# Pseudonymizing Unix Log Files

Ulrich Flegel
University of Dortmund
D-44221 Dortmund, Germany
ulrich.flegel@udo.edu

## Abstract

Unix systems in many cases record personal data in log files. We present tools that help in practice to retrofit privacy protection into existing Unix audit systems. Our tools are based on an approach to pseudonymizing Unix log files while balancing user requirements for anonymity and the service provider's requirements for accountability. By pseudonymizing identifying data in log files the association between the data and the real persons is hidden. Only upon good cause shown, such as a proceeding attack scenario, the identifying data behind the pseudonyms can be revealed. We develop a trust model as well as an architecture that integrates seamlessly with existing Unix systems. Finally, we provide performance measurements demonstrating that the tools are sufficiently fast for use at large sites.

## 1   Introduction

For diverse reasons most internet services store information about their customers' requests on a long term basis. One of the reasons to do this is a service provider's need to be able to establish accountability for certain requests. Regardless of the required effort, IP addresses can often be linked to an actual person accessing a service. Since the stored request information contains personal data, we need to consider customer demands regarding privacy. In times when companies appreciate the value of personal information, customers will increasingly appreciate the protection of their valuable personal data. Survey after survey shows that privacy concerns with respect to the use of the internet are on the rise [4, 5, 6]. A gap has been shown between what kind of identifying data users think ought to be collected versus what kind they think actually is collected when accessing a web site. The biggest disagreement is shown for some of the data commonly logged by web servers such as machine name or IP address, browser and OS [6]. Most users would rather not access a site than reveal personal information asked by the site for registration [6]. The vast majority of users strongly values being able to anonymously use internet services [6]. This situation calls for privacy enhancing technologies that can be used today and that help service providers to comply with user expectancies as well as with legal regulations concerning personal data. Service providers that can credibly assure their customers the protection of their personal data may gain a competitive advantage over those that can't. Consider as a simple example a web site where the service provision requires no user registration and no billing. There is a reasonable user expectancy to be able to access the site anonymously. Nevertheless most sites are reluctant not to retain web server logs containing client IP addresses to be able to track perpetrators.

Furthermore, prevalent statutory regulations confine the processing[1] of personal data. Widely adopted preconditions for the processing of personal data are the informed consent of the data subjects or a legal or contractual necessity. Fundamental processing principles comprise the prior announcement of the purpose of processing, processing only data relevant for the purpose and the fixation of the scope of data processing. Considering our example web site the users would not only have reason to expect, but they would also have a right to be able to use the site anonymously, based on the pertinent legal situation in many western countries. Refer to [2] and [7] for a more detailed discussion of the related legal issues. Unfortunately it is insufficient to merely legally interdict the misuse of personal data.

---

[1] Processing, in relation to personal data, covers virtually the entire data life cycle from collection, through to erasure of the data when no longer required.

Once personal data has been collected its processing is hard to control in a non-technical way, since any processing can occur on arbitrary copies of the data. Due to the scale and complexity of today's IT systems it is impractical to completely rule out the existence of trojan horse code copying personal data. Considering the example web site it is practically futile for a user to determine or even to prove that the organizational controls are ineffective wrt. the protection of his personal data.

Therefore we observe an increasing demand for strong technical enforcement of privacy principles. Multilaterally secure systems take into account and balance conflicting security requirements of all involved parties [8]. Accordingly a multilaterally secure internet service allows customers to act pseudonymously as long as violations of the service security policy are still detectable. In return the service provider is allowed to recover the identities of customers upon good cause shown, which supports the hypothesis of violations of the security policy. Translated to the example web site this means that the IP addresses and potentially other identifying data are pseudonymized when the web server logs are generated. Only if a client exploits a vulnerability of the web server its IP address and other identifying data is recoverable.

This paper demonstrates that the approach to threshold-based identity recovery we designed in [3, 2] is actually implementable and applicable in practice. Specifically the contributions of this paper are the following:

- the development of a practical architecture to integrate our pseudonymizer with existing *syslog* auditing infrastructures

- while taking into account a trust model supported by actually deployed systems, and

- the provision of tools that implement our concepts and architecture, as well as

- a detailed evaluation of the performance of our pseudonymizer, indicating that it lives up to the requirements posed by the audit data volume generated by a busy server system at the Center for Communication and Information Processing at the University of Dortmund.

In a nutshell, our solution comprises the following features:

- The audit data sources, f.i. *syslog*, are supposed to be under primary control of the personal data protection official (PPO) who enforces the personal data protection policy of the site. The users of the site trust the PPO to protect their personal data and the site security officers (SSO) assign trust in the *syslog* configuration the PPO provides.

- Audit data sources are configured to forward only audit data required for sustained service provision, such as audit data indicating proceeding attacks against the site. The audit data is intercepted by a local *pseudonymizer* before *syslog* forwards it to remote hosts or stores it on disk. The pseudonymizer also is controlled by the PPO.

- The pseudonymizer inspects the audit data and substitutes identifying features by carefully tailored pseudonyms. Exploiting Shamir's cryptographic approach to secret sharing, these pseudonyms are actually shares, which can be used for feature recovery later on, if justified by sufficient suspicion. The pseudonymized audit data retains the format of the unpseudonymized version. The pseudonymizer forwards the audit data to the local *syslog* service, which processes it and forwards the results to the remote *syslog* service of the SSO.

- The pseudonyms of a given identifying feature are linkable wrt. given suspicions. Thus, the SSO can analyze the pseudonymized audit data to find symptoms raising sufficient suspicion, justifying the recovery of the corresponding identifying features.

- Sufficient suspicion is defined by a threshold on weighted occurrences of potentially attack-related behavior, and accordingly, by a threshold on shares belonging together. Exceeding such a threshold results in the ability to recover the respective features.

- Feature recovery is conducted by the SSO using our *reidentifier* tool if the need arises. Thus, the SSO can recover identifying features if and only if the conditions of sufficient suspicion are met.

The remainder of this paper is organized as follows: Section 2 justifies our design decision to pseudonymize *syslog* audit data and introduces the

relevant concepts. In Sect. 3 we detail the design decisions for integration of the pseudonymizer, explain the trust model and outline our approach to pseudonymization. The architecture and practical use of our implementation also are illustrated in Sect. 3. In Sect. 4 we evaluate the performance of the implementation. We position our contribution wrt. related work in Sect. 5 and conclude in Sect. 6.

## 2 A Case For *syslog*-Pseudonymization

Most modern operating systems offer comparable audit components, e.g. AIX, BSDs, HP-UX, Linux, Solaris, Windows-NT and its descendants. All of these record the identity of the actor of an event or at least identifying features of users. Audit components specifically designed for compliance with the Trusted Computer System Evaluation Criteria (TCSEC) (C2 or higher) [9, 10] or the Common Criteria (`FAU_GEN1.2` and `FAU_GEN2`) [11] are required to record information identifying the originator of an event. Hence, all of these audit components record personal data and need to be considered for pseudonymization. In annex C the Common Criteria provide for the application of the Privacy class (`FPR`) to `FAU_GEN2`. The Privacy class `FPR` contains the families Anonymity `FPR_ANO`, Pseudonymity `FPR_PSE`, Unlinkability `FPR_UNL` and Unobservability `FPR_UNO`. The Pseudonymity family (`FPR_PSE`) provides accountability and `FPR_PSE2` offers reversible pseudonyms liable to specified reidentification conditions. Basically our approach is implementing `FPR_PSE2` for `FAU_GEN`. In addition to the audit components of the operating system, application processes record audit data containing personal data, such as web servers and mail exchangers.

### 2.1 Event Sources

When considering pseudonymization of audit data all relevant event data sources mentioned above should be covered. With our approach we aim at a non-invasive, easy to install tool that covers as many categories of audit data as possible. We therefore decided to build a pseudonymizer for *syslog* audit data. Due to its uniformity and availability in most significant Unixes many event sources in the Unix world leverage *syslog* for recording audit data.

Also active network components such as routers provide for remote audit recording using *syslog*. Finally there are several third party products available for Windows driven systems to integrate with an existing Unix *syslog* infrastructure. By supporting *syslog* we can pseudonymize host-based, network-based and out-of-band audit data. Host-based audit data relates to events taking place within a host system and it is also collected on that host system. *Syslog* supports host-based audit data generated by most network services and by the kernel. Examples for host-based audit data not covered via *syslog* are accounting records and system-call events. Network-based audit data is generated by components associated with network resources, where data is handled at abstraction levels below the application layer. Our tool covers network-based audit data f.i. generated by firewalls, routers and end systems protected by IP traffic filters, such as the *ip filter* kernel module [12], which logs via `ipmon` to `syslogd`. Out-of-band audit data refers to event sources not monitored by the IT system. Out-of-band data can be covered if applications are made aware of external events and if these applications use *syslog* to record these events. Event sources not using *syslog*, but directing their audit data to dedicated files are covered by our *redirector* (see Sect. 3.2.4). In practice with our tools we cover most of the audit data usually collected in Unix systems. In certain environments additional audit data is collected and needs to be pseudonymized, which is not yet covered by our tool. For a more elaborate discussion on the diverse sources for audit data in Unix systems refer to [2].

### 2.2 Audit Data

As described above, host-based, network-based and out-of-band audit data converge in *syslog*. The audit data collected via *syslog* describes events and conditions monitored by diverse components of the IT system. These components are categorized as *facilities* in *syslog* parlance. A facility that monitors an event or condition supplies a rating of its *severity* [13, 14]. The *priority* of an event record is defined as the pairing of its generating facility and its severity. The severity ranges from debugging and informational over error conditions to emergency conditions. Mostly informational and diagnostics messages from the kernel and from network services are recorded via *syslog*. Some sample audit records are shown in Fig. 1. For the sake of presentation we

```
 1 ftpd[7427]: xferlog (send): 30 pony.puf 240369 0 240369 funstuff.tar.gz b _ o g dexter ftp 0 * c
 2 ftpd[7427]: xferlog (send): 28 pony.puf 243943 0 243943 funstuff.tar.gz b _ o g dexter ftp 0 * c

 3 pppoed[15459]: invalid state a7
 4 pppoed[15459]: Failed to discover server!

 5 Pluto[13321]: Starting Pluto (FreeS/WAN Version 1.9)
 6 Pluto[13321]:   including X.509 patch (Version 0.8.1)
 7 Pluto[13321]: X.509 certificate file '/etc/x509cert.der' not found
 8 Pluto[13321]: OpenPGP certificate file '/etc/pgpcert.pgp' not found
 9 Pluto[13321]: listening for IKE messages
10 Pluto[13321]: FATAL ERROR: bind() failed in find_raw_ifaces4(). Errno 13: Permission denied
12 sendmail[23248]: gethostbyaddr(192.168.3.1) failed: 1
13 wu.ftpd[4657]: warning: /etc/hosts.allow, line 12: can't verify hostname: gethostbyname(mport80.minerva.com.au) failed
14 wu.ftpd[4658]: warning: /etc/hosts.allow, line 12: can't verify hostname: gethostbyname(mport80.minerva.com.au) failed

15 kernel: LIDS: lidsadm (22 5 inode 28746) pid 14419 user (0/0) on ttyp181: try to open /etc/lids/lids.conf for reading
16 portsentry[848]: attackalert: SYN/Normal scan from host: pony.puf/192.168.1.4 to TCP port: 21
17 tcplog[1040]:      SYN            RES2 : ftp from 192.168.1.4 port 45691
18 tcplog[1040]: FIN SYN     PSH     URG      : ftp from 192.168.1.4 port 45693
19 tcplog[1040]: FIN SYN     PSH     URG      : ftp from 192.168.1.4 port 45693
20 tcplog[1040]: FIN         PSH     URG      : port 34513 from 192.168.1.4 port 45697
21 tcplog[1040]: FIN         PSH     URG      : port 34513 from 192.168.1.4 port 45697
22 tcplog[1040]: QUESO: port 34513 from 192.168.1.4 port 45697
23 login[1510]: ROOT LOGIN on 'tty1'
24 login[12571]: ILLEGAL ROOT LOGIN on 'ttyS1'
```

Figure 1: Sample *syslog* audit records

numbered the records and omitted the time stamp and host fields (`TS` and `H` in Fig. 2). Fig. 1 contains informational audit records (numbers 1 and 2), diagnostic audit records (numbers 3 and 4), diagnostic audit records related to security (numbers 5 through 14) as well as possibly attack related audit records (numbers 15 through 24).

Also serious actions of users associated with login sessions are recorded by *syslog*. Depending on the system configuration a variety of other sorts of events may be reported to *syslog*. As an example the intrusion detection system *Snort* [15, 16] is capable of reporting attack detections to *syslog*. Basically, *syslog* audit data is collected and used for troubleshooting and limited manual intrusion detection. Also some intrusion detection systems analyze *syslog* audit data, such as the *STAT Tool Suite* [17]. Sites that wish to collect *syslog* audit data containing personal data of their users, can use our tools to avoid the legal restrictions otherwise imposed.

## 2.3 Audit Data Processing

Messages intended to be recorded using the *syslog* system are received by the `syslogd` application process either from user level application processes, from kernel level code or optionally via the network from remote sources. The ability to process audit data sent by remote sources allows for centralized

consolidation of audit data from a variety of platforms, such as routers, firewalls, intrusion detection systems and end systems driven by widespread operating systems. Unfortunately the *syslog* protocol is traditionally implemented over UDP (port number 514) without any security precautions and thus opens several opportunities for attack [14]. Therefore on most modern Unixes reception of *syslog* audit data from remote sources is disabled by default or can be disabled by the system administrator. The IETF Syslog Working Group endeavors to develop a standard to secure the *syslog* protocol [18]. In the mean time some alternative, purportedly more secure *syslog* replacements are available for Unix, such as `nsyslogd` [19]. Another option is to use IPSEC [20] to secure the *syslog* traffic.

When delivering an audit record to *syslog* the generating audit component provides the record priority, i.e. the pairing of the according facility and the record severity, as well as the name of the respective audit component. At the discretion of the audit component in addition to its name a process ID is supplied identifying the application process generating the audit record. Apart from this information the audit record comprises a description of the notable event. The event description is encoded in human readable form, but there are no standards with regard to the format of the description. The description format thus is specific to a given audit component and may vary even between different

Figure 2: Parts of a sample *syslog* audit record

versions of an audit component.

The further processing of incoming audit records can be tailored by the system administrator based on the record priority. Some implementations of `syslogd` allow finer grained selection criteria such as the name of the audit component. Basically `syslogd` sequentially evaluates rules provided by the system administrator. Each incoming audit record priority is matched against the rules. When a rule matches, the audit record is written out to a destination as specified in the matching rule. Possible destinations are files, devices, named pipes, the console of one or more users being logged on and `syslogd` processes on remote hosts. For the details of rule specification refer to [13].

