On Transaction Pseudonyms with Implicit Attributes

Stefan G. Weber

StefanGeorgWeber@gmail.com

Abstract. Transaction pseudonyms with implicit attributes are a novel approach to multilevel linkable transaction pseudonyms. We extend earlier work of Juels and Pappu on reencryption-based transaction pseudonyms, by developing new mechanisms for controlled pseudonym linkability. This includes mechanisms for cooperative, stepwise re-identification as well as individual authentication of pseudonyms. Our proposal makes use of efficient techniques from the area of secure multiparty computation and cryptographically secure PRNGs.
Key words: Pseudonymity, Privacy Protection, Accountability, Secure Multiparty Computation

1 Introduction

Privacy protection is a security requirement that is often articulated by individual users of computer applications. Moreover, it is also implied by existing data protection regulations. Digital pseudonyms, originally introduced by Chaum [4], are an important base mechanism for many privacy-enhancing technologies. In a technical sense, a (digital) pseudonym is an identifier of an entity that is used instead of the entity's real-world name [21]. Pseudonymity is the use of pseudonyms as identifiers. In a computing system, a user can act among one or multiple pseudonyms in digital transactions, e.g. in order to access a digital resource or service without disclosing her identity. The provided degree of unlinkability[1] and anonymity[2] is stronger, the more often pseudonyms are changed over time. Thus, the degree of protection is highest for pseudonyms that are only used in a single transaction.

In this article, we describe how to construct *transaction pseudonyms with implicit attributes*, a novel approach for realizing multilevel linkable transaction pseudonyms. While a transaction pseudonym[3] provides unlinkability to further pseudonyms created by the same user, multilevel linkability refers to a controlled capability to make pseudonymous users accountable again in several levels of re-identification and thus granularity.

[1] Unlinkability is a property that aims at hiding relationships between items in a system.

[2] Anonymity is the state of not being identifiable within a set of subjects, the so-called anonymity set.

[3] In the literature, transaction pseudonyms are sometimes also denoted as *short-lived pseudonyms* or simply as *changing pseudonyms*.

In order to achieve these seemingly conflicting properties, pseudonyms with implicit attributes encode specific access informations that allow defining and enforcing security policies on transaction pseudonyms:

- users are empowered to authenticate own pseudonyms after use,
- authorized parties may cooperatively re-identify pseudonyms in multiple levels of granularity, i.e. link them to attributes that are only implicitly associated with pseudonyms due to specific registration information,
- an (optional) law enforcement authority may completely disclose every transaction pseudonym.

Our approach extends earlier work of Juels and Pappu [14], which proposed reencryption-based transaction pseudonyms for privacy protection in a RFID application context. In existing proposals for transaction pseudonyms, the controlled linkability is usually limited to law enforcement or does neither consider a stepwise nor a cooperative re-identification.

The next section introduces the proposed construction principle, followed by a description of the main primitives than we build upon. Then, the mechanisms are explained in detail.

2 Construction Principle

Pseudonyms implement central reference points w.r.t. the handling of access to personal data in computing systems. Thus, pseudonyms can allow balancing conflicting accountability and privacy requirements to some extent. Since a pseudonym is an identifier of an entity that is used instead of the real-world name of the entity, it implements a certain degree of unlinkability and thus privacy protection. By using the identifier-to-pseudonym-mapping [10], a re-identification of entities' real world names is possible; thus making pseudonymized entities accountable again. Hence, a fair balance of interests depends on the provided degrees of (un-)linkability implemented by a pseudonym and by controlling who can use the pseudonym-to-identity-mapping.

Existing types of pseudonyms allow constructing particular forms of pseudonym linkability; moreover, pseudonym constructions can even be classified depending on how and by whom given pseudonyms can be re-translated into the users identity (by making use of the pseudonym-to-identity-mapping). This is also captured in the notion of linkable pseudonyms [1]. Technically, enabling such a mapping requires that a pseudonym additionally encodes some kind of or is associated to trapdoor information, to enable attribution of pseudonyms to real-world identities. According to [9], major types of pseudonyms are:

- *Reference pseudonyms*: In a simple case, a (non-changing) pseudonym can be mapped back to an identity based on existing reference information, e.g. if issued public keys are used as pseudonyms. Access to the referencing registration information then limits pseudonym disclosure.

- *Self-generated pseudonyms*: In case that pseudonyms are only generated by the user herself and no reference is stored together with identities, only the user herself can link a pseudonym, e.g. based on secret information. A self-chosen nickname is a simple example.
- *Cryptographic pseudonyms*: Given that pseudonyms are constructed by applying a cryptographic function to identity-related information, the function itself and possibly further input parameters control the re-identification. In case of relying on encryption functions, e.g. the keys that enable decryption have to be known.

We strive for a construction that integrates all these different levels of linkability. Thus, we propose to combine useful properties of every of this types. We achieve this as follows:

It is known that techniques from the area of secure multiparty computation (SMPC) can theoretically be applied to a large range of problems in the area of privacy-preserving data analysis and in the construction of privacy-preserving protocols [16]. Basically, SMPC mechanisms [30] allow implementing multiparty protocols that do not rely on a single trusted third party (TTP). The intention of these cryptographic techniques is that a number of distinct, but connected parties may jointly compute an agreed function of their inputs in a secure way. Hereby, the correctness of the output as well as the privacy of each input shall be preserved, even if some participants cheat. Thus, secure multiparty computation is a general approach to distribute the functionality and powers of a single TTP among several parties[4]. In most SMPC approaches, this is only achieved with very high computational costs [16], due to the intensive use of secret sharing [23] and operations on secret shared data in SMPC protocols. Yet, more efficient special purpose approaches to SMPC have been proposed. E.g. in the *mix-and-match* approach [13], secret sharing techniques are replaced by *operations on encrypted data*. We exploit this capability in our proposal.

