PROF.DR. REINHARD POSCH
GRAZ UNIVERSITY OF TECHNOLOGY
INST. FOR APPLIED INFORMATION PROCESSING
AUSTRIA

PROHIBITING THE EXCHANGE ATTACK CALLS FOR HARDWARE SIGNATURE *B8*

Prohibiting the Exchange Attack calls for Hardware Signature

Dipl.-Ing. Wolfgang Mayerwieser [1]
o.Univ.Prof. Dipl.-Ing. Dr. Reinhard Posch [2]

Institute for Applied Information Processing,
Graz University of Technology
address: Klosterwiesgasse 32/I, A-8010 Graz, Austria
fax: (++43 316) 82 65 88 - 20

[1]phone: (++43 316) 82 65 88 - 12 – email: wmayer@iaik.tu-graz.ac.at
[2]phone: (++43 316) 82 65 88 - 10 – email: rposch@iaik.tu-graz.ac.at

Prohibiting the Exchange Attack calls for Hardware Signature

Extended abstract

Keywords: Public key cryptography, RSA and RSA-like algorithms, exchange attack, encapsulation attack, hardware signature, device authentification, device distribution protocols.

1 Introduction

Security is becomming a major concern with an increasing number of applications of information processing. Speed of cryptographic devices is one of the upcomming demands with this aspect. The nature of most mechanisms and algorithms envolved demands for intensive computing. General purpose instruction sets of CPU devices cannot be used efficiently for up to date performance of cryptographic algorithms [1],[2]. Resulting from these facts special purpose devices for cryptographic applications are in place to an increasing extent[3]. However, if special devices are implemented and distributed it is becomming a most vulnerable task as an opponent can focus his attack at the device.

This paper discusses the possibility to implement a forgery to a special purpose device with the *exchange attack* and presents a distribution scenario for a public key encryption chip which beats this attack. The discussed distribution szenario bases on an identifying secret for the device which is used as a key. With this result all the security can be kept in a startup key as an identifying secret.

2 Defining the Exchange Attack

It is generally demanded that all security of an envolved mechanism is represented by the key. Breaking a key is too much work for any computer that can be built in the forseeable future [4] is the basic assumption. On the other hand it is the practice that the functionality of an encryption device is very often kept secret. This is also done for some good reasons. Building a second security device which is basically identical to the real one but has a triggerable trojan horse [7] in it would obviously result in a perfect attack to the device.

This puts up the question if there is a security if the internal functionality of the device is specified to an extent that it might be reproduced. If we put in other word we have to ask: "How secret must an encryption mechanism be?". The answer seems to be obvious. *The overall secrecy must be to a level so that noone can build the device again — in the optimum case not even the person that has built it in the first place.* In this optimum case the device can be authentified to be original but the authentification procedure cannot be rebuilt. However, this type of security should by the basic assumption relay only on secrecy of some key.

In the situation shown in figure 1 a manufacturer M has to deliver some security module

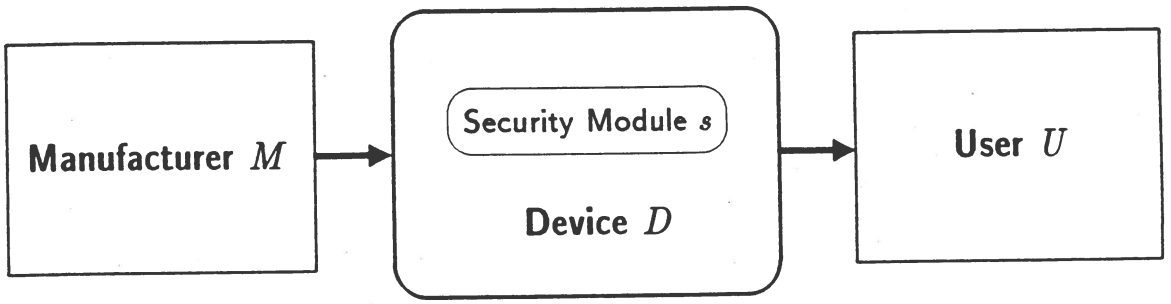

Figure 1: The distribution situation

s to the user U. This module will be incorporated into some device D by the user.