Before writing an audit record out to the specified destination `syslogd` prepends to the message content (`MSG` in Fig. 2) a local time stamp (date/time) (`TS` in Fig. 2), the (fully qualified domain) name or IP address of the local host (`H` in Fig. 2) and the name of the audit component (`AC` in Fig. 2), optionally augmented by the respective process ID (`PID` in Fig. 2). Only when forwarding the audit record to a remote host `syslogd` prepends the respective priority (`PRI` in Fig. 2) to the message, which actually encodes both facility and severity in one integer value.

## 3 Embedding the Pseudonymizer

There are several opportunities for pseudonymization of *syslog* audit records along the path from the audit component to the final recipient. While it would be possible to integrate pseudonymization with all audit components, such an approach does not scale well. Also audit records should be pseudonymized before they reach the final recipient, which may be a privacy adversary.

Firstly, a straightforward approach is to pseudonymize the output of `syslogd` by re-configuring `syslogd` to write into named pipes being read by pseudonymizers. The problem with this solution is, that a pseudonymizer is not able to

determine the priority of the local output records of `syslogd`. This information is lost, but may be required for effective log analysis.

Secondly, there are two ways to pseudonymize audit records before they enter `syslogd`. We can wrap the *syslog* API function calls, but this approach does not catch audit records received from the kernel or via the network. Instead, a *wrapper* may pick up audit records from the *syslog*-sockets, while `syslogd` is configured to receive the pseudonymized audit records from the wrapper. We implemented such a pseudonymizing wrapper (see Fig. 4a).

Thirdly, we can embed pseudonymization within `syslogd` by patching its source code or by replacing the existing `syslogd` by a pseudonymizing `syslogd`. The disadvantages of these solutions are that either the access to system-dependent source code of `syslogd` is required, which may not be readily available for all Unixes, or the existing implementation of `syslogd` may differ significantly from the replacement version. In addition to the wrapper we provide source code patches to optionally embed pseudonymization within `syslogd` (see Fig. 4b).

Some audit components write audit records directly to files without involving `syslogd`. As long as our pseudonymizer can parse these records, it may also read from these files. Sometimes it is useful to consolidate the audit records of as many audit components as possible in one place, f.i. to use the same pseudonymizer for these records. We supply a redirector, which picks up the audit records from a named pipe or a growing file and deposits them by means of the *syslog* API.

### 3.1 Trust Model

When considering audit data generated on the local host we are always confronted with the risk of integrity loss in case an attacker achieves sufficient privilege to manipulate audit data or the process of its generation. Though countermeasures have already been proposed [21] they are not yet in widespread use. Even in the face of these problems *syslog* audit data today is still one very important source of

information that is used to resolve attack situations or to gather early indications thereof.

In the same way in which an attacker can manipulate audit data or its generation, he may also corrupt the integrity of pseudonyms in pseudonymized audit data to evade later identification. We argue that we can rely on the pseudonyms generated on a host under attack as long as we can rely on the *syslog* data generated on that host. Stronger techniques for pseudonym generation are outside the scope of this approach. Refer to Sect. 5 for related work. Pseudonymization can be used to make inferences on the pseudonymized data impractical. However, we cannot avoid inferences that take into account additional information outside the scope of pseudonymization.

Our approach to pseudonymization balances conflicting security requirements of at least two parties. On the one hand the users of an IT system wish to use it anonymously to protect their privacy. On the other hand the IT system owner's representative responsible for the system security requires accountability in the case of policy violations. Other interests in identifying users are not considered here, such as direct marketing by means of user profiling.

To put the mutual security requirements for anonymity and accountability not at risk, neither of the above mentioned parties can be allowed to control the audit components, `syslogd` and the pseudonymizer. Hence we introduce a third party denoted as the personal data protection official (PPO), who controls above mentioned system components (see the PPO domain in Fig. 4). The PPO is trusted by the users to protect their pseudonymity and he is trusted by the site security officer(s) (SSO) to ensure accountability in the face of a security incident.

In the current implementation audit components (with the help of *redirectors*) feed audit records via a *wrapper* or via *syslog* into the pseudonymizer (see Fig. 4 and Sect. 3.2.3). The pseudonymizer substitutes predefined types of identifying information, henceforth denoted as *features*, by pseudonyms. We exploit Shamir's threshold scheme for secret sharing to generate the pseudonyms. Owing to the properties of the threshold scheme we use, the features behind the pseudonyms can be recovered only under certain conditions later on (see Sect. 3.2.2). Thus, attacks on the pseudonymity of the system users are anticipated and resisted as soon as the pseudonymized audit records leave the pseudonymizer. Note that audit records should be pseudonymized on the same physical component where they are generated by the audit components. This may not be possible in some situations, f.i. when consolidating audit data generated by routers or Windows driven systems as is proposed in Sect. 2.1. In such cases additional measures must be taken by the PPO to provide a secure channel between the audit component and the pseudonymizer (see Sect. 2.3).

The pseudonymized audit data is delivered to and analyzed by the SSO to acquire indications for policy violations (see the SSO domain in Fig. 4). In order to support aforementioned analysis the pseudonyms of a given identifying feature are linkable wrt. a given suspicion. Given sufficient suspicion, the pseudonyms can be used to reveal the originally substituted features in the hope to gain more information helping to identify the perpetrator. In our approach sufficient suspicion is defined by a threshold on weighted occurrences of potentially attack-related events. The recovery of a feature is possible if and only if the corresponding threshold is exceeded. Thus, the SSO's requirement for accountability is met if the definition of sufficient suspicion is satisfied, but not otherwise, conforming to the user requirement for pseudonymity. The recovery of identifying features is done by the *reidentifier*, which is – unlike the pseudonymizer – fed with the output files of `syslogd` (see Fig. 4).

## 3.2 Pseudonymizing *syslog*

The pseudonymizer receives audit records from `syslogd` containing directly or indirectly identifying features [22]. Since each feature may contribute with a specific weight to a scenario legitimating feature recovery, the pseudonymizer distinguishes different event types and associates each embedded feature type with one or more scenarios and the respective weight(s). The PPO specifies apriori knowledge about the syntax of events and features to be pseudonymized. In cooperation with the SSO, the PPO models suspicions justifying reidentification by specifying associations between features and scenarios, defining the weight(s) for the contribution of each feature to its associated scenario(s), as well as the scenario thresholds. This apriori knowledge is stored in the configuration files of our tools. Consequently, feature recovery is tied to the sce-

`identd`[1] `[` `22509`[2] `]:` `token`[3] `TWpldDmO2sq65FfQ82zX` `== uid`[4] `1000`[5] `(`[6] `deedee`[7] `)`

`su`[8] `:` `BAD SU`[9] `deedee`[10] `to` `root`[11] `on`[12] `/dev/ttyp1`[13]

Figure 3: Syntactical concepts in two sample *syslog* audit records

narios that are configured when the features are pseudonymized. See Sect. 3.2.5 for an example.

### 3.2.1 Parsing

To locate and pseudonymize identifying features in *syslog* audit records, we have to parse them. Since the syntax of the records is mainly determined by the originating audit component, recognition of different event and feature types is based on the evaluation of syntactical contexts, e.g. by means of regular expressions for pattern matching as defined in POSIX 1003.2 Sect. 2.8.

For the sake of brevity we omitted the time stamp and host fields in Fig. 3 (`TS` and `H` in Fig. 2). The *audit component* (framed fields 1 and 8 in Fig. 3) of a record indicates an application or a kernel component, optionally followed by its *process ID* (frame 2 in Fig. 3). An audit record represents an event, whose type is uniquely specified by the *event type context* (frames 3 with 4, and 9 with 12 in Fig. 3) and the audit component. Each event type may contain a number of identifying *features* (frames 5, 7, 10, 11 and 13 in Fig. 3) of a type specified by a *feature type context* and the event type. A feature (f.i. frame 5 in Fig. 3) thus is an instance of a feature type determined by context (f.i. frames 4 and 6 in Fig. 3). There may be multiple occurrences of the same feature as instances of various feature types. Pseudonyms of a given feature associated with the same attack scenario contribute to reach the scenario threshold. This is not the case for pseudonyms of a given feature, which are associated with different scenarios.

When parsing audit records we use the apriori knowledge stored in the configuration files for our tools. By sequentially matching audit component, event type context, and feature type contexts, we isolate the features of the record that are to be pseudonymized. According to the assigned weight and scenario, a number of pseudonyms is gener-

ated, replacing the respective feature. The more pseudonyms are generated for a feature, the earlier the scenario threshold is exceeded, i.e. the feature is recoverable. The pseudonymized audit records retain the format of the unpseudonymized version.

### 3.2.2 Generating Pseudonyms

The basic idea of our approach to generating pseudonyms is to have the pseudonymizer split an identifying feature into as many pseudonyms as are needed to pseudonymize audit records containing this feature[2]. The pseudonyms have the property, that given a number of pseudonyms equal to the associated scenario threshold or more, but not less, the reidentifier is able to recover the corresponding feature. Secret sharing threshold schemes are suitable to fulfill these requirements. For our purposes we exploit Shamir's threshold scheme, as described in detail in [23]. 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 more control upon that entity. Owing to different conditions of deployment of Shamir's threshold scheme, we make some modifications with regard to its application. For a more technically inclined description of these modifications and the basic algorithms used by the pseudonymizer and the reidentifier, refer to [3].

In a nutshell, the pseudonymizer encrypts each identifying feature and the corresponding decryption key is split into shares, which are used as pseudonyms for that feature. The reidentifier uses Lagrange interpolation to recover the decryption key from a sufficient number of shares belonging to the same feature. [3] describes how to map a recovered decryption key to the corresponding encrypted feature before it can be decrypted. We implemented this map-

---

[2]The secret sharing scheme we apply operates over $GF(P)$, where $P$ is prime and smaller than $2^l$. $l$ equals to the number of bits of the symmetric en-/decryption keys used for en-/decrypting identifying features. We thus can generate at maximum $P - 1$ pseudonyms per feature per scenario.

ping in a distributed fashion by using the encrypted features as labels for the pseudonyms. The result is, that the pseudonyms of a given identifying feature are linkable wrt. a given scenario. In the current implementation features are encrypted symmetrically using the Blowfish encryption scheme [24].

### 3.2.3  Architecture

We supply tools for the PPO and for the SSO, namely `redirector`, `wrapper`, `pseudonymizer`, `shared`, and `reidentifier`, `combined`, respectively. The tools are portable and have been used successfully on Solaris, OpenBSD and Linux.

According to Sect. 3.1 the audit components, `redirector`, `wrapper`, `syslogd`, `pseudonymizer` and `shared` are controlled by the PPO. The audit components are configured to generate only audit data required for sustained service provision, such as audit data indicating proceeding attacks against the site. The audit data is written to `/dev/log` (with the help of `redirectors`), to UDP port 514 and to other locations usually read by `syslogd`, f.i. `/dev/klog` for OpenBSD kernel audit data. In Fig. 4a the `wrapper` intercepts audit data at `/dev/log` and at UDP port 514[3]. Audit data that cannot be intercepted[4] is received by `syslogd` and sent on to the `wrapper` (see the dotted lines in the PPO domain in Fig. 4a). The `pseudonymizer` is fed with audit data by the `wrapper`. This solution can be used if `syslogd` shall not be replaced or modified. Optionally we provide a patch for `syslogd` such that incoming audit data is sent directly to the `pseudonymizer` without involving the `wrapper`. For this solution `syslogd` needs to be replaced by a patched version (see Fig. 4b).

The audit data contains identifying features that need to be pseudonymized. The `pseudonymizer` parses each incoming audit record, determines the appropriate scenarios and asks `shared` for pseudonyms for the features wrt. these scenarios. `shared` provides the cryptographic primitives, such as symmetric encryption and secret sharing. For encryption `shared` uses the OpenSSL crypto library [24] and the threshold secret sharing scheme

is implemented using the GNU Multiple Precision Arithmetic library (GMP) [25]. `shared` may run on the same machine as the `pseudonymizer`, or it may run remotely. This allows several distributed `pseudonymizers` to use a central `shared` with the result that pseudonyms for a given identifying feature are linkable across the corresponding distributed audit components. The protocol used for communication between the `pseudonymizer` and `shared` transfers identifiable features. Since the PPO might not control the network between the `pseudonymizer` and `shared`, the channel is protected using the OpenSSL SSL/TLS library [26] when `shared` runs remotely. The `pseudonymizer` replaces the identifying features with the pseudonyms delivered by `shared`. The pseudonymized audit data is then handed to `syslogd` (via the `wrapper`). In the recommended setup `syslogd` sends the pseudonymized audit records to a remote `syslogd` running under the control of the SSO. Note that using the recommended setup audit data is not written into disk files before it is pseudonymized and received in the SSO domain.

In compliance with Sect. 3.1 the `syslogd` receiving pseudonymized audit records from the `pseudonymizer` as well as the `reidentifier` and `combined` are controlled by the SSO. `syslogd` receives the pseudonymized audit records and handles them according to the SSO's requirements. If the SSO wishes to be able to recover identifying features some time later, `syslogd` needs to be configured to write those audit records to a log file (see Fig. 4). When needed, the `reidentifier` is started manually or automatically on an alarm. It reads the log file, determines which pseudonyms satisfy their scenario and asks `combined` to recover the respective identifying features. `combined` provides cryptographic primitives, such as symmetric decryption and Lagrange interpolation, the reverse operation for the threshold secret sharing scheme we use. For decryption `combined` also uses the OpenSSL crypto library [24] and the Lagrange interpolation again is implemented using the GMP library [25]. `combined` may run on the same machine as the `reidentifier`, or it may run remotely. The protocol used for communication between the `reidentifier` and `combined` transfers identifiable features. Since identifiable features are transferred only if a scenario is satisfied, i.e. the SSO is allowed to see the features, there is no need for a channel providing confidentiality.[5] The

---

[3]We actually use UDP port forwarding to redirect packets sent to port 514 to the port where the `wrapper` listens.

[4]For example OpenBSD kernel audit data written to `/dev/klog` cannot be intercepted because the OpenBSD `syslogd` is hard-coded to read kernel audit data from `/dev/klog`.

