

Threshold-based Identity Recovery for Privacy Enhanced Applications*

Joachim Biskup
University of Dortmund
D-44221 Dortmund, Germany
biskup@ls6.cs.uni-dortmund.de

Ulrich Flegel
University of Dortmund
D-44221 Dortmund, Germany
flegel@ls6.cs.uni-dortmund.de

ABSTRACT

Privacy and accountability are potentially conflicting organizational and legal requirements which can be approached by allowing users to act pseudonymously. The reidentification of pseudonyms should be bound to a legal purpose requiring accountability. Existing solutions entrust this function to third parties. Upon good cause shown, these parties perform reidentification on demand. The ability to perform reidentification should be technically bound to the actual existence of a legal purpose, which in some applications can be interpreted as the transgression of a threshold. We present an approach for constructing transaction-based pseudonyms as shares for a suitably adapted version of Shamir's cryptographic approach to secret sharing. Only if pseudonymous actions exceed a threshold specified by a predetermined purpose, can the actor's identity be recovered.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection—*access controls, authentication, cryptographic controls, anonymity*; K.6.5 [Management of Computing and Information Systems]: Security and Protection—*unauthorized access, anonymity*; K.4.4 [Computers and Society]: Electronic Commerce—*security, anonymity*

General Terms

Security, Design, Algorithms

Keywords

privacy, anonymity, pseudonymity, secret sharing, purpose binding

*The work described here is currently partially funded by Deutsche Forschungsgemeinschaft under contract number Bi 311/10-1.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS '00, Athens, Greece.

Copyright 2000 ACM 1-58113-203-4/00/0011 ..\$5.00

1. INTRODUCTION

Recent trends in communication media signify a shift from information broadcasting to information delivery on demand. Transactions involve an increasing number of parties, such as banks handling digital money. Some of these parties, such as infrastructure providers, are involved in an increasing number of transactions of the same person in different contexts. Also service providers handle interest data and other personal data, that is data that can be associated with or identifies real persons.

While the kind of service may inherently require such data (e.g. interest data delivered to Web search engines), personal data often is merely needed to satisfy some legal requirement of the service provider, such as billing or system security. Since such purposes may require establishment of accountability only under certain circumstances, personal data should only be disclosed if these circumstances occur.

In times when companies appreciate the value of personal information, service clients will increasingly appreciate the protection of their valuable personal data. Service providers that can credibly assure their clients the protection of their personal data, may gain a competitive advantage. Along with business related issues, service providers need to account for statutory regulations that may confine processing of personal data. Among the basic principles of informational self-determination, we find the need of either the data subject's informed consent on or the legal or contractual necessity of processing the personal data and, accordingly, a strict purpose binding for collecting, processing and communicating personal data. Pseudonymization, hiding the association between the data and real persons, can help to avoid confinements, as far as the original goals still can be achieved. Exemplary statements are from the forthcoming amendment of the German Data Protection Act [4] and the German Teleservices Data Protection Act in force [5], which specifically addresses teleservices such as those available via the Internet. Both demand collection of as few personal data as possible (Art.2 §3(4) in [5], §3a in [4]) and provisions for anonymous or pseudonymous use of service (Art.2 §4(1) in [5], §3a in [4]). Art.2 §4(4) in [5] permits user profiling exclusively under the condition that profiles are used and not combined with data related to the user. For a discussion of fundamental principles for the protection of personal data and derived paradigms for pseudonymization refer to [2].

Once personal data is available in an IT-processible form, it is hard to enforce accountability of processing, for it can be performed on copies of the data externally to the domain un-

der control. The high complexity, composability and short life cycles of today's IT systems render the application of available verification techniques to these systems impractical. It is exceedingly difficult to rule out that an IT system does not contain a trojan horse. Therefore it is insufficient to legally prohibit malpractice related to personal data. In fact, there is an increasing demand for technical enforcement of privacy principles, reflected by emerging approaches for various applications (see also [1]). Such multilaterally secure systems take into account security requirements of all involved parties and balance contrary interests in an acceptable way. While there are a few unilateral and bilateral security contexts, such as the provisions for a personal trust domain, protection of confidentiality and integrity of communication content, other important security contexts are only achievable multilaterally such as anonymity and unobservability. A multilaterally secure service that allows users to act pseudonymously still can require identity recovery for certain purposes. There are applications where the circumstances accompanying such a purpose can be described as the *transgression of a threshold* t . If reidentification is technically bound to t , we can enforce strong *purpose binding for identity recovery*.

As an example, suppose a service provider granting users a number of free trials, guaranteeing them anonymous access until continued access is liable for costs. Only after t free and anonymous accesses the service provider in this example needs a way to identify the user for billing. If users and service provider do not trust each other, we need a third party authenticating the user and pseudonymizing his actions. This party is trusted by the user to protect his privacy interests, and it is trusted by the service provider to properly generate pseudonyms allowing reidentification. To use the service the user accesses the trusted third party, which pseudonymizes his requests and forwards them to the service provider. Alternatively the service provider could provide a small pseudonymizer-applet to the users' web browser, if he can afford to trust in the user not to interfere with proper generation of pseudonyms allowing identity recovery. The applet's property to protect the user's privacy interests must be verified and certified by an independent third party. The user's requests to the service are then pseudonymized by the applet within the user's browser. Note, that the pseudonymizer-applet needs to store certain data between different sessions to ensure proper generation of pseudonyms. To motivate the user not to interfere with the applet's storage, from the user's point of view, the service quality must increase with forthcoming sessions.