Hereby, we follow basic ideas of the *mix-and-match* approach. We formulate the pseudonym generation in terms of specific non-deterministic encryption operations. Since the encryption is non-deterministic, this allows deriving multiple transaction pseudonyms by updating the inherent random factors. Also, this enables us to apply efficient concepts from the mix-and-match framework in order to realize privacy-respecting data linkage functionalities *on the pseudonym level*. Especially, we realize the mechanisms for multilevel linkability of pseudonyms based on extended mix-and-match concepts.

Additionally, we propose to make use of cryptographically secure pseudo random number generators [15] in order to control random factors inside the encryption operations. This allows implementing a further direct pseudonym-to-identity-mapping, that can be exploited by a user, in order to authenticate pseudonyms that relate to herself.

[4] A classic example of SMPC is the millionaires' problem due to Yao [30]: some millionaires (*the parties*) want to find out, who is the richest (*agreed function*) without revealing the precise amount of their individual wealth (*input privacy*).

3 Main Primitives

Having introduced the construction principle, in this section, we briefly describe the main primitives that are employed in order to implement the transaction pseudonyms with implicit attributes approach.

3.1 Cryptographically Secure PRNGs

A pseudo-random number generator (PRNG) [15] is a deterministic algorithm that generates sequences of numbers that appear random. In order to achieve this, a PRNG incorporates an internal source of entropy, which is called a seed, for deriving and computing the output. A cryptographically secure PRNG is a special kind of PRNG that produces sequences of numbers with stronger security requirements: it is practically impossible to guess or derive any forward or backward numbers by analyzing the output of a cryptographically secure PRNG. Therefore, such a PRNG is suitable for cryptographic purposes. Our constructions actually employ cryptographically secure PRNGs. For simplicity, we often refer to this tool simply as PRNG.

3.2 Threshold ElGamal Cryptosystem

A key primitive in our approach is the ElGamal cryptosystem [7], over subgroups G_q of order q of the multiplicative group Z_p^*, for large primes $p = 2q + 1$. The primes p, q and a generator g of G_q are common system parameters. ElGamal encryption is semantically secure in G_q, under certain complexity assumptions [24]. Practically, semantic security means that no partial information about a plaintext is leaking from the corresponding ciphertext. Thus, an adversary can neither recover any information about the plaintext from the ciphertext, nor distinguish whether a ciphertext is the encryption of a known plaintext or not.

More specifically, we utilize a threshold variant of the ElGamal cryptosystem, according to Pedersen [19, 20], which allows distributing cryptographic operations. It thus supports distributability of powers. In this threshold system, an ElGamal private key $s \in_R Z_q$ can be defined in two ways[5]:

- Firstly, it can be initially generated by a trusted dealer and then be secret shared (according to Shamir secret sharing [23]) among all n participating authorities.
- Secondly, it can be generated via the distributed key generation protocol of Pedersen [19], whereby no single party knows the complete private key.

In both key generation approaches, the power to decrypt is distributed among all of the participating authorities. A quorum, i.e. a minimal majority of t out of n authorities need to cooperate to perform a threshold decryption protocol, as specified in [19, 20]. Thus, a threshold ElGamal system can tolerate a

[5] In the key generation, the private key is chosen at random.

maximum of $t-1$ corrupt authorities, yet still requires a majority of t participating authorities. The total number of authorities n has to be (at least) $n = t-1+t = 2t-1$, so $t-1$ is a minority of authorities. The authorities share a common public key, $h = g^s \bmod p$. In this ElGamal setting, a message $m \in G_q$ is non-deterministically encrypted by choosing $r \in_R Z_q$ and by computing $(g^r, h^r m)$. Messages that are not in G_q can efficiently be mapped onto G_q [13]. Thus, arbitrary strings can be ElGamal encrypted.

Since different parties cooperate within distributed protocols, there is a need for a communication channel and a synchronization of individual inputs. Following the standard assumption in the literature, we assume that the communication channel is a broadcast channel with memory. This channel, which is also referred to as bulletin board, is used to store, exchange and synchronize inputs in any protocol that involves distributed computations. For example, partial decryptions and identifiers of every participating authority are provided to the bulletin board in a threshold decryption. Moreover, this broadcast channel is append-only, i.e. once the information is published, it is stored and cannot be changed or deleted afterwards. Thus, its content can later be analyzed to support audit and verifiability purposes.

The ElGamal system parameters p, q, g and the public key h as well the mentioned communication channel are also relevant and available to the following primitives, i.e. non-interactive zero-knowledge proofs, plaintext equality tests and ElGamal reencryption mixnets.