---

[5]If the SSO does not control the network between `combined` and the `reidentifier` a confidential channel can
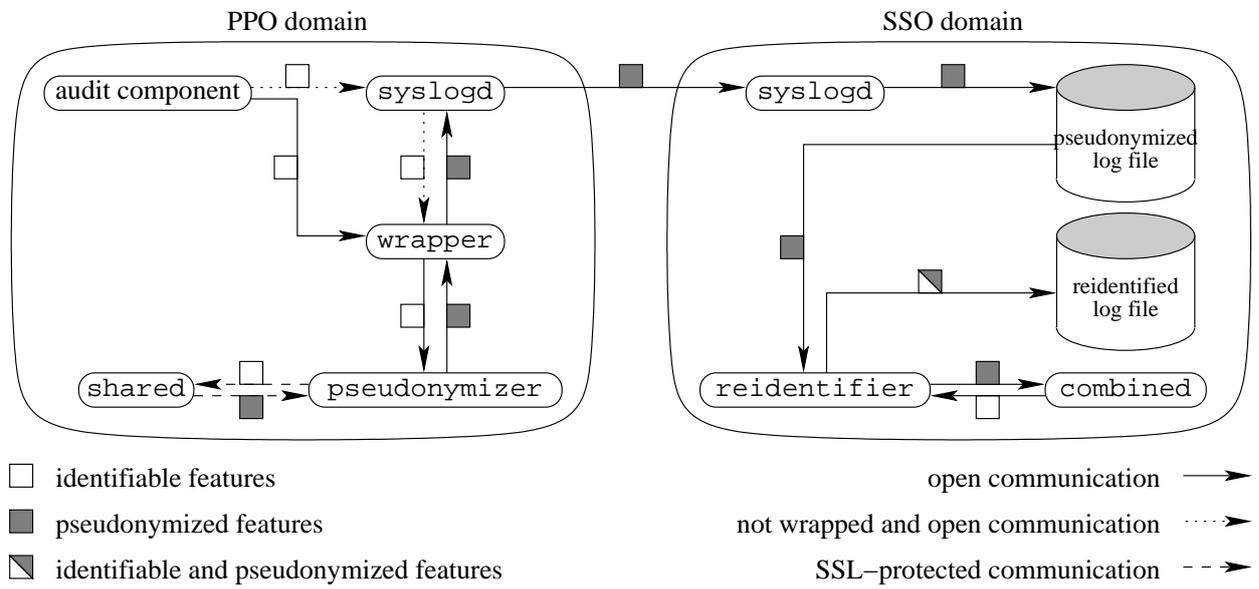
(a) Wrapped syslogd
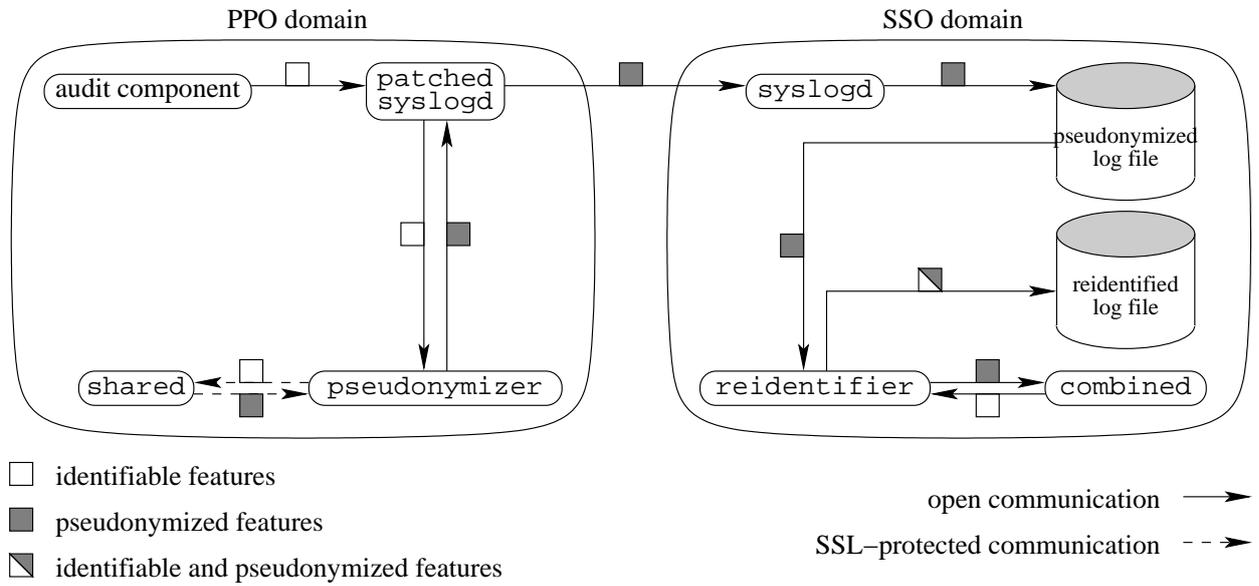


(b) Patched syslogd

Figure 4: Interplay with syslogd

`reidentifier` replaces all recoverable pseudonyms with the corresponding identifying features delivered by `combined`.

### 3.2.4 Use

For the machines where *syslog* audit data is to be pseudonymized the PPO decides either to replace the existing `syslogd` by a patched version or to wrap the existing `syslogd` using the `wrapper`. The PPO ensures that the `wrapper`, the `pseudonymizer`, `shared` and the `redirectors` start up at system boot time such that they can communicate as required and can process the audit data in background. Additionally the PPO can use the `pseudonymizer` like a filter to process archived log files on demand.

Audit components, which do not use *syslog* to record audit records can also be covered using the `redirector`. This tool is able to read from a pipe or to monitor a log file for recently appended audit records and deposits these using the *syslog* API and a priority specified by the PPO. The result is, that these audit records can be pseudonymized like described above.

In the case of a policy violation the corresponding scenario would be satisfied and the involved identifying features can be recovered from the log file. The `reidentifier` can be triggered manually or via a script when an alarm occurs. It is used like a filter to recover as many identifying features as possible. For the `reidentifier` to work, a running `combined` needs to be reachable.

### 3.2.5 Applications

Both, the `pseudonymizer` and the `reidentifier` leverage apriori knowledge provided in configuration files to locate identifying features and pseudonyms in audit data, respectively. Feature types are located using the audit component name (see the `FACILITY` keywords in Fig. 5), the event type context (see the `EVENT` keywords in Fig. 5) and the feature type contexts (see the `LEFTFEATURE` and `RIGHTFEATURE` keywords in Fig. 5). The apriori knowledge associates each feature type and corresponding pseudonyms with one or more scenarios

be provided.

(see the `GROUP` keywords next to the `RIGHTFEATURE` keywords in Fig. 5) as well as (a) corresponding weight(s) (see the `WEIGHT` keywords in Fig. 5), specifying the number of pseudonyms generated for the feature type. Also for each scenario a threshold is defined (see the `THRESHOLD` keyword at the top of Fig. 5), specifying the number of pseudonyms required for associated features to be recoverable from the corresponding pseudonyms.

In the following example we pseudonymize audit records generated by `tcplog`. The `tcplog` program logs TCP flag combinations that should not occur in regular TCP traffic. Additionally `tcplog` is able to detect `queso` OS fingerprinting scans. `queso` is a reconnaissance tool used to determine the operating system version using the OS fingerprinting technique, offering a subset of the functionality of `nmap` [27].

We suppose that the PPO of a site considers IP addresses being sensitive information since they have the potential to identify the person behind the associated activities. The PPO thus decides to pseudonymize IP addresses.[6] The SSO knows that for an intruder the information provided by `queso` is useful for determining if a given service provided by a host might be vulnerable to a particular exploit. Hence, the SSO wants to be able to identify the IP addresses originating `queso` scans against his site.

The PPO analyzes, which audit components generate event types containing IP address features, such as the sample audit records in Fig. 6. The SSO provides the PPO with information about scenarios that require feature recovery, such as the `queso` scenario. For the `tcplog` audit component the PPO in cooperation with the SSO extends the apriori knowledge stored for our tools in the configuration files as shown in Fig. 5.

When the tools have been configured for the `queso` scenario the audit records in Fig. 6 are pseudonymized by the `pseudonymizer`. The resulting audit records are shown in Fig. 7. For the sake of presentation we moved the additional records generated by the `pseudonymizer` to the bottom of Fig. 7 while retaining the relative order of the `tcplog` records and of the additional records. Note that in the `tcplog` records the IP addresses have been replaced by pointers to the additional records. The

_____

[6]The PPO additionally might want to pseudonymize the port numbers. This also is perfectly possible, but for the sake of the brevity of the example we refrain from doing so.

```
GROUP I1 THRESHOLD 6
FACILITY tcplog EVENT 'QUESO'
  LEFTFEATURE ' from ', RIGHTFEATURE ' port' GROUP I1 WEIGHT const(1)
FACILITY tcplog EVENT ' from '
  LEFTFEATURE ' from ', RIGHTFEATURE ' port' GROUP I1 WEIGHT const(1)
```

Figure 5: Sample configuration for pseudonymizing `tcplog` audit records, such as in Fig. 6

```
May 19 02:23:37 host tcplog[1567]: FIN SYN RST        URG      : port 41067 from 217.82.199.102 port 42312
May 22 13:42:06 host tcplog[1040]:     SYN                RES2 : ftp from 192.168.1.4 port 45691
May 22 13:42:06 host tcplog[1040]: FIN SYN     PSH     URG     : ftp from 192.168.1.4 port 45693
May 22 13:42:06 host tcplog[1040]: FIN SYN     PSH     URG     : ftp from 192.168.1.4 port 45693
May 22 13:42:06 host tcplog[1040]: FIN         PSH     URG     : port 34513 from 192.168.1.4 port 45697
May 22 13:42:06 host tcplog[1040]: FIN         PSH     URG     : port 34513 from 192.168.1.4 port 45697
May 22 13:42:06 host tcplog[1040]: QUESO: port 34513 from 192.168.1.4 port 45697
```

Figure 6: Sample unpseudonymized audit records generated by `tcplog`, which contain a `queso` OS fingerprinting scan

```
May 19 02:23:37 host tcplog[1567]: FIN SYN RST        URG      : port 41067 from a1 port 42312
May 22 13:42:06 host tcplog[1040]:     SYN                RES2 : ftp from a2 port 45691
May 22 13:42:06 host tcplog[1040]: FIN SYN     PSH     URG     : ftp from a3 port 45693
May 22 13:42:06 host tcplog[1040]: FIN SYN     PSH     URG     : ftp from a4 port 45693
May 22 13:42:06 host tcplog[1040]: FIN         PSH     URG     : port 34513 from a5 45697
May 22 13:42:06 host tcplog[1040]: FIN         PSH     URG     : port 34513 from a6 45697
May 22 13:42:06 host tcplog[1040]: QUESO: port 34513 from a7 port 45697
May 19 02:23:37 host pseudonymizer: Short=a1 Long=RW4gGPc2dWVLOwXgF20ENg:10:ECmA!_ph6kOhKLZBsMw!H7H2ztc Group=I1
May 22 13:42:06 host pseudonymizer: Short=a2 Long=5QgfV3!umUu_Fj8MWI2yoA:5b:d2petb50CXOun4CuLPhluM8 Group=I1
May 22 13:42:06 host pseudonymizer: Short=a3 Long=sW7bl67oOo1G5eiBpmu7hg:5c:d2petb50CXOun4CuLPhluM8 Group=I1
May 22 13:42:06 host pseudonymizer: Short=a4 Long=wdEey!pfsM4jGtz91yejZw:5d:d2petb50CXOun4CuLPhluM8 Group=I1
May 22 13:42:06 host pseudonymizer: Short=a5 Long=Fi7o9GJU!A5SwYO!dAcxsA:5e:d2petb50CXOun4CuLPhluM8 Group=I1
May 22 13:42:06 host pseudonymizer: Short=a6 Long=rog6EObIHE3V5moFBlAr8Q:5f:d2petb50CXOun4CuLPhluM8 Group=I1
May 22 13:42:06 host pseudonymizer: Short=a7 Long=it0SIYe5EYysleQNFOhaNg:60:d2petb50CXOun4CuLPhluM8 Group=I1
```

Figure 7: Sample pseudonymized audit records from Fig. 6

```
May 19 02:23:37 host tcplog[1567]: FIN SYN RST        URG      : port 41067 from a1 port 42312
May 22 13:42:06 host tcplog[1040]:     SYN                RES2 : ftp from 192.168.1.4 port 45691
May 22 13:42:06 host tcplog[1040]: FIN SYN     PSH     URG     : ftp from 192.168.1.4 port 45693
May 22 13:42:06 host tcplog[1040]: FIN SYN     PSH     URG     : ftp from 192.168.1.4 port 45693
May 22 13:42:06 host tcplog[1040]: FIN         PSH     URG     : port 34513 from 192.168.1.4 port 45697
May 22 13:42:06 host tcplog[1040]: FIN         PSH     URG     : port 34513 from 192.168.1.4 port 45697
May 22 13:42:06 host tcplog[1040]: QUESO: port 34513 from 192.168.1.4 port 45697
May 19 02:23:37 host pseudonymizer: Short=a1 Long=RW4gGPc2dWVLOwXgF20ENg:10:ECmA!_ph6kOhKLZBsMw!H7H2ztc Group=I1
```

Figure 8: Sample pseudonymized audit records from Fig. 7 with recovered features

additional records provide the actual pseudonyms, i.e. the cryptographic material and other information required for the recovery of the replaced IP addresses. Note that the actual pseudonyms for IP address `192.168.1.4` are linkable using the identifier `d2petb50CXOun4CuLPhluM8`. Generally there may occur audit records describing activity that is definitely not related to any scenario, but where features need to be hidden. These audit records can be pseudonymized while omitting the additional records, such that recovery is infeasible later on.

