A Formal Approach to Security Architectures[1]

Rainer A. Rueppel

R^3 Security Engineering, 8623 Wetzikon

Eidgenössische Technische Hochschule, Zürich

Switzerland

Abstract

We define a formal language whose symbols are security goals and mechanisms. This allows us to express every security architecture as a string. Designing a security architecture becomes the task of generating a word in the language. Analysing a security architecture becomes the task of parsing a string and determining if it belongs to the language. Since not every complete security architecture achieves its goals equally efficient, we associate a complexity parameter to every goal and mechanism. This allows us to identify complexity- reducing and complexity-increasing mechanisms.

1 Introduction

In computer systems development, formal methods are applied to tasks such as requirements analysis, system design, system verification, system validation, and system analysis and evaluation. In computer security, formal methods are applied in particular to the problem of access control [14]. For example, in the Bell-LaPadula model [1] a computer system is modelled as a finite-state machine where user actions cause state transitions; the main interest is in determining if the system can ever reach an insecure state.

In cryptology, formal methods have been applied to the analysis of protocols [11, 2, 6]. Here the main interest is in verifying the assumptions on which the security of the protocol depends, and in verifying that the protocol achieves the desired goals while not releasing any compromising information. Typically, these formal methods are too restricted to be applicable to the general task of analysing and designing a security architecture.

In the standards arena, as exemplified by the treatment of security in the OSI-model [7], there is a great uncertainty about how the various security services, security mechanisms, and the corresponding security management are to be interconnected or to be embedded into the system architecture[13, 8, 9, 10, 5].

Another aspect, although important in practice, has only received little attention in theory. This is the problem of assessing the management complexity that results from specific choices of security services and mechanisms [18].

Now suppose you are the designer of a secure system. You would like to know

1. what design choices are available at a specific layer of the security architecture,

[1]This work was supported in part by the Hasler-Foundation, Switzerland

2. what the consequences of these choices are, in particular, if the management complexity can be reduced using appropriate security techniques and mechanisms,

3. when the security architecture is complete.

The methodology presented in this paper helps to answer all of these questions. We identify security goals and mechanisms with symbols in a formal language. The various ways in which security mechanisms can be combined or can be used to achieve certain security goals are governed by a set of productions (rewriting rules). In a given stage of the design, the subset of applicable productions are exactly the available design choices. A security architecture is complete, when all security goals have been satisfied. Moreover, to each security goal and mechanism we associate a parameter, its inherent complexity (measured by the number of secure channels). This allows us to identify complexity- reducing and complexity-increasing mechanisms. It also allows us to detect over- and under-achievement of security goals. Summarizing, the presented approach has applications as a formal design tool or as an analysis and evaluation tool, revealing unstated assumptions, inconsistencies, and unintentional incompleteness. In section 1.3 we give examples for both uses.

In this paper we limit ourselves to channels and trusted authorities as architectural components in a homogeneous environment (one in which all participants have the same capabilities). But it is obvious that our approach can be extended in various ways, for instance, to include the problem of key generation. It is beyond the scope of this paper to provide an exhaustive list of productions for all security goals and mechanisms. Instead, the purpose is to show that even complex concepts such as certificate or identity- based systems are amenable to analysis in our language.

1.1 Confidentiality and Authenticity

In practice there exists a confusing multiplicity of security services. Take for instance the security architecture of the OSI model [7]. There are such services as data origin authentication, peer-entity authentication, data integrity, confidentiality, non-repudiation. Moreover, most services distinguish between connectionless and connection-oriented mode, and between selective-field message, and traffic flow security. But fundamentally, there are only two basic security goals

Confidentiality: the sender is certain that, whoever might receive his message, only the legitimate receiver can read it. Imagine a channel, to which everybody can write, but from which only the legitimate receiver can read.

Authenticity: the receiver is certain that, when he can read the message, it must have been created by the legitimate sender. Imagine a channel to which only the legitimate sender can write, but from which everybody can read.

We will use **security** to denote the combined goal of both confidentiality and authenticity.

