# Discrete Logarithm Based Protocols

Patrick Horster
Hans-Joachim Knobloch

University of Karlsruhe
European Institute for System Security
Am Fasanengarten 5
D-7500 Karlsruhe 1
FR Germany

**Abstract**

The Exponential Security System (TESS) developed at the European Institute for System Security is the result of an attempt to increase the security in heterogenous computer networks.

In this paper we present the cryptographic protocols in the kernel of TESS. We show how they can be used to implement access control, authentication, confidentiality protection, key exchange, digital signatures and distributed network security management.

We also look at the compatibility of TESS with existing standards, like the X.509 Directory Authentication Framework, and compare it to established systems like Kerberos. A comparison of TESS with the non-electronic "paper"-world of authentication and data exchange shows strong parallels.

Finally we give a short overview of the current state of development and availability of different TESS components.

## 1 Introduction

During the last years a workinggroup at the European Institute for System Security developed the network security system SELANE (SEcure Local Area Network Environment) [Baus90]. The main part of the system is a family of cryptographic protocols based on the discrete logarithm problem. After the possible scope of applications of these protocols had been extended far beyond the originally anticipated area of LAN security, a larger system called TESS (The Exponential Security System) was formed. SELANE is the part of TESS dealing with network security. Another part is an electronic signature system named EES (Exponential Electronic Signature).

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 399-408, 1991.

# 2 Protocols

## 2.1 The Discrete Logarithm Problem

One of the most important one way functions in cryptography is based on the discrete logarithm problem in the finite field GF($p$).

Given a large prime $p$ and a primitive element $\alpha \in \mathrm{GF}(p)$ it is feasible to compute the value of $y := \alpha^x$ (using $O(\log x)$ modular multiplications via the square-and-multiply algorithm). It is, however, (except for trivial cases) infeasible to compute the value of $x$, given $y$, $\alpha$ and $p$ [PoHe78]. For the solution of this discrete logarithm problem $O(\exp(const \cdot \sqrt{\log p \log\log p}))$ long integer multiplications are needed [Odly84]. Similar discrete logarithm problems can be found in finite fields GF($p^k$) of prime characteristic $p$ or in the group of points on an elliptic curve [Kobl87].

It should be noted that these are true one way functions, where no intrinsic trap door has to be kept secret, in contrast to the RSA trap door one way function [RiSA78].

For the rest of this paper we will assume that the parameters $p$ and $\alpha$ are known to all participants of the schemes mentioned.

## 2.2 The Diffie-Hellman Public Key Distribution Scheme

The Diffie-Hellman scheme [DiHe76] uses this one way property to allow two principals, let us call them $A$ and $B$, to exchange a secret key using a public channel. Each principal $i$ chooses a secret random number $x_i$ and publishes the value $y_i := \alpha^{x_i}$. Then both can compute a common key $K = y_A^{x_B} = y_B^{x_A}$.

A well-known attack on this scheme uses the fact that the authenticity of the $y_i$ is not assured. Suppose an attacker $C$ can control the communication channel between $A$ and $B$. Upon receiving $y_A$ from A he will send $y_C := \alpha^{x_C}$ to $B$ instead. Similarly he sends $y_C$ to $A$ instead of $y_B$. Using this attack he will share one key $K' = \alpha^{x_A x_C}$ with $A$ and another key $K'' = \alpha^{x_B x_C}$ with $B$ and may thus decrypt, read, modify and re-encrypt messages between $A$ and $B$. A method to detect this attack has been developed in [RiSh84].

## 2.3 The ElGamal Signature Scheme

For an ElGamal signature [Elga85] the signer chooses a random number $x \in \mathbf{Z}_{p-1}$ and computes $y := \alpha^x$. He publishes $y$ as public key and keeps $x$ secret. These values are constant for all messages to be signed. Any message to be signed must be encoded as a number $m \in \mathbf{Z}_{p-1}$, e.g. by concatenating ASCII representations of letters. To sign $m$ the signer chooses a random $k \in \mathbf{Z}_{p-1}^*$ . $k$ may never be reused to sign another message. The signer computes $r := \alpha^k$ and solves the congruence $m \equiv xr + ks \pmod{p-1}$. The triple $(m, r, s)$ is the signed message. It may be verified by checking the equation $\alpha^m = y^r r^s$.