The audit records in Fig. 7 are transferred to the SSO. When the SSO detects the `queso` scan contained in the audit records in Fig. 7 he uses the `reidentifier` to recover the IP address features associated with the satisfied `queso` scenario. The resulting audit records are shown in Fig. 8.
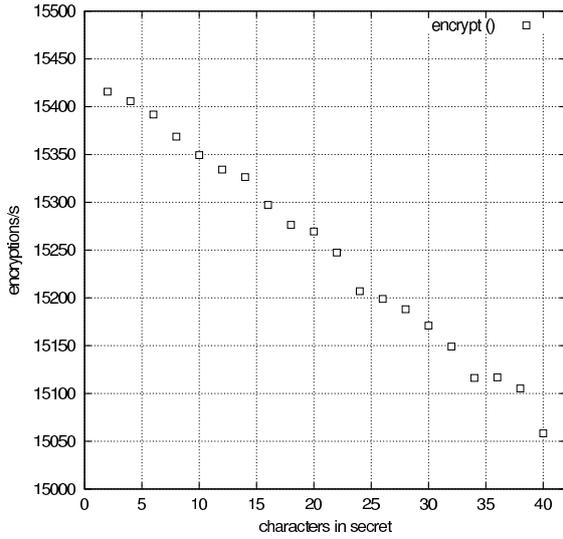
(a) OpenSSL encryption
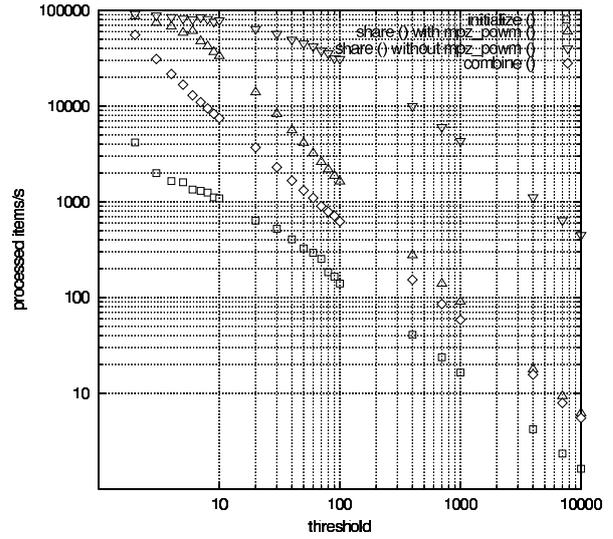


(b) `initialize()`, `share()`, `combine()`
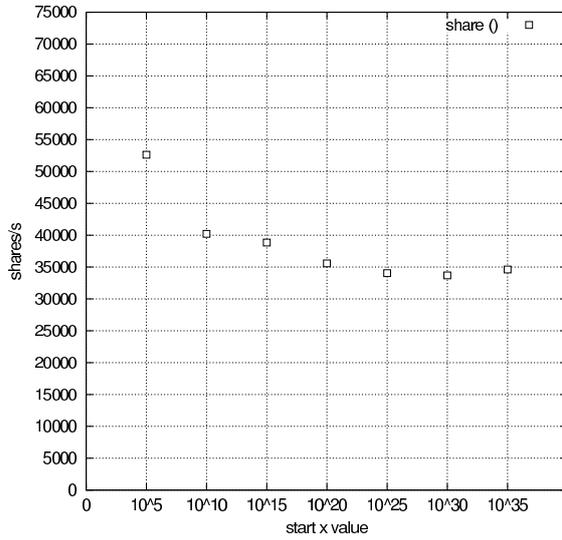


(c) GMP number size
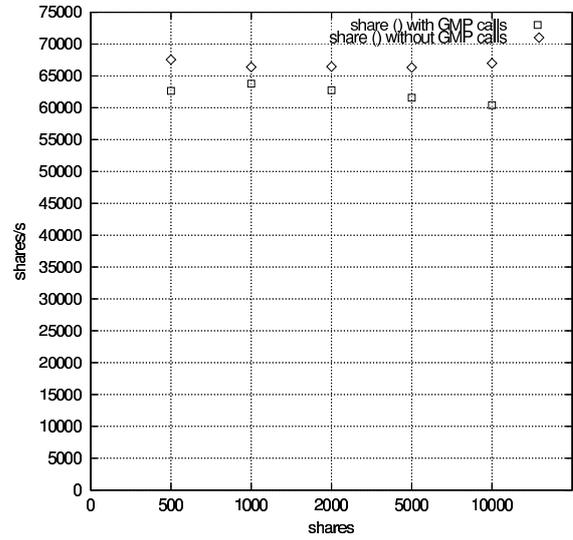


(d) GMP memory management

Figure 9: Performance measurements of the cryptographic components

Note that the pseudonym in the first audit record in Fig. 8 is not revealed since it corresponds to an IP address not involved in the `queso` scan originated at IP address `192.168.1.4`.

## 4   Performance

The `pseudonymizer` processes audit data in real-time. It is thus required to pseudonymize audit records sufficiently fast to avoid a bottleneck, which might impair overall system performance. In contrast, the `reidentifier` is sparely used and needs not to satisfy real-time requirements.

Table 1: Default values for the parameters and how they were varied

| parameter | default | varied by means of |
|---|---:|---|
| communication with `shared` | Unix domain socket | command line |
| number of bits | 128 | source code |
| number of audit records | 1000 | audit data |
| scenario threshold | 5 | configuration |
| number of scenarios | 1 | audit data & configuration |
| number of audit components | 1 | audit data & configuration |
| number of event types | 1 | audit data & configuration |
| number of features per record | 1 | audit data & configuration |
| pseudonym weight | 1 | configuration |

The performance of the pseudonymizer depends on the performance of the cryptographic primitives it uses. However, in Sect. 4.2 we show that also the performance of the pattern matching is a major factor. The performance measurements were conducted on a single processor Pentium III 650MHz machine with 256MB RAM and a 100Mbps Fast-Ethernet NIC, running OpenBSD 2.7.

## 4.1 Microbenchmarks for the Cryptographic Components

We measured the performance of the cryptographic primitives used in `shared` and `combined`. For secret sharing and Lagrange interpolation they operate over $GF(P)$ where $P$ is a prime number of 128 bits. For encryption we also use symmetric 128 bit keys. Thresholds used in the measurements default to 5. The results are averaged over 5 measurements for `encrypt()` and `share()`, over $1 + 10/t^2$ measurements for `initialize()` and over $1 + 10000/t^2$ measurements for `combine()`, where $t$ is the current threshold.

The first time `shared` processes a given feature it uses the `initialize()` routine, which generates a pseudo-random polynomial, encrypts the feature and inserts the results in an AVL tree. We found that the Blowfish encryption in the OpenSSL crypto library operates at about 15370 encryptions per second for features of eight characters. It slows down slightly to about 15150 encryptions per second for features of 32 characters (see Fig. 9a). We expect most features to be pretty short strings. The encryption speed is independent of the number of bits used for the symmetric keys. Depending on the threshold of the scenario associated with a feature, `initialize()` generates a number of pseudo-

random coefficients for the corresponding polynomial. The higher the threshold, the more pseudorandom numbers are generated (see `initialize()` in Fig. 9b). When evaluating the performance of the `share()` routine, we found some interesting run time behavior of the GMP library. `share()` basically evaluates the polynomial `initialize()` provides. Hence, the higher the threshold, the more expensive the exponentiations `share()` calculates using the GMP library call `mpz_powm()` (see `share()` with/without `mpz_powm()` in Fig. 9b). It is no surprise that GMP library calls take longer if we calculate with larger numbers (see Fig. 9c). We also found that the GMP library calls take longer the more often integer objects are reused, though `mpz_clear()` has been called to free the space the integer objects occupy (see Fig. 9d). Again, an expected result is that the Lagrange interpolation, which the `combined` uses, is an expensive calculation for large thresholds (see `combine()` in Fig. 9b).

## 4.2 Macrobenchmarks for the Pseudonymizer

We measured the performance of the `pseudonymizer` for varying parameters using synthetic audit data and configurations. All the numbers reported are averaged over 5 measurements. When varying one parameter all other parameters used default values as shown in Table 1. The synthetic audit data we used for the measurements contained audit records, which all had the same number of identifying features, each with a length of three characters. All features in the audit data were required to be pseudonymized. Each feature in a given audit record was associated with a separate scenario, assuming that the features in an audit record are different kinds of identifying

Table 2: Influence of the number of bits used for encryption and secret sharing

| scenario | number of audit records/s for | | | |
|---|---|---|---|---|
| features | 64 bits | 128 bits | 168 bits | 256 bits |
| identical | 1075 | 1064 | 1064 | 1062 |
| varying | 600 | 573 | 560 | 530 |

Table 3: Influence of the communication technique used between the `pseudonymizer` and `shared`

| | number of audit records/s for | | | |
|---|---|---|---|---|
| scenario | | local Unix | SSL over | SSL over |
| features | no `shared` | domain socket | loopback | Ethernet |
| identical | 1300 | 1064 | 444 | 400 |
| varying | 1300 | 573 | 328 | 377 |

data. The $i^{th}$ features in all audit records were associated with the same scenario. Thus, the scenarios could comprise events generated from several audit components. All scenarios had the same threshold and all features were associated with the same weight. Audit records from different audit components and for different event types were generated cyclically.

`shared` handles features differently depending on whether they have already been seen in the audit data or not. If a feature is processed for the first time, `initialize()` generates a pseudo-random polynomial and stores it together with the encrypted feature (see Sect. 4.1). In any case a polynomial is used to generate a fresh pseudonym using `share()` (see Sect. 4.1). Owing to these different cases of feature handling we used two kinds of audit data to measure the best and the worst case performance. In the best case all features associated with a given scenario were *identical*. `shared` then calls `initialize()` only once per scenario. In the worst case we had *varying* features associated with a given scenario. As a result `shared` calls `initialize()` for every feature in the audit data.

When increasing the number of audit records the `pseudonymizer` processes, we observe for varying features a mild performance penalty due to the growth of the AVL trees storing encrypted features and their corresponding polynomials (see Fig. 10a). A growing number of audit components, event types and scenarios has a linear influence on the performance since these objects are managed using lists (see Fig. 10b). The more features per audit record are pseudonymized, the more often the `pseudonymizer` asks `shared` for pseudonyms. In addition to the communication overhead more

feature type contexts need to be matched by the `pseudonymizer` (see Fig. 10c). The pattern matching is the main cause for the performance loss when many features need to be pseudonymized per audit record (see Fig. 10d). In contrast to this the performance loss caused by generating pseudonyms with larger weights is moderate (see Fig. 10e). There is no additional pattern matching or communication overhead involved. Increasing the threshold does not only slow down the generation of pseudonyms using `share()`, but also the initial generation of pseudo-random numbers for each distinct feature in a scenario using `initialize()` (see `initialize()` and `share()` with `mpz_powm` in Fig. 9b in comparison to Fig. 10f). Though the blowfish encryption operates at a speed independently from the number of bits used for the keys, we have to consider that the secret sharing slows down for larger numbers (see Fig. 9c). For reasonable key lengths we found only a negligible effect on the secret sharing performance (see the row for identical features in Table 2). In contrast to this result the impact on the initialization for secret sharing was significant, since larger pseudo-random numbers need to be generated and converted to mpz-numbers for the GMP library calls (see the row for varying features in Table 2).

`shared` is able to operate on the same or on a remote host as the `pseudonymizer` (see Table 3). Note that there is a significant SSL-overhead if there is high traffic between the `pseudonymizer` and a `shared` connected to the loopback interface. We thus recommend to use Unix domain sockets to communicate with a local `shared`.

However, in practice the parameters will only in rare cases reach values as high as shown in Fig. 9 and Fig. 10. In the next section we show that even in these
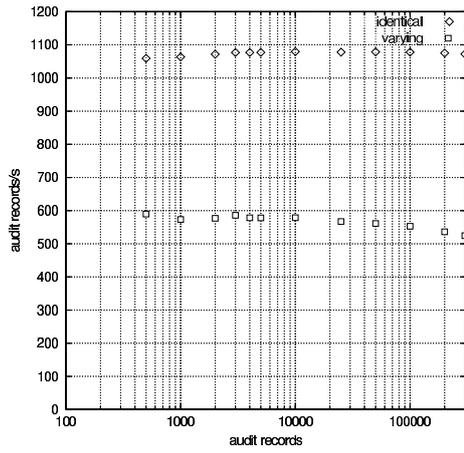
(a) Audit records
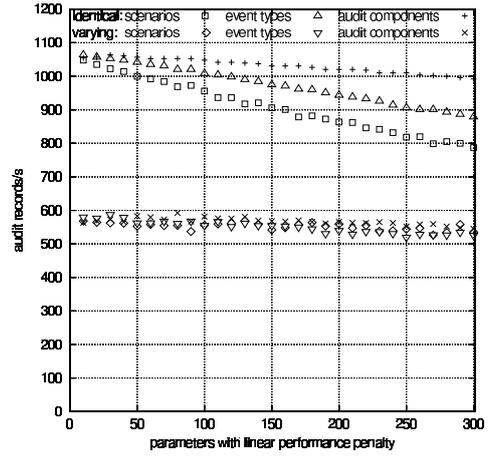
(b) Scenarios, events, audit components
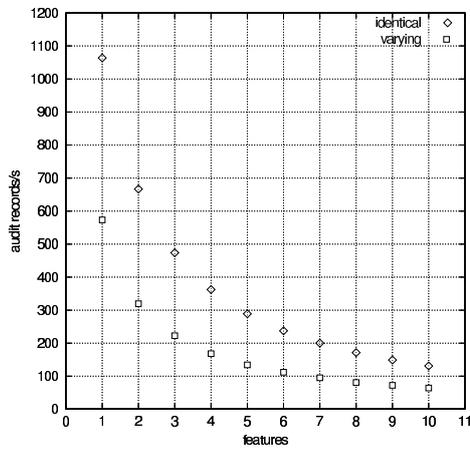
(c) Features
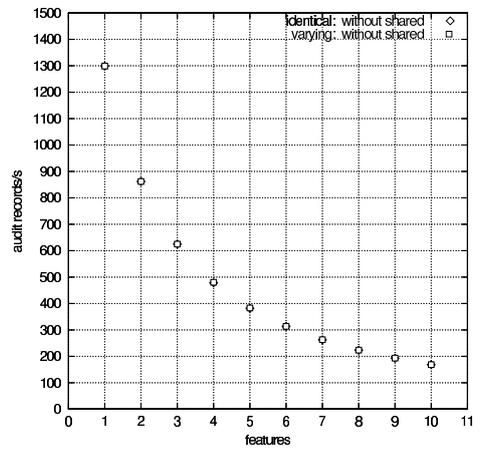
(d) Features, parsing only
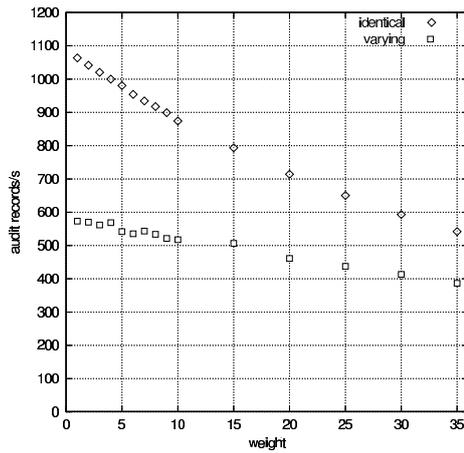
(e) Weight
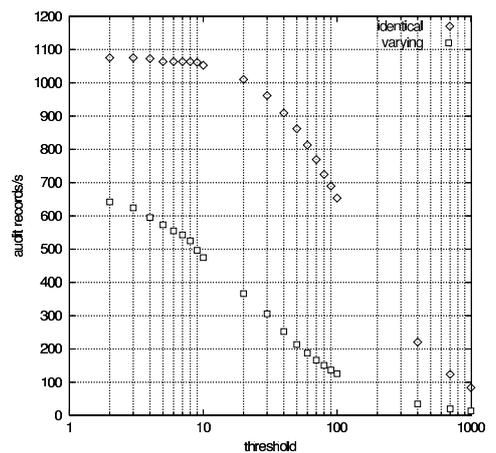
(f) Threshold

