PROTECTING PRIVACY IN MEDICAL DATABASES
Efficient Local Generation of System-Wide Unique Health IDs

Peter Schartner and Martin Schaffer
Institute of Applied Informatics, System Security Group
Klagenfurt University, 9020 Klagenfurt, Austria

Keywords: Privacy protection, anonymity, linkability, pseudonyms, nyms, medical registers, medical databases.

Abstract: In this paper we will introduce a replacement for linkable unique health identifiers: locally generated system-wide unique digital pseudonyms. The presented techniques are based on a novel technique called *collision free number generation* which is discussed in the introductory part of the article. After this brief introduction, we will pay attention onto two specific variants of collision-free number generation: one based on the RSA-Problem and one based on the Elliptic Curve Discrete Logarithm Problem. The main part of the article focuses on two applications of unique digital pseudonyms: centralized medical records and anonymous medical databases.

1 INTRODUCTION

In this paper we will present an application of digital pseudonyms in the scope of unique health identifiers (health IDs) and digital medical records. So far, unique health IDs are issued by some sort of centralized agency, in order to guarantee the system-wide uniqueness of the employed identifiers. This enables linking between the Health ID (and indirectly the patient's name) and the medical record. Hence, there are severe privacy concerns which are documented by the presence of "Health IDs & Privacy" in the media (Institute for Health Freedom, 2000; CNN, 2000) or even special web-sites which are protesting against such identifiers (Medical Privacy Coalition, 2007).
It is obvious that in some scenarios (e.g. electronic prescription) there must be a link between the health ID and the individual's name, but there are several scenarios, where such a link is not necessary or is even not wanted. Consider centralized medical records or anonymized databases for certain diseases (e.g. cancer registers). In these scenarios, the only requirement is that all data concerning a specific person is added to the right medical record. So, the real identity (i.e. the unique Health Identifier) can be replaced by a digital pseudonym, which has again to be unique (Pfitzmann and Köhntopp, 2001).

A critical problem is, how one can efficiently generate provably unique pseudonyms. The straightforward solution is to issue these pseudonyms by a centralized instance, which we have to trust completely. This, however, is an unrealistic assumption since there is no single instance whom one can trust. The ideal solution would be, that each individual generates the pseudonym on his own. Without cross-checking, however, there is a risk that some pseudonyms are chosen to be identical. This drawback can be overcome by using the technique of collision-free number generation for the establishment of pseudonyms. Pseudonyms are then

1. *generated locally*, i.e. derived from a system-wide unique identifier, and they are
2. *system-wide unique* without being linkable to the original identifier, from which they are derived.

The remainder of this paper is organized as follows. After going through some preliminaries we will briefly describe a scheme, that allows the individual to locally generate almost random numbers, which are globally (in particular system-wide) unique (Schartner and Schaffer, 2005; Schaffer et al., 2007). These numbers can be used as pseudonyms and hence they can be used to replace the original health ID. The individual's name and the corresponding ID (thus the medical record) are then computationally unlinkable. After describing the original scheme with two practical implementations, we will present two application scenarios. In the first, we will discuss centralized databases which store a medical record for each individual in order to improve the quality and security of ongoing and later treatments. In the second, we will discuss databases for medical records of specific diseases which are used to statistically investigate them.

2 PRELIMINARIES

Henceforth, l_x denotes the bit-length of x.

AES Encryption (NIST, 2001). Let k be a random secret AES-key. The AES-encryption of a binary message m is given through

$$\mathrm{AES}_k(m) = c$$

where c is the computed ciphertext. For simplicity, the decryption process is denoted similarly:

$$\mathrm{AES}_k^{-1}(c) = m$$

It is assumed that if the message is large, the encryption is carried out by using and appropriate mode of operation with a standardized padding scheme.

