Decentralized Generation of Multiple, Uncorrelatable Pseudonyms without Trusted Third Parties

Jan Lehnhardt[1] and Adrian Spalka[1,2]

[1] CompuGroup Medical Software GmbH, Dept. of Data Security and System Architecture, Maria Trost 23, 56070 Koblenz, Germany
{jan.lehnhardt,adrian.spalka}@cgm.com
[2] University of Bonn, Dept. of Computer Science III, Römerstr. 164, 53117 Bonn, Germany
adrian@iai.uni-bonn.de

Abstract. Regarding the increasing number of applications provided as external services, the importance of pseudonymous data as a means for privacy protection of user entities is growing. Along with it grows the relevance of secure and accurate generation, use and management of pseudonyms. In particular we consider the involvement of third parties in this process as potentially harmful, and therefore favor a decentralized pseudonym generation approach where the role of central components is reduced to a minimum. In this paper, we propose a pseudonym generation mechanism and focus on its implementation based on elliptic curve cryptography, in which every user entity can generate an arbitrary number of uncorrelatable pseudonyms with minimal effort, initially as well as at any later point in time. Because no sensitive information necessary for pseudonym generation is available on central components, our approach provides security as well as flexibility and usability.

Keywords: Pseudonym generation, trusted third party, decentralized, elliptic curve cryptography.

1 Introduction

The number of applications provided as external services is continuously growing. There also exist more and more applications that deal with sensitive data, on whose security high demands have to be made by their user entities, as well as by application providers. Providers must gain trust and acceptance of possible users of their applications, and may also need to meet further legal requirements in order to comply e.g. with laws of data protection. In this context, the security merit of confidentiality of data handled by external applications is usually in the focus, even more than the security merits of data integrity and availability. After all, the external applications often use the internet as the operating environment, which has to be considered as potentially harmful.

But not only threats of attacks by external adversaries have to be averted; in fact, security measures for transport encryption or data center security are considered as

S. Furnell, C. Lambrinoudakis, and G. Pernul (Eds.): TrustBus 2011, LNCS 6863, pp. 113–124, 2011.

highly developed, and are not the target of this paper. The application user entity's informational privacy may also need to be protected from the application providers themselves, if the sensitivity of the managed data demands it, e.g. in the case of medical data underlying the laws of data protection in Germany. The threat of insiders using their assigned privileges in an unintended way like e.g. corrupt data administrators cannot be excluded, no matter how noble the intentions of the application providers are or how deterrent the consequences of privilege abuse are communicated to be.

When looking at user data as a logical pair consisting of the owning user entity's identity (or rather a reference to that identity) and the actual data content, data confidentiality can be ensured by making at least one component of the pair unintelligible to unauthorized entities. It depends on the nature of the application which of the two components is chosen for this, or if it is even necessary to make both components unintelligible. For instance, making the data component unintelligible for unauthorized users (i.e. data encryption) may be the more obvious step, but the nature of the underlying application may demand that the data component remains plaintext in order to be processed by the application provider. Therefore, to ensure privacy, the association with its identity has to be concealed, rendering the data pseudonymous. The unintelligible association with the data owning entity will be referred to as the pseudonym in the following.

We note that in case of pseudonymous data, the plaintext data must not contain any inherent information that allows the association with the data owning entity. However, this is not target of this paper.

We much rather focus on the generation, management and use of pseudonyms; especially who in the system architecture is capable of linking a pseudonym to its owning entity is of interest for us, as it is a key feature in the security architecture of applications dealing with pseudonymous data. Whoever is capable of linking pseudonyms to their owning entities has access to the managed data to the full extent.

However, regardless of this aspect's importance, we identify many architectural shortcomings in its practical use. For instance, in many systems, linking a pseudonym to its owning user entity is performed by a static mapping table, which is operated by a trusted third party, often the same party that also stores the managed data. This is not a desirable solution; as we have pointed out before, we consider the use of trusted third parties as an architectural weakness because they simply can never be fully trusted (one corrupt administrator is enough). Security measures like physical separation of identity/pseudonym mappings on one hand and managed data on the other or strict administrative and jurisdictional regulations may alleviate the threat imposed by the use of trusted third parties, but they can never eliminate it.