We denote the exchange attack to take place if some opponent O succeeds to introduce

a fake security module s^* into the delivery system and to let disappear the original module s. The attack is defined to be unsuccessful if this situation is detected before an attempt to use s. In this case the user is inhibited to use s but in excess to that the exchange attack could not do any harm. However, it has to be taken into account what abuse can be made with the original device s which the opponent got hold of.

In this situation the analysis attack takes potentially place. We have to distinguish between the situation where the opponent destroys the device and the situation where the device remains in function. Basically it has to be suspected with any delivery that this last analysis attack has taken place. Thus it must be trusted that the device is secure against an analysis attack that keeps the device in function. Moreover the effect of a successful attack should be kept local to a single device.

We define also a second attack, the encapsulation attack. The encapsualtion attack has taken place if it was possible to build a replacement s' for s eventually containing s and s' diverting selected requests to the device so as not to produce the identical output as s would.

This attack can mainly monitor the traffic and its contents and is thus able to disclose information upon request. This information may be present or past traffic.

This encapsulation attack puts mainly a constraint to the key loading procedure. Some papers [5] proposed to adopt hardware to prevent from this attack but this seemes not to be feasible for electronic components in the discussed environment. Therefore, it ultimatly calls the key be loaded under control. *Suspection of the encapsulation attack demands for encrypted key loading.* This situation has to be considered as the channel between the computer and the device s is to be assumed insecure.

For further discussion we exclude the situation where it is possible to physically secure the delivery with trusted personnel. This situation is trivial from the point of view of device distribution but cannot be managed with a growing number of devices.

Attacks as previously described have to be suspected all the time but especially when

maintainance is taking place.

3 Device Distribution and Authentification

We consider only security devices with publically specified functions. In this case the full specification is accessible to the opponent so as to construct s^*. Uniqueness of a device or of a group of devices is obtained by adding an identifier I, a key type information. The goal of the distribution of a security device is to embed the secret identifier I into the device before shipping and to identify (s, I) after receipt by the user or even during operation at the user's site.

At this point it has to be stated that there are protocols and mechanisms performing authenticity like the proposal by Fiat and Shamir for the secure smart cards [6]. These proposals however do not fit well into our consideration as they assume a processor to be available at the device. We rather assume that there is the encryption device but not much more around.

Obviously it is not sufficient to implement a functionality to retrieve the identifier as it would become known by the opponent performing this operation. The only way to retrieve information that is depending on the identifier is to perform some cipher $c(var)$ with var a function $var(I)$. With the same argument it can be stated that some random input has to be added to inhibit the replay attack when $c(var(I))$ is performed.

With this notation the device authentification protocol $DAP(I, V, r) \rightarrow \{true, false\}$ uses the identifier I, some random number r and compares to some verifier V.

Security devices discussed with this paper are or contain encryption chips. Thus a minimum overhead must result from the implementation of the authentification procedure DAP. This calls for similarity between DAP and the primary encryption function $c(var)$. To avoid singularities of the encryption function $c(var)$ it seems to be the ob-

vious way to use the identifier I as a key and to implement the $DAP(I,V,r)$ as an encryption-decryption cycle $DAP(I,V,r) :: D_V(E_I(r)) = r$. In this context the encryption phase $X := E_I(r)$ is performed within the security device s. The decryption phase $D_V(X)$ as well as the comparison to the random value r are performed externally to the security device.

With this concept I and V are keys. Symmetric device authentification can be defined for the case $I = V$. Authentification is defined as to be asymmetric if $I \neq V$. In both cases the identifier I is designed to remain secret and is not disclosed during the authentification procedure. This simply calls that the encryption algorithm cannot be broken by a chosen plaintext attack [8]. However, as the verifier is equal to the identifier in the case of symmetric authentification the verifier has to be kept secret to the opponent. It must be stated that storage of the tuple $(r, I(r))$ external to s and comparison of $I(r)$ after encryption of r by s - a method which is applied to storage of secret keys - is impractical. This method would at least allow the replay attack to take place. If it is just a one time authentification after shipment transmission of the device verifier after the device has arrived at the user would inhibit effective disclosure of the verifier. But, together with the availability of this mechanism one certainly wishes to use this mechanism to authentificate the security device repeatedly during operation on a regular base.