RSA Encryption (Rivest et al., 1978). Let n be the product of two safe (or strong) primes p and q. The public key e is chosen at random (or system-wide static), such that $\gcd(e, \varphi(n)) = 1$. The corresponding private key d is computed such that $ed \equiv 1 \pmod{\varphi(n)}$ holds. Given (e,n), a message $m \in \mathbb{Z}_n$ is encrypted through the function $\mathrm{RSA}_{(e,n)}$, defined as follows:

$$\mathrm{RSA}_{(e,n)}(m) = m^e \text{ MOD } n$$

A ciphertext $c \in \mathbb{Z}_n$ can be decrypted through the function $\mathrm{RSA}_{(d,n)}^{-1}$ defined as follows:

$$\mathrm{RSA}_{(d,n)}^{-1}(c) = c^d \text{ MOD } n$$

The security of the scheme relies on the problem of factoring n and computing e-th roots modulo n. The problem of finding m, given c and (e,n) is sometimes called the RSA-Problem and defined as follows:

Definition 2.1 Let $n = pq$, where p and q are primes, $e \in \mathbb{Z}_{\varphi(n)}^*$, $m \in \mathbb{Z}_n$ and $c = m^e$ MOD n. The *RSA-Problem* is the following: given c and (e,n), find m.

Like all state-of-the-art implementations of RSA we employ optimal asymmetric encryption padding OAEP (Jonsson and Kaliski, 2002), which ist as special form of padding, securing RSA against chosen ciphertext attacks.

Elliptic Curve Cryptography (ECC). To speed up computations, discrete logarithm-based cryptosystems are often run over an elliptic curve group, or in particular on a subgroup of prime order. A good introduction to ECC can be found in (D. Hankerson, 2004). We use scalar multiplication in such groups as a one-way function. The corresponding intractable problem is defined in the following:

Definition 2.2 Let $E(\mathbb{Z}_p)$ be an elliptic curve group, where p is an odd prime. Let $P \in E(\mathbb{Z}_p)$ be a point of prime order q, where $q | \#E(\mathbb{Z}_p)$. The *Elliptic Curve Discrete Logarithm Problem (ECDLP)* is the following: given a (random) point $Q \in \langle P \rangle$ and P, find $k \in \mathbb{Z}_q$ such that $Q = kP$.

To avoid confusion with ordinary multiplication we henceforth write $\mathrm{SM}(n,P)$ to express the scalar multiplication nP in $E(\mathbb{Z}_p)$. It is currently believed that the ECDLP using $l_p \approx 192$ and $l_q \approx 180$ is secure against powerful attacks like Pollard's rho algorithm (D. Hankerson, 2004).

ECC Point Compression (IEEE, 2000). A point on an elliptic curve consists of two coordinates and so requires $2l_p$ bits of space. It is a fact that for every x-value at most two possible y-values exist. Since they only differ in the algebraic sign, it suffices to store only one bit instead of the whole y-value. We therefor define the point compression function $\mathrm{CP} : E(\mathbb{Z}_p) \to \mathbb{Z}_p \times \{0,1\}$ as follows:

$$\mathrm{CP}((x,y)) = (x, y \text{ MOD } 2)$$

Storing $(x,\tilde{y})$ instead of (x,y) requires only $l_p + 1$ bits of space. To uniquely recover (x,y) from $(x,\tilde{y}) \in \mathbb{Z}_p \times \{0,1\}$ one needs a decompression function $\mathrm{DP} : \mathbb{Z}_p \times \{0,1\} \to E(\mathbb{Z}_p)$, such that

$$\mathrm{DP}((x,\tilde{y})) = (x,y)$$

How y is uniquely recovered depends on the kind of elliptic curve that is used and on how p is chosen. For instance, consider an elliptic curve defined through the equation $y^2 \equiv x^3 + ax + b \pmod{p}$, where a, b are some appropriate public domain parameters. If $p \equiv 3 \pmod{4}$, one can recover y form $(y,\tilde{y})$ through one modular exponentiation:

$$y_{0,1} \equiv \pm(x^3 + ax + b)^{(p+1)/4} \pmod{p}$$

where $y = y_{\tilde{y}}$. This is a very efficient way to compute square roots modulo a prime and is also used in the Rabin cryptosystem.

