Unique User-generated Digital Pseudonyms*

Updated: 2007/09/12

Peter Schartner and Martin Schaffer

University of Klagenfurt, Austria
Computer Science · System Security Group
{p.schartner, m.schaffer}@syssec.at

Abstract. This paper presents a method to generate unique and nevertheless highly random pseudonyms in a distributed environment. More precisely, each user can now generate his pseudonym locally in his personal security environment, e.g. in his smart card or his personal digital assistant. There is no need for any information interchange between issuing parties or global data (especially keys), except unique identifiers for each user and each device of the system. Additionally the holder can prove, that he generated a specific pseudonym without revealing his identity and he can reveal his identity by disclosing the pseudonym. Whereas the verifier of a disclosed pseudonym can be sure, that the presenter of the pseudonym is the holder of the pseudonym (i.e. the person which originally generated it). The identifier of the user and the identifier of the user's device will be used to generate unique pseudonyms, but to ensure pseudonymity, both components will be stored in the pseudonym in encrypted form.

1 Introduction

Pseudonyms (or nyms) are identifiers of subjects. The subject that may be identified by the pseudonym is the holder of the pseudonym (see [7, 9]). From the technical point of view, a pseudonym is a bit string which is

- (locally or globally) unique as identifier and
- suitable to authenticate the holder and his data (e.g. messages sent).

Most of the applications of pseudonyms have in common, that there should be no way to correlate data (of the pseudonym) stored in different applications or to link these data to the holder of the pseudonym and his identity. So another important aspect in the scope of pseudonyms is linkability, i.e. the knowledge of the relationship between the holder and his pseudonym. This linking may be known to third parties or only to the holder of the pseudonym.

Up to date, there are two ways to generate globally unique pseudonyms for a person (here called holder):

* Originally published in Springer LNCS 3685.

Centralized Generation: This approach employs a centralized third party, which generates the pseudonym on the user's behalf. This party can easily avoid duplicates and hence the generated pseudonyms are unique. On a larger scale, we may employ several hierarchically organized issuing parties. In order to guarantee the uniqueness of the pseudonyms, these issuers either generate pseudonyms in a specific (previously specified) range, or they have to check the randomly generated pseudonym with all other issuers which causes immense communication efforts. Additionally, the holder of the certificate has to trust in the issuer, since the issuer knows the linking of the holders identity to his pseudonym.

Local (Holder-based) Generation: The other way is, that the user generates his pseudonym locally. Now, only the user knows the linking between his identity and his pseudonym. But again we need some sort of cross-checking to avoid duplicates.

In the approach presented in this paper, the holder locally generates globally unique pseudonyms, which are nevertheless highly random. There is no need for any information interchange between issuing parties or global data (especially keys), except unique identifiers for each user and each device of the system. Additionally the holder can prove, that he generated a specific pseudonym without revealing his identity and he can reveal his identity by disclosing the pseudonym. This disclosure is achieved by presenting some additional, previously unknown, information to the verifier. As a security feature, this information (the opening information) cannot be forged, so that the verifier retrieves an identity different from the identity used in the generating process of the pseudonym. Another feature of the proposed system is, that there is no way to disclose the pseudonym, if the holder does not cooperate. For several application scenarios this may seem to be a major drawback (e.g. the holder of a pseudonym has just won an auction, but does not want to pay). But there are others, where there is either no need for enforced disclosure, or where the holder of the pseudonym has a strong interest in disclosing his pseudonym at a certain point of time and hence will cooperate.

The application scenarios of pseudonyms (providing pseudonymity or anonymity), where the approach presented in this paper is suitable, include:

(Centralized) Register for Medical Records: Concerning medical records, there is a strong interest in privacy, i.e. to keep the connection between a person's name and his medical record(s) private. On the other hand many countries (like Austria and Germany) run centralized databases, in order to provide data for statistical studies. To achieve this, each medical record is sent to a Server, which keeps an anonymized medical history for each person. Hence the patients have to trust in this server, because it knows the relation between the patient's identifier and his (anonymous) record identifier. If the server has been compromised and the algorithm for mapping the patients name (or social insurance number) to his record identifer is publicly known, the privacy of all patients is at risk. In contrast to this, by applying our scheme for globally unique pseudonyms, the medical records are anonymized before sending them to the server. Hence, there is no way (except breaking the encryption algorithm) to re-map a pseudonym to a user of the system.