Note that the signed message has the triple size of the message alone. Furthermore verification of the signature depends on the knowledge of the system parameters $p$ and $\alpha$ and the signer's public key $y$, which has to be authentic.

## 2.4 The modified ElGamal Signature Scheme

A modification of the ElGamal signature was presented by Agnew, Mullin and Vanstone [AgMV90]. Instead of $m \equiv xr + ks \pmod{p-1}$ the signer may solve the congruence $m \equiv xs + kr \pmod{p-1}$. (In this case $x$ must have been chosen from $\mathbf{Z}_{p-1}^*$.) The signature $(m, r, s)$ is verified by checking the equation $\alpha^m = y^s r^r$.

The advantage of this scheme over the standard ElGamal signature is, that, in order to compute the signature by solving the congruence for $s$, the signer only needs to compute $y^{-1}$ in $\mathbf{Z}_{p-1}^*$ once, instead of computing $k^{-1}$ for every signature.

## 2.5 The Testimonial Scheme

There is a variation of a signature we call a "testimonial". Whereas a signature involves a signer and a verifier, a testimonial involves three parties, called claimer, notary and verifier. For a testimonial the notary chooses $x$ and $y$, like the signer does in the signature schemes. The claimer, who wants to have the message $m$ testified, chooses $h \in \mathbf{Z}_{p-1}^*$ random, computes $a := \alpha^h$ and passes $a$ on to the notary. Now, the notary chooses a random $k \in \mathbf{Z}_{p-1}^*$ and computes $r := a^k$. Again $h$ and $k$ may never be reused for another message. The notary solves the congruence $m \equiv xr + kb \pmod{p-1}$ and passes the triple $(m, r, b)$ to the claimer. The claimer then acknowledges the receipt of the triple, e.g. by a handwritten or electronic signature. Afterwards he computes $s = bh^{-1}$ in $\mathbf{Z}_{p-1}$. The triple $(m, r, s)$ is the testified message. It may be verified by checking the equation $\alpha^m = y^r r^s$.

Note that it is possible for the notary to issue a signature $(m, r', s')$ that may be verified instead of the testimonial $(m, r, s)$. But it seems infeasible for him to compute a signature using the particular $r$ (being acknowledged by the claimer) and $s$ or to obtain $s$ if it is not published by the claimer. And obviously nobody else (besides the claimer of course) will have a better chance to fool the claimer than the notary. The claimer may prove the knowledge of his secret $s$ using a zero-knowledge protocol or use it as his secret key in a public key system.

If $s \in \mathbf{Z}_{p-1}^*$ (i.e. $\gcd(s, p-1) = 1$) reconstructing $s$ by the notary is equivalent to computing the discrete logarithm $h$ of $a = \alpha^h$, since

$$h = bs^{-1} \bmod p - 1.$$

Thus, any algorithm which allows the notary to efficiently compute $s$ from $(a, m, r, k, b)$ leads to a probabilistic algorithm for computing the discrete logarithm $h$ of $a$ by simply trying different values $m'$ and/or $k'$, until the result will be $s' \in \mathbf{Z}_{p-1}^*$. The efficiency of this algorithm depends on the chance to find a proper $s'$. Under the assumption that the values $s'$ are equidistributed over $\mathbf{Z}_{p-1}$ for appropriately chosen $m'$ and $k'$ this chance is $\phi(p-1)/(p-1)$, where $\phi$ is the Euler totient function.

On the other hand the claimer gets no more information about the notary's secrets if the testimonial scheme is used instead of a standard ElGamal signature issued by the notary. This can be proven using similar arguments as for proving the zero-knowledge property of an interactive protocol. In the testimonial scheme the notary sends the claimer

a triple $(m, r, b)$ which satisfies the modular equation

$$\alpha^m = y^r (r^{h^{-1}})^b,$$

where $h \in \mathbf{Z}_{p-1}^*$ is chosen by the claimer. As a standard ElGamal signature the notary sends the claimer a triple $(m, r, s)$ which satisfies the equation

$$\alpha^m = y^r r^s.$$

The claimer may then compute the corresponding value $b$ for an arbitrary $h \in \mathbf{Z}_{p-1}^*$ by simply multiplying $s$ with $h$, resulting in the equation

$$\alpha^m = y^r (r^{h^{-1}})^{hs}.$$

If the factorization of $p-1$ is known, the notary can easily check whether the particular $h$ was indeed chosen from $\mathbf{Z}_{p-1}^*$ by testing if $a = \alpha^h$ is a primitive element. The claimer might also use a zero-knowledge proof of membership to convince the notary about the proper choice of $h$.

