A Novel Policy-driven Reversible Anonymisation Scheme for XML-based Services

Nils Ulltveit-Moe*, Vladimir Oleshchuk

Department of Information and Communication Technology
University of Agder
Jon Lilletuns vei 9, 4879 Grimstad, Norway

Abstract

This paper proposes a reversible anonymisation scheme for XML messages that supports fine-grained enforcement of XACML-based privacy policies. Reversible anonymisation means that information in XML messages is anonymised, however the information required to reverse the anonymisation is cryptographically protected in the messages. The policy can control access down to octet ranges of individual elements or attributes in XML messages. The reversible anonymisation protocol effectively implements a multi-level privacy and security based approach, so that only authoried stakeholders can disclose confidential information up to the privacy or security level they are authorised for. The approach furthermore supports a shared secret based scheme, where stakeholders need to agree to disclose confidential information. Last, it supports time limited access to private or confidential information. This opens up for improved control of access to private or confidential information in XML messages used by a service oriented architecture. The solution provides horizontally scalable confidentiality protection for certain types of big data applications, like XML databases, secure logging and data retention repositories.

Keywords: Reversible anonymisation, privacy, XML-encryption, XACML, deanonymiser, big data

1. Introduction

This paper [1] proposes an innovative approach for reversible anonymisation of private or confidential information in XML messages. It extends the eXtensible Access Control Markup Language (XACML) based decision cache and anonymiser for XML documents, proposed in [23], with support for reversible anonymisation of private or confidential information based on broadcast encryption. An advantage compared to the original approach, is that it supports multi-level privacy for confidential content in the XML messages, so that only authorised parties can access this information. This simplifies handling of data with multiple security levels, since these can be stored together, protected by encryption. It is assumed that all connections have basic security (e.g. encrypted using TLS/SSL), to avoid cleartext attacks on underlying communication channels.

Big data analytics often depend on small data inputs, including information about people, places and things collected by sensors, cell phones, click patterns and the like [11]. These small data inputs are aggregated to produce large datasets which analytic techniques mine for insight [11]. It is assumed that sensitive parts of information for such scenarios can be protected as close as possible to the information sources. This means that private or confidential information under these assumptions can be protected before the data is being aggregated into a big data stream.

The proposed solution allows for retrofitting security as an add-on to certain types of big data consumers, like XML document databases. It is a horizontally scalable approach that may be used for protecting the privacy and confidentiality of sensitive data produced by web services or other XML-based data sources. The approach allows for both confidentiality and integrity protection of the XML data based on XML encryption. Such a solution can for example be useful for secure logging, data retention, protecting private or confidential information in SmartGrid-based systems (e.g. Demand-Response systems) or for Security Incident and Event Management (SIEM) systems. The approach furthermore supports location-aware authorisation and anonymisation of data, by using the GeoXACML framework [1].

The reversible anonymisation enforcement scheme has been successfully demonstrated for a SIEM system anonymising XML-based Intrusion Detection System (IDS) alarms in the Intrusion Detection Message Exchange Format (IDMEF) [8]. This means that the proposed approach can be used to implement privacy-enhanced IDS

*Corresponding author
Email addresses: `Nils.Ulltveit-Moe@uia.no` (Nils Ulltveit-Moe*), `Vladimir.Oleshchuk@uia.no` (Vladimir Oleshchuk)
[1] Note that this is a preprint version, not including final comments from the reviewers. The published version of the paper can be found in Information Systems, Elsevier Volume 48, March 2015, Pages 164–178, doi:10.1016/j.is.2014.05.007 at `http://www.sciencedirect.com/science/article/pii/S030643791400091X`.

services. The proposed scheme is general, and it is envisaged that it in the future will be integrated in an Enterprise Service Bus (ESB), to provide on-demand policy controlled reversible anonymisation of information in any XML-based web service.

The original use case for the reversible anonymisation scheme is privacy-enhanced intrusion detection system based services. It can be noted that the IDMEF-based IDS alarm format is a semi-structured XML format, that supports arbitrary extensions via the IDMEF AdditionalData construct. Performing data mining of such semi-structured data can be a challenge with existing relational databases, meaning that a non-relational data representation may be required for efficient data processing. Horizontal scalability is typically required for efficient processing of IDS alarms. Horizontal scalability for IDS is often implemented by having local data repositories for IDS alarms, and using an event correlation system to reduce the number of IDS alarms sent for central processing by a Security Operations Centre. The proposed reversible anonymisation scheme supports such a use case by being inherently parallelisable, so that IDS alarms from different sensors can be processed by individual anonymisers.

The proposed solution can be considered as security control extensions for XML databases that include shared secret based authorisation (which can be used as a building block for multifactor authentication), data encryption and anonymisation. This can be used to address some of the privacy concerns on big data repositories based on XML databases. It can also be used to increase the transparency of anonymised services by supporting secure logging schemes. This means that the approach can be a first step towards solving the transparency paradox - that big data operators pervasively collect all manner of private information, however with the operations of big data itself being almost entirely shrouded in legal and commercial secrecy [11].

The paper is organised as follows: The next section discusses the reversible anonymisation scheme, including notation used, background and motivation for implementing reversible anonymisation and a high-level description of the reversible anonymisation process. Section 3 describes how the reversible anonymisation protocol is extended to support a default DENY anonymisation scheme. Section 4 describes the adaptations required to support key sharing, and section 5 describes how time-based data expiry can be implemented, in order to support time-limited data retention. Section 6 describes the results from performance tests of the anonymiser and the deanonymiser, and section 7 discusses advantages and disadvantages with the proposed approach. Section 8 discusses related work, section 9 concludes the paper and section 10 outlines future work.