Online Gambling: Here, the player wants to stay anonymous during gambling. He participates in the game by using his pseudonym. In case of a win, he discloses his pseudonym. Since he wants to receive his prize, he will be cooperative and will not try to forge the disclosed identity.

Online Retrieval of Information: By correlating the different areas, where a certain person retrieves information (e.g. patents or conference papers), one may conclude the research topic (and the state of the research) of this person. Applying pseudonyms (actually anonyms) here, would solve the problem.

Electronic Voting: Here, the voter may use his unique identifier received during the setup phase of the voting scheme to choose his pseudonym locally. Unfortunately, by now there is no way to prove the binding between the identifier and the pseudonym without disclosing the pseudonym. Nevertheless, this approach may be useful within closed systems, where each participant in the system is a legitimate voter.

Other applications may be in the field of online-subscription of newspapers or temporary identifiers in the scope of mobile phones or RFID (radio frequency identification).

2 Generation of Pseudonyms

The method presented in this paper is based on the idea of generating unique keys (or key components like primes) within isolated instances [3] which has been refined in [4, 5, 6, 14]. Figure 1 shows the operating principle of the original scheme. Here, we first generate unique identifiers ($EID_1||k_1$ respectively $EID_2||k_2$) by means of symmetric encryption, where $EID = \mathrm{E}_k(UID||Data||PAD)$. The proof of uniqueness will be given later on in this paper. These identifiers are concatenated with some bits (PP_1 respectively PP_2) in order to generate probabilistic primes. Finally, the primes are multiplied and the result gives the unique modulus of an RSA-Crypto-System consisting of two unique primes p_1 and p_2.

One may now directly use the unique Identifier ($EID_1||k_1$) as a pseudonym. But it is obvious, that this pseudonym does not hide any information (especially the user identifier – UID) without additional measures. Given a pseudonym of this form, the ID of the user which has been used to generate the pseudonym can be easily retrieved by decrypting the block EID_1 with key k_1. Replacing the symmetric encryption by asymmetric encryption (in this paper RSA) solves that problem. For simplicity of our notation, we will only display the public and private exponents (e and d) of the public and private keys (e, n) and (d, n). So for example $\mathrm{E}_e(m)$ represents the asymmetric encryption of message m with the public key (e, n).

More generally, a pseudonym P of an user identity (UID) is generated by use of a function f parameterized with at least two parameters: the user identity UID and a secret key k. In our approach, this function f has to be a bijective (one-to-one) one-way computation, more precisely an asymmetric encryption function, and the key k is the public key (e, n). Hence the pseudonym results in $P = f(UID, k) = \mathrm{E}_e(UID)||k = \mathrm{E}_e(UID)||e||n$.

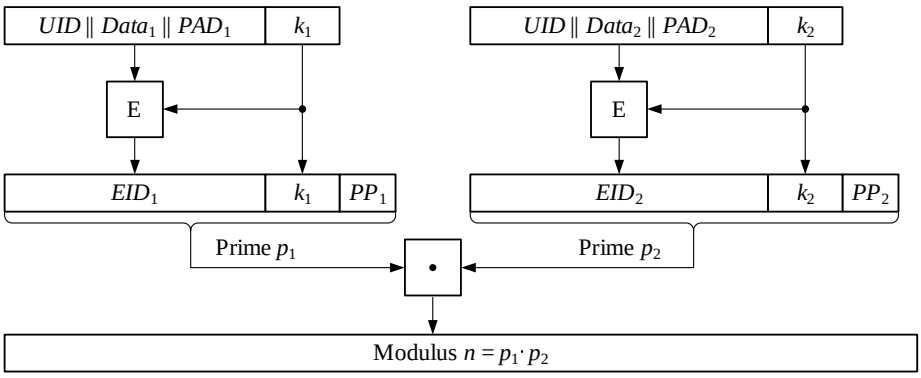

Fig. 1. Basic Idea – Generation of Unique Key Components

Since the public key (and its components) are random, two different users may accidentally choose the same key. By concatenating $\mathrm{E}_e(UID)$ and $e||n$ we ensure, that at least one of the components and hence the concatenation of the components is globally unique. For details see the proof of uniqueness given later on in this paper.

This scheme generates unique but nevertheless highly random pseudonyms in a distributed environment. More precisely, each user can now generate his pseudonym locally in his personal security environment (PSE), e.g. in his smart card or his PDA (personal digital assistant). There is no need for any global data (especially keys) or information interchange between issuing parties. The only requirement is a unique identifier (UID – user identifier) for each user of the system and a unique identifier for each PSE of the system, which may easily be managed by the use of a hierarchical issuing structure. If smart cards are used as a PSE, then the $ICCSN$ (integrated chip card serial number [10]) – a globally unique identifier which is stored in every smart card – can be used in the generating process. So we do not need to distribute or manage IDs at all.

