On the KEY PREDISTRIBUTION SYSTEM: A Practical Solution to the Key Distribution Problem

Tsutomu MATSUMOTO
&
Hideki IMAI

Division of Electrical and Computer Engineering
YOKOHAMA NATIONAL UNIVERSITY
156 Tokiwadai, Hodogaya, Yokohama, 240 Japan

August 1987

Abstract To utilize the common-key encryption for the efficient message protection in a large communication network, it is desired to settle the problem of how to distribute the common keys. This paper describes a practical solution called the ***key predistribution system*** (KPS, for short), which has been proposed by the present authors. On request, the KPS quickly brings a common key to an arbitrary group of entities in a network. Using the KPS, it is quite easy to construct an enciphered one-way communication system, as well as an enciphered two-way (interactive) communication system. For example, even in a very large public network, the KPS can be applied to realize a practical enciphered electronic mailing service directed to individuals. This paper presents secure and efficient realization schemes for the KPS. This paper also discusses the security issues and the variety of applications of them.

1. Introduction

Cryptography is a key technology for the information security of our everyday lives. Main features of today's and near future's cryptographic communications include the point that they are performed in large open networks having a lot of *entities* (or, users, subscribers, terminals, ...): we will often face to the need of cryptographic communications with what we don't know very well, in, for examples, facsimile systems, portable telephone systems, personal (card) computer networks, or, ...

For such large-network cases, both common-key and public-key approaches have had to require somewhat complex and costly protocols to overcome the so-called key distribution problem. And it has not been so easy to build a practical enciphered electronic mailing system for a large network by applying those conventional approaches.

Main purposes of this paper are to introduce the ***key predistribution system*** (KPS, for short) proposed by the present authors in [MI86a,b,c] and to present secure and efficient realization schemes for the KPS. The KPS is a kind of key distribution system for the common-key systems and, on request, brings a common key to each member of any group of specified entities in a network, without previous communications among the group

C. Pomerance (Ed.): Advances in Cryptology - CRYPTO '87, LNCS 293, pp. 185-193, 1988.

nor accesses to any public key directory or whatsoever. So, the KPS has the variety of applications which have never been realized by the conventional common-key nor public-key systems alone.

In the following, the definition, the applications, and the security of the KPS are described in **2**, in **3**, and in **4**, respectively. Then several realization schemes based on the linear algebra and/or smart cards are proposed in **5**.

2. The Key Predistribution System

Imagine a public communication network consisting of many entities (users), say, *entity*1,*entity*2,. . .,*entity n* and trusted managing center(s), say, *center*1,*center*2,. . .,*center s*, where n and s are positive integers. We assume each entity (say, *entity i*) has its own *identity* y_i : the identity y_i may be the entity's name, address, etc., or combinations of such items.

The Key Predistribution System (KPS) is a method of sharing a cryptographic key, among any group of specified entities in such a network, according to the following processes [MI86a,b,c]:

(1) Generation of Center-Algorithms
(2) Calculation and distribution of each entity's Secret-Algorithm
(3) Key sharing among the group.

Process(1) is required only when the system is setting up or renewed. At this process, each center (say, *center p*) independently generates a special Center-Algorithm $\mathbf{G}_p$ which should be kept secret by the owner (*center p*).

Process(2) is operated between an entity (say, *entity i*) and every center only when the entity joins the system or an emergency arises. In Process(2), for *entity i*, each center (say, *center p*) applies its Center-Algorithm $\mathbf{G}_p$ to the identity y_i of *entity i*, and obtains an algorithm $X_{pi} = \mathbf{G}_p(y_i)$, and sends X_{pi} only to *entity i*, with the aid of some cryptographic or physical protection mechanisms, eg., smart cards. Here, a smart card means a well-access controlled physically secure small computing device. *Entity i* combines the received algorithms $X_{1i}, X_{2i}, \ldots, X_{si}$ and keeps the resulting Secret-Algorithm X_i confidentially. For the convenience of *entity i*, the the Secret-Algorithm X_i may be generated by some interactive procedures among the centers and a smart card owned by *entity i*.