ElGamal Encryption (ElGamal, 1985). In this paper, we use a particular variant of the ElGamal encryption scheme. Let $E(\mathbb{Z}_p)$ be the elliptic curve group as described above and P a point of order

q. Then $d \in_R \mathbb{Z}_q$ denotes the private key whereas $Q = SM(d,P)$ denotes the corresponding public key. If done straightforwardly according to (ElGamal, 1985), a message $M \in E(\mathbb{Z}_p)$ can be encrypted by first choosing a random $r \in \mathbb{Z}_q$ and then computing the ciphertext-pair $(A,B) = (SM(r,P), M + SM(r,Q))$. Given (A,B) and d, M can be obtained through $B + SM(-d,A)$. However, it is not trivial to map a binary message m to a point M on an elliptic curve. An efficient method to overcome this drawback is to choose a cryptographic hash-function $\mathcal{H} : E(\mathbb{Z}_p) \rightarrow \{0,1\}^{l_m}$, where l_m denotes the bit-length of the binary message m, and define the ElGamal encryption function ElG_Q:

$$\mathrm{ElG}_Q(m) = (SM(r,P), m \oplus \mathcal{H}(SM(r,Q))), \quad r \in_R \mathbb{Z}_q$$

Given the ciphertext-pair $(A,B) \in E(\mathbb{Z}_p) \times \{0,1\}^{l_m}$ and d, m can be obtained through ElG_d^{-1}:

$$\mathrm{ElG}_d^{-1}((A,B)) = \mathcal{H}(SM(-d,A)) \oplus B$$

The correctness of this ElGamal variant is obvious.

3 COLLISION-FREE NUMBER GENERATION

In (Schaffer et al., 2007) we proposed a general method for generating system-wide unique numbers in a local environment, whilst preserving the privacy of the generating individual. The described generator, called collision-free number generator (CFNG), fulfils the following requirements:

R1 (Uniqueness). A locally generated number is system-wide unique for a certain time-interval.

R2 (Efficiency). The generation process is efficient regarding communication, time and space.

R3 (Privacy). Here we distinguish two cases:

1. *Hiding*: Given a generated number, a poly-bounded algorithm is not able to efficiently identify the corresponding generator.
2. *Unlinkability*: Given a set of generated numbers, a poly-bounded algorithm is not able to efficiently decide which of them have been generated by the same generator.

In the following subsection we summarize two general principles of collision-free number generation, formerly introduced in (Schaffer et al., 2007).

3.1 General Construction

For efficiency reasons, an identifier-based approach is used. Every generator is (once) initialized with a system-wide unique identifier, which we denote by UI. The idea is, to derive several unique numbers from UI, such that none of them is linkable to UI.

Uniqueness Generation. As a first step a routine is needed, which derives a unique number u from UI in every run of the generation process. We call such a routine *uniqueness generator* and denote it by UG. UG can be designed as follows:

$$u = \mathrm{UG}(), \quad u = UI||cnt||pad$$

where pad is a suitable padding for later use of u and cnt is an l_{cnt}-bit counter initialized by $cnt \in_R \{0,1\}^{l_{cnt}}$ and incremented modulo $2^{l_{cnt}}$ in every round. So, the output of UG is unique for at least $2^{l_{cnt}} - 1$ rounds. So far, the privacy is not preserved, since UI is accessible through u.

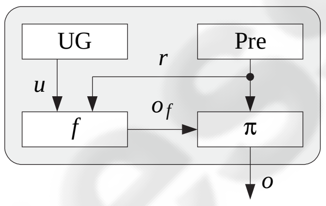

Figure 1: Collision-Free Number Generator CFNG1.

Uniqueness Randomization. A first step to provide privacy is to transform u such that the resulting block looks random. Hereby, we use an injective function f_r, where r is chosen from a set R. The idea is that u is randomized by r and hence we call f_r the *uniqueness randomization function*. We suggest to either use an injective mixing-transformation for f_r (e.g. symmetric encryption) or an injective one-way function based on an intractable problem (e.g. the Discrete Logarithm Problem). In both cases, r is chosen at random. The output of f_r is obviously not guaranteed to be unique: let u, u', $u \neq u'$ and r, r' random, where $r \neq r'$. Then $f_r(u) = f_{r'}(u')$ may hold since two different injective functions f_r and $f_{r'}$ map on the same output space. On the other hand, this problem cannot happen if $r = r'$, since f_r is injective. To generate a unique block o, sufficient information about the chosen function f_r has to be attached to its output. Hence, o can be defined as $o = f_r(u)||r$. The concatenation of two blocks by writing them in a row is an unnecessary restriction. The bits of the two blocks can be concatenated in any way. This leads to the following construction (cf. Figure 1):

$$o = \pi(f_r(u), r), \quad u = \mathrm{UG}(), \quad r = \mathrm{Pre}()$$

where π is a (static) bit-permutation function (or bit-permuted expansion function) over the block $f_r(u)||r$

and the generation of r is done by a routine called *pre-processor*, denoted by Pre. The main-task of Pre is the correct selection of r. This can include a key generation process if f_r is an encryption function.