There may even exist a further reason against the use of trusted third parties. Under certain circumstances, the user entities may be legally obligated to make their data unintelligible to other entities. For instance, in addition to data protection laws, the German criminal code names occupation groups, e.g. physicians or lawyers, that are responsible for the confidentiality of their data, even if that data are hosted by third parties, known as §203 StGB. According to that, if a corrupt administrator links a physician's managed patient data to the patients' identities and discloses the no more pseudonymous data, the physician is also liable.

We clearly favor a decentralized approach for the generation and management of pseudonyms. More clearly, in the entire lifecycle of a pseudonym from its generation over its normal use to its abandonment, there should be no architectural constraint that the association of a pseudonym with its owning entity can be made by any other entity than the owning entity itself.

A simple solution for a decentralized approach is of course letting the participating user entities choose their own pseudonyms and letting them keep the pseudonyms in their custody. Although this solution might basically work, it suffers from a few drawbacks. For instance, since the pseudonyms are chosen randomly by the user entities, duplicates cannot be ruled out. Another drawback is that entities can impersonate other entities by using their pseudonyms, not to mention that this approach imposes considerable managing effort on the user entities.

We state the following requirements to a decentralized approach for pseudonym generation and management:

1. *Ease of use.* Although the user entity's contribution should be minimal, it should be able to access its pseudonyms at any time.
2. *Confidentiality.* Pseudonym generation should be performed exclusively in the scope of the pseudonym owning entities. Linking a pseudonym to its owning entity should not be possible for any other than the pseudonym owning entity itself, unless it is explicitly permitted. Furthermore, in case user entities can own multiple pseudonyms, it should not be possible to identify different pseudonyms as belonging to the same user entity for any entity other than the pseudonym owning entity itself.
3. *Injectivity.* Although decentralized, the pseudonym generation process should avoid duplicates, even if every user entity may own multiple pseudonyms.
4. *Flexibility.* It should be possible to add new pseudonyms to the user entities with minimal effort at any point in time.

On the following pages, we will propose a pseudonym generation mechanism and its implementation based on elliptic curve cryptography (or a slight variation thereof) that fulfills the requirements stated above. In the following section 2 previous and related work will be discussed; after that, basic notions from the field of elliptic curve cryptography will be introduced in section 3 before our pseudonym generation mechanism will be presented in section 4. Section 5 covers the benefits of our proposed approach to pseudonym generation, before future development is considered in chapter 6. The paper's conclusion is drawn in chapter 7.

Our pseudonym generation mechanism is not only of theoretical relevance, but already in use by CompuGroup Medical AG (CGM), a large producer of healthcare information systems located in Germany. The CGM product family "Software Assisted Medicine" (SAM) comprises systems in which pseudonymous patient data are stored and processed in a central database. The pseudonyms of the SAM participants (physicians and patients) are generated using our mechanism. As SAM comprises multiple modules (e.g. "SAM Diabetes", "SAM COPD" etc), the patients are provided by our pseudonym generation mechanism with different (and not correlatable) pseudonyms for different modules.

2 Previous and Related Work

The authors of [1] propose a pseudonym generation mechanism which makes use of iterative Diffie-Helman (IDH) key agreement schemes, and which is well suited for ad-hoc group communication scenarios. Members agree on a group key using the protocols of the Tree-based Group Diffie-Hellman (TGDH) suite and then compute their own pseudonyms using a newly proposed pseudonym generation scheme, which is an extension of TGDH. Even though the pseudonyms are unlinkable to the user entities owning the pseudonyms in normal operation mode, a subgroup of the user entities can reveal the real identity behind certain pseudonyms after making a democratic decision to do so. The method is based on the idea of generating unique keys (respectively key components like primes) within isolated instances.