## 2.6 The Beth Zero Knowledge Identification Scheme

Soon after the invention of zero knowledge proofs Chaum, Evertse and van de Graaf published a zero knowledge scheme to prove the possession of discrete logarithms [ChEG87].

Beth later presented a zero knowledge identification scheme based on the discrete logarithm problem [Beth88]. This scheme, like most other protocols that involve authentication or identification, rests on a trusted authority. Here this trusted authority is called Secure Key Issuing Authority (SKIA). A principal $A$ is characterized by an attribute list $m_A$, containing his name and other relevant data in encoded form.

The SKIA chooses $l$ independent random numbers $x_1, \ldots, x_l \in \mathbf{Z}_{p-1}$ and computes $y_j := \alpha_j^x$ for $j = 1, \ldots, l$. It publishes the $y_j$ as public keys and keeps $x_j$ secret. These values are constant for all principals in the system. Additionally the SKIA chooses and publishes a one way hashing function $f : \mathbf{Z}_{p-1} \times \mathbf{Z} \to \mathbf{Z}_{p-1}$.

When principal $A$ is registered, the SKIA computes $m'_{A,j} := f(m_A, j)$ for $j = 1, \ldots, l$ and signs these $m'_{A,j}$ with an ElGamal signature, using the key pairs $(x_j, y_j)$ but only one pair $(k, r) = (k_A, r_A)$ for all $l$ signatures. The identification data, afterwards given to $A$, is the $(l+2)$-tuple $(m_A, r_A, s_{A,1}, \ldots, s_{A,l})$.

When $A$ wants to identify himself towards another principal $B$, he sends the values $m_A$ and $r_A$ to $B$ wich can then compute the $m'_{A,j}$.

$A$ then chooses a random $t \in \mathbf{Z}_{p-1}$, computes $z := r_a^{-t}$ and sends $t$ to $B$. $B$ chooses $l$ random values $b_1, \ldots, b_l$ from a suitably chosen subset of $\mathbf{Z}_{p-1}$, and sends them to $A$. $A$ computes $u := t + \sum_{j=1}^{l} b_j s_j \bmod p-1$ and sends $u$ to $B$. $B$ accepts the identification, if $r_A^u z \prod_{j=1}^{l} y_j^{r_A b_j} = \alpha^{\sum_{j=1}^{l} b_j m'_{A,j}}$. This step of the protocol may be repeated to increase security.

Schnorr published a similar identification scheme [Schn89], which is optimized with respect to the time/computation constraints of smartcards.

## 2.7 Four KATHY Protocols

Several protocols have been developed to overcome the weakness of the Diffie-Hellman scheme that the exchanged keys are not necessarily authentic [BaKn89, Günt89]. We call these protocols KATHY protocols, K-ATH-Y standing for KeY exchange with embedded AuTHentication.

All KATHY protocols rely on a trusted authority, which is needed to assure the principals about each others identity (not necessarily during the authentic key exchange itself). Like in section 2.6 we call this trusted authority SKIA and characterize a principal $i$ by an attribute list $m_i$.

For the basic form of KATHY the SKIA signs each principal's attribute list using the ElGamal scheme, publishes $y$ and hands the signature $(m_i, r_i, s_i)$ out to the principal $i$. If principal $A$ wants to exchange an authentic key with principal $B$, $A$ passes the values $m_A$ and $r_A$ to $B$. $B$ chooses $t \in \mathbf{Z}_{p-1}$ random, computes $v := r_A^t$ and passes $v$ to $A$. $A$ can now compute the key $K_A := v^{s_A}$. $B$ can compute the same key $K_A = (\alpha^{m_A} y^{-r_A})^t$. As $A$'s attribute list $m_A$ is directly used to compute $K_A$, $B$ can be sure that the only principal sharing the key is $A$ (besides eventually the SKIA).

In a first variation of KATHY the SKIA uses the modified ElGamal scheme to sign $m_i$. The authentic key exchange is the same as for the basic KATHY, except that $B$ computes $v := y^t$ and $K_A = (\alpha^{m_A} r_A^{-r_A})^t$. This modification enables $B$ to precompute $v$ without even knowing $A$ beforehand.