2. Reversible Anonymisation Scheme

This section first describes the background and motivation for the reversible anonymisation scheme proposed in this paper, and then outlines how the anonymisation process works from a high level perspective. The reversible anonymisation scheme is implemented as an extension of the anonymising decision cache proposed in [23].

2.1. Background and Motivation

The reversible anonymisation scheme is useful for dynamically configuring confidentiality protection of XML-based web services in a service oriented architecture. This provides a possibility to enforce the security and privacy of existing services by running these services through the anonymiser. Authorised users or services can subsequently deanonymise and use information in security levels they have clearance for. This provides a flexible, policy driven protection scheme for private or confidential information, where protection mechanisms can be added on demand.

General functionality that the reversible anonymisation protocol provides, is irreversible and reversible anonymisation of information in XML messages controlled by XACML policies. In addition, it supports key sharing, which can be used to enforce separation of duties constraints - for example so that different stakeholders need to agree to disclose confidential information to reduce the risk of insider attacks. The scheme can also be used to implement trustworthy deployment of system configurations. The approach furthermore supports time-limited access to sensitive data, which can be used to support data retention and secure logging mechanisms for XML databases.

It is expected that such a general policy-driven reversible anonymisation scheme will be useful in a range of different use cases, including outsourcing - for example to cloud-based services, e-health, e-commerce, critical infrastructures and managed security services, which is the practical use case considered in this paper.

Reversible anonymisation here means that sensitive information in the XML messages is anonymised, however necessary information required to reverse the anonymisation process is stored encrypted in the XML message. Reversible anonymisation is different from traditional pseudonymisation since no pseudonym is used to replace the anonymised XML element. Instead the encrypted information required to reverse the anonymisation is added to the XML message. There are several advantages by using such a strategy compared to a traditional pseudonymisation strategy. First, this means that there is no need to consider the cryptographic strength of a pseudonymisation scheme to avoid linkability between pseudonyms, since pseudonyms are not used. Second, the anonymised data can be any data, for example an informative text, replacing the anonymised data. This informative text may even contain nonsensitive parts of the original data (for example the most significant part of an

IP range)[2]. Third, having larger chunks of encrypted data reduces the risk of leaking information via traffic analysis.

The reversible anonymisation scheme builds key distribution into the XACML policies, so that the Policy Enforcement Point (PEP) queries the Policy Decision Point (PDP) about which public keys that are authorised to access information in the XML messages.

It is assumed that the Security Assertion Markup Language (SAML) or similar is used both to authorise the data consumer to receive anonymised messages, and also to authorise individual users for access to sensitive information in the messages. SAML supports automatic logout of users by using the single logout protocol of SAML 2.0 [19].

2.2. Outline of the Reversible Anonymisation Process

The cryptographic problem that must be solved, is how to cryptographically protect confidential information, so that only authorised personnel can access the information on a needs basis. It is assumed that information may be split into $d > 1$ different security levels (for example Restricted, Confidential, Secret). There may furthermore be one or more consumers of XML messages, where each consumer is authorised to a given subset of security levels. The solution proposed here does not assume any semantics or relationships between the security levels, apart from controlling who have access to which security levels. It is assumed that additional semantics, e.g. that access to level Secret also includes access to level Confidential and Restricted, can be enforced by the XACML policies controlling the anonymisation policy. Some stakeholders, for example law enforcement or CERT, may have a need to access information with a higher level of sensitivity than a first-line service. It is also desirable to enforce a shared secret scheme to ensure that two or more parties must agree by providing their key shares before sensitive information is disclosed (for example that a data controller and law enforcement must agree to disclose a given set of sensitive data).

The following notation is used in the paper. A pseudorandom number generator is denoted by $rnd()$, an encryption function is denoted by $Enc(key, value)$ and decryption function by $Dec(key, value)$. Furthermore, $value'$ denotes the encrypted or anonymised value, i.e. $value' = Enc(key, value)$ and $value = Dec(key, value')$. The list of notation used in the paper can be found in Table 1.

Fig. 1 gives overview over how the encryption scheme used to implement reversible anonymisation and multilevel security is implemented. The confidential information l_i in each security level $i \in \{1, ..., d\}$ consists of both a specification on how to reverse the anonymisation using XPath expressions and the confidential information identified using these XPath expressions. The information in

Index	Meaning
d	Number of security levels.
i	Security level
j	Share index of encryption key $K_{i,j}$.
N	Number of resources.
n	Number of key shares.
m	Number of users.
p	XPath expression number.
q	Match number for XPath expression p.
u	User number.
z	Number of matches of r_p.

Type	Meaning
C	Set of classified XML elements.
$\underline{e_p}$	$\underline{e_p} = (e_{p,1}, e_{p,2}, ..., e_{p,z})$ identified by XPath resource r_p on $message$.
$e_{p,q}$	XML element q identified by evaluating XPath resource r_p on $message$.
K_i	Encryption key for security level i.
KM	Set of keymap tuples consisting of encryption key K_i and security level i.
l_i	Information on security level i.
$\underline{L}$	Vector of all confidential information l_i.
Λ_u	Keymaps user u is authorised for.
AM	Authorisation map from public keys PK_u to the set of keymaps $\Lambda_u \subseteq KM$ the user u is authorised for.
PK_u	Public key of user u.
r_p	XPath expression for p identifying resources that need authorisation.
R	Set of all r_p.
S	Set of encrypted keys.
SK_u	Secret key of user u.
t_{exp}	Key expiry time.
$t_{retention}$	Data retention time.
Q	Matrix of authorised elements $e_{p,q}$ for resouce p and security level i.