Another example is activity pattern detection for detecting intruders or fraudsters. Only after sufficient activity constituting malicious behavior has been exhibited by a user, his identity should be recovered.

In this paper we propose and study a general approach to pseudonymization of events with the ability for threshold-based identity recovery. In a companion paper [2], focused on privacy respecting intrusion detection, we demonstrate how the general approach fits a specific application.

In our approach we distinguish event collectors observing events, and event recipients, to which collected events are made available for evaluation. Basically, our solution comprises the following features:

- Event collectors are supposed to be under the primary control of, or to be verified and certified by a party

which is trusted by the user, about whom events are collected. The event recipients have to assign (or to enforce) some restricted trust in the collectors.

- Event collectors, either by themselves or by appropriate extensions, forward personal data only if that data is relevant for exceeding the threshold describing the circumstances to be met for reidentification.
- Pseudonymizers are integrated with, or as in our prototype, implanted just behind the event collectors, or their extensions, respectively, and are still under the primary control of, or verified and certified by the same trusted third party, though the event recipients have to assign (or to enforce) some restricted trust in them.
- The pseudonymizers inspect collected events, and they substitute identifiers by carefully tailored pseudonyms. Exploiting Shamir's cryptographic approach to secret sharing, these pseudonyms are actually shares, which can be used for reidentification later on, if the required circumstances are met.
- The circumstances for the purpose of reidentification are defined by a threshold on weighted occurrences of relevant events, and accordingly, by a threshold on shares belonging together. Exceeding such a threshold results in the ability to reveal the secret, i.e. to recover the identity behind the pseudonyms.
- Thus the event recipients, holding the pseudonymized data including the shares, can reidentify the data if and only if the circumstances for the purpose of reidentification are met.
- Since shares are mixed together, we have to take precautions that shares are not inappropriately combined. Resulting mismatches would lead to blame innocent persons.

The rest of the paper is organized as follows. First we survey related work (Sect. 2). Then we present in more detail an application framework for our approach (section 3) and the application of secret sharing to pseudonymization (Sect. 4). We conclude with an outlook on issues for further investigation (section 5).

2. RELATED WORK

Quite some work has been done to allow anonymous use of services of varying kinds, e.g. anonymous public communication systems. We will only refer to approaches here, which in addition to anonymous use offer some degree of protection or deterrence against the misuse of anonymity. We distinguish pseudonyms wrt. their resistance against reidentification through inference on information cumulated related to a given pseudonym. While *entity-based* pseudonyms are used for all transactions of an entity, *relation-based* pseudonyms are used for transactions in the context of a given relation only, and *transaction-based* pseudonyms are used in the context of a given transaction only [10].

Some related work is described in [8], [3] and [6]. They propose modifications to authentication, based on the observation, that traditional access control requires the collection of identifying data, which is required for the resolution of extraordinary circumstances, before access is granted. Instead,

authentication components can collect information strictly necessary for the purpose of access control and require a guarantee, that identifying data can be recovered with the help of an independent third party. More related, but intrusion detection specific approaches are surveyed in [2].

The approaches in [8] and [3] generate transaction-based pseudonyms during user authentication, comprising a zero-knowledge proof, that identity recovery is possible later on. [6] presents an approach based on [7], basically interpreting offline electronic cash as certificates of pseudonyms, leading to entity-based pseudonyms, but not transaction-based pseudonyms as in [8] and [3]. Entity-based pseudonyms provide linkability as required for some intrusion detection methods, respectively, but linkable pseudonyms allow cumulation of information related to the user of the pseudonym, which can eventually enable reidentification. After an initial session login, entity-based pseudonyms could be used in traditional access control scenarios.

Our work and the approaches mentioned above integrate with pseudonyms the ability for identity recovery as a deterrence to misuse. A specific kind of misuse happens when a person shares its pseudonym(s) with further persons. Pseudonym sharing can be a serious threat to commercial services. In [9] it is proposed to design pseudonym systems such that someone who is able to access the system under a pseudonym also is able to reveal some private information about the pseudonym owner. That information would have to be of a kind such that consequences of its misuse outweigh the pseudonym owner's gains by sharing the pseudonym(s). Similarly, the protocols proposed in [13] can be extended such that pseudonym sharing either requires also the sharing of some private information, or is very inconvenient.

Since in aforementioned approaches pseudonyms are introduced before resources are accessed, the identity of actors is protected against the entire accessed service. Also, if users try to use pseudonyms that do not exhibit the properties required by the system, damage can be avoided on the service side, because the pseudonyms are used for access control. In the case of transaction-based pseudonyms this comes at the cost of major efforts required to modify the concerned access control systems traditionally used in off-the-shelf operating systems. On the other hand we were mainly interested in pseudonymous intrusion and fraud detection. If we can afford to place trust in a component to correctly introduce pseudonyms on behalf of the actors, and if we can technically separate observability of identifying accesses from pseudonymous event processing, we can easily integrate pseudonymous audit with major operating systems.