Theorem 3.1. *Let u be a system-wide unique number, f_r be an injective function and $r \in R$. Furthermore, let π be a static bit-permutation function or bit-permuted expansion function. Then $o = \pi(f_r(u), r)$ is system-wide unique, for all $r \in R$.*

Proof. Let $o' = \pi(f_{r'}(u'), r')$ and $u \neq u'$. The case where $r \neq r'$ obviously guarantees that the pairs $(f_r(u), r)$ and $(f_{r'}(u'), r')$ are distinct. Now consider the case where $r = r'$. Since $u \neq u'$ holds per assumption $f_r(u) \neq f_{r'}(u')$ holds due to the injectivity of f_r and the fact that $r = r'$. Thus, we have $(f_r(u), r) \neq (f_{r'}(u'), r')$ for all $r, r' \in R$. It remains to show that $o \neq o'$. This is obviously the case, because π is static and injective. □

Privacy Protection. Given $o = \pi(f_r(u), r)$, computing $\pi^{-1}(o) = (a, b)$ is easy since π is public. Concerning f, two cases might occur concerning the obtainable information about u and hence about UI:

1. Computing $f_b^{-1}(a) = u$ is computationally hard. Then we are done, since obtaining u is hard.
2. Computing $f_b^{-1}(a) = u$ is easy. Then an extension of the basic construction is necessary, which is sketched in the following.

To keep finding u hard, we suggest using an injective one-way function g to hide $\pi(f_r(u), r)$. We call g the *privacy protection function*. The new output o is defined as follows (cf. Figure 2):

$$o = g(\pi(f_r(u), r)), \quad u = \text{UG}(), \quad r = \text{Pre}()$$

The extended construction still outputs unique numbers, which is shown by the following corollary.

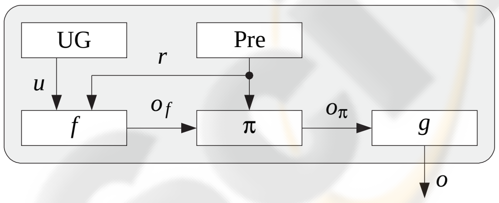

Figure 2: Collision-Free Number Generator CFNG2.

Corollary 3.1. *Let u be a system-wide unique number, f_r be an injective function and $r \in R$. Furthermore, let π be a static bit-permutation function or bit-permuted expansion function and g an injective one-way function. Then $o = g(\pi(f_r(u), r))$ is system-wide unique, for all $r \in R$.*

Proof. Let $o' = g(\pi(f_{r'}(u'), r'))$ with $u \neq u'$. By Theorem 3.1 $\pi(f_r(u), r) \neq \pi(f_{r'}(u'), r')$. Since g is injective, $o \neq o'$ for all $r, r' \in R$. □

3.2 Practical Approaches

In the following two examples are given, how CFNG1 and CFNG2 can be implemented for the generation of system-wide unique pseudonyms. The first one is based on the RSA-Problem and the second is based on AES and the ECDLP. For both variants, we will only discuss the principle operation, but no details (e.g. bit lengths) are given.
In order to guarantee the proper generation of the pseudonyms we assume that each individual has been provided with a system-wide unique identifier UI (which could be the original health identifier or the ICCSN () of a smartcard). Henceforth, we assume that all computations are done in a smartcard.