Table 1: List of notations

[2]Note however that such a strategy should be used with care for private or confidential data to avoid reducing the anonymity set unduly for the underlying data.

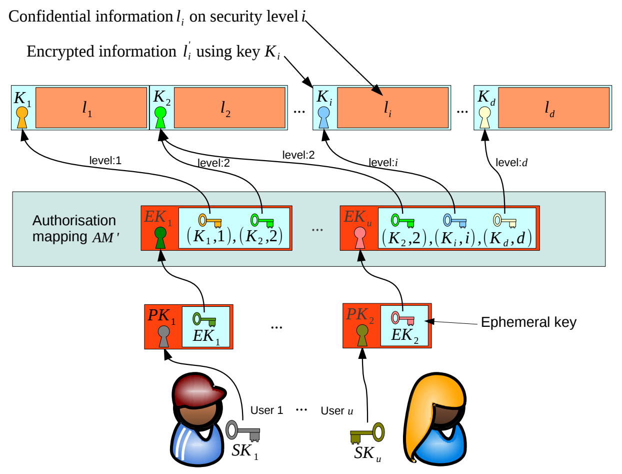


Figure 1: Overview over encryption scheme used to implement reversible anonymisation.

each security level l_i is encrypted using the corresponding encryption key K_i for the security level referenced by the security level index (or label) i. In this paper it is assumed for simplicity that the security level index is a natural number[3].

Furthermore, users can be authorised to a subset of all security levels. In Fig. 1, User 1 has access to security levels 1 and 2, and User u has access to security levels 2, i and d. The authorisation mapping (AM) for each user is encrypted using a two-stage process, where a symmetric key EK_u is used to encrypt the AM, and the public key PK_u of the authorised user u is used to encrypt the EK_u. EK_u is regenerated each time a user is authorised for access to the data, and has a configurable key renegotiation timeout. The authorisation mapping, encryption keys and the number of security levels are controlled by XACML policies. An advantage by using a separate key for encrypting the key mapping, is to avoid having to use relatively slow public key cryptography for encrypting the AM. Another advantage is that standardised XML Encryption methods can be used for key wrapping.

A given user u, can then decrypt the EK_u using her private key SK_u, which in turn can be used to decrypt the tuples (K_i, i) that the user is authorised for. The index i can subsequently be used to retrieve the corresponding encrypted information $l_i^{'}$ which then can be decrypted using the corresponding key K_i. The confidential information in l_i both contains the confidential data that is anonymised in the original IDS alarm and a specification (XPath expressions) that describes how the anonymisation for the given security level can be reversed. The approach used for encrypting information in security levels is similar to

[3]However it may also be implemented using textual labels (e.g. “secret”) if the underlying data structure is implemented as an associative array.

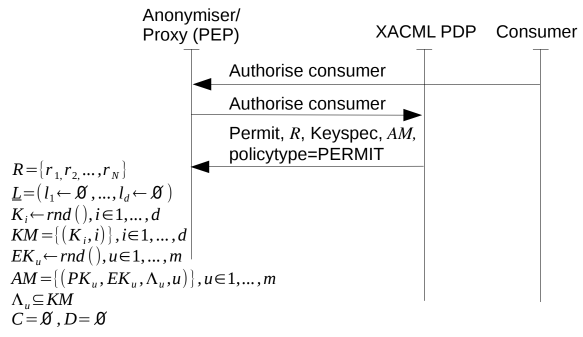

Anonymiser/ Proxy (PEP)
XACML PDP
Consumer
Authorise consumer
Authorise consumer
Permit, R, Keyspec, AM, policytype=PERMIT
$R=\{r_1, r_2, \ldots, r_N\}$
$\underline{L}=(l_1 \leftarrow \emptyset, \ldots, l_d \leftarrow \emptyset)$
$K_i \leftarrow rnd(), i \in 1, \ldots, d$
$KM=\{(K_i, i)\}, i \in 1, \ldots, d$
$EK_u \leftarrow rnd(), u \in 1, \ldots, m$
$AM=\{(PK_u, EK_u, \Lambda_u, u)\}, u \in 1, \ldots, m$
$\Lambda_u \subseteq KM$
$C=\emptyset, D=\emptyset$

Figure 2: Initial authorisation sequence.

broadcast encryption [5].

This approach gives flexibility for authorising access to information on a given security level according to operative needs. It may for example be desirable to enforce separation of duties between information considered secret by customer A and information considered secret by another customer B, by authorising different trusted CERT teams to access this information in each organisation.

2.3. Reversible Anonymisation Protocol

This subsection describes the reversible anonymisation protocol. The detailed description of the protocol is given in Fig. 6.