Manual purpose binding for reidentification can be straightforwardly achieved by placing the information needed for identity recovery at a trusted third party, which most of the time lays dormant. Only if a legal purpose requires identity recovery, and upon good cause shown, the trusted third party makes use of the recovery information. Manual purpose binding impedes automatic reidentification. To achieve automatic identity recovery, which can be a desired property of pseudonym systems in some application scenarios, technical purpose binding is required. Note, that it is not sufficient to merely trust the event recipients to adhere to purpose binding, for they are controlled by the service providers.

Different from related work, our approach allows for trans-

action-based pseudonyms that guarantee automatic reidentification independently of a third party, if certain conditions bound to a specified purpose are met. It does so at the cost of always involving trusted components, namely the event collector and the pseudonymizer, to ensure generation of valid pseudonyms.

3. APPLYING THRESHOLD-BASED IDENTITY RECOVERY

In application scenarios we have a small component henceforth denoted as *pseudonymizer* which is trusted by the users as well as the service provider. Trust can be achieved e.g. by involving an independent third party that verifies that the pseudonymizer protects the user's and the service provider's interests regarding anonymity and accountability, respectively. Integrity loss wrt. the pseudonymizer's functionality and configuration should at least be detectable by the user and the service provider. The component complementary to the pseudonymizer is denoted as *reidentifier*. Since in our approach the reidentifier cannot cheat by recovering an identity before the respective threshold is exceeded, it can be controlled by the service provider and users do not need to trust its functionality. To make sense, service providers must be confined to pseudonymized events for the observation of users.

As an application for threshold-based identity recovery, for our prototype we chose to pseudonymize standard OS audit data. We want to cover with one non-invasive approach to pseudonymizer placement as many existing sources for audit data as possible. For our first prototype we chose the *syslog* audit format, which is widely used in Unix systems. Notable network services audit via *syslog* with varying levels of detail. Busy web sites, mail exchangers, or TCP wrappers on hosts with huge login user populations, can generate remarkable amounts of audit records containing personal data. Via *syslog* we can collect host-based event data from applications and the kernel. Using additional kernel modules we can also collect network-based event data. Out-of-band data sources are covered by their supporting applications. Consequently we expect to support applications for threshold-based reidentification in pseudonymous fraud and intrusion detection and other scenarios, as long as the service software produces *syslog*-style audit data. For a thorough discussion of this work refer to [2].

Threshold-based identity recovery can be used in application scenarios where users should act anonymously until a legal purpose requires reidentification, while the purpose comes into effect under conditions or circumstances which can be described as the transgression of a threshold.

We model the *circumstances* as sets of distinguishable events. Such an *event* exhibits an *event designator* allowing event-discrimination, an *event type* from a set of types specifying the kinds of events relevant to the circumstances and specifying the number and types of involved entities, and one or more identifying features describing the entities actually involved in this event. The loose concept of entities here includes, but is not limited to subjects and objects. While the same feature can identify a user in some event contexts, it would not in others. We thus define a *feature type* as the combination of an event type and a *feature designator*, hence a feature is an instance of a feature type. As a result we can pseudonymize features in events that allow

indirect identification of users, but can ignore them in other events. Certainly we have to pseudonymize directly identifying features (*identities*) in all events.

The pseudonymizer receives and handles events before they enter domains where they can be controlled by the user or the service provider. While handling events, the pseudonymizer shall be able to differentiate event types, associate events with different circumstances requiring reidentification, and prioritize events related to the same circumstances. The pseudonymizer accomplishes these requirements based on a priori knowledge about event types, feature types and circumstances. According to the application scenario, the knowledge specifies, which event types occur under which circumstances and the priority of their features' contribution. The same knowledge is available to reidentifiers.

The basic idea of our approach is cryptographically enabling reidentification if an entity has caused at least as many events related to the same circumstances as specified by a threshold. Since application scenarios may require reidentification under several different circumstances described by differing numbers of events of various types, we allow for the grouping of event types, and for assignment of an own threshold t_g to each group. It may be the case that different feature types of a given event contribute with a specific weight to different circumstances. We therefore actually assign *feature types* to *groups* I_g , which represent the circumstances A_g . In addition we assign a weight function $w_{f,g}()$ to each feature type f , representing the priority of f 's specific contribution to A_g .

The knowledge about event types and circumstances is thus represented by triplets $\langle f, I_g, w_{f,g}() \rangle$ per each feature type f . Each triplet expands to a quadruple:

$$\langle \text{event type, feature designator, } I_g, w_{f,g}() \rangle$$

Different types of identifying features may have an identical instance, which represents an identity of aforementioned entities. In other words, there may be multiple occurrences of the identity of the same entity in different feature types of one or more events. We thus can associate the same entity, though not the same feature type, with different circumstances.

As a result, pseudonyms of a specific entity, which are delivered from a given I_g , contribute to the same circumstances and therefore contribute to reach the threshold t_g . This is not the case for pseudonyms of a specific entity, which are delivered from different I_g 's.