If Process(1) and Process(2) have been set up, a group of entities can operate Process(3) to have a common cryptographic key whenever they like. Each member of the group computes the same key by inputting into its Secret-Algorithm the identities of all members except itself. Let $X_i^{(e)}$ denote the Secret-Algorithm of *entity i* used for key sharing among a group of e entities ($e \geq 2$). Then, for example, if *entityA* and *entityB* want to share a cryptographic key k, both entities compute k in the following manner:

$$k = X_A^{(2)}(y_B), \quad k = X_B^{(2)}(y_A).$$

If three entities *entityA*, *entityB*, and *entityC* want to have the same cryptographic key l, they obtain l as follows:

$$l = X_A^{(3)}(y_B, y_C), \quad l = X_B^{(3)}(y_C, y_A), \quad l = X_C^{(3)}(y_A, y_B).$$

In the key predistribution system, once the prerequisite Process(1) and Process(2) are accomplished, each entity does not have to communicate with any centers nor any entities for the purpose of acquiring cryptographic keys. The only necessary items are the other member's identities. This is the most significant feature of the KPS.

Such an advantage could not be attained by any conventional cryptographic methods in a large communication network. Indeed, the key translation / distribution center methods [ISO87], which seem to be very popular, require extra communications on channels with confidentiality and authenticity. And even for the public-key distribution / encryption / signature systems [DH76], extra authentic communications are indispensable. They may be maintained by the central public-key-file manager and/or the use of certified public keys based on some digital signature schemes.

This feature of the KPS makes it possible to readily build an one-way end-to-end encryption system even in a very large network. That is, without extra crypto-processing in the network, one can send enciphered electronic mails to anybody who joins the system. The target of the mail may consist of several entities. Of course, the KPS is still effective for small networks and for enciphered two-way (interactive) communications such as telephones.

For the KPS, only what distinguish the entities are the identities of them. So, depending on the configuration of the identities, the KPS can provide a variety of key sharings. And these identities are the only public items used in the KPS. Note that the Center-Algorithms and the Secret-Algorithms are kept secret by their owners.

Remark The name "Key Predistribution System" comes from the fact that the notion of KPS includes the following primitive method: in Process(1) each center generates keys for all possible groups, then in Process(2) each entity receives from the centers a list of all keys for the groups including the entity as a member, and in Process(3) each member of a group reads out the common key from its list according to the identities of the group members. It is apparent that this method is secure as long as the centers are trusted. But for large networks, this primitive approach is impractical since the memory size of the list becomes extraordinary big. This suggests that one of the points we must attention in the research of the KPS is how to reduce the memory size and the computational complexity of each entity's Secret Algorithm with keeping enough security level.

Remark In the point that the identities play important roles, the notion of KPS is like the ID-based cryptosystem; the independent work of A. Shamir's [S84]. But we believe that the KPS is the first unified treatment of the method of key sharing among a group of entities without previous communications.

How to construct the KPS ? The following is a very general answer. Select an e-input symmetric function g. Then, for each identity y_i generate a certain $(e-1)$-input algorithm $X_i^{(e)}$ satisfying

$$X_i^{(e)}(\xi_1,\ldots,\xi_{e-1}) = g(y_i,\xi_1,\ldots,\xi_{e-1}).$$

In **5** we will construct some schemes by adopting e-linear mappings as the g in this idea. And we feel that there are other useful schemes for the KPS constructed by some combinatoric or geometric or algebraic approaches.

3. Applications of the KPS

The key predistribution system has wide applications. Indeed it can be effectively used for any fields to which the common-key systems can be applied. Therefore, some combinations of the KPS and common-key cryptographic techniques can bring the functions including the keeping confidentiality and authenticity of messages, the peer entity authentication, the access control supporting, etc.

Moreover, we believe that the KPS has many potential applications to the smart-card-based systems, since the KPS can be elegantly realized even by today's not-so-powerful smart cards, as we will demonstrate later, and since the KPS seems to be the best fit cryptographic partner for the smart cards.

In the following we introduce some of these applications.

For $i = 1, 2$, let (E_i, D_i) be a common-key cryptosystem, i.e., E_i, D_i be a pair of algorithms satisfying

$$D_i(k; E_i(k; M)) = M,$$

for any key k and any message M. For (E_i, D_i), fast and strong stream/block algorithms are suitable for practical purposes.