Figure 10: Performance measurements of the `pseudonymizer`

(a) All web servers by hour of day

(b) All audit components by hour of day

(c) All web servers by day of month

(d) All audit components by day of month
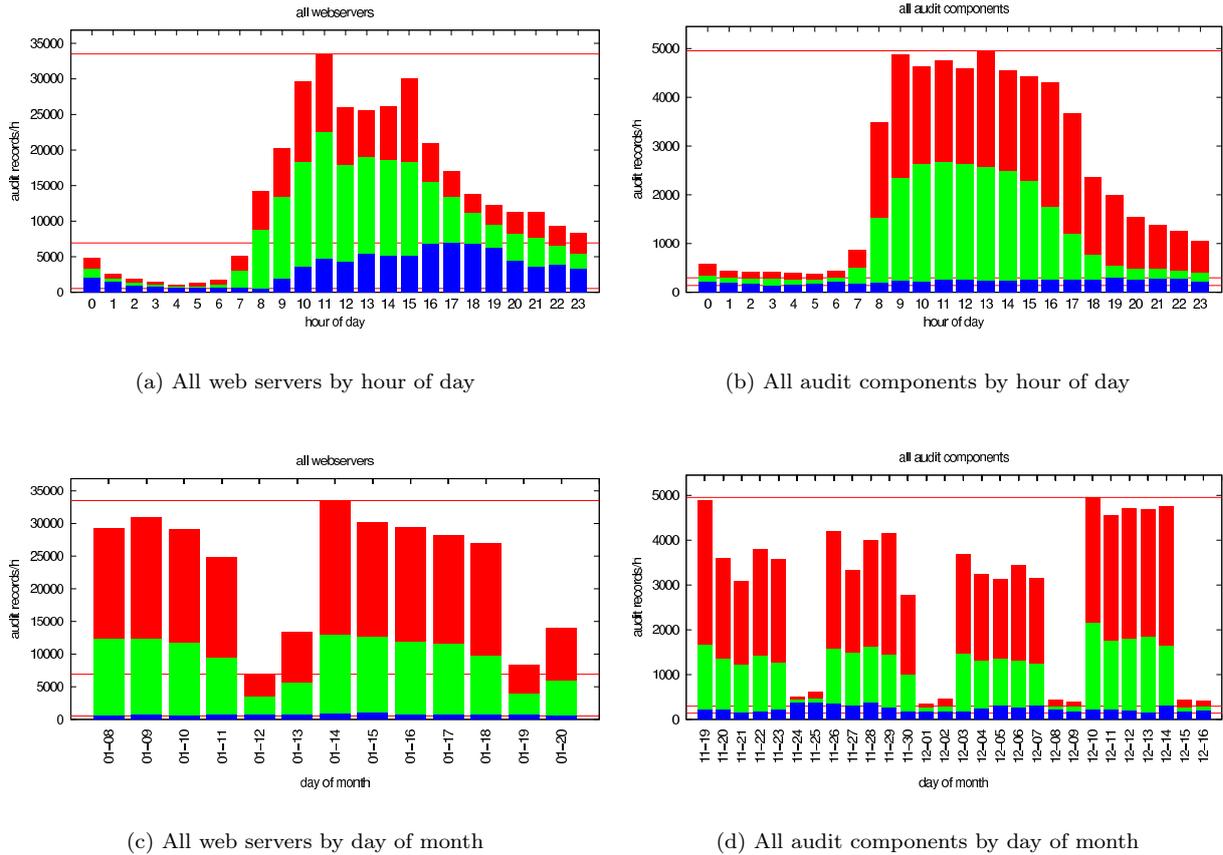
Figure 11: Server statistics: audit records generated per hour

rare cases the performance of the `pseudonymizer` is sufficient to cope with the audit record volume of a large site.

## 4.3 How Fast is Fast?

To be able to judge whether our tools perform fast enough in a real world server environment we evaluated *syslog* and *Apache* audit records from a central server at the Center for Communication and Information Processing at the University of Dortmund. The machine hardware consists of a SUN Ultra Enterprise 4000 with 3GB RAM, six Ultra SPARC 168MHz CPUs, three disk arrays totaling 396GB and a 100Mbps full-duplex Fast-Ethernet uplink. For backup services the machine is also connected to an IBM 3494 storage tape roboter. The server runs Solaris 7 and about 1050 users are registered, which are employees and students of the University of Dortmund. During working hours an average of 25 users is logged on simultaneously. The main services provided by the machine are:

**Web:** 37 world accessible *Apache* web servers

**FTP:** about 112000 transfers per month with a volume of 12GB

**Email:** about 45000 emails per month (SMTP, IMAP, POP)

**DBMS:** one Oracle DBMS for several web applications and a Cambridge chemical structure DB

**Education:** education system ARCVIEW

**Software:** large compiled software archive for several platforms, exported per NFS

**Backup:** stores about 4.5 Terabytes of data of 450 clients

We evaluated all *Apache* access audit records collected over a period of 13 days for all 37 web servers
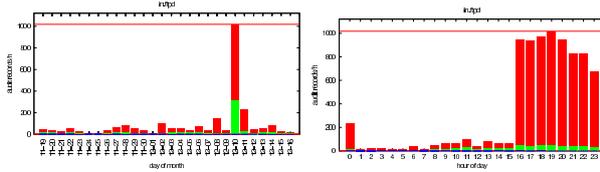
Figure 12: The day when FTP went wild

Table 4: Maximum number of audit records generated per second

| audit component | number of audit records | |
| --- | --- | --- |
| | per hour | per second |
| *Apache* | 33506 | 9.31 |
| *syslog* | 4956 | 1.38 |
| $\sum$ | 38462 | 10.68 |

(see Fig. 11a and Fig. 11c). We have also evaluated all *syslog* audit records collected over a period of four weeks (see Fig. 11b and Fig. 11d). The *syslog* audit records were collected at other days than the *Apache* audit records. The *syslog* audit records are generated by the following audit components: `sendmail`, `imapd`, `ipop3d`, `in.ftpd`, `in.lpd`, `in.telnetd`, `in.rlogind`, `in.rshd`, `in.rexecd` and `in.comsat`.

For both, *syslog* and *Apache* we counted the number of audit records written per hour ($ar/h$). For each of the 24 hours of a day we determined the minimum, the maximum and the average $ar/h$ over all days for which we had audit data available (see Fig. 11a and Fig. 11b). For each of the days for which we had audit data available we also determined the minimum, the maximum and the average $ar/h$ over all hours of that day (see Fig. 11c and Fig. 11d).

We noticed that in the later hours the maximum *syslog* $ar/h$ deviates significantly from the average *syslog* $ar/h$. This is due to an abnormally high number of `in.ftpd` audit records at the $10^{th}$ of December after 4pm (see Fig. 12). The root cause of this event is out of the scope of this paper.

The crucial point here is, that the `pseudonymizer` is able to keep up with the number of audit records generated even in situations where some audit component generates an abnormally high number of audit records. See Table 4 for the maximum numbers of audit records we measured.

## 5    Related Work

Our tools can be applied to server log data in order to hide information that may directly or indirectly identify the users that accessed the server. There are other well known services available to achieve this at the price of installing some client-side software or requiring the user to perform additional steps in order to achieve anonymity. More importantly, many of these anonymity tools do not provide for conditional identity recovery.

On the web the P3P framework helps users to match their privacy requirements against the privacy policies of the sites they visit. The amount of personal data being divulged can be tailored to the profile of the sites being visited [28]. Recent proposals support the user in the task of delivering real and pseudonymized personal data by automating the management of several virtual identities. These are tailored for various different classes of situations within which a user interacts with web sites [29]. While this approach allows for fine grained control over what personal data is revealed at the application layer, identifying information from lower layer protocols is not protected. For this purpose there are also many different anonymizing services for accessing the web, such as the *Anonymizer*, *Anonymouse*, *Proxymate* formerly known as *LPWA*, *Crowds*, *Onion Routing* and *JAP* [30]. In addition to filtering the transmission wrt. personal data, these services usually involve one or more intermediaries and aim at decoupling the client from the target server. Thus, identifying data is hidden before it reaches the target server. Similar examples for anonymous mail transfer are *Anonymouse* or remailers such as *Cypherpunk* and *Mixmaster*. Since the techniques employed, f.i. MIXes, are entirely different from the ones used for pseudonymizing log data, we do not go into the details here. These services are surveyed and compared in [31, 32, 33]. For the same reasons we do not consider services for anonymous publishing, file sharing, voting and bidding. An introduction to basic privacy enhancing technologies such as DC nets, MIX nets, anonymous credentials and electronic payment is given in [32].

The techniques mentioned above are conducted under the control of the user himself and/or at least one third party trusted by the user wrt. anonymization. Thus, no trust is required in the target service provider. On the other hand these approaches require the user to take vari-

ous actions to be able to communicate anonymously.

Pertinent legislation in various western countries requires providers of some classes of target services to support anonymous use of their service. To comply with these requirements, service providers need technical means, independent from actions their clients may take to protect their privacy. Existing services based on Unix systems often record personal data in log files. Reconfiguring the system such that it does not record certain kinds of log events is not always an option. The information supplied by some log events may be mission critical and may also contain personal data. One straightforward and easily deployable approach is to anonymize or pseudonymize identifying data after it has been collected, but before it is recorded in log files.

The *Anonymouse log file anonymizer* [31] is implemented as a customizable Perl script that replaces identifying information by default values or more coarse values. Anonymized log files may still be analyzed. In case some event happens, which requires further investigation wrt. the user's identity, it would be useful to be able to recover the original data, which has been anonymized. This feature is not supported by the *Anonymouse log file anonymizer*. The situation is similar for the aforementioned MIX technologies. Indeed, even in the case of misuse of a service by an anonymous user it is infeasible for the provider or for investigating authorities to establish accountability. Contrary to this our tools provide for recoverability of pseudonymized data under certain conditions.

Due to this property our work is strongly related to privacy enhanced intrusion detection systems. Most of these systems use pseudonymization to hide identifying information after it has been generated and recorded in audit data. Some approaches have been published for the *Intrusion Detection and Avoidance* system (IDA) [34, 35, 22, 32], the *Adaptive Intrusion Detection* system (AID) [36, 22, 37, 32], a Firewall audit analyzer [38] and for the *Aachener Network Intrusion Detection Architecture* (ANIDA) [39]. These approaches have been reviewed and compared to our work in [1]. Intrusion detection systems (IDS) do not only give warnings in case of attacks, they also record evidence for further investigation. In case an attacker is to be prosecuted, it is desirable to extract identifying information from the recorded data. Privacy enhanced IDS support the recovery of pseudonymized information. A common weakness of existing privacy enhanced IDS is that they do not technically bind the recovery of identifying information to a specific purpose such as the investigation of a proceeding attack. Some privacy enhanced IDS require a third party to cooperate in the recovery process. The users have to trust that party to allow recovery only for given legitimate purposes. Other privacy enhanced IDS provide no strong protection against recovery for other purposes [1]. In contrast to this our tools instantly allow for recovery of pseudonymized data if predefined conditions bound to a specific purpose are met, but not otherwise.

Similar mechanisms are found in fair offline electronic payment systems. Such systems allow for anonymous payment as long as no payment unit has been spent more often than permitted. In the case of double spending of an electronic coin, the perpetrator can be traced [40, 41, 42, 43]. Some of the recovery techniques used are similar to the scheme our approach employs. Others use group signatures to achieve recoverability [44, 45].

Payment units in fair offline electronic payment systems are also similar to anonymous credentials, which can be used for anonymous authentication [46, 47, 48, 49, 50]. Many schemes for anonymous authentication support identity recovery while employing different techniques. Some schemes [51, 52, 53, 54, 55] have been related to our approach in [3]. Other recent work is described in [56, 57, 58, 59]. Since the resources of the target service are accessed using anonymous and pseudonymous credentials or payment units, the target service does not learn identifying information about the users during authentication. The users do not need to trust the target service provider regarding their anonymity. Also, malformed credentials, which do not exhibit the properties required by the target service can be rejected, avoiding further damage. On the other hand, before they can access any target service anonymously, users need to register with some third party to generate credentials and payment units. Since no such infrastructure has yet been widely adopted, in the meantime service providers can pseudonymize log data to offer privacy enhanced service access.

## 6   Conclusion

We motivated the need for technical enforcement of privacy principles and surveyed related work on privacy enhancing technologies. Though many of the related approaches allow for a strong trust model, they require the user to take additional steps and/or install client-side software. Also many approaches require a widely accepted infrastructure for issuing and managing credentials and payment units. None of these infrastructures is yet in widespread use, i.e. service providers cannot adopt widely accepted standards.

In contrast to this situation service providers are expected to respect and protect the privacy of their clients. We supply tools that help to retrofit privacy protection into existing Unix systems by means of pseudonymization of identifying features in audit data. Different from existing log file anonymizers or privacy enhanced intrusion detection systems, our approach balances requirements regarding anonymity and accountability. That is, pseudonymized identifying features can be revealed only if given definitions of scenarios requiring to do so are satisfied, but not otherwise. A trusted third party ensures that these scenarios meet the requirements regarding anonymity and accountability. This party controls the technical components that enforce the proper pseudonymization, such that later on identifying features can be recovered.

We described the trust model and architecture of the tools and gave some examples on their use. Then we presented measurements of the performance of the `pseudonymizer`. Finally we demonstrated measurements of the audit data volume generated by a real world system, indicating that the performance of the `pseudonymizer` should be sufficient for application at sites with even larger audit data volume.

## Acknowledgments

We gratefully acknowledge the efforts of Hans Bornemann, Michael Heyer and Dieter Stolte in providing data about their networks and machines. Most of all Sven Bursch, Kai Grundmann, Stefan Magerstedt and Dennis Real deserve credit for their tireless work on the code as well as the measurements of the tools.

## References

[1] Joachim Biskup and Ulrich Flegel. On pseudonymization of audit data for intrusion detection. In Federrath [60], pages 161–180.

[2] Joachim Biskup and Ulrich Flegel. Transaction-based pseudonyms in audit data for privacy respecting intrusion detection. In Hervé Debar, Ludovic Mé, and S. Felix Wu, editors, *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000)*, number 1907 in LNCS, pages 28–48, Toulouse, France, October 2000. Springer.

[3] Joachim Biskup and Ulrich Flegel. Threshold-based identity recovery for privacy enhanced applications. In Sushil Jajodia and Pierangela Samarati, editors, *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 71–79, Athens, Greece, November 2000. ACM SIGSAC, ACM Press.

[4] Privacy survey results, January 2002. `http://www.cdt.org/privacy/survey/findings/`.

[5] Louis Harris & Associates Inc. IBM multinational consumer privacy survey. Technical Report 938568, IBM Global Services, 1999.

[6] Jarek Rossignac et al. GVU's 10th WWW User Survey, December 1998. `http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/graphs/graphs.html#privacy`.

[7] Steven R. Johnston. The impact of privacy and data protection legislation on the sharing of intrusion detection information. In Lee et al. [61], pages 150–171.

[8] Kai Rannenberg, Andreas Pfitzmann, and Günter Müller. IT security and multilateral security. In Müller and Rannenberg [62], pages 21–29.

[9] National Computer Security Center. US DoD Standard: Department of Defense Trusted Computer System Evaluation Criteria. DOD

5200.28-STD, Supercedes CSC-STD-001-83, dtd 15 Aug 83, Library No. S225,711, December 1985. `http://csrc.ncsl.nist.gov/secpubs/rainbow/std001.txt`.

[10] National Computer Security Center. Audit in trusted systems. NCSC-TG-001, Library No. S-228,470, July 1987. `http://csrc.ncsl.nist.gov/secpubs/rainbow/tg001.txt`.