The tuples defined above form a decision tree such as in Fig. 1 with a virtual root connected to all specified event types. Each event type is connected to all feature designators contained in tuples with the given event type. I_g and $w_{f,g}()$ are stored with their respective feature designator. For an incoming event the pseudonymizer determines in the decision tree the matching event type. Then the feature types are determined according to the feature designators in the subtree of the event type.

The pseudonymizer determines each identifying feature, henceforth also denoted as identity, retrieves the respective data structure denoted by I_g and checks, whether the identity is already a member of I_g . In case it is not, for an *identity update* an entity entry for the identity is allocated and initialized in I_g .

Finally, it generates $w_{f,g}()$ pseudonyms for the identity and in the event replaces the identity by the generated pseudonyms. The higher the priority of a specific feature type's

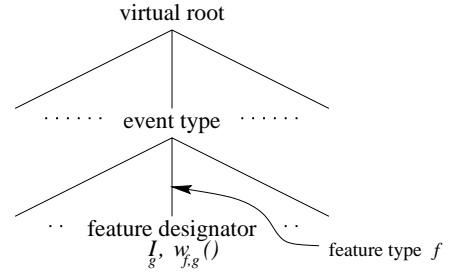


Figure 1: The decision tree used for assigning groups I_g and weights $w_{f,g}()$ to feature types f , based on a priori knowledge

contribution to A_g , according to $w_{f,g}()$, the more pseudonyms are generated for the identity.

Similarly proceeding, reidentifiers can relate incoming pseudonyms with their respective I_g .

4. APPLYING SECRET SHARING TO PSEUDONYMIZATION

The basic idea of our approach is to have a pseudonymizer, acting as representative of the anonymity group of all its users, split an identifying feature id_{i_g} into as many pseudonyms as are needed to pseudonymize events containing id_{i_g} , at maximum¹ $P - 1$ pseudonyms. The pseudonyms shall have the property, that given any t_g , but not less, pseudonyms of id_{i_g} taken from pseudonymous events, a reidentifier is able to recover id_{i_g} . In particular, secret sharing schemes are suitable to fulfill these requirements. For our purposes we exploit Shamir's threshold scheme as described in detail in [12], with some modifications. Shamir's threshold scheme has some desirable properties: it is perfect, ideal and it does not rely on any unproven assumptions. New shares can be computed and issued without affecting preceding shares, and providing an entity with more shares than others, bestows stronger influence upon that entity.

Owing to different conditions of deployment of Shamir's threshold scheme, we make some modifications with regard to its application. Firstly in our scenario we don't have a group of participators, of which each confidentially receives and stores one or more shares, until t_g of them pool their shares for secret recovery. Instead, we have one or more reidentifiers, each of them receiving *all* shares. A reidentifier is always in the position to recover the secret, which is the identity id_{i_g} associated with a polynomial $p_{i_g}(x)$, as soon as it received t_g compatible shares, which are the pseudonyms of id_{i_g} . Additionally, since from the point of view of the pseudonymizer all reidentifiers are potential adversaries, the confidentiality requirements regarding shares cease to apply.

While in conventional applications of secret sharing schemes it is feasible to estimate the number of shares preliminarily to be issued, the same is impractical in our scenario, for it is unknown which identities in the near future will require pseudonym generation. We thus take a stepwise approach to the choice of x -coordinates and to share generation, and we distribute $p_{i_g}(x)$ paired with its respective x . We have to preclude linkability wrt. the x -coordinates of shares within a group of feature types. If we chose the same

¹The secret sharing scheme we apply operates over $GF(P)$.

x -coordinate for shares of different identities being members of the same group I_g , which is allowed by Shamir’s scheme, it were obvious, that the shares belong to different identities. Accordingly, we choose unique x -coordinates for all shares wrt. a given I_g , e.g. by counting². Since we choose the polynomials p_{i_1} and p_{i_2} independently for the same identity $id_{i_1} = id_{i_2}$ in two groups I_1 and I_2 , we are free to use the same x -coordinates in different groups.

Normally, certain products of pairwise combinations of the x -coordinates of shares pooled for secret recovery can be precomputed and issued to the participants in order to improve the performance of secret recovery using Lagrange interpolation. Since the number of shares going to be issued for a given identity is unknown beforehand, one would have to issue an increasing number of precomputed intermediary results together with each share. In order to profit from these intermediary results, a reidentifier would have to store them all, because it generally does not know, of which shares it is going to make use of, and of which not. We therefore refrain from this optimization.

4.1 Mapping Secrets to Identities

As the assignment of identities to different feature type groups I_g and respective thresholds t_g is just a matter of applying rules (see Sect. 3), for our forthcoming discussions we regard a given I_g and omit the index g wherever applicable.