One problem with using the $ICCSN$ is, that this number may be used during the authentication of the smart card (e.g. to derive the individual authentication key of the card) or to manage black-lists of revoked or lost smart cards. In this case, the card has to hold a user identifier, which cannot be linked to the holder of the card. Nevertheless, by now only the need for a globally unique identifier shall be emphasized, one concrete mechanism for such an identifier will be presented in section 5.

The principle to generate unique and highly random pseudonyms is quite easy (see figure 2 and algorithm 1):

1. The user (respectively his PSE) generates a key-pair for an asymmetric encryption algorithm.
   For the ease of description, we will focus on the RSA-System [11] in the remainder of this paper. Other asymmetric encryption schemes will work as well. Hence the PSE generates the modulus n, the public exponent e, and

the private exponent d. In the generation process, there is no need for the private exponent d. This parameter is only needed for later disclosure of the pseudonym.

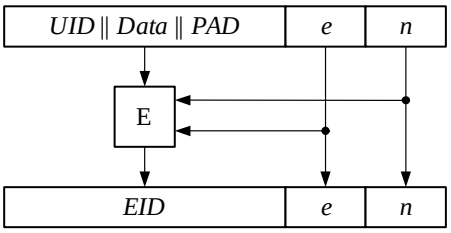

Fig. 2. Generation of a Unique Pseudonym

2. The unique identifier (UID) is concatenated with some additional data ($Data$) and some padding (PAD) and is finally encrypted with the public-key (e, n). In the remainder of this paper we will call this block holding the user identifier the UID-Block.
   The data field has to contain a serial-number of the pseudonym, which has to be incremented automatically each time a pseudonym is generated by the PSE. If a user employs different PSEs, the data-field has to store a device identifier as well. By this, we can guarantee, that different devices generate different pseudonyms.
   Additionally, the data field may contain the (unique) identifier of the Application (AID) requesting the pseudonym. By this, the user holds different pseudonyms for different applications and there is no way for correlating data of different applications. If these application-specific pseudonyms are used, the PSE has to store the pseudonym along with the AID for later usage.
3. The result of this encryption process, the so called encrypted ID (EID), is concatenated with the public-key. In case of RSA this results in $EID||e||n$, which forms the unique and highly random pseudonym $P = EID||e||n$.
   The proof of uniqueness is given in the next section. Concerning the randomness of the pseudonym, it is obvious that the second half is completely random, because we chose e, p_1 and p_2 (and hence n) at random. The first half is an encrypted block. Since the key used for encryption was chosen at random, the encryption function works as a strong pseudo-random function.

2.1 Proof of Uniqueness

The proof of uniqueness of the generated pseudonyms is straight forward and is based on the following facts:

input : UID, Data
output: P

(1) generate two random primes $p, q \in_R \mathbb{P}$
(2) generate a random public key e with $((p-1)(q-1), e) = 1$
(3) compute the private-key $d = e^{-1}$ MOD $(p-1)(q-1)$
(4) generate the pseudonym $P = \mathrm{E}_e(UID||Data||PAD)||e||n$
(5) `return` P

Alg. 1: Generation of a Unique Pseudonym

Fact 1: Each issued user identifier (UID) is unique. A hierarchical structure of the identifiers may be used, in order to simplify the management of the identifiers.

Fact 2: $\mathrm{E}_e(m_1) \neq \mathrm{E}_e(m_2) \Leftrightarrow m_1 \neq m_2$, since $\mathrm{E}_e(m)$ is a bijective (one-to-one) function for some constant public key (e, n).

To prove the uniqueness of the pseudonym generated by two different users, we have to distinguish two cases:

1. Both users (respectively their PSEs) accidentally generate (choose) the same public key (e, n). In this case, the second halves of the pseudonyms (namely $e||n$) are equal for both users. But fact 1 and fact 2 guarantee, that the first halves, namely $\mathrm{E}_e(UID_1||Data_1||PAD_1)$ and $\mathrm{E}_e(UID_2||Data_2||\ PAD_2)$, differ in at least one bit, since UID_1 and UID_2 differ in at least one bit.
2. The second case is quite easy to prove: the users generate (choose) different keys, and hence, the second halves of the generated pseudonyms (namely $e_1||n_1$ and $e_2||n_2$) differ in at least one bit. So we do not need to care about the first halves, which may be accidentally equal (different plaintexts encrypted with different keys may result in the same ciphertext). *Note:* This proof obviously holds also for symmetric encryption (see [14]).