[11] Common Criteria Implementation Board. *Common Criteria for Information Technology Security Evaluation — Part 2: Security functional requirements, Version 2.1.* Number CCIMB-99-032. National Institute of Standards and Technology, August 1999. `http://csrc.ncsl.nist.gov/cc/ccv20/p2-v21.pdf`.

[12] Darren Reed. IP Filter. `http://coombs.anu.edu.au/~avalon/ip-filter.html`, 2001.

[13] *syslog.conf.* Manual Page.

[14] C. Lonvick. *RFC 3164: The BSD syslog Protocol*, August 2001. `http://www.ietf.org/rfc/rfc3164.txt`.

[15] Martin Roesch. Snort – lightweight intrusion detection for networks. In *Proceedings of LISA'99: 13th Systems Administration Conference*, pages 229–238, Seattle, Washington, November 1999. The Usenix Association, Usenix.

[16] Martin Roesch. *Snort Users Manual – Snort Release: 1.8.2*, October 2001. `http://www.snort.org/docs/SnortUsersManual.pdf`.

[17] Giovanno Vigna, Richard A. Kemmerer, and Per Blix. Designing a web of highly-configurable intrusion detection sensors. In Lee et al. [61], pages 69–84.

[18] Internet Engineering Task Force. Security issues in network event logging (syslog), July 2001. `http://www.ietf.org/html.charters/syslog-charter.html`.

[19] Darren Reed. Nsyslogd, 2001. `http://coombs.anu.edu.au/~avalon/nsyslog.html`.

[20] S. Kent and R. Atkinson. *RFC 2401: Security Architecture for the Internet Protocol*, November 1998. `http://www.ietf.org/rfc/rfc2401.txt`.

[21] Bruce Schneier and John 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, September 1998. IBM Emergency Response Team. `http://www.zurich.ibm.com/~dac/Prog_RAID98/Table_of_content.html`.

[22] Michael Sobirey, Simone Fischer-Hübner, and Kai Rannenberg. Pseudonymous audit for privacy enhanced intrusion detection. In L. Yngström and J. Carlsen, editors, *Proceedings of the IFIP TC11 13th International Conference on Information Security (SEC'97)*, pages 151–163, Copenhagen, Denmark, May 1997. IFIP, Chapman & Hall, London.

[23] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[24] *OpenSSL cryptographic library*, December 2001. `http://www.openssl.org/docs/crypto/crypto.html`.

[25] Torbjörn Granlund. *The GNU Multiple Precision Arithmetic Library.* GNU, 3.1.1 edition, September 2000. `http://www.gnu.org/manual/gmp/index.html`.

[26] *OpenSSL SSL/TLS library*, December 2001. `http://www.openssl.org/docs/ssl/ssl.html`.

[27] Marco de Vivo, Eddy Carrasco, Germinal Isern, and Gabriela O. de Vivo. A review of port scanning techniques. *Computer Communication Review*, 29(2):41–48, April 1999.

[28] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) specification, September 2001. `http://www.w3.org/TR/2001/WD-P3P-20010928/`.

[29] Marit Köhntopp and Oliver Berthold. Identity management based on P3P. In Federrath [60], pages 141–160.

[30] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web mixes: A system for anonymous and unobservable internet access. In Federrath [60], pages 115–129.

[31] Claudia Eckert and Alexander Pircher. Internet anonymity: Problems and solutions. In Dupuy and Paradinas [63], pages 35–50.

[32] Simone Fischer-Hübner. *IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms.* Number 1958 in Lecture Notes in Computer Science. Springer, 2001.

[33] Oliver Berthold, Hannes Federrath, and Marit Köhntopp. Project "Anonymity and unobservability in the internet". In *Proceedings of the Workshop on Freedom and Privacy by Design / Conference on Freedom and Privacy*, pages 57–65, Toronto, Canada, April 2000. ACM.

[34] Simone Fischer-Hübner and Klaus Brunnstein. Opportunities and risks of intrusion detection expert systems. In *Proceedings of the International IFIP-GI-Conference Opportunities and Risks of Artificial Intelligence Systems ORAIS'89*, Hamburg, Germany, July 1989. IFIP.

[35] Simone Fischer-Hübner. *IDA (Intrusion Detection and Avoidance System): Ein einbruchsentdeckendes und einbruchsvermeidendes System (in German).* Reihe Informatik. Shaker, 1993.

[36] Michael Sobirey. Aktuelle Anforderungen an Intrusion Detection-Systeme und deren Berücksichtigung bei der Systemgestaltung von AID$^2$ (in German). In Hans H. Brüggemann and Waltraud Gerhardt-Häckl, editors, *Proceedings of Verläßliche IT-Systeme*, DuD-Fachbeiträge, pages 351–370, Rostock, Germany, April 1995. GI, Vieweg.

[37] Michael Sobirey. *Datenschutzorientiertes Intrusion Detection (in German).* DuD-Fachbeiträge. Vieweg, 1999.

[38] Emilie Lundin and Erland Jonsson. Anomaly-based intrusion detection: privacy concerns and other problems. *Computer Networks*, 34(4):623–640, October 2000.

[39] Roland Büschkes and Dogan Kesdogan. Privacy enhanced intrusion detection. In Müller and Rannenberg [62], pages 187–204.

[40] Holger Petersen. Faires elektronisches Geld (in German). In *Mit Sicherheit in die Informationsgesellschaft*, pages 427–444, Bonn, Germany, April 1997. Bundesamt für Sicherheit in der Informationstechnik, SecuMedia Verlag, Ingelheim.

[41] George Davida, Yair Frankel, Yiannis Tsiounis, and Moti Yung. Anonymity control in e-cash systems. In R. Hirschfeld, editor, *Proceedings of the First International Conference on Financial Cryptography (FC'97)*, number 1318 in Lecture Notes in Computer Science, pages 1–16, Anguilla, British West Indies, February 1997. Springer.

[42] Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable electronic coupon protocol with anonymity control. In M. Mombo and Y. Zheng, editors, *Proceedings of the Second International Workshop on Information Security (ISW'99)*, number 1729 in Lecture Notes in Computer Science, pages 37–46. Springer, November 1999.

[43] David Pointcheval. Self-scrambling anonymizers. In Frankel [64], pages 259–275.

[44] Jaques Traoré. Group signatures and their relevance to privacy-protecting offline electronic cash systems. In J. Pieprzyk, R. Safavi-Naini, and J. Seberry, editors, *Proceedings of the 4th Australasian Conference on Information Security and Privacy (ACISP'99)*, number 1587 in Lecture Notes in Computer Science, pages 228–243, Wollongong, NSW, Australia, April 1999. Springer.

[45] Greg Maitland and Colin Boyd. Fair electronic cash based on a group signature scheme. In S. Qing, T. Okamoto, and J. Zhou, editors, *Proceedings of the Third International Conference on Information and Communications Security (ICICS 2001)*, number 2229 in Lecture Notes in Computer Science, pages 461–465, Xian, China, November 2001. Springer.

[46] Stuart Schechter, Todd Parnell, and Alexander Hartemink. Anonymous authentication of membership in dynamic groups. In M. Franklin, editor, *Proceedings of the Third International Conference on Financial Cryptography (FC'99)*, number 1648 in Lecture Notes in Computer Science, pages 184–195, Anguilla, British West Indies, February 1999. Springer.

[47] Els Van Herreweghen. Secure anonymous signature-based transactions. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proceedings of the 6th European Symposium on Reserach in Computer Security (ESORICS*

*2000)*, number 1895 in Lecture Notes in Computer Science, pages 55–71, Toulouse, France, October 2000. Springer.

[48] Stuart G. Stubblebine and Paul F. Syverson. Authentic attributes with fine-grained anonymity protection. In Frankel [64], pages 276–294.

[49] Ben Handley. Resource-efficient anonymous group identification. In Frankel [64], pages 295–312.

[50] Dimitris Gritzalis, Konstantinos Moulinos, John Iliadis, Costas Lambrinoudakis, and Steven Xarhoulakos. Pythia: Towards anonymity in authentication. In Dupuy and Paradinas [63], pages 1–17.

[51] Joe Kilian and Erez Petrank. Identity escrow. In *Proceedings of the Conference on Advances in Cryptology (CRYPTO'98)*, pages 196–185, 1998.

[52] Dan Boneh and Matt 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, November 1999. ACM SIGSAC, ACM Press.

[53] Yuen-Yan Chan. On privacy issues of internet access services via proxy servers. In Rainer Baumgart, editor, *Proceedings of the Congress on Secure Networking – CQRE[Secure]'99*, number 1740 in LNCS, pages 183–191, Düsseldorf, Germany, November 1999. secunet, Springer.

[54] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography (SAC'99)*, pages 184–199, Kingston, Ontario, Canada, August 1999. Springer.

[55] Stuart G. Stubblebine, Paul F. Syverson, and David M. Goldschlag. Unlinkable serial transactions: Protocols and applications. *ACM Transactions on Information and System Security*, 2(4):354–389, November 1999.

[56] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, number 2045 in Lecture Notes in Computer Science, pages 93–118, Austria, May 2001. Springer.

[57] Jan Camenisch and Anna Lysyanskaya. Efficient revocation of anonymous group membership certificates and anonymous credentials. `http://eprint.iacr.org/2001`, December 2001.

[58] Ariel Glenn, Ian Goldberg, Frédéric Légaré, and Anton Stiglic. A description of protocols for private credentials. `http://eprint.iacr.org/2001`, October 2001.

[59] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, Massachusetts, 2000.

[60] Hannes Federrath, editor. *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, number 2009 in LNCS, Berkeley, California, July 2000. ICSI, Springer.

[61] Wenke Lee, Ludovic Mé, and Andreas Wespi, editors. *Proceedings of the Fourth International Workshop on Recent Advances in Intrusion Detection (RAID 2001)*, number 2212 in LNCS, Davis, California, October 2001. Springer.

[62] Günter Müller and Kai Rannenberg, editors. *Multilateral Security in Communications*. Information Security. Addison Wesley, 1999.

[63] Michel Dupuy and Pierre Paradinas, editors. *Proceedings of the IFIP TC11 16th International Conference on Information Security (IFIP/Sec'01)*, Paris, France, June 2001. IFIP, Kluwer Academic Publishers.

[64] Y. Frankel, editor. *Proceedings of the 4th International Conference on Financial Cryptography (FC'00)*, number 1962 in Lecture Notes in Computer Science, Anguilla, British West Indies, February 2000. Springer.

# Appendix – Example Application

In this appendix we show an example for pseudonymizing audit data not generated via *syslog*, but by an *Apache* web server. We used the **redirector** to deposit the audit records via *syslog* and pseudonymized them the same way as other *syslog* audit records. For the sake of presentation we omitted the time stamp, host, audit component and the respective process ID fields (TS, H, AC and PID in Fig. 2) in Fig. 13, 15, 16, 18 and Fig. 19. That is, we only retained the order of the audit records and the message content fields (MSG in Fig. 2) containing the *Apache* audit records. We show only the audit records and configurations for the access log data. In a real system other log data generated by *Apache* would also have to be pseudonymized.

We show two different ways to pseudonymize the audit records in Fig. 13 (see Fig. 14 and Fig. 17). We only pseudonymize the first field in the audit records in Fig. 13 containing the name or IP address of the client machine. Some of the audit records in Fig. 13 show symptoms of attacks a *NIMDA* worm at IP address 217.82.228.10 initiated. These attacks exploit several vulnerabilities in some web server implementations, but to which our web server was not vulnerable.

Using the configuration in Fig. 14 the client IP address features contained in the 16 event types a *NIMDA* scan generates are pseudonymized with a weight of 5. Accordingly the threshold of the scenario associated with these pseudonyms is set to 80, allowing for feature recovery after 16 *NIMDA* audit records have been processed by the **reidentifier**. The client IP address features in all other audit records are pseudonymized with a weight of 1, thus being recoverable after 80 audit records have been processed by the **reidentifier**.

The pseudonyms are shown in Fig. 15. Note that due to their higher weight the pseudonyms in *NIMDA* audit records are longer than the ones in the other audit records. The recovered IP address features in *NIMDA* audit records are shown in Fig. 16. Since all pseudonyms in Fig. 16 are associated with the same scenario *I1*, any pseudonyms for IP address 217.82.228.10 in non-*NIMDA* audit records would also be revealed.

There are several differences between the configurations in Fig. 14 and Fig. 17. Firstly the IP address features in the *NIMDA* audit records are associated with an other scenario (*I1*) than the ones in the non-*NIMDA* audit records (*I2*). The result is, that even if the pseudonyms in *NIMDA* records are recovered, the pseudonyms for the recovered IP address features in non-*NIMDA* audit records will not necessarily be recovered. Also the threshold for *NIMDA* audit records now is adjustable separately from the threshold of the non-*NIMDA* audit records. In contrast to the configuration in Fig. 14, using the configuration in Fig. 17 we can freely adjust when non-*NIMDA* records will be recoverable. Additionally, we can pseudonymize all IP address features with a weight of 1 now.