3.2.1 RSA-based CFNG1

In order to generate a system-wide unique pseudonym N, the smartcard generates an RSA key pair consisting of the public exponent e, the random private exponent d and the random modulus n (consisting of two appropriately chosen large prime factors). Now, the smartcard calculates the pseudonym N as follows:

$$N = \text{RSA}_{(e,n)}(u)||(e,n)$$

Setting $f = \text{RSA}$, $r = (e, n)$ and $\pi = ||$, Theorem 3.1 can be applied and thus N is system-wide unique. Moreover, the encryption process is hard to invert, given only N. Hence, obtaining u from N is infeasible for a poly-bounded algorithm which leads to a sufficient privacy-protection.

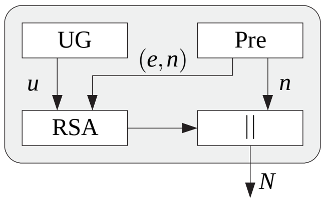

Figure 3: RSA-based Pseudonym Generation.

To keep the bit-length of N as short as possible, every user might use the same public key e. In this case, there is no need to embed e in N:

$$N = \text{RSA}_{(e,n)}(u)||n$$

Figure 3 shows the principle design of the RSA-based CFNG. Here Pre denotes a key generator, which generates the key (e, n) and padding material for the encryption process. The system parameters includes the bit-length of n and of the padding.

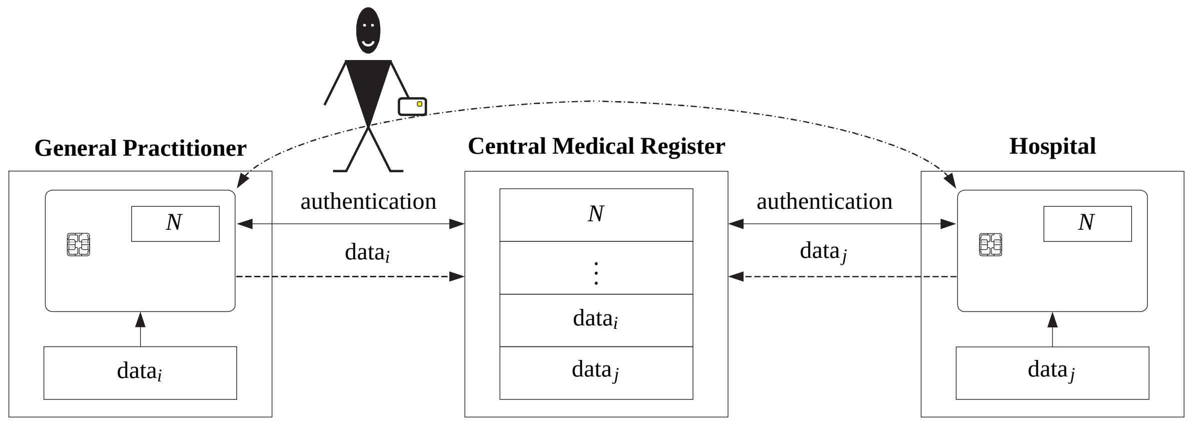

Figure 4: Centralized Medical Registers.

3.2.2 ECC-based CFNG2

In this variant, we employ a symmetric encryption algorithm to conceal the identity of the patient. As above, the randomly chosen key used for this encryption process (this time k) has to be concatenated to the output the ciphertext. However, this construction is not sufficient for privacy protection, since one can easily invert the encryption process (it is symmetric). According to the design of CFNG2, one has to choose a privacy protection function g. For efficiency, we use scalar multiplication in an elliptic curve group as a one-way function. The pseudonym N is defined as:

$$N = \mathrm{SM}(\mathrm{AES}_k(u)||k, P)$$

Figure 5 shows the principle design of the ECC-based generation process. Here, the system parameters include the system-wide constant point P, and the bit lengths of the primes p and q and the key k.

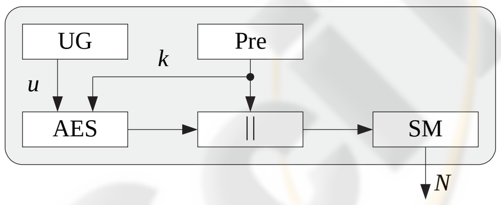

Figure 5: ECC-based Pseudonym Generation.