In the second variation of KATHY the SKIA also uses the modified ElGamal scheme to sign $m_i$. After two principals $A$ and $B$ exchanged $(m_A, r_A)$ and $(m_B, r_B)$, they compute a common key $K_{AB} = (\alpha^{m_A} r_A^{-r_A})^{s_B} = (\alpha^{m_B} r_B^{-r_B})^{s_A}$. This key is authentic for both, however it is fixed for each pair of principals and all communication between them.

In the third variation of KATHY the SKIA uses the testimonial scheme to testify $m_i$, where principal $i$ is the claimer. After $i$ has computed $s_i$, the authentic key exchange works exactly the same way as in the basic KATHY protocol. Now the SKIA can not recompute the key $K_A$ using only the information that passes the public channel, even if it memorizes all the data about $i$ during the testimonial.

## 2.8 EES

The electronic signature package comprises the original and modified ElGamal signature schemes. Additionally, extending the possible applications, the KATHY setup may be used to generate electronic signatures.

If principal $A$ wants to sign a message $M$ he chooses $k \in \mathbf{Z}_{p-1}^*$ random and computes $t := r_A^k$. Then he solves the congruence $M \equiv s_A t + ku \pmod{p-1}$.

$(M, m_A, r_A, t, u)$ is the authentically signed message. It may be verified by checking the equation $r_A^M = (\alpha^{m_A} y^{-r_A})^t t^u$.

For the KATHY variation using the modified ElGamal scheme $t$ must be computed as $t := y^k$ and the congruence to solve is $M \equiv s_A u + kt \pmod{p-1}$. This leads to the verification equation $y^M = (\alpha^{m_A} r_A^{-r_A})^u t^t$.

Note that the verifier of such a signature only needs to be confident in the public SKIA parameters, not in the parameters of each signing principal.

### 2.9 SELANE

Only in the second KATHY variation both principals can be sure about the authenticity of the exchanged key. For network security applications, however, a different key is desired for each communication session.

The solution is to use one of the three other KATHY protocols twice (or two variations once), once for each direction of the communication. This may be done in parallel for both directions to reduce the overall running time.

This solution yields two keys $K_A$ and $K_B$, each authentic for one of the principals. A simple interlock protocol ensures the mutual key authenticity. Principal $A$ chooses a random value $z_A$, encrypts it using key $K_A$ and sends ist to $B$. $B$ does the same with a random value $z_B$ and key $K_B$. Upon reception, both principals decrypt the random value, re-encrypt it with the other of the two keys and return it. After successful decryption and comparison with the original value, $A$ can be sure that the authentic $B$, who knows key $K_B$, is also the one with whom he shares key $K_A$ and vice versa.

In the case of a cleartext communication over an insecure network, a third party might impersonate one of the communicating principals after the authentication. Therefore it is necessary to use a message authenticator algorithm or to encrypt the whole communication to keep up the authenticity of the session.

At present SELANE uses a stream cipher algorithm to encipher the communication with authentic keys. However, any other symmetric cipher, like DES, could be used instead.

## 3 Established Systems

### 3.1 X.509

The CCITT recommendation X.509 [CCIT88] describes a certificate-based system, where a certification authority (CA) authenticates registered users' public keys. To this purpose the CA issues certificates, which are essentially digital signatures of the users' names and public keys. Different CAs may mutually certify their public keys, resulting in a connected graph of CAs. The initial point of trust for a particular user is the CA which registered him. For reasons of simplicity this graph is often looked upon as a tree (with optional additional connections).

In contrast to that the KATHY protocols are in the terminology of Girault [Gira91] not certificate-based but self-certified public key systems. (Note that different versions of the KATHY protocols obtain either trust level 1 or trust level 2 as defined by Girault).

It is possible to group SKIAs hierarchically, in a way that higher level SKIAs' signatures authorize lower level SKIAs to act as such. These lower level SKIAs use the KATHY protocol with the signature scheme described in section 2.8 instead of the standard ElGamal signature. In this case knowledge of the top level SKIA's public parameters is sufficient to authenticate all principals registered at lower level SKIAs. Thus, the initial point of trust for all users is the top level SKIA.