In our approach we do not share identities directly. Instead, we assign a secret unique wrt. I to each identity and then share this secret. For reidentification after secret recovery we need a pseudonymizer-generated function (or table) that maps secrets to identities, and in doing so, fulfills some basic security requirements:

- Since the party using the reidentifier needs to control the host on which the reidentifier operates, and that party in our attacker model is a potential adversary with regard to the pseudonymity of events, we cannot rely on the operating system to protect the confidentiality of the mapping. In its basic version the mapping from secrets s_i to identities id_i is accomplished by storing cryptograms $c_i := E(k_i, id_i)$, regarding the decryption key k_i as the secret s_i , matching the encryption key k_i , pseudo-randomly chosen and unique wrt. I . The uniqueness requirement frustrates the recovery of $id_j, j \neq i$ before t shares for id_j have been issued. After an identity update (see Sect. 3) I is to be made available to the reidentifiers.
- Once recovered, an identity shall not be indefinitely linkable to forthcoming events caused by the respective entity. We have to trade the anonymity of entities off against the linkability of events wrt. the identity of the causing entities. Since this linkability is required in some application scenarios (e.g. intrusion detection) to some degree, we carefully have to trade off anonymity against accountability. Other parameters include but are not limited to performance and storage requirements for handling growing numbers of shares and limitations to the number of issueable shares, and the performance penalty imposed by limiting measures

²Counting is cheaper than choosing unique x -coordinates pseudo-randomly.

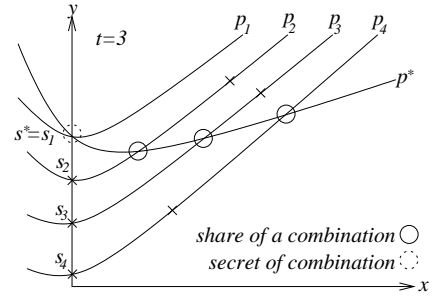


Figure 2: A mismatch: the solution s^* of a combination of t shares, not all stemming from the same polynomial, matches a secret s_1 , though it should not

as well as their consequences. To limit linkability we change the mapping after expiry of an *epoch*, but during an epoch, identities are not deleted from the mapping. Epoch expiry effects either that the mapping is discarded and rebuilt on demand during the next epoch, or that appropriate rekeying of all identities is performed.

4.2 The Mismatch Problem

Provided the pseudonymizer issues shares for id_i as unmarked pairs $(x, p_i(x))$, a reidentifier cannot determine which shares within a group I belong to the same identity, unless it tries to recover an identity from each combination of t shares.

When ‘blindly’ choosing combinations of t shares, a reidentifier will not always draw t shares stemming from the same polynomial. Consider a reidentifier, which, as depicted in Fig. 2, chooses three shares stemming from three different polynomials p_2, p_3 and p_4 . The solution p^* of the linear equations in this case matches in $s^* = p^*(0)$ the secret of p_1 . If the reidentifier recovers the respective identity id_1 , correct accountability could not be established. Accordingly, if the solution of a combination of t shares matches a secret s_i in I , it is denoted as a *valid match* if all shares are *compatible*, i.e. stem from the same polynomial p_i , otherwise it is called a *mismatch*.

4.3 Tackling the Mismatch Problem

A straightforward approach to the mismatch problem discussed in Sect. 4.2 is to supply reidentifiers with information enabling them to determine, whether two shares stem from the same polynomial. One, but not the best, method to accomplish this is marking all shares stemming from the same polynomial p_i , and thus from the same identity id_i within I , with an identical label.

Following we give the steps necessary for pseudonym generation by secret sharing, performed by the pseudonymizer (see also the upper left section of Fig. 3): For each incoming event perform the steps provided in Sect. 3 to extract the embedded identifying features or identities id , and determine for each id the respective group I and the pertinent weight $w_f()$. For each triplet $(id, I, w_f())$ perform the following steps:

Identity update: If the data structure for the designated I not yet exists, create it and initialize its x -coordinate counter x . If id is not yet member of I , create an entity entry e_i in

I , choose a pseudo-random encryption key \tilde{k}_i uniquely wrt. I , encrypt id as $c_i := E(\tilde{k}_i, id)$ and store c_i in $e_i := \langle c_i \rangle$. As required by treating the respective decryption key k_i as a secret s_i to be shared, annotate the new entity entry e_i with: $id_i = id$, $s_i = k_i$, an initialized share counter m_i , a label l_i chosen uniquely wrt. I , and $t - 1$ pseudo-randomly chosen coefficients $a_{i,1}, \dots, a_{i,t-1}$ for the polynomial $p_i(x) = s_i + \sum_{j=1}^{t-1} a_{i,j} \cdot x^j$. Make I , but not the annotations, available to the reidentifiers.

Secret sharing: determine the label l_i from e_i 's annotations, and perform the following steps $w_f()$ times: increment the x -coordinate counter x of I and the share counter m_i in e_i 's annotations; then compute the next share for s_i as $r_{i,m_i} := \langle x, p_i(x) \rangle$, and finally embed r_{i,m_i} in the event in place of the feature id_i . When all $w_f()$ shares have been embedded, make the rewritten event and the label³ $l_{i,m_i} := l_i$ available to the reidentifiers.

Following we give the steps necessary for reidentification by Lagrange interpolation of t compatible shares, performed by the reidentifier (see also the upper right section of Fig. 3).