In [2], an approach is presented in which user entities generate globally unique pseudonyms locally, which are claimed to be highly random. No information interchange is needed, especially not keys, except for unique identifiers that are issued initially for participating entities. Following the authors' approach, user entities generate RSA key pairs locally at random, encrypt their unique identifier using that random key pair and concatenate the RSA key pair's public key exponent and modulus to the encryption result. The authors claim that the resulting value can be proven to be unique. Proof of ownership for a pseudonym performed by a user entity is implemented by a challenge/response algorithm in which the user entity is proving its pseudonym ownership by using the private key of its generated RSA key pair.

Although [3] deals also with pseudonyms that are cryptographically derived in cooperation with the owning user entity from the entity's identity number, the security architecture of the presented system relies considerably on trusted third parties that are assigned central administration tasks for the pseudonyms. For instance, a proof of pseudonym ownership is performed by certificates that are issued to the users by trust centers.

[4], [5] and [10] are also examples for pseudonym management systems that rely on trusted third parties, although the authors of [5] claim that the involvement of the trusted third party is reduced to a minimum in comparison to previous approaches. Here, the user entities create each an asymmetric key pair, whose private key is kept secret and whose public key is registered with a certification authority (CA). The CA issues credentials that state that the user entities are valid user entities. After that, user entities can register with other organizations and collaboratively compute pseudonyms for interaction with these organizations, after they have authenticated themselves using their CA credential.

The authors of [6] and [7] propose a „privacy manager" applied in the field of Cloud Computing; however, the authors focus rather on encrypted (or "obfuscated", as they call it) data than on pseudonymous data, which forms more or less an additional line of defense. The pseudonym generation mechanism is performed by static mapping tables or by symmetric encryption functions, with no statement being made on key management for the symmetric encryption functions.

The authors of [8] propose a proxy-based public key infrastructure, which is to be applied in the field of Mobile Location-Based Services (LBS). There, users can operate pseudonymously and encrypt their communications using mediated identity-based encryption. Users are claimed to have to own only one private key in order to

protect their communications when they access LBS under different pseudonyms, which can be either randomly picked one-time pseudonyms or long-term pseudonyms that are cryptographically computed, albeit this has to be done in collaboration with a trusted third party.

In [9] a novel cryptographic primitive is proposed which is called an "Incomparable Public Key System" in order to protect the anonymity of message receivers while using Multicast. The primitive enables message receivers to create many anonymous identities without disclosing that these identities belong to the same user entity. This is done by creating many different public keys that all correspond to the same private key and that are claimed to be not correlatable. An implementation of the primitive is presented by a variation of the ElGamal cryptosystem.

[11] deals with anonymization of network traffic data, where (source and destination) IP addresses of captured IP packets are anonymized. As the approach is clearly a centralized one and requires secret knowledge by a trusted third party, it is of minor interest for us.

In the field of location based services (LBS), the authors of [12] propose an approach to ensure privacy of LBS users without having to rely on trusted entities, yet without a significant loss in the quality of service. The authors describe a decentralized matching service that unfortunately takes trivially encoded (not encrypted) information from both LBS users and LBS providers (i.e. pseudonymous entity locations and pseudonymous entity identities) and creates triggers that fire when users and their objects of interest are in the vicinity of each other.

3 Basic Notions of Elliptic Curve Cryptography

In this chapter, we recall some basic notions necessary for the description of elliptic curve cryptography and discuss the interpretation of elliptic curve point multiplication as an injective one-way function.

Let:

- p be a prime number, $p > 3$, and $\boldsymbol{F}_p$ the corresponding finite field
- a and b integers

Then the set E of points (x, y) such that

$$E = \{(x, y) \in \boldsymbol{F}_p \times \boldsymbol{F}_p \mid y^2 = x^3 + ax + b \} \tag{F1}$$

defines an elliptic curve in $\boldsymbol{F}_p$. For reasons of simplicity, we skip the details on E being non-singular and, as well, we do not consider the formulae of elliptic curves over finite fields with $p = 2$ and $p = 3$. The subsequent statements apply to these curves, too. The number m of points on E is its order. Let $P, Q \in E$ be two points on E. Then the addition of points

$$P + Q = R \text{ and } R \in E \tag{F2}$$

can be defined in such a way that E forms an Abelian group, i.e., it satisfies the rules of ordinary addition of integers. By writing