The following sections discuss handling and authentification of security devices with these two methods. The question is: "Is this problem different from the generally known authentification problem at all?" Basically it seems to be obvious that there is no major difference. However, it has to take place consistently in the smallest functional unit performing encryption.

If it took place just at the device level above, each part of the security device would have to be secured against exchange. This consideration certainly puts a strong argument for using single component security devices.

At the component level it has to be assumed that authentification mechanisms as they are known cannot always be built in. The present work is a response presenting a simpler protocol that makes a valid authentification for this special purpose.

4 Handling with symmetric ciphers

As stated in the previous section the enciphering mechanism of the security device will in general be the same as the enciphering mechanism used for the distribution process. This section discusses the case where a symmetric cipher is used to implement device distribution and operation security. The basic idea is to implement a secret as a key into the device s. After receipt of the device by the end user this secret is checked for identity with the expected secret by execution of the DAP. To achieve this situation this secret authentifying key must be communicated to the user's authentification module sc.

The goal when using the DAP protocol is to make disclosure of the secret key impossible. As symmetric keys never should be stored in clear the verifier itself must be kept secret. Usage of different data every time a device authentification is performed clearly beats the replay attack. The related mechanism consists in the following steps:

1. sc chooses a random number r and sends it to s

2. s sends $E_I(r)$

3. $DAP : r = D_V(E_I(r))$

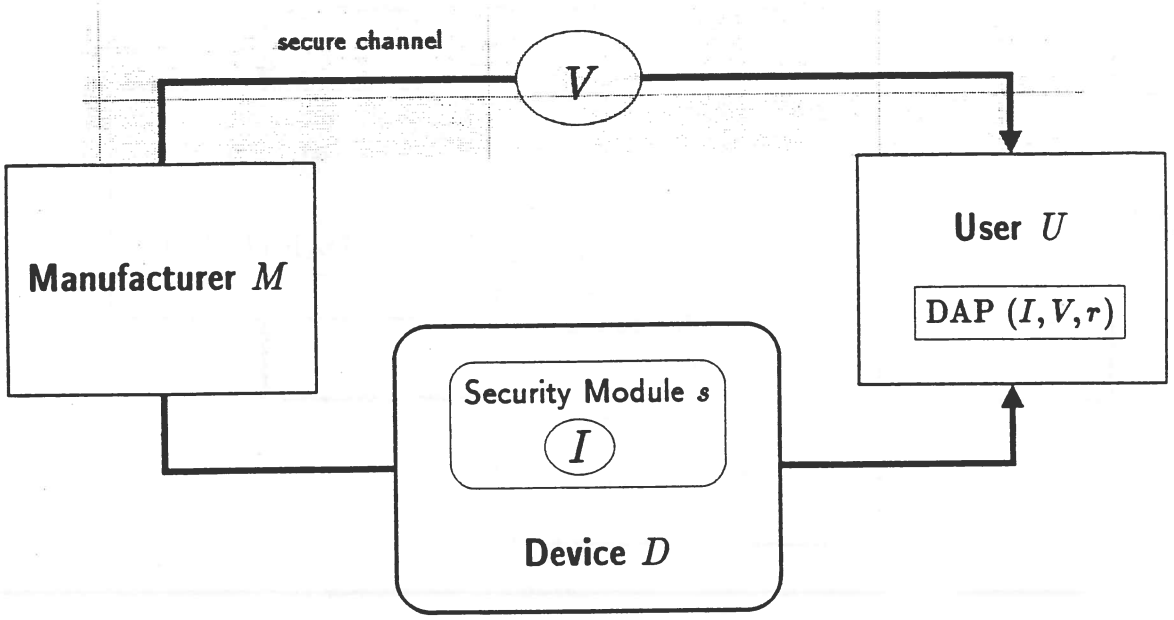

Figure 2: The symmetric cipher situation

For this procedure I is equal to V. This implies that I must be available external to s. As the authentification procedure should be available during lifetime of s an external secure controlling device sc must be installed. This device sc holds V as a secret. If either s or sc discloses the secret, authenticity is no longer guaranteed.