Initially, the consumer sends a SAML assertion with proof of authenticity to log in to the Anonymiser/Proxy PEP, as shown in Fig. 2. The PEP will then ask the XACML PDP for authorisation of the consumer. If the response is *Permit*, then the XACML response will contain a set of obligations that amongst others contain the set of $N \geq 0$ resource identifying expressions (XPath expressions) $R = \{r_1, r_2, ..., r_N\}$ which identify information that needs authorisation. The reply furthermore contains the Keyspec, which specify keys such as the public keys PK_u, and information on how to generate ephemeral keys EK_u, $u \in \{1, ..., m\}$. It also specifies the number of security levels d as well as how to generate the unique encryption keys K_i, $i \in \{1, ..., d\}$ that are used. Then the vector of confidential information $\underline{L}$ is initialised to the empty vector for all security levels, i.e: $\underline{L} = (l_1 \leftarrow \emptyset, ..., l_d \leftarrow \emptyset)$, and the encryption keys are initialised to a random number using the specified key generation algorithm, i.e: $K_i \leftarrow rnd()$, $i \in \{1, ..., d\}$.

The encryption keys are used to generate a set of keymaps $KM = \{KM_i | i \in 1, ..., d\}$, where each keymap $KM_i = (K_i, i)$ is a tuple consisting of the encryption key K_i and an index i referring to the confidential information l_i on security level i. The ephemeral keys EK_u for each $u \in \{1, ..., m\}$ are subsequently generated.

The initial XACML response also contains a key authorisation mapping $AM = \{(PK_u, EK_u, \Lambda_u, u) | u \in$

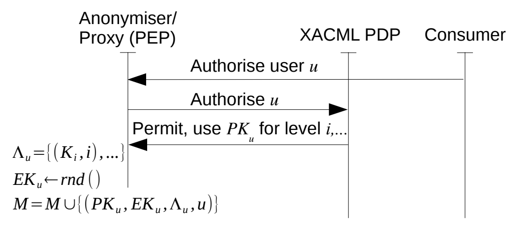

Anonymiser/ Proxy (PEP)
XACML PDP
Consumer
Authorise user u
Authorise u
Permit, use PK_u for level i,...
$\Lambda_u = \{(K_i, i), ...\}$
$EK_u \leftarrow rnd()$
$M = M \cup \{(PK_u, EK_u, \Lambda_u, u)\}$

Figure 3: Authorise user.

$1, ..., m\}$, which describes who (i.e. which user u's public keys PK_u) that are authorised to access data at which security levels (via the ephemeral keys EK_u). Here $\Lambda_u \subseteq KM$ is the subset of keymaps indicating which security levels i the user is authorised for.

It is assumed that deployment of XACML policies describing the key mapping is being controlled by the information owner, so that the key owners themselves will not be allowed to modify this mapping. Finally, the set of classified elements, denoted as C, and elements explicitly declassified, denoted as D, are initialised to the empty set. The relative complement $C \backslash D$ describes the set of XML elements or attributes that needs to be anonymised.

The process of subsequent authorisation of a user u is shown in Fig. 3. The user asks the Anonymiser to be authorised for access to a set of one (or more) security level(s) $L_u \subseteq \{1, ..., d\}$. If the analyst has access according to the security policy, then the XACML responds with *Permit*. The response contains an obligation with the public key of the analyst PK_u and an authorisation mapping showing that this analyst is authorised for the set of security levels L_u. The PEP first generates the ephemeral key EK_u and then adds this decision to the key authorisation mapping: $AM = AM \cup \{(PK_u, EK_u, \Lambda_u, u)\}$, where $\Lambda_u = \{KM_i | i \in L_u\}$ so that the user is authorised for accessing confidential information in l_i.

After that, the authorisation of elements and attributes in the XML message starts, as shown in Fig. 4. First each resource identifying XPath expression r_p, $p \in \{1, ..., N\}$ is evaluated on the XML message to get a vector of z matching XML elements:

$$\underline{e_p} = (e_{p,1}, e_{p,2}, ..., e_{p,z}) \leftarrow XPATH(r_p, message). \quad (1)$$

The PEP then needs to authorise each of the elements in $\underline{e_p}$ by querying the PDP for which security level(s) each element is authorised for. If access to the element $e_{p,q}$ is granted, then the XACML Response contains a *Permit* decision with an obligation to anonymise the given element which also contains which security level(s), denoted *levels*, that are authorised to reverse the anonymisation.

The anonymiser then iterates through all levels $i \in levels$ and stores the value of each element $e_{p,q}$ in a matrix $Q_{i,q}$, so that each row vector Q_i contains the confidential

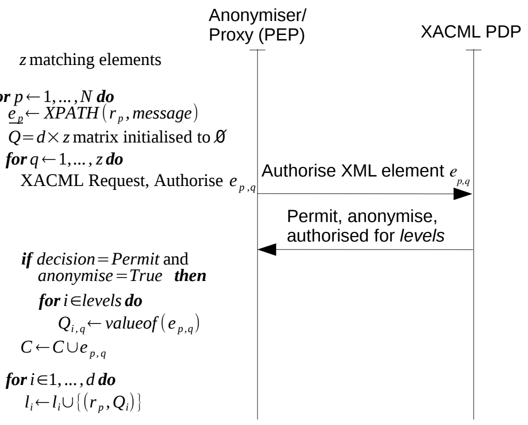

Anonymiser/ Proxy (PEP)
XACML PDP
z matching elements
for $p \leftarrow 1, ..., N$ **do**
$\underline{e_p} \leftarrow XPATH(r_p, message)$
$Q = d \times z$ matrix initialised to $\emptyset$
for $q \leftarrow 1, ..., z$ **do**
XACML Request, Authorise $e_{p,q}$
Authorise XML element $e_{p,q}$
Permit, anonymise, authorised for *levels*
if $decision = Permit$ and $anonymise = True$ **then**
for $i \in levels$ **do**
$Q_{i,q} \leftarrow valueof(e_{p,q})$
$C \leftarrow C \cup e_{p,q}$
for $i \in 1, ..., d$ **do**
$l_i \leftarrow l_i \cup \{(r_p, Q_i)\}$