1.2 The Underlying Logic

There is a strict logic supporting our constructions. Suppose we use a symmetric cryptosystem to encrypt the basic data channels. Then we write

$$Secu(N^2) \rightarrow sym(secu, N^2), Secu(N^2)$$

where $Secu(N^2)$ on the left of the production denotes the goal of securing N^2 (data) channels, $sym(secu, N^2)$ denotes the symmetric cryptosystem used for security, and $Secu(N^2)$ on the right of the production denotes the goal of securing N^2 (key) channels. Clearly, security on the data links can only be achieved if the keys for the symmetric cryptosystem are securely transmitted. Hence, security of the data link implies security of the key link. Similarly, for all other productions, **achievement of the security goal on the left-hand side of the rewriting rule implies achievement of the security goal(s) on the right-hand side of the rewriting rule.** And since, while building a security architecture, we replace lower layer security goals by higher layer security goals, we achieve the original goal only if we resolve all goals by some suitable means during the construction process.

1.3 Examples

We want to illustrate our approach with some simple and self-explanatory examples.

1.3.1 Designing a Security Architecture

Suppose your task is to design a system with N participants which achieves **homogeneous individual authentication** (what we mean is, that every participant upon reception of a message can convince himself about the authenticity of the message). Your idea might be to use a symmetric cryptosystem for the data channel, which leaves you with the task of finding means to distribute the keys securely. What are your options here? Should you use another symmetric or an asymmetric cryptosystem, or a specialized key distribution protocol like Diffie-Hellman, or a tamper-proof device? Applying the construction rules given by our language and making the corresponding decision you might arrive at the following security architecture:

$$\begin{aligned} Auth(N) &\rightarrow sym(secu, N^2), Secu(N^2) \\ Secu(N^2) &\rightarrow asym(secu, N), Auth(N) \\ Auth(N) &\rightarrow phys(auth, N), ta(auth, N), asym(auth, 1), Auth(1) \\ Auth(1) &\rightarrow phys(auth, 1) \end{aligned}$$

First line: You have decided to achieve the original security goal $auth(N)$ using a symmetric system. This implies that you use (roughly) N^2 secret keys, one for each pair of participants. If more than two participants were to share the same key, the receiver could no longer authenticate the sender. Thus, the new security goal is $secu(N^2)$, that is, the secure distribution of N^2 keys. The first line also indicates that your choice has increased the complexity of the problem from N to N^2.

Second line: Now you decide to use a public-key system to solve the key distribution problem, which leaves you with the problem of authentically distributing the public keys, indicated by the new goal $auth(N)$. This choice reduces the complexity of the management problem from N^2 to N. Actually, it seems as if you have not gained anything. Since you are back at your original goal. This is not true since public keys do not change as often as the data in the basic link.

Third line: Now you decide to employ a trusted center (ta) which certifies the public keys. The participants use physically secure channels to the ta (for instance, they go there with their public keys), the center signs the public keys using its asymmetric system and distributes them back to the participants. For the certificates to be verifiable, the participants need to receive an authentic copy of the center's public key, indicated by the new goal $Auth(1)$. Note that, by employing a center, you have reduced the management complexity from N to 1. Note also that you do not have to entrust your secrets to the center.

Fourth line: In order to distribute the center's public key you decide to use an authentic physical technique. For instance, you publish it every morning in the Newspaper. There is no more security goal to be achieved. So you know that your architecture is complete.

1.3.2 Analyzing a Security Architecture

A certain vendor has a product where the data links are encrypted using a symmetric cryptosystem. For the key distribution he uses the following protocol:

1. A and B initialize a session; independently they choose secret keys k_1 and k_2.
2. Both A and B encrypt their secret keys using the public key P_{KMC} of the center and send them to the center:

$$\begin{aligned} A \rightarrow KMC: &\quad P_{KMC}(k_1) \\ B \rightarrow KMC: &\quad P_{KMC}(k_2) \end{aligned}$$

3. The KMC decrypts k_1 and k_2 using its secret key S_{KMC}; then the KMC selects a session key k_s for A and B and sends k_s back to them on the symmetric channels established by k_1 and k_2:

$$\begin{aligned} KMC \rightarrow A: &\quad E_{k_1}(k_s) \\ KMC \rightarrow B: &\quad E_{k_2}(k_s) \end{aligned}$$

4. A and B decrypt k_s with their independent keys k_1 and k_2 and are now ready to have a private session using the symmetric cryptosystem.