This hierarchy tree may be mapped to the X.509 directory tree and makes it possible, if desired, to start all authentication paths at the root of the directory tree.

## 3.2 Kerberos

The Kerberos authentication scheme [MNSS87] uses a modified form of the Needham-Schroeder key distribution protocol [NeSc78]. Here we will only describe the basic ideas of the scheme. Our description may differ in some details from the actual implementation of Kerberos. These differences are, however, irrelevant for the review of Kerberos in the context of this paper. For a more complete description of the Kerberos scheme the reader is refered to the original publications.

The basis of the scheme is that every principal $u$ has a secret key $K_u$ in common with the trusted authority, which is here called *Kerberos authentication server*. $K_u$ is exchanged upon registration of the user and known only to $u$ and the authentication server.

When principal $c$ wants to communicate with principal $s$, he contacts the authentication server, denoting his name and the name of $s$. The authentication server creates a session key $K_{c,s}$. It encrypts $K_{c,s}$ using the secret key $K_s$. This encryption is called *seal* and denoted by $\{K_{c,s}\}K_s$. Then the authentication server seals $K_{c,s}$ and $\{K_{c,s}\}K_s$ using $K_c$. $\{K_{c,s}\}K_s$ is called a *ticket*. In addition to the session key a ticket contains both principal's names, addresses, a time stamp, the ticket life time and other data, mainly used to prevent replay attacks or to make the handling of the authentication system easier for the user.

The authentication server passes $\{K_{c,s}, \{K_{c,s}\}K_s\}K_c$ to $c$, who unwraps the seal with $K_c$. Then it passes the ticket on to $s$, which can decrypt $K_{c,s}$ using its own secret key. Thereafter $K_{s,c}$ is the common secret between $c$, $s$ and the authentication server.

In the standard Kerberos scheme the session key is used only to encrypt a test message, here called *authenticator*, from $c$ to $s$. The authenticator consists of the name and address of $c$ and a time stamp. Additionally the session key may be used for mutual authentication or to encrypt a session or compute a message authentication code.

Areas administered by different authorities are called *realms*. Every realm has its own authentication server. Authentication between principals of different realms is possible by registering the authentication server of one realm as a principal at the authentication server of the other realm.

The major drawback of Kerberos is the existence of the authentication server. It must be physically secure, however it must have network access as it is needed online for every authentic connection within its realm. Discreditation of the authentication server would be fatal to the whole authentication system.

It is possible to replicate the authentication server to provide fault tolerance, but this replication introduces the need for a protocol to keep the database containing the users' secret keys consistent among all instances of the authentication server. Additionally, replication increases the necessary efforts to protect the authentication server from unauthorized access.

Finally it is possible for the administrator of the authentication server to pretend any desired identity and to decipher any network communication even if the session key is used to encrypt a complete session. To obtain privacy even with respect to the Kerberos administrator, it is necessary to put an independent key distribution and authentication system on top of Kerberos.

One obvious advantage of SELANE over Kerberos is, that it does not need a trusted online authentication server and thus does not depend on the security and fault tolerance of such a server to establish authentic communication.

Another important advantage occurs if the KATHY variation with testimonials is used. The SKIA can not decrypt authentic communication, whereas the Kerberos server always can do this, as it itself generates the session keys.

### 3.3 Conventional passports

In many points the presented protocols are directly equivalent to what can be found in the "paper"-world of authentication and data exchange.

The SKIA's role is that of a passport office. To recognize an authentic passport (signature/testimonial on an attribute list) one only has to know how the imprint of an authentic seal of the passport office (public key $y$ of the SKIA, the seal itself corresponds to the private key $x$) looks like, not to ask the passport office itself (like one has to do in Kerberos).

The KATHY variation using testimonials is equivalent to the case of a passport, which is not valid until its bearer has signed it. It is possible to use different variations of the KATHY protocols for the two directions of the authentication similar to two persons presenting different types of passports for mutual inspection.

## 4 Implementation

The basis of the current implementation of TESS is a toolbox comprising arithmetic functions in GF($p$) and $\mathbf{Z}^*_{p-1}$, including tests for primality and primitivity, and some symmetric cipher systems, including DES. This toolbox is coded in C with optimized assembler parts for several popular processors. An optional hardware accelerator box for arithmetics and en-/deciphering has been developed. This box can be connected to various standard interfaces.