3.3 Non-Interactive Zero Knowledge Proofs

Zero knowledge proofs (ZKPs) [12] are basically generalized challenge-response authentication protocols which are used to guarantee correctness of and verify participation in distributed cryptographic operations and protocols. A special feature of ZKPs is that they disclose no further information beyond that a statement is true. Non-interactive zero knowledge proofs (NIZKPs) [2] are variants which allow rendering the challenge-response process non-interactive, i.e. they can be executed by a single party. This is achieved by applying the Fiat-Shamir heuristic [8] and thus produces transcripts of the NIZKPs. See e.g. [6] for a broader discussions of zero knowledge techniques. A NIZKP it thus comparable to a digital signature, it can be stored and may also be verified, after its execution. Thus, incorrect or inappropriate actions can be deduced by assessing the transcripts. NIZKPs are a common cryptographic approach to implement verifiability. They are involved in some of the distributed cryptographic operations in order to assure that only correct inputs of individual parties are considered for computing a function. It is important to notice that zero knowledge techniques are integral parts of many protocols, yet, we mostly abstract from details throughout this article.

3.4 Plaintext Equality Tests

A plaintext equality test (PET) [13] is a primitive for pairwise blind comparison of ciphertexts of non-deterministic threshold cryptosystems like ElGamal. A PET allows testing whether two ciphertexts represent the same plaintext by performing algebraic operations on ciphertexts, but without revealing the plaintext. Plaintext equality tests exploit properties of the algebraic division of two ciphertexts, which is introduced next:

Let $(x_1, y_1) = (g^{r_1}, h^{r_1} m_1)$ and $(x_2, y_2) = (g^{r_2}, h^{r_2} m_2)$ be two ElGamal ciphertexts with plaintexts m_1 and m_2. In case that m_1 and m_2 are equal, the algebraic division of both, $(x_3, y_3) = (x_1/x_2, y_1/y_2) = (g^{r_1 - r_2}, h^{r_1 - r_2} m_1/m_2) = (g^{r_3}, h^{r_3} m_3)$ is an encryption of 1, since $m_1/m_2 = m_3 = 1$.

In a PET, the ciphertext (x_3, y_3) is firstly blinded by raising each component to a random exponent $z \in Z_q$, $(x_4, y_4) = (x_3^z, y_3^z) = (g^{r_3 z}, h^{r_3 z} m_3^z)$, and then decrypted to the blinded value, m_3^z. The decryption reveals $m_3^z = 1^z = 1$ in case that the plaintexts are equal and a random integer $(m_1/m_2)^z$ otherwise[6]. Due to this, only negligible information beyond whether the plaintexts are equal is leaking due to the execution of the protocol. In order to prevent misuse of the power to decrypt, PETs can be performed in a distributed setting, i.e. harnessing a threshold ElGamal system. In this case, blinding and decryption are realized as distributed operations.

PETs and applications thereof are key primitives to implement multilevel linkability of transaction pseudonyms.

Figure 1 shows the interplay of the primitives relevant to distributed (threshold) computations and operations. Every authority submits the result of her partial operation to the bulletin board, accompanied by a NIZKP. After verifying the proofs, the bulletin board combines the inputs in order to produce the result of the operation, e.g. the result of a PET or a threshold decryption.

3.5 ElGamal Reencryption Mixnets

A mixnet, originally introduced by Chaum [4], is a primitive that can be used to anonymize sets of ciphertexts by a set of mix servers. Together, the mix servers form the mixnet. In our work, we build on ElGamal reencryption mixnets [18], which basically reencrypt and permute ciphertexts in order to anonymize them. In this setting, reencryption can be executed without a private key, i.e. it is not required to decrypt the ciphertext in order to produce a reencryption. Moreover, we assume that the mixnet is additionally verifiable, i.e. it provides NIZKPs of correctness of the operations. This can e.g. be achieved by employing the verifiable mixnet proposed by Furukawa et al. [11], describing the details of the cryptographic verifiability mechanisms.

[6] Given that the group order of the underlying group is prime, then $(m_1/m_2)^z$ is a random non-identity group element [17, 22, 3]. This is true in our setting, i.e. the underlying group G_q is of prime order q.

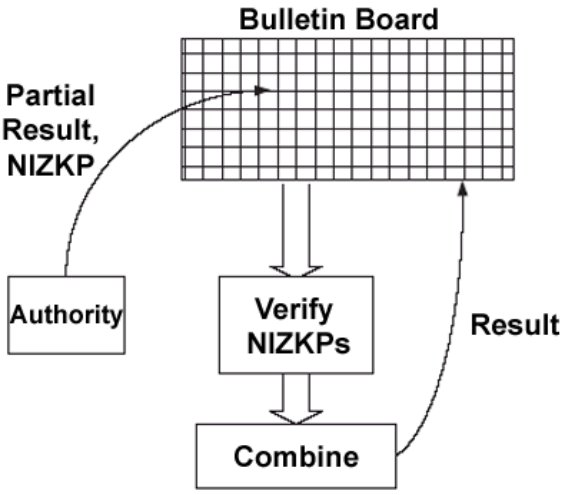

Fig. 1. Distributed computations and bulletin board

Basically, a mixnet consists of a set of mix servers M_i, that consecutively process ciphertexts. In our work, the operation of a reencryption mixnet consists of the following two main steps:

1. *Submission of Inputs:* A batch of ElGamal encrypted inputs $(g^r, h^r m_i)$ is submitted to the mixnet. Every input is an ElGamal encryption of a message m_i.
2. *Mixing Phase:* Mix server M_i receives the batch of ciphertexts output by the previous mix server M_{i-1} (or the initial input respectively). M_i reencrypts each ciphertext $c_i = (g^r, h^r m_i)$ by selecting a random value $r^{'} \in Z_q$ and computing $c_i^{'} = (g^r * g^{r^{'}}, h^r * h^{r^{'}} m_i) = (g^{r+r^{'}}, h^{r+r^{'}} m_i)$. Then, it permutes the batch of all ciphertexts randomly and passes it to the next mix server M_{i+1}. The last mix server outputs the batch of ciphertexts.