Setting $f = AES$, $r = k$ and $g = SM$, Corollary 3.1 can be applied and system-wide uniqueness is achieved. Notice that $AES_k(u)||k$ is highly random and hence inverting the ECC point multiplication process is hard due to the ECDLP. Thus, the requirement for privacy is fulfilled sufficiently for practical use.

4 UNIQUE PSEUDONYMS IN MEDICAL REGISTERS

4.1 Centralized Medical Records

In the first scenario, we employ a centralized medical database which stores a medical record for each individual. From the individual's point of view it is essential, that

- there is no link between the medical record and the real identity,
- no unauthorized person gets read or write access to the medical record, and
- additional data which is stored in the course of time is accumulated in the correct medical record.

4.1.1 Initialization

In this application we will employ smartcards which are equipped with an ECC-based CFNG2 for efficiency reasons. Hence, the digital pseudonym is of the form $N = \mathrm{CP}(\mathrm{SM}(AES_k(u)||k, P))$, where $u = UI||cnt||pad$ (cf. Section 3.1) and UI the patient's identifier. Notice that we apply the point compression function CP to make N as short as possible. Whenever additional medical data has to be stored in the medical record, this pseudonym will unambiguously identify the record belonging to the holder of the pseudonym.

4.1.2 Data Storage

Prior to adding some data to his medical record, the patient has to prove that he owns the necessary rights. To protect the privacy of the patient, the authentication process (cf. Figure 4) must not include secret information on server side (this excludes symmetric encryption based authentication). An idea is to consider

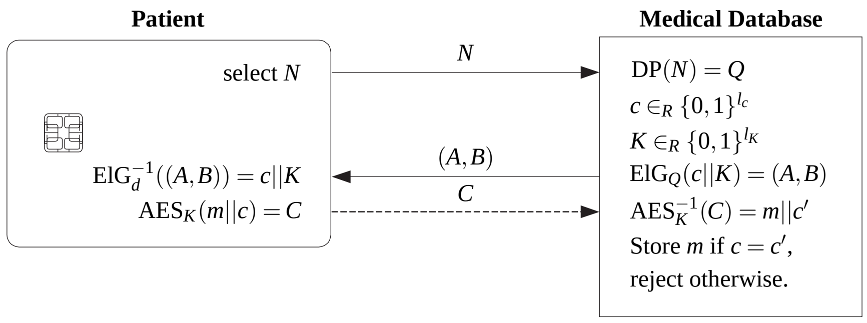

Figure 6: Authentication and Data Storage.

$Q = (x,y)$, where $N = CP(Q)$, as the public key of the ElGamal encryption scheme described in Section 2. The corresponding private key is $d = AES_k(u)||k$, since $N = CP(Q)$ and $Q = \mathrm{SM}(AES_k(u)||k,P)$. This enables an indirect asymmetric authentication and key-exchange protocol, that provides anonymity of the patient, confidentiality of the sent medical data and freshness of the sent messages (cf. Figure 6):

1. To start the upload of medical data, the patient's smartcard contacts the server that stores the medical records, and sends his pseudonym N.
2. The server generates a random challenge c of length l_c and a session key K of length l_K, such that $l_c + l_K = l_m$ and l_K with respect to the involved symmetric encryption algorithm (here AES). Then he encrypts $c||K$ with Q using ElG_Q as described in Section 2. The resulting ciphertext-pair (A,B) is returned to the smartcard.
3. The smartcard decrypts (A,B) by using ElG_d^{-1} with the private key d and retrieves c and K. The medical data m (extended by the server's challenge) is encrypted to $C = \mathrm{AES}_K(m||c)$ and sent to the server.
4. The server decrypts the encrypted message, compares the received challenge to the sent one and updates the medical record if they are identical. Otherwise he rejects the update request.

The authentication process only succeeds, if the patient knows d, such that $Q = \mathrm{SM}(d,P)$. The probability of computing a correct C (that contains the same challenge as generated by the server) without knowing d is negligible.

4.1.3 Data Retrieval

In principle, the data retrieval process is similar to the data storage process. At the beginning the patient needs to authentication himself to the server. Thereby, the server generates a session key. The data retrieval message is encrypted with the session key and provides the server with some data to identify the requested section of the medical record. The server selects this section, encrypts it with the session key and returns it to the patient's smartcard. Here, the message is decrypted and the medical data can be processed by the specific application.