The center broadcasts its public key P_{KMC} continuously to all N participants.
Translating this information into our language we arrive at the following security architecture:

$$\begin{aligned} Secu(N^2) &\rightarrow sym(secu, N^2), Secu(N^2) \\ Secu(N^2) &\rightarrow sym(secu, N), ta(secu), Secu(N) \\ Secu(N) &\rightarrow asym(conf, 1), Auth(1) \\ Auth(1) &\rightarrow cont(1) \end{aligned}$$

Parsing this architecture, we get an error message at layer 3: "impossible to achieve $Secu(N)$ with $asym(conf, 1)$". The flaw is that any participant (in fact anybody in possession of the center's public key P_{KMC}) could impersonate any other participant in the system. Thus, the above architecture neither achieves confidentiality (the sender cannot be certain about the receiver of his message) nor authenticity (the receiver cannot be certain about the sender of a received message) and, as a consequence, security breaks down.

2 A Formal Language for Security Architectures

We define a language for security architectures by specifying a grammar where

1. the terminals are security mechanisms.
2. the nonterminals are security goals.
3. the productions give the available constructions, that is, the different ways in which security goals may be replaced (achieved) by security mechanisms and, possibly, other security goals.

As is customary, we designate the nonterminals by symbols which start with upper- case letters, and the terminals by symbols which start with lower- case letters.

2.1 Security Goals

For the moment we wish to deal only with homogeneous security goals, that is, with goals where each participant in the network is given the same capabilities, except possibly for a distinguished participant such as a trusted center. We will distinguish between distributed and centralized security goals, which both appear frequently in a network of N participants. They will form the nonterminals in our formal language.

2.1.1 Distributed Security Goals

$Conf(N)$ denotes homogeneous individual confidentiality. Every participant has associated a (unidirectional) channel from which he can read, but to which everybody else can only write.

$Auth(N)$ denotes homogeneous individual authenticity. Every participant has associated a (unidirectional) channel to which he can write, but from which everybody else can only read.

$Secu(N^2)$ denotes homogeneous individual security. Every pair of participants share a private channel.

Note that

$$Secu(N^2) \rightarrow Auth(N), Conf(N)$$

that is, achieving homogeneous individual confidentiality and authenticity is equivalent to achieving homogeneous individual security.

2.1.2 Security Goals Involving a Center

$Conf(ta, 1)$ denotes distinguished (or centralized) confidentiality. The center has associated a channel to which everybody can write, but from which only the center can read.

$Auth(ta, 1)$ denotes distinguished (or centralized) authenticity. The center has associated a channel from which everybody can read, but to which only the center can write.

$Secu(ta, N)$ denotes distinguished (or centralized) security. The center shares with every participant a private channel.

Because $Conf(ta, 1)$ and $Auth(ta, 1)$ do not amount to $Secu(ta, N)$ we must introduce two more security goals:

$Auth(ta, N)$ denotes individual authenticity towards the center. Every participant has associated a channel to which only he can write and, from which only the center can read.

$Conf(ta, N)$ denotes individual confidentiality from the center. Every participant has associated a channel from which only he can read and, to which only the center can write.

Note that

$$Secu(ta, N) \rightarrow Auth(ta, N), Conf(ta, N)$$

2.2 Elementary Components and Constructions

In order to achieve our security goals we must employ protection mechanisms and, possibly, trusted authorities. We will distinguish the following elementary channels:

1. Physical channel: the message is protected by some physical means.
2. Symmetric channel: the message is protected by a symmetric cryptographic technique.
3. Asymmetric channel: the message is protected by a public-key technique.

2.2.1 The physical channel

A physical channel is typically used to achieve security, that is, for confidentiality and authentication jointly. Supposedly secure physical channels are couriers, fibre optics, tamper-proof devices etc. We shall write

$phys(secu, N^2)$ to denote a homogeneous system with a secure physical channel between each pair of participants. The corresponding production is

$$Secu(N^2) \rightarrow phys(secu, N^2)$$

But it is also possible to use a physical channel for authentication only. Supposedly authentic physical channels are hardcopy, newspapers, books and *WORMs* (write-once-read-many-times devices) such as compact disks. Write access to the channel is determined by the ability to print. We shall write

$phys(auth, N)$ to denote a homogeneous system with an authentic physical channel associated to each participants. The corresponding production is

$$Auth(N) \rightarrow phys(auth, N)$$

The distinguishing feature of a physical system is that is does not require any further (secure) distribution of information in order to achieve its security goals.