The pseudonyms are shown in Fig. 18. Note that due to the reduced weight the pseudonyms in *NIMDA* audit records are not as long as in Fig. 15. The recovered IP address features in *NIMDA* audit records are shown in Fig. 19. Since in Fig. 16 pseudonyms in *NIMDA* audit records are associated with scenario *I1* and any pseudonyms for IP address 217.82.228.10 in non-*NIMDA* audit records are associated with scenario *I2*, the latter would not be revealed.

```
pd9519434.dip.t-dialin.net - - [05/Nov/2001:01:11:17 +0100]  "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.html HTTP/1.0" 200 11804
pd9519434.dip.t-dialin.net - - [05/Nov/2001:01:11:20 +0100]  "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.css HTTP/1.0" 200 344
pd9004967.dip.t-dialin.net - - [05/Nov/2001:01:12:13 +0100]  "GET /issi/archive/literature/tech-reports/643.html HTTP/1.1" 200 8825
pd9519434.dip.t-dialin.net - - [05/Nov/2001:01:12:14 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 200 0
217.82.228.10 - - [05/Nov/2001:01:12:14 +0100]  "GET /scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:14 +0100]  "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 208 "-" "-"
pd9519434.dip.t-dialin.net - - [05/Nov/2001:01:12:14 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 0
pd9004967.dip.t-dialin.net - - [05/Nov/2001:01:12:14 +0100]  "GET /issi/archive/literature/tech-reports/643/unilab.gif HTTP/1.1" 200 1885
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
pd9519434.dip.t-dialin.net - - [05/Nov/2001:01:12:15 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 83588
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:15 +0100]  "GET /~magerste HTTP/1.0" 301 248
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:16 +0100]  "GET /~magerste/ HTTP/1.0" 200 844
217.82.228.10 - - [05/Nov/2001:01:12:16 +0100]  "GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:16 +0100]  "GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
pd9004967.dip.t-dialin.net - - [05/Nov/2001:01:12:16 +0100]  "GET /issi/archive/literature/tech-reports/643/unido.gif HTTP/1.1" 200 311
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:17 +0100]  "GET /icons/image2.gif HTTP/1.0" 200 309
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 265 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:17 +0100]  "GET /icons/blank.gif HTTP/1.0" 200 148
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:18 +0100]  "GET /~magerste/popel.jpg HTTP/1.0" 200 21994
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:18 +0100]  "GET /~magerste/cocacola3.gif HTTP/1.0" 200 5970
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:19 +0100]  "GET /~magerste/cocacola2.gif HTTP/1.0" 200 16724
p5087bfc7.dip.t-dialin.net - - [05/Nov/2001:01:12:20 +0100]  "GET /~magerste/cam1.jpg HTTP/1.0" 200 9519
```

Figure 13: Sample audit records generated by *Apache*, which contain a *NIMDA* scan

```
GROUP I1 THRESHOLD 80
FACILITY redirector EVENT '.?GET \/c\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/MSADC\/root\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/d\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%255c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/_vti_bin\/\.\.\%255c\.\.\/\.\.\%255c\.\.\/\.\.\%255c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/_mem_bin\/\.\.\%255c\.\.\/\.\.\%255c\.\.\/\.\.\%255c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/msadc\/\.\.\%255c\.\.\/\.\.\%255c\.\.\/\.\.\%255c\/\.\.\%c1\%1c\.\.\/\.\.\%c1\%1c\.\.\/\.\.\%c1\%1c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%c1\%1c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%c0%2f\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%c0%af\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%c1%9c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%%35%63\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%25%35%63\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/\.\.\%252f\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET \/scripts\/root\.exe\?\/c\+dir HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(5)
FACILITY redirector EVENT '.?GET (.*) HTTP\/'
        LEFTFEATURE '.*', RIGHTFEATURE '  - -  \[' GROUP I1 WEIGHT const(1)
```

Figure 14: Sample configuration 1) for pseudonymizing *Apache* audit records, such as in Fig. 13

```
a1  - -  [05/Nov/2001:01:11:17 +0100]  "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.html HTTP/1.0" 200 11804
a2  - -  [05/Nov/2001:01:11:20 +0100]  "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.css HTTP/1.1" 200 344
a3  - -  [05/Nov/2001:01:12:13 +0100]  "GET /issi/archive/literature/tech-reports/643.html HTTP/1.1" 200 8825
a4  - -  [05/Nov/2001:01:12:14 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 200 0
a5  - -  [05/Nov/2001:01:12:14 +0100]  "GET /scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-" "-"
a6  - -  [05/Nov/2001:01:12:14 +0100]  "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 208 "-" "-"
a7  - -  [05/Nov/2001:01:12:14 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 0
a8  - -  [05/Nov/2001:01:12:14 +0100]  "GET /issi/teaching/lectures/643/unilab.gif HTTP/1.1" 200 1885
a9  - -  [05/Nov/2001:01:12:15 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 83588
a10 - -  [05/Nov/2001:01:12:15 +0100]  "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
a11 - -  [05/Nov/2001:01:12:15 +0100]  "GET /magerste HTTP/1.0" 301 248
a12 - -  [05/Nov/2001:01:12:15 +0100]  "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
a13 - -  [05/Nov/2001:01:12:15 +0100]  "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a14 - -  [05/Nov/2001:01:12:16 +0100]  "GET /magerste/ HTTP/1.0" 200 844
a15 - -  [05/Nov/2001:01:12:16 +0100]  "GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
a16 - -  [05/Nov/2001:01:12:16 +0100]  "GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
a17 - -  [05/Nov/2001:01:12:16 +0100]  "GET /issi/archive/literature/tech-reports/643/unido.gif HTTP/1.1" 200 311
a18 - -  [05/Nov/2001:01:12:17 +0100]  "GET /icons/image2.gif HTTP/1.0" 200 309
a19 - -  [05/Nov/2001:01:12:17 +0100]  "GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 265 "-" "-"
a20 - -  [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a21 - -  [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a22 - -  [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a23 - -  [05/Nov/2001:01:12:17 +0100]  "GET /icons/blank.gif HTTP/1.0" 200 148
a24 - -  [05/Nov/2001:01:12:18 +0100]  "GET /magerste/popel.jpg HTTP/1.0" 200 21994
a25 - -  [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a26 - -  [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 215 "-" "-"
a27 - -  [05/Nov/2001:01:12:18 +0100]  "GET /magerste/cocacola3.gif HTTP/1.0" 200 5970
a28 - -  [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
a29 - -  [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%255%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a30 - -  [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a31 - -  [05/Nov/2001:01:12:19 +0100]  "GET /magerste/cocacola2.gif HTTP/1.0" 200 16724
a32 - -  [05/Nov/2001:01:12:20 +0100]  "GET /magerste/cam1.jpg HTTP/1.0" 200 9519

Short=a1  Long=U10MvwFCP1kVNDj_iSKr8w:1:7Eu59a18FQEryoQdJHIprU9RvV1vnevU1w1k Group=I1
Short=a2  Long=5JDMjaZdMV4MjCTRgJBAOw:2:7Eu59a18FQEryoQdJHIprU9RvV1vnevU1w1k Group=I1
Short=a3  Long=EJ09GJbgR3IMnVK0W5IZZg:1:6UB1kZzFW1Dj3X_7dZKV5cQWavS1WV0ZUtHH Group=I1
Short=a4  Long=1jKtPmzNeuf6ZW1GKZr4tA:3:7Eu59a18FQEryoQdJHIprU9RvV1vnevU1w1k Group=I1
Short=a5  Long=8akiD31LN8mdCwcqyWQsFA:1:guZh3iTmKs7UTvL9tNt1kw:2:dIhCjutkdFjCPD3yXeYtdA:3:peroQEqMRYaNFvVbAJpmSg:4:U1Pt10dmZDVDx1fgGYjNg:5:57ywFJVt4vEp5ld4en0 Group=I1
Short=a6  Long=p07EVdVMa5gSmcGWVCh1xA:6:HUjyIzyU7BGhz42pQc8ESg:7:zIB4RtHixvxdfDtf11BS1Q:8:qVKsTtkQsoFURrcK851BCA:9:6iZNbbAH31umbH0t84D_uA:a:57ywFJVt4vEp5ld4en0 Group=I1
Short=a7  Long=B5VS78wDTB10QCcuzE8yKQ:4:7Eu59a18FQEryoQdJHIprU9RvV1vnevU1w1k Group=I1
Short=a8  Long=odp85zv7DncD9T5hHv_uRg:2:6UB1kZzFW1Dj3X_7dZKV5cQWavS1WV0ZUtHH Group=I1
Short=a9  Long=D_22SdQJaQrWIk1onTLNvg:b:1tclNMPodNL7H4cYez6jg:c:VhKTL1X_jEhvJLi3y_duLg:d:FrrJw6n5Fr4USRlKcdcO1g:e:MramryQbKMnC11DF4zr01g:f:57ywFJVt4vEp5ld4en0 Group=I1
Short=a10 Long=RnRjPB4NkHaEQ83jA1Q644A:1:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a11 Long=E1NOF9ee9LObcVSsEHCh9g:10:aKmjO6ZsdLyZUCvLjYN0dA:11:yW1aK8EpZYS_P27NBMBIkA:12:1yvkfFLz1I1Epg2F9AC4_KQ:13:rXzAXTY1R0JpVPanVy2nqw:14:57ywFJVt4vEp5ld4en0 Group=I1
Short=a13 Long=U5p150kHmH0EvYcJu1p82w:15:fip1J7Vs1EEmoKLasqpqcoA:16:mwbu0aVdVuxY26wD9SILQ:17:0CyGs5OzFXj1ybxcfMacEA:18:BP1EZBqY1dCFLvUvXjXELA:19:57ywFJVt4vEp5ld4en0 Group=I1
Short=a14 Long=17GjCsMog3t7mzSt6HxUwA:2:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a16 Long=104wMS8cxP9Dy4yrxmYT1AQ:1a:HorZvFCtfSrEl1rIgBL5BQ:1b:gO14vM_RreaLv5HeyToXg:1c:IWKtK9XFj_wDUDACB6rOTA:1d:_Tuq5tF1Ao11h79jDHoOFA:1e:57ywFJVt4vEp5ld4en0 Group=I1
Short=a17 Long=k3xdmAJrhADxzo1V8ApmJw:3:6UB1kZzFW1Dj3X_7dZKV5cQWavS1WV0ZUtHH Group=I1
Short=a18 Long=yVUDu4mYzdVpdK_ikYcFoQ:3:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a19 Long=RoSDxJMOnRc2YU124P_pfA:24:12FjvrLG2pSsfo193c8o1A:25:ns5r7lLy_nam3fZKbjRtwg:26:0TnRf1om49NcqSt0WRsKyQ:27:u1SBhuYAgmtnbq2LXQxfQqA:28:57ywFJVt4vEp5ld4en0 Group=I1
Short=a20 Long=gxZUDt_ef8NS_A1_w7MnvA:29:wXvYntVabPARwCDEXmnrsDA:2a:U86Cj7o3v_WEzBelhYcdvw:2b:IkzZsvv6utas24Bt-jHEjNg:2c:KUIWsevVc8GtRPLZK1Q8rA:2d:57ywFJVt4vEp5ld4en0 Group=I1
Short=a21 Long=W9JwUYx6h1GYrRdTmTd41A:2e:115FE2ZNnxzZy5hq54o1A:2f:0puOS5omDXk0bkFzBDvgF1cQ:30:i9KAPSjSc1BDT5mLyAf84A:31:jdU2t659_WbSJYLbxrEoYQc:32:57ywFJVt4vEp5ld4en0 Group=I1
Short=a23 Long=_rYpbDj0njq9T2cLhDa_dw:4:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a24 Long=PsQ4eHf8j18R48PtabT7Vu:5:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a25 Long=XOjjZc4d11bElbq3J1p9Gg:38:S8z4VBeURfLe1DGBanJfeQ:39:Ny5ZbG4c1cn12k9AbcRvZg:3a:myLEAIOJ9dpJ8YSkph9Jhw:3b:T1BSqLAtNpgBzy8hTV63_g:3c:57ywFJVt4vEp5ld4en0 Group=I1
Short=a26 Long=CPVGr1_c6f9R1iMGca_cv:3d:CMNlugFCoXOjC2PK1nrEpg:3e:HAce1z3Q9D_MMtygPFnOXw:3f:FAZWUDQHovupkReIva4FFu:40:t3jLaa8gO1Dz73Z5Xrk6tA:41:57ywFJVt4vEp5ld4en0 Group=I1
Short=a27 Long=_dofgnUD0xES5OjNGh8q68Q:6:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a29 Long=j69P3FNrdznIA1CKjc2_Gcw:42:Mhbpt1MhqyPXyweulrkmpA:43:my91XNU8WqbMrzHFVF4R_w:44:Dxporf_5Rp8eEwk7bKJ9TA:45:z1fIgrZjymv6DRxpr1O21w:46:57ywFJVt4vEp5ld4en0 Group=I1
Short=a30 Long=vVlLlN17G88DNhf11gnOmg:4c:Jps1iif15aFIQBSDhtwUag:4d:JqGpD8PaRfsG1y1PIySQ4w:4e:tdTEkIpfOrF_barTzy_8Lg:4f:7rUNfeW11Fzt5jzLRkgfCg:50:57ywFJVt4vEp5ld4en0 Group=I1
Short=a31 Long=cQ2T9rRRL5MB19ZdW1cdw:7:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
Short=a32 Long=lUu5c3A1H6kEtkO0QCvEroQ:8:1vZ7EXaWvh7r013VPjhJokgoROmkarch61OX Group=I1
```

Figure 15: Sample audit records from Fig. 13 pseudonymized using configuration 1) in Fig.14

a1  - - [05/Nov/2001:01:11:17 +0100]  "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.html HTTP/1.1" 200 11804
a2  - - [05/Nov/2001:01:11:20 +0100]  "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.css HTTP/1.0" 200 344
a3  - - [05/Nov/2001:01:12:13 +0100]  "GET /issi/archive/literature/tech-reports/643.html HTTP/1.1" 200 8825
a4  - - [05/Nov/2001:01:12:14 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 200 0
217.82.228.10 - - [05/Nov/2001:01:12:14 +0100]  "GET /scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-" "-"
a7  - - [05/Nov/2001:01:12:14 +0100]  "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 208 "-" "-"
a8  - - [05/Nov/2001:01:12:14 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 0
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /issi/archive/literature/tech-reports/643/unilab.gif HTTP/1.1" 200 1885
a10  - - [05/Nov/2001:01:12:15 +0100]  "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
a11  - - [05/Nov/2001:01:12:15 +0100]  "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 83588
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /magerste HTTP/1.0" 301 248
a14  - - [05/Nov/2001:01:12:16 +0100]  "GET /magerste/ HTTP/1.0" 200 844
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100]  "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a17  - - [05/Nov/2001:01:12:16 +0100]  "GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.1" 200 311
a18  - - [05/Nov/2001:01:12:17 +0100]  "GET /icons/image2.gif HTTP/1.0" 200 309
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a23  - - [05/Nov/2001:01:12:17 +0100]  "GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a24  - - [05/Nov/2001:01:12:17 +0100]  "GET /icons/blank.gif HTTP/1.0" 200 148
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100]  "GET /magerste/pope1.jpg HTTP/1.0" 200 21994
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100]  "GET /scripts/..%%35%32%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
a27  - - [05/Nov/2001:01:12:18 +0100]  "GET /magerste/cocacola3.gif HTTP/1.0" 200 5970
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%25%2%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100]  "GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a31  - - [05/Nov/2001:01:12:19 +0100]  "GET /magerste/cocacola2.gif HTTP/1.0" 200 16724
a32  - - [05/Nov/2001:01:12:20 +0100]  "GET /magerste/cam1.jpg HTTP/1.0" 200 9519