Let (H, U, V) be an integrity mechanism, i.e., H, U, V be a triple of algorithms such that V takes either OK or NG and that for any message M there hold

$$U(H(M)) = M, \quad V(H(M)) = OK$$

and that if $N = H(M)$ is altered into some N' then in high probability,

$$V(N') = NG.$$

An error-detecting code can be used as (H, U, V).

Electronic Mail: Suppose that *entityA* wants to send *entityB* a message M in an enciphered form.

(Step 1) $\{entityA\}$: Generate a random number r. Compute

$$F = E_1(X_A^{(2)}(y_B); r)$$

$$C = E_2(r; H(M))$$

and send the mail $(y_B; y_A; F; C)$ to *entityB* or to the *entityB*'s mail box.

(Step 2) $\{entityB\}$: Receive $(y_B; y_A; F'; C')$ from *entityA* or from the mail box. Compute

$$r' = D_1(X_B^{(2)}(y_A); F')$$

$$N' = D_2(r'; C')$$

$$M' = U(N')$$

$$T = V(N')$$

and judge that $M' = M$ and the sender is really *entityA* iff $T = OK$.

Electronic Broadcasting Mail: Suppose that *entityA* wants to send *entityB*1, *entityB*2, ..., *entityBe* a message M in an enciphered form.

There are two generalizations of above Electronic Mail.

One is to utilize $X_A^{(e+1)}$ instead of $X_A^{(2)}$. That is, *entityA* broadcasts the mail

$$(y_{B1}, y_{B2}, \ldots, y_{Be}; y_A; F^*; C)$$

where $F^* = E_1(X_A^{(e+1)}(y_{B1}, y_{B2}, \ldots, y_{Be}); r)$.

The other is to use F_j for *entityBj* for $j = 1, \ldots, e$. That is, in this method *entityA* broadcasts the mail

$$(y_{B1}, y_{B2}, \ldots, y_{Be}; y_A; F_1, F_2, \ldots, F_e; C)$$

where $F_j = E_1(X_A^{(2)}(y_{Bj}); r)$, and *entityBj* does the similar procedure in (Step 2).

The advantages of the former method are that the length of the mail is independent from the number of the receivers and that each receiver can check whether the mail was really directed to the specified receivers. On the other hand, the memory size and the computational complexity of Secret-Algorithm $X_A^{(e+1)}$ may rapidly increase according to $e + 1$. Opposed to this, although the increment of the mail length is linear in e, the latter method requires relatively cheap Secret-Algorithm $X_A^{(2)}$ only. Especially, the load of each receiver is not affected by the increase of receivers. We think that the latter method is much practical than the former.

Remark In the above mailing methods, no crypto-processing is required on the mails transferred through the network. Thus those enciphered electronic mails can be treated just as usual cleartext mails. This means the readiness of implementation.

Remark Since the above mailing methods are based on common-key systems, sender authentication is naturally implemented. This is an advantage. On the other hand, there may be a case where the sender wants to send an anonymous message. To attain this end, it is sufficient to prepare another identity (pen name) and the corresponding Secret-Algorithm.

Implementation with Smart Cards and Terminals: If Secret-Algorithms and other crypto-algorithms are stored and worked in their owner's powerful smart cards, there is no problem. We do hope that such powerful and compact devices will be available in near future. However, since at present any smart card is not so powerful, it seems to be practical to implement the KPS and related crypto-functions by some combinations of smart cards and terminals.

Smart cards are assumed to have Secret-Algorithms and some private informations, but not to have enough computing power. Terminals are assumed to have enough computing ability. One of the problems is how to work the Secret-Algorithm using both facilities so that the smart card obtains the key but that the terminal cannot get it. An example of solutions will be given in **5.4**.

Another problem is how to efficiently do the encipherment and the decipherment by using the smart card having the key and the terminal, so that the key cannot be known to the terminal. For the case of above mentioned Electronic (Broadcasting) Mail, one of the possible solutions is to implement the algorithms E_1 and D_1 and the random number generation in the smart cards, and to implement (E_2, D_2) and (H, U, V) in the terminals.

4. Security of the KPS

The security of the KPS highly depends on the centers. If there is only one center, it can do everything since it can readily compute any key for any group in the network. But if we pose several centers, none of keys can be disclosed by the center side as long as there is at least one trusted center.