Figure 4: XML Element Authorisation

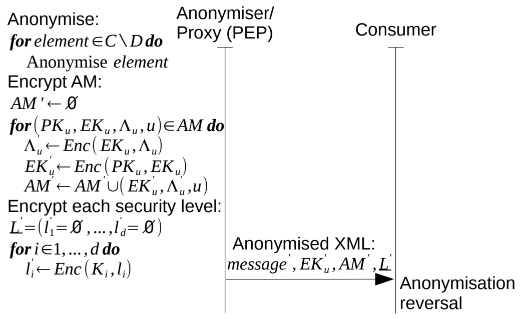

Anonymise:
for $element \in C \backslash D$ **do**
Anonymise *element*
Encrypt AM:
$AM' \leftarrow \emptyset$
for $(PK_u, EK_u, \Lambda_u, u) \in AM$ **do**
$\Lambda_u' \leftarrow Enc(EK_u, \Lambda_u)$
$EK_u' \leftarrow Enc(PK_u, EK_u)$
$AM' \leftarrow AM' \cup (EK_u', \Lambda_u', u)$
Encrypt each security level:
$\underline{L}' = (l_1' = \emptyset, ..., l_d' = \emptyset)$
for $i \in 1, ..., d$ **do**
$l_i' \leftarrow Enc(K_i, l_i)$
Anonymiser/ Proxy (PEP)
Consumer
Anonymised XML:
$message', EK_u', AM', \underline{L}'$
Anonymisation reversal

Figure 5: Anonymisation enforcement and reversal for default PERMIT policies.

elements of $\underline{e_p}$ that security level i is authorised for, and $\emptyset$ otherwise. After that, the current element $e_{p,q}$ is added to the list of classified elements C, i.e: $C \leftarrow C \cup e_{p,q}$, that later will be anonymised.

The last part of element authorisation is to iterate through all security levels $i \in \{1, ..., d\}$ and add a tuple (r_p, Q_i), consisting of resource identifier r_p and confidential information for the given resource identifier Q_i to the set of confidential information l_i for security level i, i.e: $l_i \leftarrow l_i \cup \{(r_p, Q_i)\}$. This essentially means that a specification on how to undo the anonymisation has been stored in l_i.

When all elements matching the set of resources R have been authorised, the algorithm proceeds to the enforcement part, as shown in Fig. 5. Enforcement starts with anonymising all classified elements in C that have not been explicitly declassified (that are not part of D) i.e. $C \backslash D$. The anonymiser then loops through the authorisation mapping tuples in AM, and encrypts Λ_u with EK_u, and then encrypts EK_u using the user u's public key PK_u. After that, the anonymiser adds a tuple consisting of the encrypted ephemeral key $EK_u^{'} \leftarrow enc(PK_u, EK_u)$, the encrypted encryption keys and security level references $\Lambda_u^{'} \leftarrow enc(EK_u, \Lambda_u)$ and a reference to the user u that can decrypt the authorisation mapping to the set of encrypted authorisation maps $AM^{'}$, i.e: $AM^{'} \leftarrow AM^{'} \cup \{(EK_u^{'}, \Lambda_u^{'}, u)\}$.

Finally, the confidential information l_i in each security level i is encrypted using the respective symmetric encryption key K_i, to protect the sensitive information from disclosure by unauthorised parties. The encrypted authorisation mapping $AM^{'}$, the ephemeral key(s) $EK_u^{'}$ and the vector of encrypted security levels $\underline{L}^{'} = \{l_1^{'}, ..., l_d^{'}\}$ are finally enveloped in an IDMEF AdditionalData element of $message^{'}$. The anonymised anonymised IDMEF XML message ($message^{'}$) is then sent to the receiver. It must be noted that the proposed solution is not restricted to IDMEF based IDS alarms. It works for any XML schema that supports an extension mechanism where the encrypted data and XML signatures can be inserted.

On the receiving side, the deanonymiser goes through the $(\Lambda_u^{'}, u)$ tuples in $AM^{'}$ to search for a user u that matches the current $user$ as shown in Fig. 7. If this is found and decryption of the encrypted ephemeral key $EK_u^{'}$ using the secret key SK_{user} succeeds, then the deanonymiser will decrypt Λ_u using EK_u, and goes through all key maps $(K_i, i) \in \Lambda_u$ and decrypt the confidential information at the given security level: $l_i \leftarrow Dec(K_i, l_i^{'})$.

The deanonymiser can then loop through all tuples $(r_p, Q_i) \in l_i$, use XPath searches with the expression r_p on $message^{'}$ to retrieve the elements that need to be deanonymised, i.e. $\underline{e_p} \leftarrow XPATH(r_p, message^{'})$ and finally loop through all elements q of $\underline{e_p}$ to replace the original content using $e_{p,q} \leftarrow Q_{i,q}$ if $Q_{i,q} \neq \emptyset$, which reverses the anonymisation.

The XACML policy allows the encryption keys to be regenerated at regular time intervals, to reduce the risk of key recovery attacks and also to reduce the amount of confidential information that can be accessed with a given encryption key.