Within our proposal, mixnets help to build up anonymous reference sets used in the re-identification of pseudonyms. In particular, mixnets anonymize sets of ElGamal ciphertexts that represent transaction pseudonyms. These sets are used to blindly compare a given transaction pseudonym with the resulting reference set, based on variants of PETs, in order to perform a re-identification operation. The underlying concept of reencryption is also employed in order to derive transaction pseudonyms.

3.6 Primitives in Combination

Having introduced the main primitives, we now sketch the conceptual interplay of the primitives. Figure 2 shows the relation of PRNGs, mixnets and PETs. In

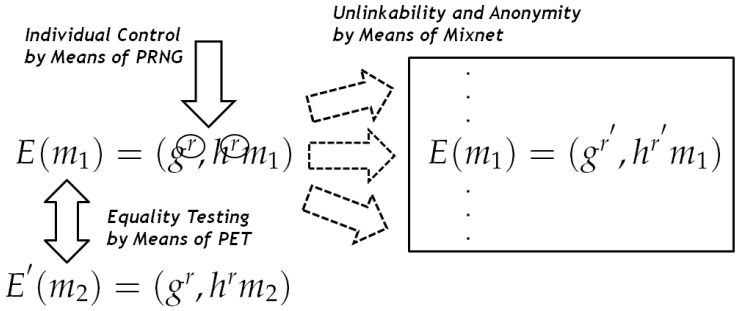

Fig. 2. Interplay of primitives

this figure, $E(m)$ represents the ElGamal encryption of an arbitrary plaintext m. By means of a PRNG, the random factors used in the encryption can individually be controlled. A mixnet can implement unlinkability and anonymity within a set of ciphertexts. A PET can be used to detect the equality of plaintexts.

In particular, the given primitives in combination allow making use of the several degrees of (un-)linkability that are given in ElGamal settings.

4 Setting and Main Protocols

We introduce the basic concepts of our approach in the following. The provided protocols implement functionalities for the generation, authentication, linking, partial re-identification and complete disclosure of transaction pseudonyms.

4.1 Parties

We consider the following authorities and entities in our setting:

- A *User* is an entity that intents to act pseudonymously. A user provides a transaction pseudonym as identifier during the relevant digital interactions instead of her identity.
- A *Registration Authority* (RA) is an entity that is responsible for registration processes. This authority interacts with users in order to register them to the system and sets up a registration list, which contains registration information.
- *Linkability Brokers* (LBs) are authorities that are able to cooperatively link and partially re-identify pseudonyms that are stored in a data repository.
- A *Law Enforcement Authority* (LEA) is an authority that is able to completely re-identify a pseudonym. This is called disclosure of pseudonyms. The misuse of the privacy protection granted by pseudonyms can be prevented by allowing such a trusted party to revoke pseudonymity in certain cases.

We assume that each entity is equipped with an appropriate computing device, that is able to store cryptographic keys, execute operations and provides communication facilities (i.e. access to the bulletin board).

4.2 Registration and Generation of Transaction Pseudonyms

In this section, we introduce the concepts for pseudonym generation. Basically, we propose to encode a static reference inside malleable pseudonyms, by generating pseudonyms as reencryptions of the reference value under the public key of a threshold ElGamal cryptosystem. The resulting construction is what we call a *transaction pseudonym with implicit attribute*. This notion reflects that every transaction pseudonym is implicitly associated with information - the implicit attributes - that can logically be derived from the registration context. Thus, implicit attributes are defined on the semantics of available registration information and support the partial re-identification of pseudonymous users.

Firstly, in our approach, every user has to participate in a registration process, which is also shown in Figure 3. In this process, each user receives a base pseudonym, that enables her to derive transaction pseudonyms. We assume that the user is in possession of a personal device which also implements a cryptographically secure PRNG. In order to register, the user interacts with a trusted registration authority (RA). The registration consists of the following main steps:

I. Each user is added to an integrity protected registration list. She receives a distinct role pseudonym, which associates the user with some meaning in the application context, e.g. with a role relevant to the issuing organization. For example, a user can be registered as $RoleX\#13$[7]. The user's unique real-world identity, e.g. $JohnDoe$[8], is encrypted and stored together with the role pseudonym on the registration list. Also, the user receives a distinct reference value, which is added to the respective entry of the registration list.
II. The RA derives a base pseudonym by encrypting the unique reference value. Thus, a base pseudonym encodes a distinct reference.
III. The RA transfers the base pseudonym to the user's personal device.
IV. The user generates and registers a seed in the PRNG of her device to enable it for pseudonym generation.

In the registration phase, the encryptions are executed under the public key belonging to the linkability brokers[9] relevant to the application context, e.g. they

[7] A registration list may contain additional information that can be used to define (implicit) attributes. Such additional information is then included in additional columns of the registration list.

[8] We assume that the real world identity can be presented as a string, denoted as ID in the following. This string could also contain legal identification numbers, in order to make it unique.

[9] In the description, we assume that cryptographic keys have been generated and distributed before. The linkability brokers thus *share* the private key associated with the single public key. Due to the use of SMPC, variants with different distributions of power are possible.