In Process(2) of the KPS, the Secret-Algorithm for *entity i* should be passed only to *entity i*. Thus rigorous entity identification must be adopted. But this does not mean the necessity of individual identification. As we have mentioned, entities are distinguished only by their identities.

Generally speaking, in any key predistribution system, information or knowledge on the Center-Algorithm $\mathbf{G}_p$ is distributed into the Secret-Algorithms $X_1, X_2, \ldots, X_n$. Therefore, if enough number of entities (breakers) collaborate, they may obtain important information which enables to completely or partially determine the keys shared by other groups.

Of course, if the center side embed the Secret-Algorithm in a well access-controlled physically secure computing device and pass it to its owner (an entity), then the Secret-Algorithm can be used for key sharing while it cannot be read even by its owner. Thus if complete physical security is available, any key predistribution system is secure. But, today, it is rather difficult to expect complete physical security. For example, a smart card, one of the hopeful candidates for such devices, does not seem to have complete physical security.

So in practical situations, the KPS should be constructed so that valuable information on $\mathbf{G}_p$ or X_t cannot be derived unless the breakers includes a lot of entities or unless a huge amount of computation is completed. We refer to the former as the information-theoretic security and the latter as the complexity-theoretic security.

5. Linear Schemes for the KPS

In this section, we present a class of realization schemes for the KPS. Its name is the *linear schemes*. The information-theoretic security of a linear scheme is reduced to the linear-independency of a certain set of vectors.

5.1 Definition of Linear Schemes

To simplify the description, we assume here that there is only one center in the network considered. For the general case, see [MI86d].

Let q be a prime power and m, h be positive integers. Let $Q = GF(q)$ and Q^m denote the vector space consisting of all m-row vectors over Q.

Assume that each entity's (say, *entity i*'s) identity y_i belongs to a set I and that $y_i \neq y_j$ if $i \neq j$.

And let R denote an one-way algorithm implementing an injection from I to Q^m.

The *center* selects (m, m) symmetric matrices $G_1, G_2, \ldots, G_h$ over Q randomly and independently from other entities.

The *center* generates the Center-Algorithm G which generates the algorithm X_i :

$$X_i(\xi) = x_i R(\xi)^T, \quad \xi \in I$$

for each $y_i \in I$. Here, x_i is an (h, m) matrix defined by

$$x_i{}^T = [G_1 R(y_i)^T, \ldots, G_h R(y_i)^T].$$

Each entity (say, *entity i*) receives its own Secret-Algorithm X_i from the *center*.

If *entityA* and *entityB* want to share a common cryptographic key, *entityA* computes $X_A(y_B)$ and *entityB* computes $X_B(y_A)$ independently. They are h-vectors over Q. It is simple to check that both vectors are the same.

If we use symmetric e-linear mappings instead of the above mentioned G_l's (symmetric bilinear mappings), we have similar schemes for key sharing among e entities.

5.2 Security of Linear Schemes

Since we assumed that there is only one center in the network, the *center* must be trusted.

To completely break the linear scheme is to determine the matrix

$$G = [G_1, \ldots, G_h].$$

By the definition of matrix x_i, at least *rank* G entities should cooperate to completely determine G. The value of *rank* G is approximately m in high probability.

Next we discuss the possibility of determining other Secret-Algorithms by some entities.

Let E denote the set of all entities in the system. It is readily derived that "even if all entities in a subset E_B of E cooperate and use $\{x_j \mid j \in E_B\}$, they cannot have any valuable information on determining a key for arbitrary pair of entities in $E - E_B$" is equivalent to "for each $i \in E - E_B$, $R(y_i)$ is linearly independent from $\{R(y_j) \mid j \in E_B\}$". Thus the information-theoretic security of the linear scheme is reduced to the linear-independency of the vector set

$$U = \{R(y_i) \mid i \in E\}.$$

So, there is a tight relation between the linear schemes and the theory of linear error-correcting codes. Consult [MI86d] for more precise discussions.

One direction is to use well known algebraic or algebraic-geometric codes. But this approach becomes rather costly when we use a big security parameter m. The other direction is to utilize random codes. By using a technique similar to one for deriving the famous Gilvert-Varsharmov bound, we can see that almost all algorithms R bring good codes and good U for our purpose. Thus we prefer to use one-way algorithm R which runs very quickly. Candidates of R may include the iterated DES-like algorithms.