Pseudonyms generated by a specific user may either be generated by the use of the same PSE or by use of different PSEs:

1. Pseudonyms generated by the same PSE will differ in at least one bit, because the serial numbers of the pseudonyms will differ in at least one bit.
2. Pseudonyms generated by different PSEs will differ in at least one bit, because the device identifiers of the PSEs will differ in at least one bit.

3 Proof of Ownership

One central problem of pseudonyms is to prove, that a certain pseudonym has been generated by a certain person. In principle, this can be achieved straight forward by disclosing the pseudonym. In our case, we do not want to disclose our identity, we simply want to prove, that we have generated the pseudonym.

Since only the generator of the pseudonym knows the factorization of n, only he can calculate d. The verifier who holds a pseudonym $P = \mathrm{E}_e(UID||Data||PAD)||e||n$ knows e and n and can simply run a challenge-response protocol, where the holder of the pseudonym has to prove the knowledge of d. To achieve this, the verifier encrypts some (random) challenge r with the public key (e, n) and sends $c = \mathrm{E}_e(r)$ to the prover. The prover decrypts the encrypted challenge, retrieves $r' = \mathrm{D}_d(c)$ and returns r' to the verifier. If r' matches r the verifier is convinced, that the prover has generated the pseudonym.

Since the verifier chooses the challenge, he might try to trick the prover by sending $c = \mathrm{E}_e(UID||Data||PAD)$. In this case, the prover would return the value $r' = \mathrm{D}_d(\mathrm{E}_e(UID||Data||PAD)) = UID||Data||PAD$ which would reveal his identity UID. So the prover has to dismiss the encrypted challenge c if it matches $\mathrm{E}_e(UID||Data||PAD)$.

Within our scheme of pseudonyms, two users may accidently choose the same public key (e, n) and hence the same value of d. In this case, they can obviously forge the proof of ownership of each other's pseudonym. Regarding key components of 1024 bits, this is a very rare scenario. To overcome this drawback, one may use the original scheme of generating unique key-components by use of trustworthy smartcards presented in [3] which has been refined in [4, 5, 6, 14]. By applying this scheme, all primes and hence all public and private keys will be pairwise different (see also figure 1).

4 Disclosure of Pseudonym

In order to disclose his pseudonym (and to reveal his identity), the user simply presents his private exponent d. Now, the encrypted identifier EID may be decrypted and the resulting plaintext holds the user identifier UID (see figure 3 and algorithm 2).

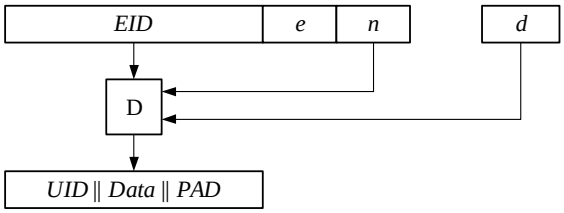

Fig. 3. Unique Pseudonyms – Disclosure

5 Forgery of Pseudonyms

Here we will investigate two attack scenarios and present solutions which prevent the following attacks:

input : pseudonym P, private exponent d
output: UID || `OK` / `NOK`

(1) retrieve EID and n form P
(2) compute $UID = \mathrm{D}_d(EID)$ *// retrieve the UID*
(3) `return` UID || `OK`

Alg. 2: Disclosure of a Unique Pseudonym (1)

1. When disclosing his pseudonym, the user sends a modified value of his private exponent d' such that $UID' = \mathrm{D}_{d'}(EID)$ and $UID' \neq UID$.
2. Another User (with identifier UID') who knows the identifier of a specific user (UID) generates a pseudonym $P = \mathrm{E}_e(UID||Data||PAD)||e||n$ in order to impersonate the user with UID.

5.1 Disclosure of a False Identity

If somebody reveals a private key d (and the primes p and q building the modulus $n = p \cdot q$) to a verifier in order to disclose his pseudonym, this private key cannot be manipulated (forged) so that the verifier retrieves an identity different from the identity used to generate the pseudonym. This is simply given by the fact, that exactly one value of d fulfills the requirement $e \cdot d \equiv 1 \pmod{\varphi(n)}$, with $\varphi(n) = (p-1)(q-1)$.