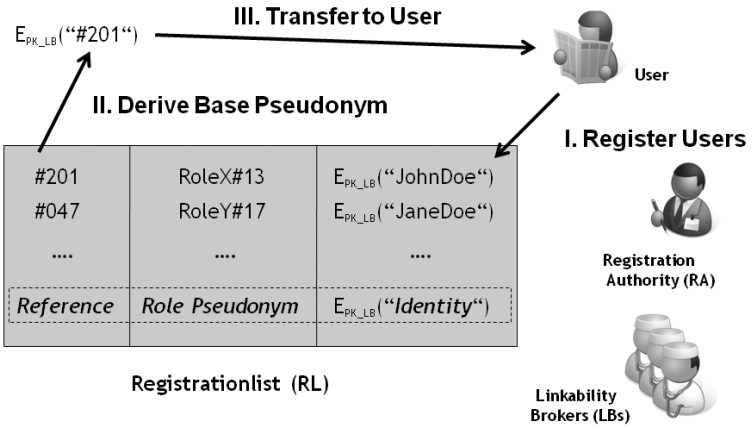

Fig. 3. Registration process

may belong to the issuing organization, or even include a party that represents interests of the users, as follows:

- The real-world identity ID is encrypted as: $E_{PK_{LB}}(ID) = (g^r, h^r ID)$.
- To generate the base pseudonym for a user, the registration authority encrypts the chosen reference value, which we denote RV, as: $P = E_{PK_{LB}}(RV) = (g^{r_s}, h^{r_s} RV)$. Moreover, the random value r_s, the start value for pseudonym generation, is also transferred to the user and stored on her device.

The user is now able to derive *transaction pseudonyms* from her base pseudonym in the following way:

1. The seeded PRNG is used to generate a sequence of random numbers.
2. Each random number r_i is used to compute a randomization factor $F_{r_i} = (g^{r_i}, h^{r_i})$.
3. F_{r_1} is used to construct the *first* transaction pseudonym by multiplying it with the base pseudonym: $P_B = P_0 = (g^r, h^r RV) * (g^{r_1}, h^{r_1}) = (g^{r+r_1}, h^{r+r_1} RV)$.
4. Further transaction pseudonyms are created by repeated multiplications: $P_{i+1} = P_i * F_{r_{i+1}}$.

By this procedure, which is also shown in Figure 4, a user creates a set of different transaction pseudonyms that all contain the same reference value. The procedure includes two factors that initialize pseudonym generation: an organizationally defined one, the base pseudonym, as well a personally defined one, the seed. The generated pseudonyms are used instead of static identifiers during the relevant digital interactions.

4.3 Authentication of Transaction Pseudonyms

Due to the construction, presented in the last section, users are also enabled to authenticate a transaction pseudonym that is stored in a data repository after

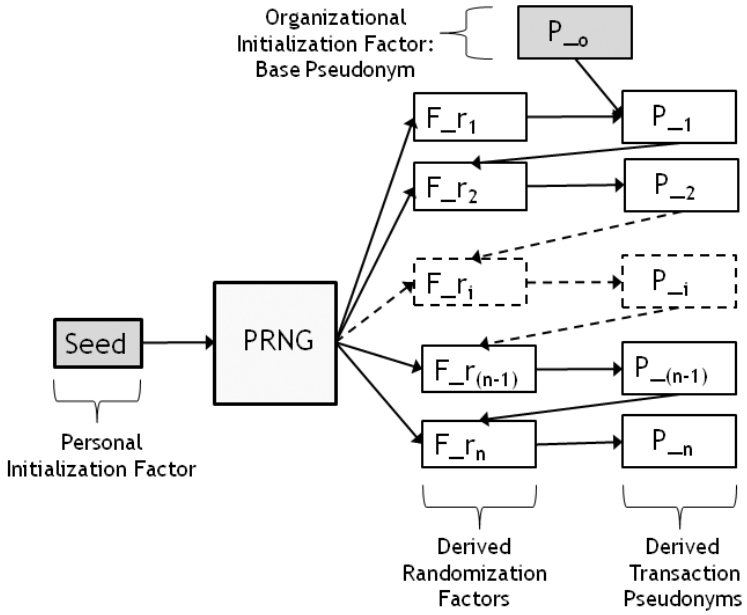

Fig. 4. Overview of pseudonym generation

it was used. Therefore, a user needs to show that she is in possession of the base pseudonym and the correct aggregated random factor, which allows reproducing a recorded pseudonym, thereby authenticating it.

4.4 Linking and Partial Re-Identification

In this section, we describe the concepts for linking and re-identifying transaction pseudonyms. Basically, we harness the possibility to execute algebraic operations on the non-deterministic encrypted ciphertexts that represent transaction pseudonyms.

We assume that the transaction pseudonyms are recorded in a data repository after they were used. Then, the linkability brokers are able to execute the following *two basic operations*:

1. *Linking:* check if two recorded pseudonyms relate to the same entity but without revealing the actual identity of the entity;
2. *Partial Re-Identification:* check if one entry relates to a group of entities with a common organizational role or function, i.e. if an implicit attribute is satisfied.

Overview of Linking The linking operation is implemented by executing a *plaintext equality test* on the pseudonym values of two recorded pseudonyms.

