution logs themselves can be important objects to be securely stored. In preparation for such a situation, the authors of [Matsuura, 97] 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 inspection mechanism since the framework in [Matsuura, 97] mostly conforms to the protocol requirements described in Section 2. After an introduction of the backgrounds, this section shows an ESDB structure and describes two of the phases in the structure.

3.1 Backgrounds

Aimed for 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 security systems have to be stored by users in a secure way. This might probably make individuals and organizations fear the consequences of losing their cryptographic keys. As a solution for this problem, Key Recovery Systems (KRSs) are now of great interest and discussed from various viewpoints [Denning, 96], [Walker, 96], [Ganesan, 96], [Abelson, 97].

Potential scenarios of storage requirements can be described as follows. We may want to enjoy on-line shopping in a secure manner. 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 enjoy family life as much as possible. SOHO (Small Office Home Office) systems help us with saving the commuting time and so on, and enable us to have home lunch, for example. Since the secure communication system for SOHO application is most probably chosen by the company or organization, it would be different from that for home application chosen by each employee. Thus we have another secrets. 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 the very favorite contents strictly private, we want to keep them in a secure way. If they are artistic objects signed by the authority and obtained with an extremely high cost, they are no longer normal contents but treasures or assets, we should say. 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 human 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.

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 the users, which would change user burden; it might be increased in some cases, and decreased in others, depending on the system. Service 4 categorizes ESDB as a system which alleviates the burden.

3.2 Flow of On-Line ESDB

If the valuables are cryptographic keys, we may be able to use a KRS as an on-line backup system [Denning, 96], [Walker, 96], [Ganesan, 96]. In order to deposit secret materials including valuables of a larger size, we here consider an on-line ESDB.

The ESDB is constructed in conformance with the requirements and recommendations described in Section 2. The basic flows are Deposit Flow and Withdrawal Flow. The former consists of authentication phase, deposit phase, and acknowledgment phase. Likewise, the latter consists of authentication phase, withdrawal phase, and acknowledgment phase. 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 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 execution time 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,

ID(X): ID of an entity X,

E(k; M): message M in an encrypted form with a session key k,

SIG(X; M): message M digitally-signed by the secret key of an entity X,

and

||: concatenation.

E denotes an encryption algorithm of a private-key cipher such as DES [NBS, 77], SPEED [Zheng, 97b], or MISTY [Matsui, 97]. It is assumed that a public-key infrastructure is available for digital signature.

The customer Bob first generates a random number r1. 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(k1; r1 || ID(Bob))

where k1 represents the session key arbitrarily chosen by Bob and ind is the indicator which tells the recipient that k1 is used in this session. On receiving this, Alice decrypts it with k1 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 r2 and sends a reply message to Bob by using the session key k1. The reply message is

E(k1; r1|| r2 || TS(Alice)).

This reply message is decrypted by Bob. He then confirms that the random number r1 is identical to the original and that Alice's time-stamp TS(Alice) is valid. If both successful, he accepts the reply and the random number r2 as a valid one.

In deposit/withdrawal phase, k1 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(k1; M || r1|| r2 || TS(Bob) || TS(Alice))

where M consists of part 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(k1; SIG(Alice; M || r1|| r2 || TS(Bob) || TS(Alice) || E(k2; k3, ind))),

which is generated if Alice accepts TS(Bob) as a valid time-stamp and admits the whole session has been successfully executed. k3 is a new key which is generated by Alice and ind indicates the order of the updated current keys: (k2, k3) or (k3, k2).

3.5 Discussion

In order to protect users from attacks by a malicious entity which pretends to be an authorized party, time-stamps are very useful; if needs arise, they can be used to tell whether the correct users are responsible for the results of the malicious entities' behavior or not. For this purpose, time-stamps SHOULD be somehow included in or attached to 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 security protocols such as key-distribution/agreement, for example, similar functions are often supported by either time-stamps or fresh random numbers called nonces (examples designed for ATM networks can be found in [Zheng, 98]). 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 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 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 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. 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. Conclusions

Towards active authorized procedures in an open network, this paper introduced an inspection mechanism for server-and-client protocols which are 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 application example, an on-line digital-valuable storage system 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.

REFERENCES

[Menezes, 96] Menezes, A., van Oorschot, P., and Vanstone, S. (1996), Handbook of applied cryptography, CRC Press, Inc.

[Zheng, 97a] Zheng, Y. (1997), 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, 165-179.

[Matsuura, 97] Matsuura, K., Zheng, Y., and Imai, H. (1997), Electronic safe-deposit box, Abstracts of the 20th Symposium on Information Theory and Its Applications (SITA97), 385-388.

[Denning, 96] Denning, D. E. and Branstad, D. K. (1996), A taxonomy for key escrow encryption systems, Comm. ACM, 39, 34-40.

[Walker, 96] Walker, S. T., Lipner, S. B., Ellison, C. M., and Balenson, D. M. (1996), Commercial key recovery, Comm. ACM, 39, 41-47.

[Ganesan, 96] Ganesan, R. (1996), The Yaksha security system, Comm. ACM, 39, 55-60 .

[Abelson, 97] 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. (1997), The risk of key recovery, key escrow, and trusted third-party encryption, http://www.crypto.com/key_study.

[NBS, 77] National Bureau of Standards (1977), Data Encryption Standard, Federal Information Processing Standards Publication, FIPS PUB 46, U. S. Department of Commerce.

[Zheng, 97b] Zheng, Y. (1997), The SPEED Cipher, Proc. of Financial Cryptography'97, Lecture Notes in Computer Science, Springer-Verlag.

[Matsui, 97] Matsui, M. (1997), New encryption algorithm MISTY, Preproceedings of 4th International Workshop of Fast Software Encryption, 53-67.

[Zheng, 98] Zheng, Y. and Imai, H. (1998), Compact and Unforgeable Key Establishment over an ATM Network, to be presented at 17th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'98), San Francisco, March 29 - April 2.