2.2.2 The symmetric channel

When a symmetric channel is used for confidentiality then the sender encrypts the message M under control of a secret key K, and the data written on the channel is $E_K(M)$. When a symmetric channel is used for authentication then the sender computes a message authentication code under control of a secret key K which is then appended to the message, and the data written on the channel is $M, A_K(M)$ where A is the authentication algorithm. When a symmetric channel is used for security then the properties confidentiality and authenticity must both be enforced. One often reads the statement that authentication is implicitly provided by encryption. In general this is not true as illustrated by the one-time pad. Therefore it is advisable to use two algorithms, one for confidentiality, the other one for authentication, both with independent keys, in order to achieve security in a symmetric channel. Who can access a symmetric channel is determined by the corresponding keys. We write

$sym(secu, N^2)$ to denote a homogeneous symmetric system that allows for a secure logical channel between each pair of participants. In such a system, N^2 secret keys are needed, one for each channel.

$sym(secu, N)$ to denote a centralized system that allows for a secure symmetric channel between every participant and the center. In such a system N keys are needed, one for each channel.

We have only used the mode *secu* in the above system descriptions, since with a symmetric cryptosystem possession of the key implies the ability to read and write from the associated channel, and hence, no distinction is necessary between authenticity, confidentiality, and security (as far as security management is concerned). If a symmetric cryptosystem is used to achieve a security goal, we are then faced with the problem of distributing the corresponding keys in a secure fashion. This amounts to a new security goal. The most obvious production is

$$Secu(N^2) \rightarrow sym(secu, N^2), Secu(N^2)$$

where we establish a secure channel between every pair of participants using a symmetric cryptosystem and, as a consequence, must establish another secure channel between every pair of participants to distribute the keys. When the goal is only confidentiality or authenticity then the use of a symmetric cryptosystem will result in an overachievement (in the sense that the mechanism used does more than is required by the goal). The following two productions illustrate the situation.

$$\begin{aligned} Conf(N) &\rightarrow sym(secu, N^2), Secu(N^2) \\ Auth(N) &\rightarrow sym(secu, N^2), Secu(N^2) \end{aligned}$$

Moreover, they are complexity-increasing since, by using N^2 symmetric channels, we must then distribute N^2 keys securely.

2.2.3 The asymmetric channel

When an asymmetric channel [4, 16] is used for confidentiality then the sender encrypts the message M with the public key E_α of the recipient α, and the data written on the channel is $E_\alpha(M)$. Only the recipient α has full access to the asymmetric channel defined by E_α since only he can decrypt the ciphertexts with his secret key D_α. But everybody with access to E_α can write messages on the channel. When an asymmetric channel is used for authentication then the sender A signs the message M with his private signature key S_A, and the data written on the channel is $S_A(M)$. Everybody with access to the public verification key V_A can check the signature (read messages form the channel) but only A has the capability to also write messages on the asymmetric channel. We shall write

$asym(mode, N)$ to denote a homogeneous asymmetric system that associates with every participant his own asymmetric channel. Assuming that the participants create their asymmetric channels themselves, the remaining problem is to distribute the N public keys in an authentic fashion among the participants. According to the use of the asymmetric system we set $mode = auth, conf$, or $secu$.

The corresponding productions are

$$\begin{aligned} Conf(N) &\rightarrow asym(conf, N), Auth(N) \\ Auth(N) &\rightarrow asym(auth, N), Auth(N) \end{aligned}$$

where the new goal $Auth(N)$ denotes the problem of distributing the public keys authentically among the participants. A secure channel between two communicants may be built by composing two asymmetric channels, one for authentication and one for confidentiality. First the sender A signs the message M under his private signature key S_A and then he encrypts $S_A(M)$ under the recipients B public encryption key E_β to produce $E_\beta(S_A(M))$. The corresponding production is

$$Secu(N^2) \rightarrow asym(secu, N), Auth(N)$$

where, in fact, we have allowed $asym(secu, N)$ to stand for two different asymmetric channels (one for confidentiality and one for authenticity) for every participant. The new goal $Auth(N)$ is to distribute the public keys authentically among the participants. Note that this is a complexity-reducing production since we replace the management of N^2 secure channels by the management of N authentic channels.

2.2.4 The trusted authority

A participant may instead of directly establishing a secure channel with another participant send his message to a trusted authority, and let the authority forward his message to the intended recipient in the desired secure mode. Introducing a trusted authority means switching from a distributed to a centralized concept in the corresponding layer of the security architecture. We shall write

$ta(secu, N)$ to denote an authority which is trusted with handling the secrets of the participants.