Suppose that $P_a = (g^{r_a}, h^{r_a} RV_a)$ and $P_b = (g^{r_b}, h^{r_b} RV_b)$ represent two entries of that kind. If they relate to the same entity, they contain the same reference value. In order to verify this, the pseudonyms can be algebraically divided: $P_c = P_a/P_b = (g^{r_a - r_b}, h^{r_a - r_b} RV_a/RV_b)$. Given that RV_a equals RV_b, this is an encryption of the value "1". By performing a threshold decryption, the linkability brokers reveal a value which is either "1", indicating that the reference is matching, or a meaningless different value.

Overview of Partial Re-Identification The second operation, the partial re-identification, is an extension of the procedure above to a global instead of pairwise comparison of pseudonyms. It allows testing if a given transaction pseudonym relates to an implicit attribute, i.e. a certain organizational unit, a function or even place. The implicit attributes are defined on the semantics of available registration information. This test does not disclose which of the possibly involved users is the originator of the pseudonym. Thus it implements a partial re-identification.

An overview of the method of partial re-identification is presented next. It consists of building up anonymous reference sets, supported by mixnets, and mechanisms for blind set lookups. The main steps are also presented in Figure 5. Again, it is based on operations of an ElGamal threshold cryptosystem. Basically, it works as following:

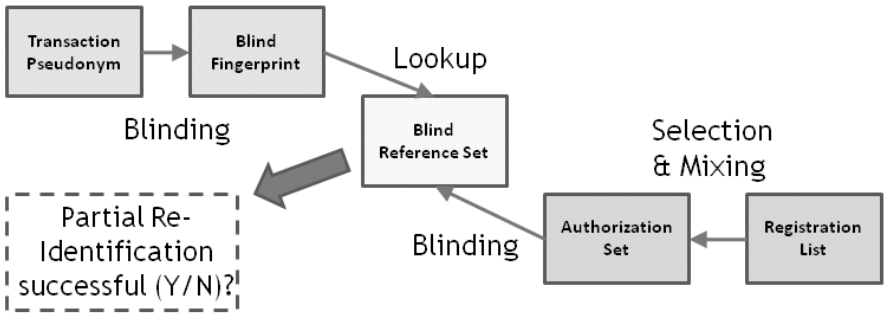

Fig. 5. Steps of partial re-identification

1. *Distributed Key Generation:* Firstly, the participating linkability brokers jointly generate a shared key z, which is used for blinding purposes in later steps.
2. *Selection:* Then, the registration authority selects all entries of the registration list that are relevant to the implicit attribute that shall be checked[10]

[10] The selected entries define the implicit attributes. Given that further information is attached to the registration list, more meaningful implicit attributes can be set up.

(see Figure 6 for an example) and creates encryption of each chosen reference value.

3. *Mixing:* Next, all encryptions are processed by a reencryption mixnet. This creates an anonymized list of the ciphertexts, i.e. the positions of the individual entries in the list as well as the ciphertext representations are changed. The result is called an authorization set.
4. *Blinding:* The linkability brokers cooperatively apply their shares of z to each ciphertext in the authorization set. This process achieves blinding of the reference inside the ciphertext.
5. *Distributed Decryption:* After that, each blinded ciphertext is jointly decrypted. This yields a blinded reference, which is used as a deterministic yet blind fingerprint of the originally chosen reference value.
6. *Reference Set Definition:* All processed blinded references define a blind reference set. The set can be used for matching without leaking the original reference value, by comparing only the blind fingerprints, which is called a set lookup. Defined reference sets can be used in multiple lookups.
7. *Lookup:* In order to execute the lookup, the authorities derive a blind fingerprint of the chosen transaction pseudonym. Then, the lookup on the blind reference set is performed.

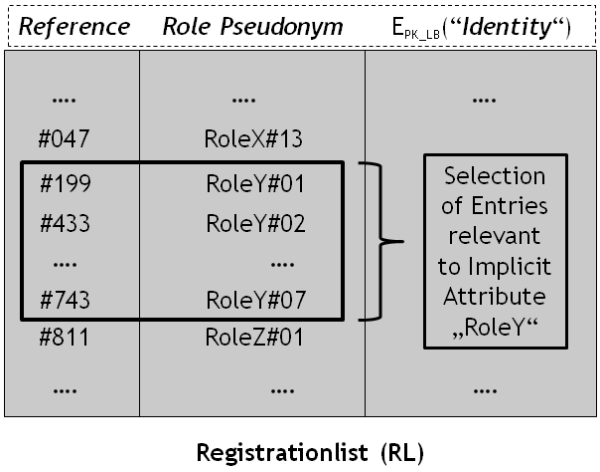

Fig. 6. Example of selection of registration list entries

Partial Re-Identification in Detail Having outlined the abstract steps, we now describe the method in detail. The whole scheme makes use of secret sharing techniques according to Shamir [23] as well as of the distributed key generation

protocol according to Pedersen [19]. Firstly, to jointly generate the secret shared key z used for blinding, the linkability brokers employ the distributed key generation protocol due to Pedersen. In this protocol, each linkability broker LB_j receives a share z_j of the key z. Also, each broker is publicly committed to the share z_j by a public value $\rho_{z_j} = g^{z_j}$, due to the execution of the protocol.