Note: This is contrary to the variant that employs symmetric encryption, where the key may be changed (attack based on a plaintext-ciphertext-pair) in order to retrieve a different identity.

The complete procedure for disclosure of a pseudonym is given in algorithm 3. The algorithm runs on the inputs P, d, p and q and returns the user identifier UID if all checks concerning the correctness of d have been passed.

input : pseudonym P, private exponent d, primes p and q
output: UID || `OK` / `NOK`

(1) retrieve EID, e and n form P
(2) `if` $(p \cdot q \neq n)$ `then` *// check the primes*
(3) `return` 0 || `NOK`
(4) `if` $(e \cdot d \not\equiv 1 \pmod{\varphi(n)})$ `then` *// check the public exponent*
(5) `return` 0 || `NOK`
(6) compute $UID = \mathrm{D}_d(EID)$ *// retrieve the UID*
(7) `return` UID || `OK`

Alg. 3: Disclosure of a Unique Pseudonym (2)

5.2 Forgery of a User's Pseudonym (Impersonation)

Another central problem of pseudonyms (presented in this paper) is, that a pseudonym which has been disclosed to a verifier may be used by the verifier to impersonate its original holder. This is possible, because after disclosure the verifier knows d, p, and q. Hence he can act like the original holder; he may use and disclose the 'stolen' pseudonym to proof the ownership, which enables him to impersonate the original holder.

A straight-forward solution for this problem is to sign the ID-Block ($UID||Data||PAD$) and to replace the original ID-Block by the new ID-Block $UID||Validity||Data||\text{SIGN}_{d_s}(UID||Validity||Data||PAD)||PAD$. Here $\text{SIGN}_{d_s}(m)$ is the signature of the hash value of m using the private signing key (d_s, n_s) with the according public verification key (e_s, n_s). The value $Validity$ holds the time of generation (i.e. the time, the pseudonym was requested by some application) and the time-to-live of the pseudonym. These values may be used to check the freshness of a presented (and disclosed) pseudonym, after the signature has been verified by use of the public key which may be retrieved from a certificate issued by a trusted certification authority. A drawback of this approach is, that only the application that requested the pseudonym is able to verify the value of $Validity$. All other applications which also use the pseudonym do not know the point in time when the pseudonym has been requested (and generated). Hence, they cannot verify the freshness.

If the freshness of the pseudonym cannot be checked by the application using the pseudonym, we need some other mechanism to avoid the reuse of a previously disclosed pseudonym. Now, the verifier of the pseudonym simply checks if the presenter (i.e. the supposed holder) of the pseudonym has generated the signature within the pseudonym. This can again be checked by verifying, that the holder knows the according private key (d_s, n_s). As above, we will employ a challenge-response protocol to accomplish this proof.

The certificates used to sign the ID-Block, show a method to retrieve a unique user identifier. This identifier may be the distinguished name of the certificate issuer concatenated with the distinguished name of the owner of the certificate. It would be more practical to replace the distinguished name of the owner of the certificate by the serial number of the certificate. Since the verifier of a disclosed pseudonym knows the issuer and the serial number, he may retrieve the according certificate and use the public key to check (as described above), if the supposed holder of the pseudonym knows the private signing key. If we use this method, there is no need to include a signature in the pseudonym. In order to verify a presented pseudonym, it is sufficient to check that the presenter of the pseudonym knows the private key belonging to the certificate holding the identifier of the pseudonym's holder.

6 Analysis of the Proposed Scheme

For security reasons (i.e. to withstand factorization) the length of a modulus (which is the product of two primes) has to be significantly larger than 512 bit

(N.B. on December 3, 2003, the 174 digit (576 bit) RSA Challenge Number has been factored [12], whereas the next challenge, a 193 digit (640 bit) number has not been factored yet [13]).

Pseudonyms may be analyzed/classified according to the following criteria:

- Involved mechanisms (e.g. symmetric/asymmetric encryption, hash-function, MAC or digital signature)
- Pre-computation (Are pre-computations possible? Which values may be pre-computed?)
- Generation efforts (Which values have to be calculated at the time, the pseudonym is generated?)
- Length of pseudonym
- Proof of ownership (with or without disclosure)
- Disclosure (local/global, Key Escrow)
- Security (forging of pseudonyms, non-repudiation)

The last three points have been discussed in previous sections of this paper. Now we would like to analyze the length of the proposed pseudonyms and the (pre-)computation efforts.