For the distribution itself either s or sc must obay a physically secure way not to allow the opponent the exchange with a set of devices with corresponding secrets I^* and V^* respectively. Since both are physical devices it can be stated that the distribution of security devices requires physically secure distribution in the case where symmetric ciphering is used.

5 Handling with asymmetric ciphers

This section discusses the enhancement using asymmetric ciphers when applied authentification protocols for encryption devices. In all cases authentification by the end user

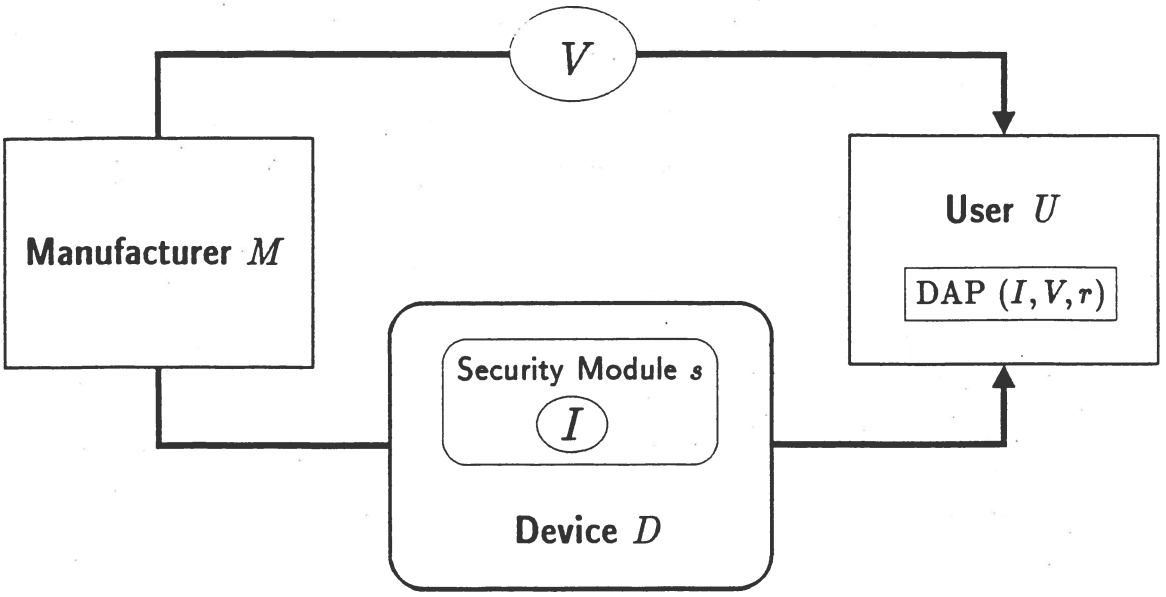

Figure 3: The asymmetric cipher situation

is considered.

Uniqueness of s is given by the hardware itself and the pair (I, V) with $I \neq V$. The verification procedure in case of the RSA encryption algorithm could look like DAP : $r = (((r^I) \bmod D)^{V^*}) \bmod D$. We would distinguish two variants:

1. A group of devices of the type s has the same secret identification I.

   This still is no limitation as for the factory to deliver to different customers with different verifiers V. Each customer could get a unique pair $V_i = (V_i^*, D_i)$. The cryptographic instructions then are designed as $LOAD(D)$ (load the D part of the key), $C = CRY(M)$ (encrypt M by the loaded key pair (k, D)), $LDKCRY(k')$ (decipher k' to yield k and load the result as key k), and $RESET$ (set the identifier I to become the key k).