$$P + P = [2]P \tag{F3}$$

we define the k-times addition of P as $[k]P$, the point multiplication. Now EC-DLP, the elliptic curve discretionary logarithm problem, states that if

$$Q = [k]P \tag{F4}$$

then with suitably chosen a, b, p and P, which are known to the public, and the as well known to the public point Q it is computationally infeasible to determine the integer k. The order n of a point P is the order of the subgroup generated by P, i.e. the number of elements in the set

$$\{P, [2]P, \ldots, [n]P\} \tag{F5}$$

With all this in mind, we define an elliptic curve cryptographic (ECC) system as follows. Let:

- E be an elliptic curve of order m
- $B \in E$, a point of E of order n, the base point

Then

$$D = \{a, b, p, B, n, co(B)\} \tag{F6}$$

with $co(B) = m / n$ defines a set of domain ECC-parameters. Let now g be an integer and

$$O = [g]B \tag{F7}$$

Then (g, O) is an ECC-key-pair with g being the private key and O the public key. For we rely on findings of Technical Guideline TR-03111, Version 1.11, issued by the German Bundesamt für Sicherheit in der Informationstechnik (BSI), one of the best accredited sources for cryptographically strong elliptic curves, we can take that $m = n$, i.e. $co(B) = 1$, and hence reduce (F6) to

$$D = \{a, b, p, B, n\} \tag{F8}$$

Now we can define our one-way function. Let D be a set of domain parameters concordant with (F7). Then

$$f\colon [2, n - 1] \rightarrow E, f(k) = [k]B \tag{F9}$$

i.e. the point multiplication (F7), is an injective one-way function.

4 The Pseudonym Generation Mechanism

Our pseudonym generation approach will be outlined in this section. First, the basic idea is described before an implementation is presented using elliptic curve cryptography.

4.1 Basic Idea

The basic idea of our pseudonym generation mechanism is that every participating user entity owns a personal secret s which must be unique among the set of participating entities and which must never leave the entity's scope. To perform the pseudonym generation, the user entity adds a public parameter V that may also be in the entity's custody or may be retrieved from a central component.

Note that it is still fair to call our approach decentralized although we do in fact use central components for, like in this case, providing public parameters or for validation of the personal secret's uniqueness, which will be discussed later on. The crucial aspect of a decentralized approach is from our perspective not to place any information on a central component that enables any entities other than the pseudonym owning entity to link a pseudonym to its owning entity.

The user entity's personal secret s and the public parameter are input into an injective cryptographic one-way function h whose output $h(s, V) = PS$ is considered the pseudonym. Because h is injective, the public parameter V is constant and the entity's personal secret is unique, h's output PS is unique as well, though only according to V: If two user entities compute their pseudonyms using their unique personal secrets and two different values for V, it cannot be ruled out that the pseudonyms computed by h are equal.

Hence, by variation of V, referred to as a set of uncorrelated values $\boldsymbol{V} = \{V_1, ..., V_n\}$, a set of pseudonyms $\boldsymbol{PS} = \{PS_1 = h(s, V_1) , ... , PS_n = h(s, V_n)\}$ can be generated by the user entity itself in a decentralized manner, as the entity's personal secret s is known by noone except the user entity. Each of the entity's multiple pseudonyms is unique according to the value of V_i used for computation, and it is also unique among the set of the user entity's pseudonyms.

A useful application of the mechanism is to provide a defined set of public parameter values $\boldsymbol{V}$ on a central component from which values V_i can be retrieved by user entities. By doing this, it can be centrally controlled how many pseudonyms each user entity can own.

The uniqueness of the entities' secrets must be ensured, which can be done in an initial step by letting a new user entity choose its secret s, providing the entity with a special public value V_0, letting the user entity compute $h(s, V_0) = PS_0$ and storing the computed pseudonym PS_0 on the central component. If PS_0 is unique according to V_0, the user entity's secret s must be unique as well because of h's injectivity. Now the entity can be provided with any of the actual values V_i in order to compute its own pseudonyms.