The confidential information is stored in random order, using random identifiers to avoid revealing explicitly which security level (or grading) the confidential information has. Each security level in addition contains a nonce (not shown in the figures), to make it harder to correlate sensitive information between IDS alarms. This nonce can also be used by the deanonymiser to detect and avoid data replay attacks.

```
function DEANON($user, SK_{user}, msg^{'}, AM^{'}, \underline{L}^{'}$)
    for $(EK_u^{'}, \Lambda_u^{'}, u) \in AM^{'}$ do
        if $u = user$ and $EK_u \leftarrow Dec(SK_{user}, EK_u^{'})$
        then
            if $\Lambda_u \leftarrow Dec(EK_u, \Lambda_u^{'})$ then
                for $(K_i, i) \in \Lambda_u$ do
                    $l_i \leftarrow Dec(K_i, l_i^{'})$
                    for $(r_p, Q_i) \in l_i$ do
                        $\underline{e_p} \leftarrow XPath(r_p, msg^{'})$
                        for $q \leftarrow 1, ..., z$ do
                            if $Q_{i,q} \neq \emptyset$ then
                                Restore content:
                                $e_{p,q} \leftarrow Q_{i,q}$
    return $msg^{'}$
end function
```

Figure 7: Anonymisation reversal.

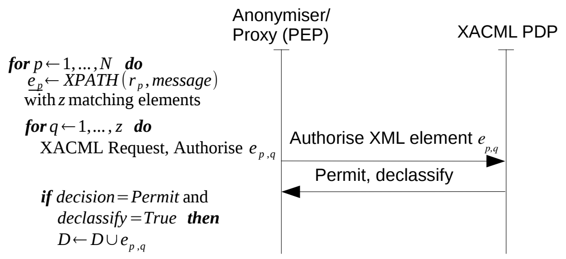


Figure 8: Element declassify operation for default DENY scheme.

3. Supporting Default DENY Protocol

The reversible anonymisation protocol described so far is a default PERMIT protocol. This means that any information in the IDS alarms, which is not explicitly being authorised by the cache specification, by default is being permitted. This strategy has the deficiency that parameters which are unknown by the policy will not be anonymised. This can be problematic from a privacy perspective.

A better strategy is then to support *privacy by default* [2], by introducing a default DENY protocol. This is also similar to common practices in computer security for firewall design, which typically use a default DENY scheme. The remainder of this section outlines how the building blocks for multi-level privacy/security can be used to implement a default DENY reversible anonymisation protocol for the anonymiser.

A default DENY scheme can be implemented by minor modifications of the proposed scheme. The initial authorisation need to specify whether a default PERMIT or default DENY protocol is being used. Subsequent authorisation of other parties after the initial authorisation is done in the same way as for the default PERMIT scheme shown earlier.

Authorisation of individual elements can then be performed in two ways, depending on the outcome of the XACML element authorisation:

1. If the outcome of the XACML Response is PERMIT with an Obligation to *anonymise* information, then the *levels* specify that the information for this docu-

```
Initial XACML authorisation:
1: $R = \{r_1, r_2, ..., r_N\}$
2: $KM = \{(K_i, i) | K_i \leftarrow rnd(), i \in 1, ..., d\}$
3: $AM = \{(PK_u, EK_u, \Lambda_u, u) | \Lambda_u \subseteq KM, EK_u \leftarrow rnd(), u \in 1, ..., m\}$

4: function ANONYMISE($message, defaultpolicy, defaultlevel$)
5:     $\underline{L} = (l_1 = \emptyset, ..., l_d = \emptyset)$    ▷ Confidential information per security level.
6:     $\underline{L}^{'} = (l_1^{'} = \emptyset, ..., l_d^{'} = \emptyset)$    ▷ Encrypted confidential information.
7:     $C = \emptyset$    ▷ Set of classified elements to be anonymised.
8:     $D = \emptyset$    ▷ Set of elements to declassify.
9:     for $p \leftarrow 1, ..., N$ do
10:        $\underline{e_p} \leftarrow XPath(r_p, message)$    ▷ $\underline{e_p} = \{e_{p,1}, ..., e_{p,z}\}$ XPath matches
11:        Evaluate XPATH expressions to select scope parameters for $e_p$
12:        Create a $d \times z$ matrix $Q$, initialised to $\emptyset$ storing authorised elements.
13:        for $q \leftarrow 1, ..., z$ do
14:            Authorise element $e_{p,q}$ (XACML request)
15:            $(decision, anonymise, declassify, levels) \leftarrow XACMLReq(scope(e_{p,q}))$
16:            if $decision = "Permit"$ then
17:                if $anonymise = True$ then
18:                    for $i \in levels$ do    ▷ $e_{p,q}$ may be authorised for $\geq 0$ levels.
19:                        $Q_{i,q} \leftarrow valueof(e_{p,q})$    ▷ Copy value of element.
20:                    $C \leftarrow C \cup e_{p,q}$    ▷ Anonymise element
21:                else if $declassify = True$ then
22:                    $D \leftarrow D \cup e_{p,q}$    ▷ Declassify element
23:                else
24:                    $C \leftarrow C \cup e_{p,q}$    ▷ Anonymise element
25:            else
26:                $C \leftarrow C \cup e_{p,q}$    ▷ Anonymise element
27:        for $i \in 1, ..., d$ do
28:            $l_i \leftarrow l_i \cup \{(r_p, Q_i)\}$
29:    if defaultpolicy=PERMIT then
30:        for $element \in C \setminus D$ do
31:            Anonymise $element$
32:    else if defaultpolicy=DENY then
33:        Fetch all elements containing text and all attributes.
34:        $R_{defaultlevel} = \{"//*[name()]/*[normalize\text{-}space(text())]", "//@*"\}$
35:        for $r_{defaultlevel} \in R_{defaultlevel}$ do
36:            $allElements \leftarrow XPath(r_{defaultlevel}, message)$
37:            $Q_{defaultlevel} = []$
38:            for $element \in allElements$ do
39:                $Q_{defaultlevel}.append(valueof(element))$
40:                if $element \notin D$ or $element \in C$ then
41:                    Anonymise $element$
42:            $l_{defaultlevel} \leftarrow l_{defaultlevel} \cup \{(r_{defaultlevel}, Q_{defaultlevel})\}$
43:    $AM^{'} \leftarrow \emptyset$
44:    for $(PK_u, \Lambda_u) \in AM$ do
45:        $AM^{'} \leftarrow AM^{'} \cup \{(Enc(PK_u, EK_u), Enc(EK_u, \Lambda_u), u)\}$    ▷ Encrypt AM
46:    for $i \in 1, ..., d$ do
47:        $l_i^{'} \leftarrow Enc(K_i, l_i)$    ▷ Encrypt each security level using encryption key
48:    return $(AM^{'}, \underline{L}^{'})$
49: end function
```