$ta(auth, N)$ to denote an authority which is trusted with handling the participants' data in an authentic fashion.

$ta(conf, N)$ to denote an authority which is trusted with handling the participants' data in a confidential fashion.

The classical centralized security concept where the trusted authority shares a secure channel with each participant and acts as a central agent relaying information forth and back is described by the following production:

$$Secu(N^2) \rightarrow ta(secu, N), Secu(ta, N)$$

When the goal is only authenticity or confidentiality, then the following productions apply:

$$\begin{aligned} Auth(N) &\rightarrow Auth(ta, N), ta(auth, N), Auth(ta, 1) \\ Conf(N) &\rightarrow Conf(ta, 1), ta(conf, N), Conf(ta, N) \end{aligned}$$

An asymmetric channel can be used, if the goal is confidentiality towards the center, or authenticity from the center:

$$\begin{aligned} Auth(ta, 1) &\rightarrow asym(auth, 1), Auth(ta, 1) \\ Conf(1, ta) &\rightarrow asym(conf, 1), Auth(ta, 1) \end{aligned}$$

where $Auth(ta, 1)$ stands for the authentic distribution of the *ta*'s public key to all the participants. For more constructions which involve a trusted authority, please compare the sections on certificate systems and identity- based systems.

2.3 Compound Constructions

2.3.1 Diffie-Hellman Key Agreement

The Diffie-Hellman key agreement [3] (also called exponential key exchange) is a protocol which allows two participants, without prior exchange of secret information, to establish a common secret key using public messages only. The resulting secret key is used for encryption with their symmetric cryptosystems. The main problem is to guarantee authenticity of the transmitted public messages. The corresponding production is

$$sym(secu, N^2), Secu(N^2) \rightarrow sym(secu, N^2), expka(N), Auth(N)$$

where $expka(N)$ denotes the exponential key agreement. The complexity is N since we need N authentic channels, one for each participant. Note that the above production is context- sensitive; the exponential key agreement can only be applied in combination with a symmetric cryptosystem. Note also, that the exponential key exchange is complexity- reducing. Since in this paper we are not interested in the internal functioning of the protocol, the term $expka(N)$ may stand for any protocol exhibiting the same characteristics as the Diffie- Hellman protocol (compare [17]).

2.3.2 The Certificate System

The certificate system [12] is an asymmetric system used by the trusted authority for authentication of a participant's individual data. It is utilized as follows:

1. The participants register their data with the trusted authority.
2. The trusted authority signs the participant's data with its secret key S_{ta} to produce a certificate for the data.
3. The certified data is made available to all participants.

The major application of the certificate channel is the authentic distribution of the participant's public keys. The certificate channel is modelled by the following production:

$$Auth(N) \rightarrow phys(auth, N), ta(auth, N), asym(auth, 1), Auth(ta, 1)$$

A system with N authentic public keys may be implemented using N authentic physical transmissions from each participant to a trusted authority *ta*, which distributes the public keys via an asymmetric channel used for authenticity (in other words, it signs the public keys). But authenticity of the public keys of the participants is only achieved if the *ta*'s public key is distributed authentically and if the trust placed in the *ta* is justified. Note that we do not entrust any secret information to the *ta*. Note also that we have reduced the management complexity from N to 1 at the expense of a trusted authority.

2.3.3 The Identity-based Asymmetric System

The basic idea due to Shamir [19] is that the identity of each participant may directly serve as the public key giving partial access to the participant's asymmetric channel. But, if participant A was able to compute his private key from his ID_A (his public key) then so would be any other participant. Therefore we need a trusted authority *ta* which under control of its own secret key can do this computation. The concept of an identity-based asymmetric channel is as follows:

1. The participants identify themselves and register with the *ta*. This requires physical appearance at the *ta*.
2. The *ta* computes the secret key S_A from ID_A with the help of its own secret key K_{ta} and hands it back to A in a tamper-proof device. This creates a secure physical channel form the *ta* to the participant A.
3. A is now in the position to sign messages with S_A and everybody with access to ID_A is able to verify A's signatures. Equivalently, if the *ta* computes a secret decryption key D_A for A, then A is able to decrypt whereas everybody with access to ID_A is only able to encrypt messages for A.