In the following, we describe the complete protocol for *distributed blinding*, which proceeds analog to the distributed decryption protocol [19, 20] of the ElGamal threshold cryptosystem. This protocol can be used to blind an arbitrary element $x \in G_q$ using the shared key z. The following steps are executed in order to cooperatively apply z to the chosen pseudonym(s)[11]:

1. Each linkability broker computes $b_j = x^{z_j}$, a partial blinding of x, by applying its secret z_j. Also, each broker publishes b_j together with a NIZKP for assuring
$$\log_g \rho_{z_j} = \log_x b_j$$
The latter is realized using a non-interactive proof of knowledge for equality of discrete logs [5]. The proof assures that the broker indeed utilized the correct share to produce the partial blinding[12].
2. For any subset Λ of t linkability brokers with valid zero-knowledge proofs, the complete blinded value x^z is reconstructed using the discrete Lagrange interpolation
$$x^z = \prod_{j \in \Lambda} b_j^{\lambda_{j,\Lambda}} \bmod p$$
where
$$\lambda_{j,\Lambda} = \prod_{l \in \Lambda \setminus \{j\}} \frac{l}{l-j} \bmod q$$
are the appropriate Lagrange coefficients.

Now, let $RV_i \in G_q$ be the algebraic representation of a reference value, and $(g^r, h^r RV_i)$ a pseudonym derived from RV_i, with $r \in_R Z_q$. The linkability brokers produce the deterministic fingerprint through the following steps:

1. To each component of $(g^r, h^r RV_i)$ the distributed blinding protocol is applied, blinding it to a fix secret shared exponent $z \in Z_q$:
$((g^r)^z, (h^r RV_i)^z) = (g^{rz}, h^{rz} RV_i^z)$.
2. The blinded pseudonym is jointly decrypted to the blinded reference RV_i^z using the distributed decryption protocol of the threshold ElGamal cryptosystem.

---

[11] Every pseudonym is effectively encoded as two elements of G_q, whereas the second element is directly derived from a reference value. This second element is directly manipulated by the presented mechanisms.

[12] Otherwise, the use of a fake share would lead to an incorrectly blinded value, which would interfere with blind matching purposes. In generally, such a security mechanism is part of every threshold operation.

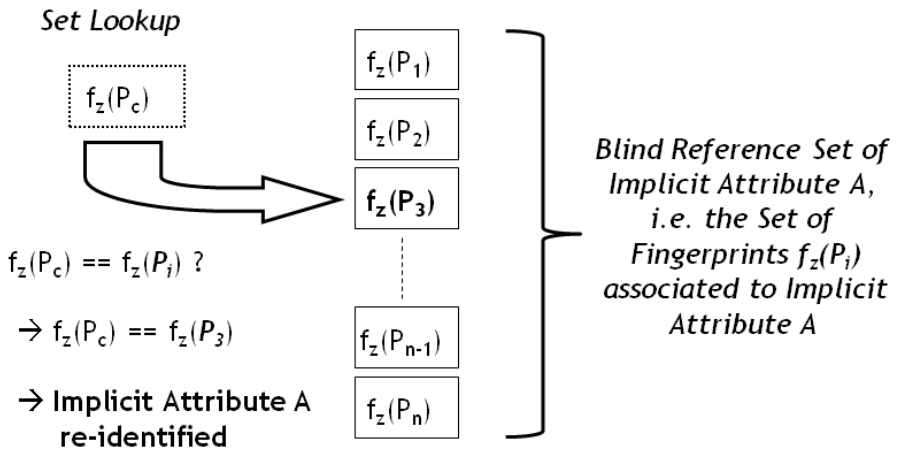

Fig. 7. Lookup

Now, RV_i^z represents a deterministic fingerprint produced with a key z. It is used to perform a lookup that reidentifies implicit attributes, by blindly comparing reference values with a blind reference set that is associated to the attribute. Especially, the presented mechanism supports the re-identification of a recorded transaction pseudonym in several levels of granularity. Hereby, the granularity only depends on the chosen authorization or reference sets. By setting up several different sets and consecutively performing lookups, a multilevel linkability check of a chosen pseudonym is implemented[13]. A lookup is also depicted in Figure 7.

4.5 Complete Disclosure of Pseudonyms

As a complement to the operations for linking and partial re-identification of transaction pseudonyms, we describe the operations for the complete disclosure of pseudonyms in the following. The given approach can be instantiated such that linkability brokers as well as a single law enforcement authority may execute a complete disclosure. In case of the linkability brokers, the disclosure works as follows:

– Firstly, the linkability brokers cooperatively decrypt a transaction pseudonym, which is recorded in a data repository. This yields the plaintext of the distinct reference value encoded in the pseudonym.

[13] Due to the underlying set-based representation, each level of granularity also corresponds to a degree of anonymity as defined by an anonymity set.

– Next, the linkability brokers select the corresponding entry on the registration list[14]. Then, they cooperatively decrypt the deposited ciphertext to reveal the real-world identity.

The disclosure functionality for the law enforcement authority is executed analogously. Basically, we assume that the law enforcement authority is in possession of the same private key as the linkability brokers, by initially having played the role of the trusted dealer in the key generation process. Then, the disclosure proceeds as follows:

– Firstly, the law enforcement authority decrypts a transaction pseudonym, which is recorded in a data repository. This yields the plaintext of the distinct reference value encoded in the pseudonym.
– Secondly, the authority selects the corresponding entry on the registration list[15]. Then, it decrypts the deposited ciphertext in order to disclose the real-world identity.