Figure 6: Reversible anonymisation.

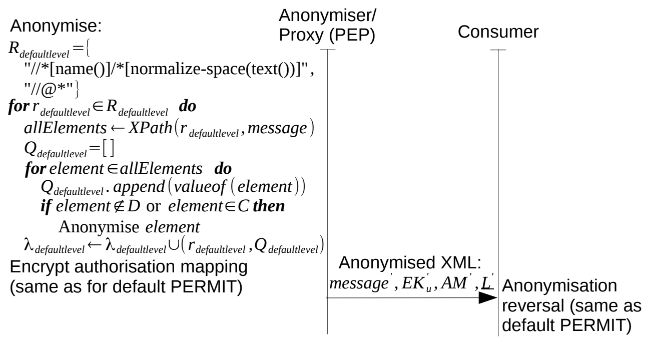

Anonymise:
$R_{defaultlevel} = \{$
"//*[name()]/*[normalize-space(text())]",
"//@*"$\}$
for $r_{defaultlevel} \in R_{defaultlevel}$ **do**
$allElements \leftarrow XPath(r_{defaultlevel}, message)$
$Q_{defaultlevel} = [\,]$
for $element \in allElements$ **do**
$Q_{defaultlevel}.append(valueof(element))$
if $element \notin D$ or $element \in C$ **then**
Anonymise *element*
$\lambda_{defaultlevel} \leftarrow \lambda_{defaultlevel} \cup (r_{defaultlevel}, Q_{defaultlevel})$
Encrypt authorisation mapping (same as for default PERMIT)
Anonymiser/ Proxy (PEP)
Consumer
Anonymised XML:
$message', EK'_u, AM', \underline{L}'$
Anonymisation reversal (same as default PERMIT)

Figure 9: Default DENY scheme.

ment element should remain anonymised, and moved from the default security level and to the security levels specified in *levels*. This is the same operation as shown in Fig. 4.

2. If the outcome is PERMIT with an Obligation to *declassify* information, then this means that the given element should be declassified, i.e. it should not be anonymised in the original IDMEF message. This operation is shown in Fig. 8.
3. If the outcome is DENY, or PERMIT with unknown or undefined parameters and the default policy is DENY, then nothing should be done, since the default DENY policy protects the information of the XML message.

This approach allows the default DENY scheme to implement policies supporting anonymisation of information by moving certain information to a different security level. The scheme also supports declassification of information that should remain visible in the XML message. Elements that are not explicitly authorised remain in the default security level for the DENY policy.

Declassification of information is controlled by the XACML policy. This means that resource elements are authorised as normal, however the authorisation decision for XACML elements that are declassified contains an obligation to *declassify* the given information instead of *anonymising* it.

The anonymisation enforcement part of the default DENY scheme is described in Fig. 9. All information which needs to be anonymised by default, denoted by $R_{defaultlevel}$, can be identified using two XPath resource expressions. The first expression $//*[name()]/*[normalize\text{-}space(text())]$ selects the text attribute of all XML elements trimmed for whitespace, and the second expression $//@*$ selects the value of all XML attributes in the XML document being anonymised[4].

[4]These two expressions have not been combined to one using the XPath or ("|") operator, to ensure that the sequence of matches is well defined.

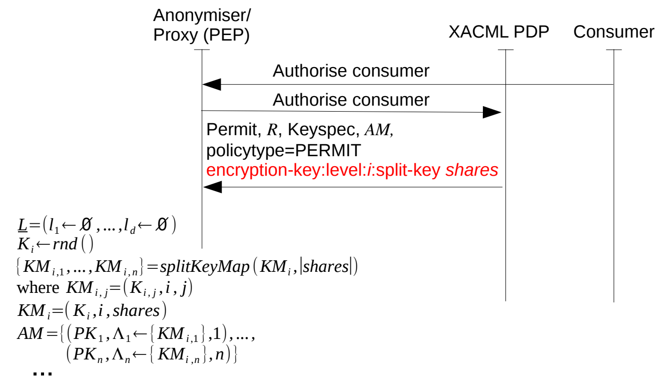

Anonymiser/ Proxy (PEP)
XACML PDP
Consumer
Authorise consumer
Authorise consumer
Permit, R, Keyspec, AM, policytype=PERMIT
encryption-key:level:*i*:split-key *shares*
$\underline{L} = (l_1 \leftarrow \emptyset, \ldots, l_d \leftarrow \emptyset)$
$K_i \leftarrow rnd()$
$\{KM_{i,1}, \ldots, KM_{i,n}\} = splitKeyMap(KM_i, |shares|)$
where $KM_{i,j} = (K_{i,j}, i, j)$
$KM_i = (K_i, i, shares)$
$AM = \{(PK_1, \Lambda_1 \leftarrow \{KM_{i,1}\}, 1), \ldots, (PK_n, \Lambda_n \leftarrow \{KM_{i,n}\}, n)\}$
...