4.1.4 An Extension

So far, the pseudonym is of the form $N = \mathrm{CP}(\mathrm{SM}(\mathrm{AES}_k(u)||k,P))$, and provides full access to the medical record. The medical record could be split into several (unlinkable) parts, if we append SID_i (the identifier of section i) to UI and generate

$$N_i = \mathrm{CP}(\mathrm{SM}(\mathrm{AES}_{k_i}(u_i)||k_i,P))$$

where u_i contains $UI||SID_i$. A pseudonym N_i will now identify and grant access to a specific section of the medical record, which enables access control and unlinkability on a finer level.

4.1.5 Security & Efficiency

The pseudonym is generated by a collision-free number generator that protects the privacy of the generating instance. In the current context this means that computing UI from a pseudonym is computationally hard, i.e. requires solving the ECDLP. The generation process is very efficient since only one symmetric encryption and one scalar multiplication is necessary.
The authentication process reveals no information about UI. The user can only respond correctly (apart from some negligible cheating probability) to the challenge, if he knows the pre-image of the pseudonym. The authentication process only requires the exchanging of three messages between server and smartcard. The smartcard only performs one scalar multiplication, one run of a hash algorithm and some symmetric encryption. The server performs one run of the point decompression function DP which equals

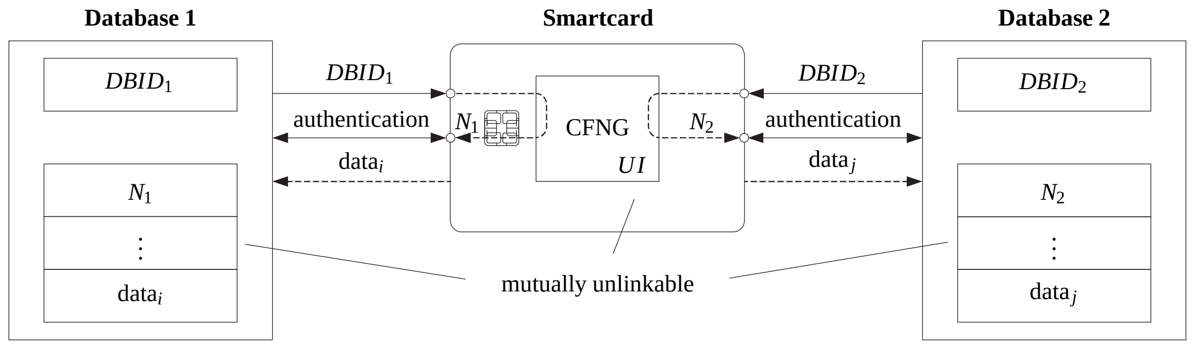

Figure 7: Anonymous Medical Databases.

to one modular exponentiation if $p \equiv 3 \pmod 4$. Furthermore, two scalar multiplications are done and some symmetric decryption. All computations only require minimal temporary space.

4.2 Anonymous Medical Databases

Centralized medical databases are quite commonly used to provide some anonymized data for statistical investigations of certain diseases like cancer, influenza, or tuberculosis. Here, data like date of diagnosis, applied treatment, medication, chronology, and mortality are of special interest. The privacy problem with these medical databases is how to provide the anonymity of the patient. Of course, the server storing the database may be trusted, but especially in the scope of medical records, the sensitivity for privacy-endangering technologies is very high. So the best method is to remove the identifying information as soon as possible. Unfortunately, the medical data concerning the disease of a specific patient is most commonly generated over a larger period of time and hence we need some identifier to link the separate data records. Here again, we can use collision-free number generators. The authentication of the data storing party (cf. section 4.1.2) will be moved to the application level. The smartcard of the patient provides only the unique identifier of the medical record, which now depends on the patient's identifier (UI) and the identifier of the database (*DBID*).

4.2.1 Initialization

The smartcard issuer has to generate appropriate system parameters, which are stored in the smartcard. Every time a new medical database is connected (cf. Figure 7), the smartcard receives the database identifier *DBID* and generates the corresponding pseudonym. This pseudonym will be stored in the smartcard for later usage.