Short=a1 Long=U10MvwFCP1kVNDj_1SKr8w:1:7Eu59aI8FQEryoQcJHIprU9RvV1vnevU1w1k Group=I1
Short=a2 Long=5JDMjaZdWV4MjCTRgJBA0w:2:7Eu59aI8FQEryoQcJHIprU9RvV1vnevU1w1k Group=I1
Short=a3 Long=EJ09GJbgR3IMnVK0W5IZZg:1:6UB1kZzFW1Dj3X_7d2KV5cQWavS1WV0ZUtNH Group=I1
Short=a4 Long=1jKtPmzNeuf6ZW1GKZr4tA:3:7Eu59aI8FQEryoQcJHIprU9RvV1vnevU1w1k Group=I1
Short=a7 Long=B5VS78wDTB10QCcuzE5yKQ:4:7Eu59aI8FQEryoQcJHIprU9RvV1vnevU1w1k Group=I1
Short=a8 Long=odp85zv7Dncb9Tbhhv_uRg:2:6UB1kZzFW1Dj3X_7d2KV5cQWavS1WV0ZUtNH Group=I1
Short=a10 Long=smo62aU0oCANOE8zTbrudg:5:7Eu59aI8FQEryoQcJHIprU9RvV1vnevU1w1k Group=I1
Short=a11 Long=RnRjPB4NkHaEQ3ja1Q644A:1:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a14 Long=1TGjCsMog3t7nzS5t6HkNwA:2:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a17 Long=k3xdmAJrhADzzolV8ApmJv:3:6UB1kZzFW1Dj3X_7d2KV5cQWavS1WV0ZUtNH Group=I1
Short=a18 Long=yVUDud4nYzQVpdK_ikYcFoQ:3:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a23 Long=_rYpbDjOnjg9T2cLNDs_dw:4:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a24 Long=PyeQ4sHf8j18R48Ptab7Yv:5:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a27 Long=_doFgnDtxEsSO1jNGh8q68Q:6:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a31 Long=chQzT9vRRL5MB192dW1cdw:7:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1
Short=a32 Long=IUu5c3A1H6kEtKOQCvErcQ:8:tvZ7EXaWwh7r013VPjhJokgoROmkarch610X Group=I1

```
GROUP I1 THRESHOLD 16
GROUP I2 THRESHOLD 100
FACILITY redirector EVENT ',?GET \/MSADC\/root\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/c\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/d\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%255c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/_vti_bin\/\.\.%255c\.\.\.\.%255c\.\.\.\.%255c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/_mem_bin\/\.\.%255c\.\.\.\.%255c\.\.\.\.%255c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/msadc\/\.\.%255c\.\.\/\.\.%255c\.\.\/\.\.%255c\.\.\.\.%c1%1c\.\.\.\.%c1%1c\.\.\.\.%c1%1c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%c1%1c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%c0%2f\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%c0%af\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%c1%9c\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%%35%63\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%25%35%63\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/\.\.%252f\.\.\/winnt\/system32\/cmd\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET \/scripts\/root\.exe\?\/c\+dir HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I1 WEIGHT const(1)
FACILITY redirector EVENT ',?GET (.*) HTTP\/'
          LEFTFEATURE '*', RIGHTFEATURE ' -- \[' GROUP I2 WEIGHT const(1)
```

Figure 17: Sample configuration 2) for pseudonymizing *Apache* audit records, such as in Fig. 13

```
a1   - -  [05/Nov/2001:01:11:17 +0100] "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.html HTTP/1.0" 200 11804
a2   - -  [05/Nov/2001:01:11:20 +0100] "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.css HTTP/1.0" 200 344
a3   - -  [05/Nov/2001:01:12:13 +0100] "GET /issi/archive/literature/tech-reports/643.html HTTP/1.1" 200 8825
a4   - -  [05/Nov/2001:01:12:14 +0100] "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 200 0
a5   - -  [05/Nov/2001:01:12:14 +0100] "GET /scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-" "-"
a6   - -  [05/Nov/2001:01:12:14 +0100] "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 208 "-" "-"
a7   - -  [05/Nov/2001:01:12:14 +0100] "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 0
a8   - -  [05/Nov/2001:01:12:14 +0100] "GET /issi/archive/literature/tech-reports/643/unilab.gif HTTP/1.1" 200 1885
a9   - -  [05/Nov/2001:01:12:15 +0100] "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
a10  - -  [05/Nov/2001:01:12:15 +0100] "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 83588
a11  - -  [05/Nov/2001:01:12:15 +0100] "GET /magerste HTTP/1.0" 301 248
a12  - -  [05/Nov/2001:01:12:15 +0100] "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
a13  - -  [05/Nov/2001:01:12:15 +0100] "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a14  - -  [05/Nov/2001:01:12:16 +0100] "GET /magerste/ HTTP/1.0" 200 844
a15  - -  [05/Nov/2001:01:12:16 +0100] "GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
a16  - -  [05/Nov/2001:01:12:16 +0100] "GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
a17  - -  [05/Nov/2001:01:12:16 +0100] "GET /issi/archive/literature/tech-reports/643/unido.gif HTTP/1.1" 200 311
a18  - -  [05/Nov/2001:01:12:17 +0100] "GET /icons/image2.gif HTTP/1.0" 200 309
a19  - -  [05/Nov/2001:01:12:17 +0100] "GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 265 "-" "-"
a20  - -  [05/Nov/2001:01:12:17 +0100] "GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a21  - -  [05/Nov/2001:01:12:17 +0100] "GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a22  - -  [05/Nov/2001:01:12:17 +0100] "GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a23  - -  [05/Nov/2001:01:12:17 +0100] "GET /icons/blank.gif HTTP/1.0" 200 148
a24  - -  [05/Nov/2001:01:12:18 +0100] "GET /magerste/popel.jpg HTTP/1.0" 200 21994
a25  - -  [05/Nov/2001:01:12:18 +0100] "GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
a26  - -  [05/Nov/2001:01:12:18 +0100] "GET /scripts/..%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
a27  - -  [05/Nov/2001:01:12:18 +0100] "GET /magerste/cocacola3.gif HTTP/1.0" 200 5970
a28  - -  [05/Nov/2001:01:12:19 +0100] "GET /scripts/..%35c../..%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
a29  - -  [05/Nov/2001:01:12:19 +0100] "GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a30  - -  [05/Nov/2001:01:12:19 +0100] "GET /magerste/cocacola2.gif HTTP/1.0" 200 16724
a31  - -  [05/Nov/2001:01:12:19 +0100] "GET /magerste/cocacola2.gif HTTP/1.0" 200 16724
a32  - -  [05/Nov/2001:01:12:20 +0100] "GET /magerste/cam1.jpg HTTP/1.0" 200 9519

Short=a1  Long="FniTV0fiLgpm3M10QZbmZSKg:1:4aJtIJOAHWauxrbr8_6z6l2ZRtUPvja7FvEH Group=I2
Short=a2  Long="NtMFb8eFT6SNKxJyHDgDfA:2:4aJtIJOAHWauxrbr8_6z6l2ZRtUPvja7FvEH Group=I2
Short=a3  Long="4qBNWilauKATwqtpgKUZVw:1:FFWEV7t1l5GgVHA98V1grFPiaWU44RThYHXJ Group=I2
Short=a4  Long="UIo2WKiLQ8oh3B1M2zO2Kvw:3:4aJtIJOAHWauxrbr8_6z6l2ZRtUPvja7FvEH Group=I2
Short=a5  Long="G7KdCMPxIfcO_u9gLX8rRQi:1:wjjArj_FdL9UMxkqW5E Group=I1
Short=a6  Long="Pq6iZo117oikBrqFz4_IVQ:2:wjjArj_FdL9UMxkqW5E Group=I1
Short=a7  Long="hdHyJJF6mvLZco4S2RQTzg:4:4aJtIJOAHWauxrbr8_6z6l2ZRtUPvja7FvEH Group=I2
Short=a8  Long="AvE8_Cd1YeCUVanCLnbLSA:2:FFWEV7t1l5GgVHA98V1grFPiaWU44RThYHXJ Group=I2
Short=a9  Long="UH0MKtbqHufx01pqL_K4Hw:3:wjjArj_FdL9UMxkqW5E Group=I1
Short=a10 Long="8PvFO6Sc9eMhAXyD31S2Iw:5:4aJtIJOAHWauxrbr8_6z6l2ZRtUPvja7FvEH Group=I2
Short=a11 Long="hYfKTIUzz4nEK3GoDUUgIHg:1:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a12 Long="n_3jp9lRrdkD4kt6FVW5CoQ:4:wjjArj_FdL9UMxkqW5E Group=I1
Short=a13 Long="di1lzzxR1OjLyZj9TfMMswA:5:wjjArj_FdL9UMxkqW5E Group=I1
Short=a14 Long="pei56lNYeMpEvmAAuxW5cA:2:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a15 Long="WZTWDai0htkibI11JhWMckw:6:wjjArj_FdL9UMxkqW5E Group=I1
Short=a16 Long="tbtEU_v8r299MtKj6gsPxQ:7:wjjArj_FdL9UMxkqW5E Group=I1
Short=a17 Long="HLhQ4QLLBLq_MiqG4Yx5jw:3:FFWEV7t1l5GgVHA98V1grFPiaWU44RThYHXJ Group=I2
Short=a18 Long="v5lN0y6kG6RunCDFbitAtv:3:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a19 Long="ZasYWfHppbpqd5PvStktOCI6g:8:wjjArj_FdL9UMxkqW5E Group=I1
Short=a20 Long="TrwS9noSYgEd1CpYHBn4i1A:9:wjjArj_FdL9UMxkqW5E Group=I1
Short=a21 Long="1VJ8TYM_nYe754pSSCTB7w:a:wjjArj_FdL9UMxkqW5E Group=I1
Short=a22 Long="4SCbAYkoGE5FRE4WrWb11Q:b:wjjArj_FdL9UMxkqW5E Group=I1
Short=a23 Long="9DkmmXInxB7RBruhd1HiJwg:4:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a24 Long="YBF6TjBwH98Y1aoSfjJstg:5:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a25 Long="vsvod_sSGR1hD5vvmlPnSw:c:wjjArj_FdL9UMxkqW5E Group=I1
Short=a26 Long="Mtmrf1CD1DMp3RuqLwAhC4DQ:e:wjjArj_FdL9UMxkqW5E Group=I1
Short=a27 Long="0qqBSe24GZ6OHrfWk_KVgg:6:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a28 Long="p_7_vtFDBFg3nPcgMLbsXg:f:wjjArj_FdL9UMxkqW5E Group=I1
Short=a29 Long="rqcmomLniLZo4naaqfT6Ezg:7:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a30 Long="bd3UCd2doORnCfQxEvU5Q:10:wjjArj_FdL9UMxkqW5E Group=I1
Short=a31 Long="rqcmomLniLZo4naaqfT6Ezg:7:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
Short=a32 Long="0PmatKb_QeNZfE13vOX9rA:8:dUpDGcysnqnY1airzB0Pna62b_SeqQDoUwzh Group=I2
```

Figure 18: Sample audit records from Fig. 13 pseudonymized using configuration 2) in Fig.17

```
a1      - - [05/Nov/2001:01:11:17 +0100] "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.html HTTP/1.0" 200 11804
a2      - - [05/Nov/2001:01:11:20 +0100] "GET /issi/teaching/lectures/96ws/Informationssysteme/manual.css HTTP/1.0" 200 344
a3      - - [05/Nov/2001:01:12:13 +0100] "GET /issi/archive/literature/tech-reports/643.html HTTP/1.1" 200 8825
a4      - - [05/Nov/2001:01:12:14 +0100] "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 200 0
217.82.228.10 - - [05/Nov/2001:01:12:14 +0100] "GET /scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:14 +0100] "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 208 "-" "-"
a7      - - [05/Nov/2001:01:12:14 +0100] "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 0
a8      - - [05/Nov/2001:01:12:14 +0100] "GET /issi/archive/literature/tech-reports/643/unilab.gif HTTP/1.1" 200 1885
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100] "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
a10     - - [05/Nov/2001:01:12:15 +0100] "GET /ir/teaching/lectures/is_ws01-02/Uebungsblaetter/infos.pdf HTTP/1.0" 206 83588
a11     - - [05/Nov/2001:01:12:15 +0100] "GET /magerste HTTP/1.0" 301 248
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100] "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:15 +0100] "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a14     - - [05/Nov/2001:01:12:16 +0100] "GET /magerste/ HTTP/1.0" 200 844
217.82.228.10 - - [05/Nov/2001:01:12:16 +0100] "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:16 +0100] "GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:16 +0100] "GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 249 "-" "-"
a18     - - [05/Nov/2001:01:12:16 +0100] "GET /issi/archive/literature/tech-reports/643/unido.gif HTTP/1.1" 200 311
a18     - - [05/Nov/2001:01:12:17 +0100] "GET /icons/image2.gif HTTP/1.0" 200 309
a       - - [05/Nov/2001:01:12:17 +0100] "GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 265 "-" "-"
a       - - [05/Nov/2001:01:12:17 +0100] "GET /icons/blank.gif HTTP/1.0" 200 148
a24     - - [05/Nov/2001:01:12:17 +0100] "GET /magerste/popel.jpg HTTP/1.0" 200 21994
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100] "GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 231 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:18 +0100] "GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
a27     - - [05/Nov/2001:01:12:18 +0100] "GET /magerste/cocacola3.gif HTTP/1.0" 200 5970
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100] "GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100] "GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
217.82.228.10 - - [05/Nov/2001:01:12:19 +0100] "GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"
a31     - - [05/Nov/2001:01:12:19 +0100] "GET /magerste/cocacola2.gif HTTP/1.0" 200 16724
a32     - - [05/Nov/2001:01:12:20 +0100] "GET /magerste/cam1.jpg HTTP/1.0" 200 9519
Short=a1  Long="FnlV0fIgpm3M10QZbmZSKg:1:4aJtIJOAHWauxrbr8_6z6I22RtUPvja7FvEH Group=I2
Short=a2  Long="NtMFb8eFT65NKkJyHDgPfA:2:4aJtIJOAHWauxrbr8_6z6I22RtUPvja7FvEH Group=I2
Short=a3  Long="4qBNWllauKATvqtpg6UZVw:1:FFWEV7t1l5GgVHA98V1grFPiaWU44RThYHKJ Group=I2
Short=a4  Long="UIoZWKLQ8oh3B1M2zO2Kuw:3:4aJtIJOAHWauxrbr8_6z6I22RtUPvja7FvEH Group=I2
Short=a7  Long=hdHyJJF6mvLZco4S2RdQTzg:4:4aJtIJOAHWauxrbr8_6z6I22RtUPvja7FvEH Group=I2
Short=a8  Long=AwEB_Cd1Y6cUVamCLnbLSA:2:FFWEV7t1l5GgVHA98V1grFPiaWU44RThYHKJ Group=I2
Short=a10 Long=8PvFO6Sc9sMhAXyD31S2lw:5:4aJtIJOAHWauxrbr8_6z6I22RtUPvja7FvEH Group=I2
Short=a11 Long=hYfKTIUzz4nEK3GoDUQIHg:1:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
Short=a14 Long=rpei56lNYeMpEvnAAuxW5cA:2:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
Short=a17 Long=HLhd4QLLBLq_M1q04YxSjv:3:FFWEV7t1l5GgVHA98V1grFPiaWU44RThYHKJ Group=I2
Short=a18 Long=v6lN0y6kG6RunCDFbitAtv:3:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
Short=a23 Long=90kxmxiWx87RBruhdtHJwg:4:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
Short=a24 Long=YBF6TjBwH98Y1aoSfjJstg:5:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
Short=a27 Long=0qpBSe24GZ60HrfWk_KVgg:6:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
Short=a31 Long=rqcmomLniLZo4maagfT6Ezg:7:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
iShort=a32 Long=0PmaatKb_QeNZfEl3vOX9rA:8:dUpDGcysnqnY1a1rzB0Pna62b_SeqQDoUwzh Group=I2
```

Figure 19: Sample pseudonymized audit records from Fig. 18 with recovered features