5 Conclusion

In this article, we described *transaction pseudonyms with implicit attributes*, our approach to multilevel linkable transaction pseudonyms. The proposal extends earlier work of Juels and Pappu on reencryption-based transaction pseudonyms. By making use of threshold ElGamal encryption, PRNGs and secure multiparty computation mechanisms according to the mix-and-match framework, we achieved multilevel linkability of transaction pseudonym for users, linkability brokers and law enforcement authorities.

Transaction pseudonyms with implicit attributes have multiple applications, e.g. to pseudonymous auditing [25, 29], identity management [28], communication privacy [27] as well as privacy protection in pervasive computing settings [26].

References

1. Biskup, J., Flegel, U.: Threshold-Based Identity Recovery for Privacy Enhanced Applications. In: ACM Conference on Computer and Communications Security (CCS '00). pp. 71–79. ACM (2000)
2. Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive Zero-Knowledge. SIAM Journal on Computing 20, 1084–1118 (1991)
3. Catalano, D., Cramer, R., Damgard, I., Crescenzo, G.D., Pointcheval, D., Takagi, T.: Contemporary Cryptology. Birkhäuser (2005)

[14] If the access to the registration list is restricted, LBs are unable to completely disclose a transaction pseudonym. In this case, they have to cooperate with the RA in order to access the matching encrypted identity. Thus, the RA may implement a further security mechanism.

[15] Since a law enforcement authority is dedicated to completely disclose pseudonyms, we implicitly assume that this authority has full access to the registration list.

4. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
5. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Advances in Cryptology - CRYPTO '92. pp. 89–105. Springer (1993)
6. Cramer, R., Damgard, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Advances in Cryptology - CRYPTO '94. pp. 174–187. Springer (1994)
7. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
8. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Advances in Cryptology - CRYPTO '86. pp. 186–194. Springer (1986)
9. Fischer-Hübner, S.: IT-Security and Privacy - Design and Use of Privacy-Enhancing Security Mechanisms. Springer (2001)
10. Flegel, U.: Privacy-Respecting Intrusion Detection. Springer (2007)
11. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: Advances in Cryptology - CRYPTO '01. pp. 368–387. Springer (2001)
12. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. SIAM Journal on Computing 18(1), 186–208 (1989)
13. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Ciphertexts. In: Advances in Cryptology - ASIACRYPT '00. pp. 162–177. Springer (2000)
14. Juels, A., Pappu, R.: Squealing Euros: Privacy Protection in RFID-Enabled Banknotes. In: Conference on Financial Cryptography (FC '03). pp. 103–121. Springer (2003)
15. Koeune, F.: Pseudo-Random Number Generator. In: Encyclopedia of Cryptography and Security, pp. 485–487 (2005)
16. Lindell, Y., Pinkas, B.: Secure Multiparty Computation for Privacy-Preserving Data Mining. Journal of Privacy and Confidentiality 01(01), 59–98 (2009)
17. MacKenzie, P.D., Shrimpton, T., Jakobsson, M.: Threshold Password-Authenticated Key Exchange. J. Cryptology 19(1), 27–66 (2006)
18. Park, C., Itoh, K., Kurosawa, K.: Efficient Anonymous Channel and All/Nothing Election Scheme. In: Advances in Cryptology - EUROCRYPT '93. pp. 248–259. Springer (1993)
19. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party (Extended Abstract). In: Advances in Cryptology - EUROCRYPT '91. pp. 522–526. Springer (1991)
20. Pedersen, T.P.: Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem. Ph.D. thesis, University of Aarhus, Department of Computer Science, Denmark (1992)
21. Pfitzmann, A., Hansen, M.: A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. http://dud.inf.tu-dresden.de/Anon_Terminology.shtml (Dec 2009), v0.32
22. Redz, A.: On Equality Testing Protocols and their Security. Tech. rep., KTH Stockholm (2003)
23. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
24. Tsiounis, Y., Yung, M.: On the Security of ElGamal Based Encryption. In: Workshop on Practice and Theory in Public Key Cryptography (PKC '98). pp. 117–134. Springer (1998)

25. Weber, S.G.: Harnessing Pseudonyms with Implicit Attributes for Privacy-Respecting Mission Log Analysis. In: Conference on Intelligent Networking and Collaborative Systems (INCoS '09). pp. 119 – 126. IEEE CS (2009)
26. Weber, S.G.: Multilaterally Secure Pervasive Cooperation - Privacy Protection, Accountability and Secure Communication for the Age of Pervasive Computing. IOS Press (2012)
27. Weber, S.G., Kalev, Y., Ries, S., Mühlhäuser, M.: MundoMessage: Enabling Trustworthy Ubiquitous Emergency Communication. In: International Conference on Ubiquitous Information Management and Communication (ICUIMC '11). pp. 29:1–29:10. ACM Press (2011)
28. Weber, S.G., Martucci, L.A., Ries, S., Mühlhäuser, M.: Towards Trustworthy Identity and Access Management for the Future Internet. In: 4th International Workshop on Trustworthy Internet of People, Things & Services (Trustworthy IoPTS '10) in conjunction with Internet of Things conference (IoT '10) (2010)
29. Weber, S.G., Mühlhäuser, M.: Multilaterally Secure Ubiquitous Auditing. In: Intelligent Networking and Collaborative Systems and Applications, SCI 329, pp. 207–233. Springer (2010)
30. Yao, A.C.: Protocols for Secure Computations (Extended Abstract). In: Symposium on Foundations of Computer Science (FOCS '82). pp. 160–164. IEEE CS (1982)