4.2 Elliptic Curve Cryptography Implementation

Elliptic curve cryptography is well suited for an implementation of our pseudonym generating mechanism that has been presented in the previous subsection[1]. In the following, the implementation is described in detail.

[1] We would like to point out that elliptic curve cryptography is not the only way of an implementation of our pseudonym generation mechanism. An implementation based on element exponentiation on finite, cyclic groups, i.e. an implementation that relies on the intractability of DLP instead of EC-DLP, is also possible. We favored elliptic curve cryptography because it offers more security with smaller key lengths and because we have been using it already in several other projects.

Rather than a personal secret, each user entity owns in this implementation an elliptic curve key pair $K = (g, O)$, with K's private key g being an integer and the corresponding public key O being a point on the used elliptic curve E. The elliptic curve E is defined by a set of public domain parameters $D = \{a, b, p, B, n\}$ (see section 3). Note that the cofactor for the given domain parameter set must equal 1 in order to be suitable for our purpose.

The private key g represents the user entity's personal secret s, and $O = [g]B$ represents the value PS_0 that is used for checking the uniqueness of the user entity secret, while E's base point B corresponds to the initial public parameter value V_0 described in subsection 4.1. Public key O, that has been computed by multiplying base point B by private key g, corresponds to PS_0 and can be checked for uniqueness in a PKI hosted on a central component. By doing this, it can be proven that the corresponding private key g is unique as well, since the point multiplication on the elliptic curve (which corresponds to the cryptographic one-way function h in subsection 4.1) is injective.

The public parameter variation from subsection 4.1 is implemented by a variation of E's base point B, i.e. by replacing B (which is feasible because E's cofactor equals 1) with other base points $B_i \in \boldsymbol{B} = \{B_1, ..., B_n\}$ picked randomly and uncorrelated from each other from E, which produces different and as well uncorrelated results when an entity multiplies one of the base point variations B_i by its respective private key:

$$[g]B_i = O_i \tag{F10}$$

The resulting curve point O_i is considered the user entity's pseudonym according to base point B_i. Because the base point variations B_i are uncorrelated, the pseudonym values are so as well. Furthermore, because of the point multiplication's injectivity, the computed pseudonyms O_i are unique in user entity's pseudonym set, and each pseudonym O_i is unique among all other pseudonyms that have been computed by other user entities using the same base point B_i and their own private key g.

Note that uniqueness of any given pseudonym O_i can only be guaranteed according to the base point B_i that has been used to compute O_i, because it cannot be ruled out that two different base points, multiplied by two different private keys, result in the same curve point.

5 Benefits of the Pseudonym Generation Mechanism

In this section, we will point out the benefits our presented pseudonym generation mechanism offers to the field of applications dealing with pseudonymous data. We will show that the requirements we expressed in section 1 are all met.

5.1 Ease of Use

The efforts of a user entity when accessing a pseudonym are limited to accessing the entity's private key g, retrieving base point B_i from the central component and performing the point multiplication of B_i by g, resulting in pseudonym $O_i = [g]B_i$. The "base point context", i.e. the information to which base point any given pseudonym

relates must be clear at all times, so when using a pseudonym, a user entity must provide this information as well.

Note that there is no distinction between a pseudonym O_i's initial generation and accessing it for normal use, because on all occasions, the pseudonym is generated anew by performing the point multiplication $O_i = [g]B_i$. These operations should be easy to implement; if the central component providing the base points is unreachable at times, the user entities can be provided with cached copies of the base points.

5.2 Confidentiality

Our pseudonym generation mechanism offers high confidentiality. Given that all base points B_i provided on the central component, including the original base point B of the used elliptic curve E, are not correlated to each other, and (of course) that the user entity's private key g is kept confidential, there is neither a feasible way to link a user entity to one of its pseudonyms PS nor to identify two pseudonyms O_i and O_j to belong to the same user entity.

Linking a pseudonym $O_i = [g]B_i$ to its owner would mean to link it to the entity's original public key $O = [g]B$. Since B_i and B are uncorrelated, the only way to achieve this would be to somehow determine g from the value pairs (B, O) and/or (B_i, O_i), meaning to solve EC-DLP which is, as of now, computationally infeasible.