4.2.2 Data Storage

During the authentication the smartcard retrieves the session key from the database server. In order to store data, all personal data of the current medical record is removed and the remaining data is encrypted by use of the session key (established during the authentication process). The resulting ciphertext is sent to the database server. The message is decrypted and the medical data is stored in the record identified by the pseudonym.

4.2.3 Data Retrieval

This application scenario does not provide data retrieval for the individual patient. Only medical institutions may retrieve data from the centralized medical register. The process to achieve this in an authentic and confidential goes beyond the scope of this paper.

4.2.4 Security & Efficiency

Security and efficiency can be considered analogously to the previous application scenario. Unlinkability is again achieved through the privacy protection that holds through the design of the used ECC-based collision-free number generator.

5 CONCLUSIONS

In this paper, we introduced a replacement for linkable unique health identifiers: locally generated but nevertheless system-wide unique digital pseudonyms. To achieve this replacement, we used so called collision-free number generators, which have been briefly discussed in den introductory part of this article. Here we presented two variants: one based on the RSA-Problem, the other based on the Elliptic Curve Discrete Logarithm Problem. Both variants fulfill the requirements uniqueness of the generated pseudonyms, privacy in terms of hiding the originators identifier and unlinkability of individual data sets. Since the second variant is more efficient in terms of computational costs and space it has been suggested for the use within smartcards.

Beside proposing two practical generators, a protocol has been proposed through which a smartcard can efficiently authenticate anonymously to a medical database (with respect to a pseudonym). Based on this protocol medical registers can be extended and updated anonymously by the patient. Furthermore such an authentication protocol can be used to make entries in several databases. For each database, a fresh pseudonym is used so that entries of different databases are mutually computationally unlinkable.

Concerning collision-free number generation further information on implementation issues and extended constructions can be found in (Schaffer and Schartner, 2007) and (Schaffer, 2007) respectively.

REFERENCES

CNN (2000). National Health Identifier: Big Help or Big Brother? http://www.cnn.com/HEALTH/bioethics/9807/natl.medical.id.

D. Hankerson, A. Menezes, S. V. (2004). *Guide to Elliptic Curve Cryptography*. Springer.

ElGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In Blakley, G. R. and Chaum, D., editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer.

IEEE (2000). *IEEE 1363-2000: IEEE Standard Specifications for Public-Key Cryptography*. IEEE.

Institute for Health Freedom (2000). What's Happening with the "Unique Health Identifier" Plan? http://www.forhealthfreedom.org/Publications/privacy/UniqueId.html.

Jonsson, J. and Kaliski, B. (2002). Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specification. http://www.rsa.com/rsalabs/node.asp?id=2125.

Medical Privacy Coalition (2007). Eliminate Unique Health Identifier. http://www.medicalprivacycoalition.org/unique-health-identifier.

NIST (2001). FIPS PUB 197: Specification of the Advanced Encryption Standard (National Institute of Standards and Technology). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

Pfitzmann, A. and Köhntopp, M. (2001). Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology. In Federrath, H., editor, *Proceedings of Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9. Springer.

Rivest, R. L., Shamir, A., and Adleman, L. M. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126.

Schaffer, M. (2007). *Collision-Free Number Generation: Efficienty Constructions, Privacy Issues, and Cryptographic Aspects*. PhD-Thesis, Klagenfurt University.

Schaffer, M. and Schartner, P. (2007). Implementing Collision-Free Number Generators on JavaCards. Technical Report TR-syssec-07-03, University of Klagenfurt.

Schaffer, M., Schartner, P., and Rass, S. (2007). Universally Unique Identifiers: How to ensure Uniqueness while Preserving the Issuer's Privacy. In Alissi, S. and Arabnia, H. R., editors, *Proceedings of the 2007 International Conference on Security & Management – SAM'07*, pages 198–204. CSREA Press.

Schartner, P. and Schaffer, M. (2005). Unique User-Generated Digital Pseudonyms. In Gorodetsky, V., Kotenko, I. V., and Skormin, V. A., editors, *Proceedings of Mathematical Methods, Models, and Architectures for Computer Network Security – MMM-ACNS 2005*, volume 3685 of *Lecture Notes in Computer Science*, pages 194–205. Springer.