Reidentification: choose a combination of t shares $\langle x_{i_1}, y_{i_1} \rangle, \dots, \langle x_{i_t}, y_{i_t} \rangle$, all of them exhibiting the same label; then compute $s^* := p^*(0) = \sum_{j=1}^t y_{i_j} \prod_{1 \leq o \leq t, o \neq j} \frac{x_{i_o} - x_{i_o}}{x_{i_j} - x_{i_o}}$ and subsequently use s^* as $k_i = s_i$ to yield $id_i := D(k_i, c_i)$ from the current I , determined as described in Sect. 3.

4.4 Variations and Extensions

The approach from Sect. 4.3 has a weakness concerning the unlinkability of events. Events caused by the same identity in a feature type group I , are linkable wrt. the labels, even before t pseudonyms have been issued⁴. In other words: if we always mark them, we merely have relation-based, not transaction-based pseudonyms. Subsequently we suggest and compare three further approaches to mismatch handling avoiding this problem.

4.4.1 Mismatch Avoidance by Deferred Marking

Instead of immediately letting reidentifiers know which shares are compatible, we can defer this information until t compatible shares have been issued. Before this condition holds, reidentifiers can't recover the respective secret, anyway. The pseudonymizer thus issues the first $t - 1$ pseudonyms for an identity without disclosing their labels. When issuing pseudonym number t , it additionally delivers the labels of the first t pseudonyms. Subsequently labels can be issued timely with their pseudonym.

4.4.2 Mismatch Avoidance without Marking

Instead of marking labels, the pseudonymizer can take precautions, altogether avoiding the occurrence of mismatches. To enable the pseudonymizer and reidentifier to check for matching combinations of t shares, we have to provide a verifier v_i with each entity entry in I such that $e_i = \langle v_i, c_i \rangle$. In its basic version, verifiers are just the value of a secure one-way hash function $H()$ applied to the secret s_i , e.g. a message digest. To check, whether a solution s^* for a combination of shares matches, the solution is hashed

³Since the labels for all shares of the same e_i are identical, it is sufficient to mark collectively distributed shares with just one label.

⁴It would therefore be sufficient to choose x -coordinates unique wrt. each p_i .

and the result is matched against the verifiers in I (see also the lower section of Fig. 3).

Before issuing a share, the pseudonymizer checks, if all combinations of t incompatible shares, each including the new share, are free of mismatches. If a mismatch is found, the share is discarded and another one is chosen by incrementing x . Since the reidentifier cannot look at the labels to find compatible shares, it has to test if any combination of t shares, each including the incoming share, matches using the v_i .

Searching (mis)matching combinations is quite expensive and can be implemented recursively, falling back on intermediary results. The mismatch handling approaches based on marked pseudonyms do not require 'blindly' combining shares and therefore are more efficient than this approach for both, the pseudonymizer and the reidentifier.

4.4.3 Mismatch Verification without Marking

We have to keep in mind, that pseudonymizers probably need to reside on production machines to be able to protect their users' privacy. It is thence imperative to minimize the performance penalty imposed upon the machine by the pseudonymizer. Reidentifiers instead could use dedicated machines. We can modify the aforementioned mark-free protocol by not having the pseudonymizer avoid mismatches, but having the reidentifier validate matches with the pseudonymizer's help. Note, that we trade performance for the ability to reidentify entities while being offline.

The pseudonymizer in this case takes no precautions to avoid mismatches and just issues unmarked pseudonyms. As in the protocol above, the reidentifier 'blindly' searches for matching combinations of t pseudonyms. It then verifies the validity of a match by providing the pseudonymizer with a reference to I , the share combination, a fresh nonce and the secure one-way hash value of the concatenation of the nonce and the matching solution. Note, that the requirement of a fresh nonce prevents reidentifiers using verifiers from I to validate share combinations instead of solving the respective equations. The pseudonymizer accepts the query if the nonce is fresh, by means of the hash value searches I for the appropriate entity entry e_i , and uses the respective polynomial to test whether the supplied shares are compatible. If the shares all stem from that polynomial, the pseudonymizer signals the validity of the match to the reidentifier, in all other cases, it signals a mismatch. Note, that the pseudonymizer must retain the annotations of I even after the current epoch expires.

For each mismatch handling approach we already have pointed out the specific benefits and drawbacks. In Table 1 we give a brief overview of all approaches. The rows refer to the properties being compared: *privacy* refers to the unlinkability of events from an entity related to specific circumstances before t shares have been issued. If records are unlinkable, we assign a '+'; if *offline validation* can be implemented, a '+' is assigned; the other rows denote the *performance penalty* imposed by the pseudonymizer ' P ' and the reidentifier ' R ', respectively. It can be seen, that the approach *mismatch avoidance by deferred marking* has the most desirable properties.