5.3 Injectivity

As the user entities' private keys are all distinct (because their initial public keys O that are stored in the central PKI are) and because point multiplication on elliptic curves is injective, the generated pseudonyms are also distinct according to the base point used during their computation.

Regarding the pseudonym set $(PS_1 = [g]B_1, \ldots, PS_n = [g]B_n)$ of a user entity, it is obvious that since all used base points are distinct, the pseudonyms the user entity computes by multiplying each base point by the same private key g must be distinct as well.

5.4 Flexibility

The set of used base points can easily be expanded by adding further base points to the central component. Each new element B_{n+1} added to the set of base points $\{B_1, \ldots, B_n\}$ provided on the central component expands the set of pseudonyms of each user entity by one additional pseudonym for that user entity. Because this can be done any time and not only in the beginning during some registration process, it shows that our pseudonym generation mechanism offers flexibility, as it can be centrally controlled how many pseudonyms each user entity can have, and this number can be adapted at any point in time.

On the other hand, another scenario for the management of base points could be that every user entity generates its own base points, and whenever a user entity communicates a pseudonym, it communicates the base point used to compute it along. However, this approach shows some operational drawbacks. First, if the entity's base points are published on the central component, they have to be linked to their owning entity, and since pseudonyms have to be associated with the base point used for the

pseudonym's computation, it would be possible to link a pseudonym to its owning user entity. Thus, all pseudonyms owned by the same entity could be identified as such as well. A second reason is that it would display some kind of redundancy if the user entities have to carry their own base points for pseudonym generation, because they could directly carry their pseudonyms.

6 Future Development

In this section future development of our pseudonym generation mechanism and possible issues are considered.

6.1 Authentication

A useful extension to our approach deals with the mechanism's resistance against spoofing and impersonation of pseudonyms. If an adversary uses a made-up pseudonym or has eavesdropped an actual pseudonym from a regular user entity and uses it to impersonate the user entity, it should be possible for an entity located e.g. in the central component to identify such attempts of unintended behavior.

Every pseudonym $O_i = [g]B_i$ is actually a public key that forms, together with the user entity's private key g, a valid asymmetric key pair according to base point B_i. Hence, when a user entity deploys data linked to a pseudonym O_i, the pseudonym can be authenticated via a challenge/response authentication protocol based on e.g. ECDSA. Although this may be a protection against pseudonym impersonation, it is not against spoofing, because if an adversary just makes up a private key g and computes the corresponding public key O_i (according to a base point B_i) the central component will not be able to distinguish the spoofed pseudonym from a regular one, and the adversary will also be able to authenticate the made up pseudonym against the central component.

So the cost of a functioning authentication mechanism is that all regular pseudonyms have to be cataloged, albeit anonymously, in the central component. Therefore, when a new user entity is introduced to the system and chooses its unique private key, all the entity's pseudonyms must be computed and provided to the central component, which stores these pseudonyms anonymously[2].

Provided with this information, the central component is able to detect spoofed pseudonyms as well as impersonated ones. However, extending the set of base points with additional elements becomes more complicated in such a scenario. The authentication aspect will clearly be target of further investigations.

6.2 Base Point Manipulation

Another possible issue of our approach applies to the choice of the base points. It must be ensured that they are not correlatable to each other, because if they are, the computed pseudonyms are so as well. If e.g. $B_1 = [2]B_2$, then $O_1 = [2]O_2$ holds for all

[2] By doing this, the central component would be technically able to correlate all pseudonyms of the user entity, and the user entities would be forced to trust that the central component stores the pseudonyms in a responsible manner.

pseudonyms O_1 and O_2 of all participating user entities, and they are easily correlatable for a dishonest entity located at the central component.

A distributed, collaborative computation of the set of base points outside of the scope of the central component by a group of user entities that each contribute a piece of information, yet are unaware of the pieces of information contributed by the other entities might solve the problem. Another solution for this issue is not only varying the base point on the same curve, but the varying the entire used elliptic curve among a predefined set of publicly accredited elliptic curves that all have about the same order so that it is easy to find an integer g that is a valid private key for all curves. This aspect will too be target of further investigations.