Support for smart cards as storage devices for secret data is available for a wide range of PCs and workstations. Additional hardware is supported that uses inductively coupled personalized tags, which may be attached e.g. to a watch, to indicate the presence of a certain user in front of his terminal/workstation.

The TESS toolbox has been used to implement the most interesting of the protocols presented in section 2.

Combining these results, a prototype of SELANE has been realized for a UNIX server with client applications on UNIX and several PC types. The possible footholds for the authentication service in UNIX are the same as for Kerberos. For easy handling the users' authentication data are stored in PIN protected smart cards. Optionally the PIN is taken from an inductively coupled tag.

Future development will address the computation of the session key within a smart card. We expect that this will be feasible with the next smart card generation which is announced to be available within 1991 from several manufacturers.

# References


[AgMV90] G. B. Agnew, R. C. Mullin, S. A. Vanstone, *Improved digital signature scheme based on discrete exponentiation*, Electronics Letters 26, 1990, pp. 1024-1025.

[BaKn89] F. Bauspieß, H.-J. Knobloch, *How to Keep Authenticity Alive in a Computer Network*, Adv. in Cryptology - EUROCRYPT '89, Springer, Berlin 1990, pp. 38-46.

[Baus90] F. Bauspieß, *SELANE - An Approach to Secure Networks*, Abstracts of SECURICOM '90, Paris 1990.

[Beth88] Th. Beth, *Zero-Knowledge Identification Scheme for Smart Cards*, Adv. in Cryptology - EUROCRYPT '88, Springer, Berlin 1988, pp. 77-84.

[CCIT88] CCITT, Recommendation X.509: *The Directory - Authentication Framework*, Blue Book - Melbourne 1988, Fascicle VIII.8: Data communication networks: directory, International Telecommunication Union, Geneva 1989, pp. 48-81.

[ChEG87] D. Chaum, J. H. Evertse, J. van de Graaf, *An Improved Protocol for Demonstrating Possession of Discrete Logarithms and some Generalizations*, Adv. in Cryptology - EUROCRYPT '87, Springer, Berlin 1988, pp. 127-141.

[DiHe76] W. Diffie, M. E. Hellman, *New Directions in Cryptography*, IEEE Trans. on Information Theory 22, 1976, pp. 644-654.

[Elga85] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Trans. on Information Theory 31, 1985, pp. 469-472.

[Gira91] M. Girault, *Self-Certified Public Keys*, Adv. in Cryptology - EUROCRYPT '91, this volume.

[Günt89] C. G. Günther, *Diffie-Hellman and El-Gamal Protocols with One Single Authentication Key*, Adv. in Cryptology - EUROCRYPT '89, Springer, Berlin 1990, pp. 29-37.

[HoKn91] P. Horster, H.-J. Knobloch, *Protocols for Secure Networks*, Abstracts of SECURICOM '91, Paris 1991.

[Kobl87] N. Koblitz, *Elliptic Curve Cryptosystems*, Math. of Computation 48, 1987, pp. 203-209.

[MNSS87] S. P. Miller, B. C. Neuman, J. I. Schiller, J. H. Saltzer, *Section E.2.1: Kerberos Authentication and Authorization System*, MIT Project Athena, Cambridge, Ma., 1987.

[NeSc78] R. M. Needham, M. D. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Comm. of the ACM 21, 1978, pp. 993-999.

[Odly84] A. M. Odlyzko, *Discrete Logarithms in Finite Fields and their Cryptographic Significance*, Adv. in Cryptology - EUROCRYPT '84, Springer, Berlin 1985, pp. 224-314.

[PoHe78] S. C. Pohlig, M. E. Hellman, *Am Improved Algorithm for Computing Logarithms Over GF(p) and its Cryptographic Significance*, IEEE Trans. on Information Theory 24, 1978, pp. 106-110.

[RiSA78] R. L. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Comm. of the ACM 21, 1978, pp. 120-126.

[RiSh84] R. L. Rivest, A. Shamir, *How to Expose an Eavesdropper*, Comm. of the ACM 27, 1984, pp. 393-395.

[Schn89] C. P. Schnorr, *Efficient Identification and Signatures for Smart Cards*, Adv. in Cryptology - CRYPTO '89, Springer, Berlin 1990, pp. 239-251.