4.4.4 Extended Threat Model

Hitherto we regarded reidentifiers as potential adversaries and equated them with other attackers. In some application

Table 1: A comparison of all proposed mismatch handling approaches

	avoidance by marking	avoidance by deferred marking	avoidance no marking	verification no marking
privacy	-	+	+	+
offline validation	+	+	+	-
performance penalty P	+ low	+ low	- high	+ low
performance penalty R	+ low	+ low	- high	- high

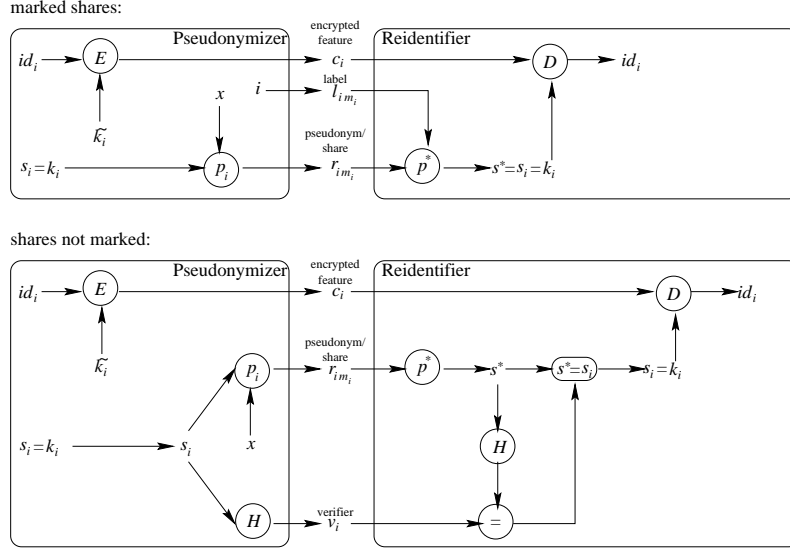


Figure 3: The flow of data between pseudonymizer and reidentifier

scenarios it might be profitable to raise further obstacles for attackers being no reidentifiers. To do so, we initially perform a secure key exchange, such that both, pseudonymizer and reidentifier, know an encryption key k_e . The reidentifier knows the respective decryption key k_e . Secrets being shared then are not defined as the keys k_i needed for decrypting the c_i . Instead, k_i is encrypted under k_e to form a secret $s_i := E(k_e, k_i)$. We also pondered about alternatively encrypting shares. For mismatch handling without marking it is expensive to widely protect shares in the memory storage of reidentifiers against attackers, for each share were to be decrypted before interpolating a share combination. If we abandon in-memory protection of shares by decrypting them on reception, as an alternative we might protect the entire transmission channel using available solutions.

In case no labels are used, we alternatively could increase the efforts of an attacker by using a keyed one-way hash-function $H^k()$ instead of $H()$ such that $v_i := H^k(k_v, s_i)$, $s_i = k_i$, where the symmetric key k_v has been securely exchanged between pseudonymizer and reidentifier.

4.4.5 Coarsening the Number of Potential Actors

Since we use a kind of transaction-based pseudonyms, it would be advantageous to hide the number of actually involved entities in order to make it harder for an adversary to apply external knowledge. For obvious reasons this cannot be achieved when using non-deferred marking for mismatch handling. We introduce dummy entities in I , which

are treated like entries for real identities⁵, except that they are annotated as dummies. The id_i for dummies shall be randomly drawn from the identity database on the pseudonymizer host. In case no mismatch handling is done, however we have to choose fictive identities which do not match any real identities in the host's identity database. Otherwise in the case of a mismatch we couldn't discern dummies from real identities. Fictive identities also adhere to the operating system's rules concerning identity naming. Further on we considered using dedicated dummy names, pseudo-random bit-strings for dummy naming and invalid verifiers v_i , but all of these are easier to spot as dummies in case a brute force attack on k_i is tried.

We can make real identities indistinguishable from dummies, as long as they have not been reidentified. In order to do this, we have to provide for the ability to completely change the apparition of all entity entries in I , without needing to change the information within the entries. Since all items used in our approach so far are invariant during an epoch, we introduce some independently and pseudo-randomly chosen seed values, which can be varied during an epoch. We use randomized encryption for the identity cryptograms $c_i := E^r(k_i, rand_i, id_i)$ using the pseudo-random seed $rand_i$. Also we have to make sure all c_i exhibit the same size, e.g. by appropriate padding. In case shares are not marked, we additionally choose for and distribute with each I a pseudo-random value $vrand$ to randomize the ver-

⁵Note, that this also applies to mismatch avoidance without marking.

ifiers v_i by hashing $vrand$ together with the secret s_i . If we change just $vrand$ and $crand_i$ for an e_i in I , an attacker cannot distinguish the effect on c_i and v_i from the effect by varying additional items such as the identity (id_i) and keys (k_i, k_e, k_v) respectively.

The maximum number of potential actors is bounded⁶ by the number rid_{max} of identities in the system’s database. If rid_{max} is large and the expected maximum number of actors per epoch is $u \ll rid_{max}$, and the expected number of shares generated per actor is m , then the maximum number of potential actors is bounded by $u \cdot m \leq rid_{max}$. Heuristically we allocate rid_{max} entity entries for I if $u \approx rid_{max}$, otherwise we allocate $u \cdot m \leq rid_{max}$ entity entries. If the actual number of actors exceeds $u \cdot m \leq rid_{max}$, I needs to be extended appropriately by $u \cdot m$ entity entries. Note, that the number of actors is known to be $(u \cdot m \cdot n) + 1, n = 1, 2, \dots$ each time I is extended. On initialization, all entries are dummies, and we generate the cryptograms c_i and verifiers v_i as usual.