Pseudonym: $P = \mathrm{E}_e(UID||AID||PAD)||e||n$

Involved Mechanisms: asymmetric encryption (here RSA)

Pre-computations: e, d, and n.

Generation Efforts: Needs one asymmetric encryption. This may be done in advance as well, if the pseudonym does not contain any data concerning the application which requests the pseudonyms (e.g. the application identifier AID).

Length: $|P| = |n| + |e| + |n| = 2|n| + |e|$, where $|n|$ is the block-length of the cipher (which is equal to the length of the modulus n) and $|e|$ is the length of the public exponent.

A variant of the proposed scheme uses a common public exponent e for all users of the system. Hence, there is no need to include e in the pseudonym, and the modified pseudonym results in $P = \mathrm{E}_e(UID)||n$. The bit-length of this type of pseudonyms is only slightly smaller than the length above (N.B. e will be most commonly some small number, like 3, 17 or $2^{16} + 1$).

7 Resumee, Problems, Extensions and Future Research

In this paper we presented a scheme for generating digital pseudonyms, which does not apply any centralized issuers or any online-communications between issuers. The holder of the pseudonym can generate his pseudonym locally in his personal security environment (e.g. in his smart card or his personal digital assistant). The proposed method generates unique and nevertheless highly random pseudonyms in a distributed environment and with considerable computation efforts. On the one hand, the holder of a pseudonym can prove that he generated the pseudonym without disclosing it. On the other hand, the verifier of a

disclosed pseudonym can be sure, that the presenter of the pseudonym is the original holder (i.e. the person who generated it).

However, there are still some open problems and possible extensions. These questions, which are scope of ongoing research include:

Enforcement of Disclosure: One of the major drawbacks of our approach is, that the disclosure of the pseudonym is completely under control of the holder. In specific application scenarios, this is an appreciated feature. In other scenarios we would like some mechanism which ensures, that a pseudonym can be disclosed under certain (previously specified) circumstances. Escrowing of the private key (d, n) is a straight-forward solution for this problem.

Certification of the Private Exponent d: A user may certify his private exponent d (note NOT his public key which would include n) at a certification authority. So he can later on prove that a specific pseudonym belongs to him.

Other Types of Pseudonyms: Different mechanisms and different types of common components influence the properties (pre-computations, generation efforts, length of pseudonym, proof of ownership, disclosure and security) of the generated pseudonym.

Pseudonyms by means of Unique Primes: Here, we will combine the original scheme of generating unique primes and the proposed scheme for unique pseudonyms in order to overcome the drawback discussed in section 3.

Proof of Binding between ID and Pseudonym: By now, the only way to prove the binding between the ID of a user presenting a certain pseudonym, is to disclose the pseudonym.

References

1. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
2. Juho Heikkila and Ursula Holmstrm. Secure digital pseudonyms for privacy and liability, Master Thesis, Pennsylvania State University, November 15, 2002.
3. Patrick Horster. Dublettenfreie Schlüsselgenerierung durch isolierte Instanzen. *Chipkarten, DuD-Fachbeiträge, Vieweg Verlag*, 1998.
4. Patrick Horster and Peter Schartner. Bemerkungen zur Erzeugung dublettenfreier Primzahlen. *Proceedings of Sicherheitsinfrastrukturen*, 1998.
5. Patrick Horster, Peter Schartner, and Petra Wohlmacher. Key Management. *Proceedings of the IFIP TC11 14th international Information Security*, pages 37–48, 1998.
6. Patrick Horster, Peter Schartner, and Petra Wohlmacher. Special Aspects of Key Generation. *Information Technology: Science-Technique-Technology-Education-Health, Printed Scientific Works, Kharkov*, pages 345–350, 1998.
7. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199, 1999.
8. Stig Fr. Mjolsnes. Privacy, cryptographic pseudonyms, and the state of health. *Lecture Notes in Computer Science* 739, 1993.

9. Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 1–9, 2000.
10. Wolfgang Rankl and Wolfgang Effing. *Smart Card Handbook, 3rd edition*. John Wiley & Sons, 2003.
11. Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21:120–126, February 1978.
12. RSA Laboratories. RSA-576 is factored!. http://www.rsasecurity.com/rsalabs/node.asp?id=2096
13. RSA Laboratories. The RSA Challenge Numbers. http://www.rsasecurity.com/rsalabs/node.asp?id=2093
14. Peter Schartner. *Security Tokens – Basics, Applications, Management, and Infrastructures*. IT-Verlag, 2001.