7 Conclusion

In this paper, we proposed a decentralized pseudonym generation mechanism that uses in a general approach injective, cryptographic one-way functions and, in a more concrete implementation, elliptic curve cryptography. Although the mechanism is comfortable and easy to use, it offers high security that is of equal strength as the discrete logarithm problem on elliptic curves, ECDLP.

The mechanism does not have rely on the trustworthiness of third parties in the process, but nevertheless guarantees uniqueness of the pseudonyms generated by the user entities, even when an arbitrary number of pseudonyms is granted to each user entity. By providing user entites with a set of public parameters (base points in the elliptic curve implementation) on a central component when pseudonyms need to be computed, the number of pseudonyms per user entity can be centrally controlled, yet this does not affect the confidentiality of the association between a user entity and its pseudonym. The set of pseudonyms per user entity can be adapted at any point in time by adding further elements to the centrally managed set of public parameters (respectively base points).

Future extensions of the mechanism comprise a pseudonym authentication component, which enables application providers to ensure that no spoofed or impersonated pseudonyms are used, as well as a decentralized base point generation scheme to prevent corrupt entities located on the central component to use manipulated base points that enable adversaries to correlate pseudonyms.

References

1. Manulis, M., Schwenk, J.: Pseudonym Generation Scheme for Ad-Hoc Group Communication Based on IDH. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 107–124. Springer, Heidelberg (2005)
2. Schartner, P., Schaffer, M.: Unique User-Generated Digital Pseudonyms. In: Gorodetsky, V., Kotenko, I., Skormin, V. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 194–205. Springer, Heidelberg (2005)
3. Mjølsnes, S.F.: Privacy, Cryptographic Pseudonyms, and The State of Health. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 493–494. Springer, Heidelberg (1993)

4. Franz, E., Liesebach, K.: Supporting Local Aliases as Usable Presentation of Secure Pseudonyms. In: Fischer-Hübner, S., Lambrinoudakis, C., Pernul, G. (eds.) TrustBus 2009. LNCS, vol. 5695, pp. 22–31. Springer, Heidelberg (2009)
5. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H., Adams, C. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
6. Pearson, S., Shen, Y., Mowbray, M.: A Privacy Manager for Cloud Computing. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 90–106. Springer, Heidelberg (2009)
7. Mowbray, M., Pearson, S.: A Client-Based Privacy Manager for Cloud Computing. In: COMSWARE 2009: Proceedings of the Fourth International ICST Conference on COMmunication System Software and Middleware. ACM, New York (2009)
8. Candebat, T., Dunne, C.R., Gray, D.T.: Pseudonym Management using Mediated Identity-Based Cryptography. In: DIM 2005: Proceedings of the 2005 Workshop on Digital Identity Management, pp. 1–10. ACM, New York (2005)
9. Waters, B.R., Felten, E.W., Sahai, A.: Receiver Anonymity via Incomparable Public Keys. In: CCS 2003: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 112–121. ACM, New York (2003)
10. Au, R., Vasanta, H., Kwang, K., Choo, R., Looi, M.: A User-Centric Anonymous Authorisation Framework in E-commerce Environment. In: Janssen, M., Sol, H.G., Wagenaar, R.W. (eds.) ICEC 2004: Proceedings of the 6th International Conference on Electronic Commerce, pp. 138–147. ACM, New York (2004)
11. Schmoll, C., Chatzis, N., Henke, C.: Protecting User Privacy with Multi-Field Anonymisation of IP Addresses. In: SIN 2010: Proceedings of the 3rd International Conference on Security of Information and Networks, pp. 38–45. ACM, New York (2010)
12. Jaiswal, S., Nandi, A.: Trust No One: A Decentralized Matching Service for Privacy in Location Based Services. In: MobiHeld 2010: Proceedings of the second ACM SIGCOMM workshop on Networking, Systems, and Applications on Mobile Handhelds, pp. 51–56. ACM, New York (2010)