On identity updates, we choose a dummy entity entry e_i to be replaced, choose a new key pair \tilde{k}_i, k_i and new seeds $crand_i$ and $vrand$, and compute c_i (and v_i). To hide, which dummy e_i was replaced, we also change the random seeds $crand_j, j \neq i$ of all other entries, effectively requiring computation of all cryptograms (and verifiers). For mark-free mismatch handling or if the implementation of labels allows, we also may randomize the order of I ’s entries. We make the new I available to the reidentifiers.

If an attacker observes within an epoch the actual number upd of updates of I , and the actual number m' of shares issued from I , he can infer from $\frac{m'}{upd} > m$ that there are fewer than u actors. To prevent this, we generate appearingly random “red herring” updates of I such that $\frac{m'}{upd} \leq m$. For “red herring” updates, we change the random seeds of I ($vrand$) and of all its entries ($crand_i$). Accordingly, we may also randomize the order of I ’s entries, and subsequently make I available to the reidentifiers.

5. OPEN ISSUES AND FURTHER RESEARCH

There are a number of promising areas for future research related to our approach. Practical limitations imposed by trust requirements could be overcome by integrating with pseudonyms a proof of validity, guaranteeing that identity recovery is possible later on. Regarding anonymity we are interested in further methods for reducing the granularity of reidentifications, as well as potential benefits and required properties of dummy events. If we can do without the decoupling of identities and secrets, we don’t need to solve the problems associated with the secret-to-identity mapping. Our approach is extendible to allow associating a specific feature type with more than one group, which may be needed for modeling certain purpose circumstances. Also exploiting secret sharing access structures for modeling the weight functions seems auspicious. In addition we will focus on intrusion detection related issues. In this context a very promising area is the derivation of pseudonymization parameters from attack contexts extracted from knowledge data in intrusion detection systems. Other directions worth investigating are pseudonym linkability and other issues concerning epoch transition and distributed pseudonymization. A final area of investigation is the complementation of audit pseudonymization with detectability of integrity loss. This might be achieved by marrying our approach with the one of Schneier and Kelsey [11].

⁶In special cases we cannot hide the number of actors, e.g. if several active identities sequentially trigger the issuing of a bulk of t or more shares each.

6. REFERENCES

- [1] J. Biskup. Technical enforcement of informational assurances. In S. Jajodia, editor, *Proceedings of the 12th international IFIP TC11 WG 11.3 Working Conference on Database Security*, pages 17–40, Chalkidiki, Greece, July 1998. IFIP, Kluwer Academic Publishers.
- [2] J. Biskup and U. Flegel. Transaction-based pseudonyms in audit data for privacy respecting intrusion detection. In *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000)*, Toulouse, France, Oct. 2000. Springer.
- [3] D. Boneh and M. Franklin. Anonymous authentication with subset queries. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 113–119, Kent Ridge Digital Labs, Singapore, Nov. 1999. ACM SIGSAC.
- [4] Bundesministerium des Inneren. Entwurf zur Änderung des BDSG und anderer Gesetze (in German), July 1999.
<http://www.datenschutz-berlin.de/recht/de/bdsg/bdsg0607.htm>.
- [5] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie. Federal act establishing the general conditions for information and communication services — information and communication services act. *Federal Law Gazette I*, 52:1870, June 1997.
http://www.datenschutz-berlin.de/recht/de/rv/tk_med/iukdg_en.htm.
- [6] Y.-Y. Chan. On privacy issues of internet access services via proxy servers. In R. Baumgart, editor, *Secure Networking – CQRE[Secure]’99*, number 1740 in LNCS, pages 183–191, Düsseldorf, Germany, Nov. 1999. secunet, Springer.
- [7] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Proceedings of the Conference on Advances in Cryptology (CRYPTO’88)*, LNCS, pages 319–327, Santa Barbara, CA, Aug. 1988. Springer.
- [8] J. Kilian and E. Petrank. Identity escrow. In *Proceedings of the Conference on Advances in Cryptology (CRYPTO’98)*, pages 196–185, 1998.
- [9] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography (SAC’99)*, pages 184–199, Kingston, Ontario, Canada, Aug. 1990. Springer.
- [10] B. Pfitzmann, M. Waidner, and A. Pfitzmann. Rechtssicherheit trotz Anonymität in offenen digitalen Systemen (in German). *Datenschutz und Datensicherheit*, 14(5-6):243–253, 305–315, 1990.
- [11] B. Schneier and J. Kelsey. Cryptographic support for secure logs on untrusted machines. In *Proceedings of the First International Workshop on the Recent Advances in Intrusion Detection (RAID’98)*, Lovain-la-Neuve, Belgium, Sept. 1998. IBM Emergency Response Team. http://www.zurich.ibm.com/~dac/Prog_RAID98/Table_of_content.html.
- [12] D. R. Stinson. *Cryptography — Theory and Practice*, chapter Secret Sharing Schemes, pages 326–331. Discrete mathematics and its applications. CRC Press, first edition, 1995.
- [13] S. G. Stubblebine, P. F. Syverson, and D. M. Goldschlag. Unlinkable serial transactions: Protocols and applications. *ACM Transactions on Information and System Security*, 2(4):354–389, Nov. 1999.