   This basically results in a securely distributable and operatable device s. The main drawback is that the manufacturer has the secret I. Reverse engineering is to some extent possible as getting the identifier I from one device discloses also

secret information of other devices.

2. Each device has a unique secret I_i. In this case only nondestructive reverse engineering remains possible. Still this is only a tamper to a single device. The most secure way to implement such secure devices would be the use of battery powered secrets I_i. Such secrets can be easily implemented and an attempt to tamper with the device can to some extent be monitored resulting in either power off or shortening both being destructive to the secret. To the cryptographic instructions an instruction $LDICRY(I')$ should be added. This function is similar to the $LDKCRY(k')$ with the exception that I is loaded from the deciphered I' value. Illegal usage of this instruction must be assumed as there should not be a privilege for the usage of any cryptographic instruction. However, abuse of this instruction just means that the device is destroyed logically.
   Adding this instruction makes the original I to be only a distribution identifier and the ultimate user can replace with its company secret after delivery.

6 A Device Distribution Protocol

In the previous section the cryptographic instructions that are meant to be available have been presented. Application of these instructions is assumed in this section to give a securely distributed device. The principal goals of such a distribution would be:

1. The user can be sure that the authenticated device has been issued by the manufacturer. With respect to this point the protocol is a signature to the hardware.

2. All keys are loaded in a protected mode. A key interception is not feasible.

3. The initial key load phase authenticates the device and can be used as a 'master login' to the device.

4. No restriction is made to the key. After initialisation the device is general purpose.

5. The device has a personal security. Master access keys are different for different users.

6.1 Multi key encryption (multi RSA)

For the distribution algorithm proposed with this work a multi key asymmetric encryption is used. Assume $D = p \cdot q$ and the plaintext P. Then intermediate ciphers $C_1 \ldots C_{n-1}$ and the final ciphertext C_n are defined. The keyset then is $k_p, k_1 \ldots k_n$.
The enryption steps with this assumption are $C_1 = P^{k_p} \bmod D$, $C_{j+1} = C_j^{k_j} \bmod D$.
The decryption step is $P = C_n^{k_n} \bmod D$.
All but one keys can be quite freely chosen. However, the same measures as with a two key RSA encryption have to be observed. The last key can be computed as the multiplicative inverse of the product of the chosen keys modulo $\Phi(D)$, where $\Phi(D) = (p-1)(q-1)$.
In the special case $n = 2$, $P = (((P^{k_p} \bmod D)^{k_1} \bmod D)^{k_2} \bmod D)$, $C_1 = (P^{k_p} \bmod D)$, $C_2 = (C_1^{k_1} \bmod D)$, and $I = k_p$ are assumed.

6.2 The distribution protocol

With the goals defined and the methods given above we now introduce the following distribution protocol.

The initialisation phase is shown in figure 4. For each newly fabricated security device