The one-way property of R is required for increasing the computational load of searching the members and the targets of the breakers.

5.3 Features of Linear Schemes

The greatest features of the linear schemes are their simplicity and efficiency. Indeed, the memory size of each Secret-Algorithm is the sum of $hm \log_2 q$ [*bit*] and the memory size of R. And the computational complexity of each Secret-Algorithm is the sum of $O(hm)$ $[Q - operation]$ and the complexity of R. The memory size and complexity of R can be bounded sufficiently low if we use R suggested in **5.2**.

The class of linear schemes is wider than its appearance. For example, if we select (multi-variate) polynomial functions instead of the e-linear mappings, we cannot extend the class. The reasons are that any polynomials are rewritten as linear polynomials by some translations of the set of indeterminates and that such translations can be absorbed in the R. Moreover some scheme can be interpreted as the composition of a linear scheme and some back-end algorithm such as the discrete exponentiation.

5.4 Practical Linear Schemes with Smart Cards

In this subsection we try to demonstrate the usefulness of the linear schemes by an example. Let $m = lh$ and divide x_A as

$$x_A = [x_{A1}, \ldots, x_{Al}],$$

where x_{Aj}'s are (h, h) matrices. Select permutation matrices $P_{A1}, \ldots, P_{Al}$ and a non-singular matrix E_A and compute

$$V_{Aj} = P_{Aj}^{-1} E_A^{-1} x_{Aj}$$

for $j = 1, \ldots, l$.

We divide the Secret-Algorithm for *entityA* into two parts. One is a set of R and V_{Aj}'s stored in a memory card, which is non-intelligent but has relatively large memory capacity. The other part is a set of E_A and P_{Aj}'s stored in a smart card, which is physically protected and has certain computing ability.

When *entityA* use these, R and V_{Aj}'s will be loaded into its terminal, which may be its personal computer, and performed in the terminal. That is, if *entityA* enters some identity, say y_B into the terminal, then the terminal applies R to the input and computes l h-vectors s_j and then multiplies V_{Aj}'s to s_j's and and outputs the resulting h-vectors t_j.

These outputs are transferred into the smart card to be processed into the common key. In the smart card, every t_j is at first permuted according to P_{Aj} then added into a single h-vector u, which is then multiplied by E_A into the h-vector key k.

This method succeeds in reducing the load of the smart card with keeping the secrecy of the Secret-Algorithm from the terminal.

When $q = 2$, $l = 128$, and $h = 64$, the required size of the memory card is 512 *Kbits* + (some for R) and that of the smart card is 52 *Kbits*. In this case, we can have a KPS by which an h-bit key is brought to any pair of entities in a network consisting of up to 10^{12} entities. The system is primarily protected by the physical security of each smart card. And the system cannot be completely broken unless at least 8192 entities break their own smart cards and collaborate with each other. And any breakers (collaboration of entities which have known their own Secret-Algorithms) cannot obtain any keys for the entities outside of the breakers, unless the number of members is greater than about 256.

References

[DH76] Diffie,W. and Hellman,M.E.,"New directions in cryptology," *IEEE Trans. on Information Theory*, Vol. IT-22, No. 6, November 1976, pp.644-654.

[ISO87] ISO/TC68, "Banking-Key management (wholesale)," ISO/DIS 8732, February 1987.

[MI86a] Matsumoto,T. and Imai,H., *Patent application*, July, 1986.

[MI86b] Matsumoto,T. and Imai,H., "The Third Key Distribution System," *Proceedings of the 1986 Workshop on Cryptography and Information Security, Yokohama, Japan*, August 27, 1986, pp.39-41.

[MI86c] Matsumoto,T. and Imai,H., "The Key Predistribution System," *IECE Technical Report* TGIT86-54, Institute of Electronics and Communications Engineers of Japan, Vol.86, No.145, September 18, 1986, pp.29-34.

[MI86d] Matsumoto,T. and Imai,H., "A Key Predistribution System Based on Linear Algebra," *Proceedings of the* 9^{th} *Symposium on Information Theory and Its Applications, Akakura, Japan*, October 29, 1986, pp.713-718.

[S84] Shamir,A., "Identity-Based Cryptosystems and Signature Schemes," *Advances in Cryptology: Proceedings of CRYPTO84*, Springer LNCS 196, 1985, pp.47-53.