We conclude that an identity-based system is modelled by the production

$$Auth(N) \rightarrow phys(auth, N), ta(secu, N), phys(secu, N), (Auth(N))$$

The goal of establishing asymmetric channels for each participant may be implemented by N authentic physical channels, a trusted authority which computes (and hence knows) the participants' secrets, and N secure physical channels to distribute the secret keys. Note that the apparent advantage of an identity-based system largely stems from the assumption that the ID's have been globally distributed in an authentic fashion. If this assumption is in doubt, then we are left with the security goal of authentically distributing N ID's to all the participants.

3 Conclusions

We have defined a formal language whose symbols are security goals and mechanisms. As a consequence, the design choices are made explicit as productions of the language, and completeness is achieved when all the goals are satisfied. Moreover, since one of the most important aspects of a security architecture is the resulting security management complexity, we have associated a complexity parameter to every goal and mechanism. This allows to assess which mechanisms are most effective to achieve a given goal. The presented methodology has applications as a formal design tool or as an evaluation tool. Moreover, it can be extended in various ways, for instance, to cover aspects such as key generation.

Acknowledgements

I wish to thank James L. Massey, Christian Waldvogel, Xuejia Lai, and Peter Landrock for stimulating discussions.

References

[1] D.E. Bell, L.J. LaPadula, "Secure computer system: unified exposition and Multics interpretation", Tech. Report MTR-2997, Mitre Corp., Bedford, Mass., Mar. 1976.

[2] M. Burrows, M. Abadi, R. Needham, "A logic of authentication", Tech. Report 39, DEC Systems Research Center, 1989.

[3] W. Diffie and M.E. Hellman, "Multiuser cryptographic techniques," in S. Winkler, Ed., AFIPS Conference Proceedings Vol. 45: National Computer Conference, New York, NY, June 7-10, 1976, pp. 109-112. Montvale, NJ: AFIPS Press, 1976.

[4] W. Diffie and M.E. Hellman, "New directions in cryptography," IEEE Transactions on Infomation Theory, Vol. IT-22, No. 6, November 1976, pp. 644-654. key management scheme for implementing the Data Encryption Standard", IBM Systems Journal, Vol. 17, No. 2, 1978, pp. 106-125.

[5] W. Fumy, M. Leclerc "Integrating key management protocols into the OSI architecture", Proc. of Symposium on Computer Security 90, Rome, Italy, Nov. 22-23, 1990.

[6] K. Gaarder, E. Snekkenes, "Applying a formal analysis technique to the CCITT X.509 strong two-way protocol", to appear in JCrypt. Special Issue on Applications of Cryptology.

[7] ISO 7498-2, "Information processing systems - Open Systems Interconnect - Basic reference model - Part 2: Security Architecture", First Ed. 1989.

[8] IEC/ISO JTC1/SC27 Working Draft: "Key management part 1: overview", 1990.

[9] IEC/ISO JTC1/SC27 Working Draft: "Key management part 2: key management using symmetric cryptographic techniques", 1990.

[10] IEC/ISO JTC1/SC27 Working Draft: "Key management part 3: key management using public key techniques", 1990.

[11] R. Kemmerer, "Analyzing encryption protocols using formal verification techniques", Advances in Cryptology - Proc. of Crypto'87, Springer - Verlag, 1988.

[12] L.M. Kohnfelder, "A method for certification," MIT Laboratory for Computer Science, Cambridge, MA May 1978.

[13] P. Landrock, W. Fumy, "User identification and key management using public key techniques", Proc. of Symposium on Computer Security 90, Rome, Italy, Nov. 22-23, 1990.

[14] J. McLean, "The specification and modeling of computer security", IEEE Computer, Jan. 1990.

[15] R.M. Needham and M.D. Schroeder, "Using encryption for authentication in large networks of computers, "Communications of the ACM, Vol. 21, No. 12, December 1978, pp. 993-999.

[16] R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, Vol. 21, No. 2, February 1978, pp. 120-127.

[17] R.A. Rueppel, "Key agreements based on function composition", Advances in Cryptology: Proceedings Eurocrypt'88, Springer - Verlag, 1988.

[18] R.A. Rueppel, "Security management", Proc. of Symposium on Computer Security 90, Rome. Italy, Nov. 22-23, 1990.

[19] A. Shamir, "Identity - based cryptosystems and signature schemes", Advances in Cryptology: Proceedings Crypto'84, Springer - Verlag, 1985.