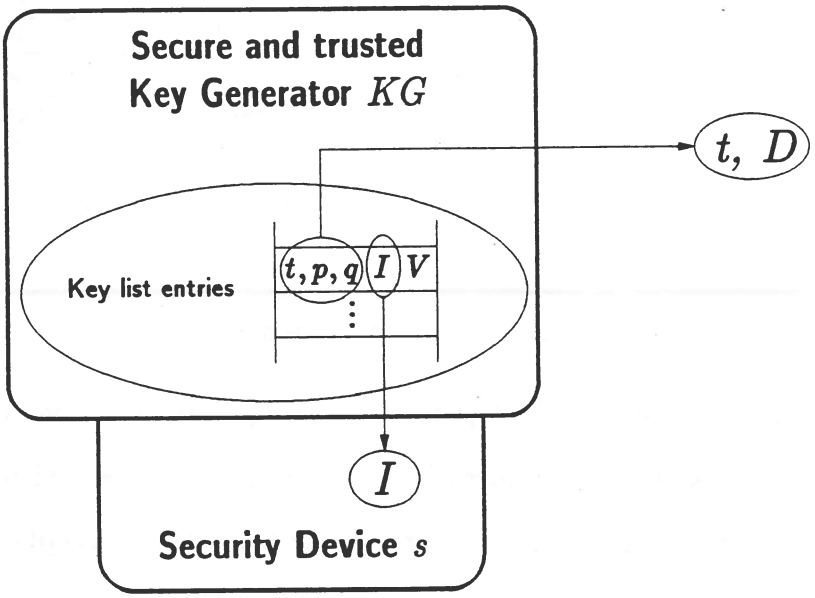

Figure 4: The initialisation phase

s a unique tag t is chosen (for example its serial number). This tag is stored in a sealed key generator together with the two primes p and q, the identifier I and the verifier V, where $I \cdot V \equiv 1 \pmod{\Phi(D)}$ and $\Phi(D) = (p-1)(q-1)$. During initialisation the security device s is passed through this key generator. While the identifier I is directly transmitted into the security device s and must not become known anywhere else, the tag t and the D part of the key ($D = p \cdot q$) are simply mailed with the chip to the recipient. It is, of course, critical for safety that the factorisation of D is kept secret inside the key generator.

Figure 5 shows the authentication scheme. It additionally provides a key which can be used for encrypted key loading.

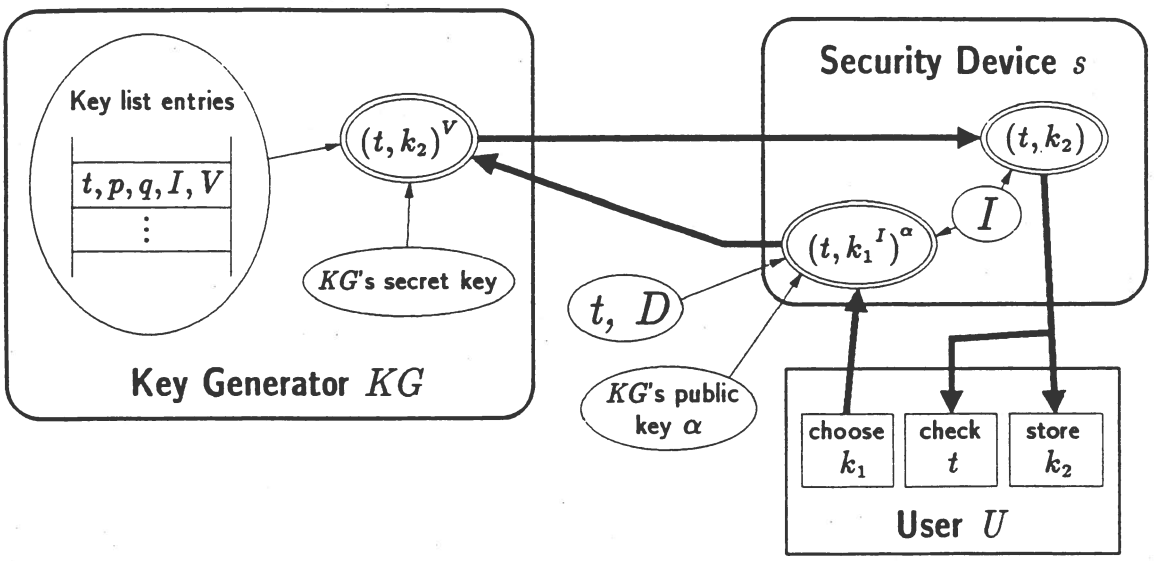

Figure 5: The authentication scheme

At the beginning, the user chooses the first part of a key, k_1. This value is encrypted by the security device - using its private identifier I - and concatenated with the tag t. The key generator uses this tag to find the correct key list entry, decrypts k_1 using the verifier V and calculates k_2, where $I \cdot k_1 \cdot k_2 \equiv 1 \pmod{\Phi(D)}$. The key generator sends the pair (t, k_2), encrypted with the verifier V. After decrypting this pair with the identifier I the user stores k_2 as the second part of the key and compares the received tag t with its own: If and only if the security device has been fed by the trusted key generator during the initialisation phase, the user receives the correct tag. This method relies on the fact that an intruder cannot both intercept and manipulate the hardware distribution and the communication between the user and the key generator. For this reason, communication from the security device to the key generator is also done in a secure way.

After completing the authentication procedure, the user might wish to destroy any information of his key list entry inside the key generator which is crucial for security (in particular p, q and I). To serve this purpose, a mechanism based on a repudiation protocol should be implemented. This procedure can be replaced using the $LDICRY(I')$.

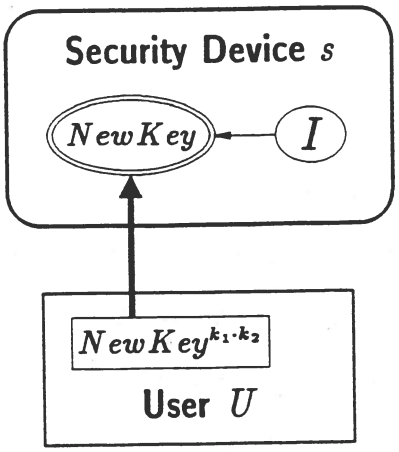

Figure 6: Encrypted key loading

With the multi encryption scheme described above, the user can perform encrypted key loading into the security device by calculating $NewKey^{k_1 \cdot k_2}$ using software on his computer, as shown in figure 6. Since $k_1 \cdot k_2$ is not reduced modulo D, this calculation contains overhead which may be ignored due to the fact that key loading is seldom done. The security device can compute $NewKey$ by decrypting $NewKey^{k_1 \cdot k_2}$ with the identifier I. Note that this method of key loading puts no constraints on the value of $NewKey$.

For ease of discussion it has not been mentioned yet that for a practical implementation of this procedure attention has to be paid to the fact that k_1 and $\Phi(D)$ must not have a common divisor. Since the user does not know the factorisation of D (and thus $\Phi(D)$), he cannot check whether this situation comes up. Therefore, if the key generator detects that $\gcd(k_1, \Phi(D)) \neq 1$ the device is marked as defect. This is the situation of guessing keys and can be ignored in practice.

7 Conclusion

The user can identify the device and the communication with the key distribution, and he has a method for loading arbitrary keys.

The proposed method gives a security domain oriented operation.

There is no further need to protect against replay attack with this procedure. Once the device is distributed the user will replace for the identifier I.

The proposed method is intended to be used for devices that are produced in a rather high quantity still giving optimum personal security.

As a general issue it can be stated that as long as the encryption speed of the device can compete with the application in view public key cryptography is highly in advantage.

Even if trusted distribution is a demand not before class A1 security this feature is much more crucial for the encryption device itself than for the whole system.

References

[1] Lippitsch P., Posch K.C., Posch R., Schindler V.: *A scalable RSA design with encryption rates from 200 Kbit/sec to 1.5 Mbit/sec*, 32nd International Science Week, Damascus, (Dec. 1992).

[2] Posch K.C., Posch R.: *Residue number system: a key to fast RSA encryption*, Proceedings of Fourth IEEE Symposium on Parallel Distributed Processing, Arlington, Texas, (Dec. 1992).

[3] Beth, Th., et al. (Eds.): *Public-Key Cryptography. State of the art and future directions.*, E.I.S.S. workshop Oberwolfach, Germany, July 3–6, 1991, Final report, (Springer, 1992).

[4] Denning, D.E.: *Cryptography and Data Security*, (Addison-Wesley, Reading MA., 1982).

[5] Brands, S., Chaum, D.: *Distance-Bounding Protocols*, Pre-Proceedings of Eurocrypt '93, Lofthus, Norway, (May 1993).

[6] Fiat A., Shamir A.: *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Proc. Advances in Cryptology - CRYPTO 86, Ed. A. Odlyzko, Lecture Notes in Computer Science 263, (Springer, 1987).

[7] Pfleger, C.P.: *Security in Computing*, (Prentice Hall, 1989).

[8] Seberry J., Pieprzyk J.: *Cryptography: An Introduction to Computer Security*, (Prentice Hall of Australia, 1989).