Figure 10: Key sharing scheme.

The algorithm then iterates through these resources and selects the matching elements using XPath. Then the enforcement part loops through all matching elements for each resource in $R_{defaultlevel}$ and adds the value of the elements to the list of elements in the default security level $Q_{defaultlevel}$. In addition, elements that either are explicitly classified or elements that are not declassified are anonymised. Subsequently the data required to reverse the default security level is stored in $l_{defaultlevel}$, by executing $l_{defaultlevel} \leftarrow l_{defaultlevel} \cup (r_{defaultlevel}, Q_{defaultlevel})$.

The remainder of the default DENY anonymisation scheme, including anonymisation reversal, is equivalent to the default PERMIT scheme. The complete anonymisation algorithm that combines the default PERMIT and default DENY schemes is shown in Fig. 6.

4. Adaptations Required to Support Key Sharing

Key sharing is implemented based on a threshold encryption scheme. The easiest key sharing scheme to adapt, is the scheme of Karnin, Greene and Hellman [3], assuming that all n shares must be known to reveal the secret (i.e. $t = n$), and assuming that the PEP acts as a trusted dealer.

Assume that the secret key space is all numbers from 0 to $2^{keysize}$ where $keysize$ is the size of the encryption key in bits. Key sharing can then be implemented by letting the anonymiser choose $n - 1$ random shares of the same size as the original encryption key, and calculate the last share as the chosen encryption key minus the sum of the chosen random shares modulo $2^{keysize}$. The encryption key can then be reconstructed by adding up all the shares modulo $2^{keysize}$.

To support secret key sharing, the encryption key definition in the XACML policy needs a *split-key* operator and the implementation must be extended to support addressing of key shares. A key sharing scheme can then be set up by authorising stakeholders to encryption key shares instead of encryption keys, as illustrated in Fig. 10.

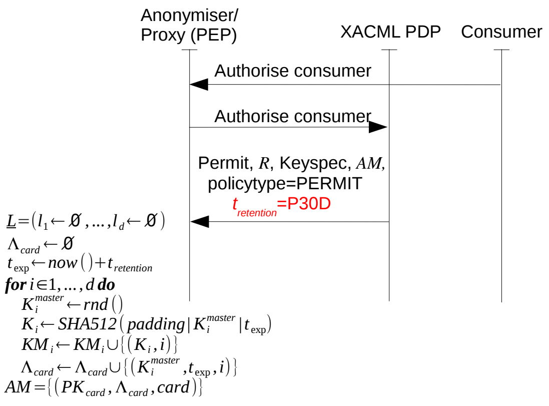


Figure 11: Time-based data expiry initialisation.

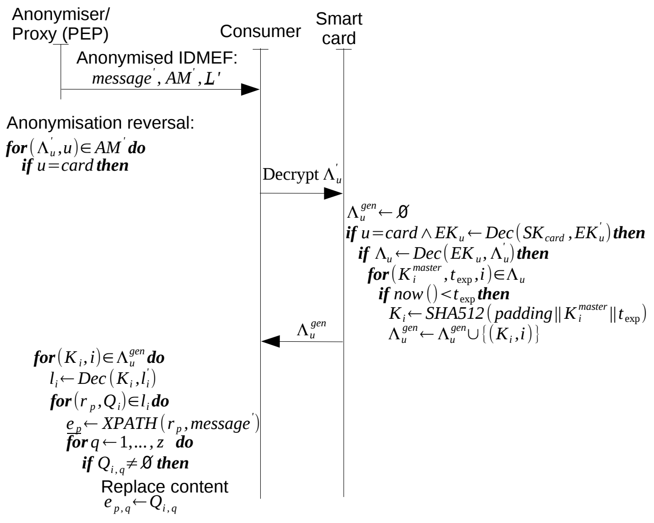


Figure 12: Decryption for time-based data expiry.

On receiving the initial XACML authorisation response, encryption keys K_i are generated as normal, and these are used to encrypt the sensitive information in $(l_1, ..., l_d)$. The XACML Response also contains an Obligation to split the encryption key K_i into $n = |shares|$ subkeymaps $\{KM_{i,1}, ..., KM_{i,n}\}$. Each subkeymap $KM_{i,j}$ consists of a tuple $KM_{i,j} = (K_{i,j}, i, j)$, where $K_{i,j}$ is share number j of the encryption key for security level i and n is the number of key shares. The key map KM_i is also extended to contain a reference to all *shares*, i.e: $KM_i = (K_i, i, shares)$. The authorisation mapping AM will then contain one key mapping share for each public key, i.e:

$$\begin{aligned} AM &= \{(PK_1, \Lambda_1 \leftarrow \{KM_{i,1}\}, 1), ..., \\ &\quad (PK_n, \Lambda_n \leftarrow \{KM_{i,n}\}, n)\} \end{aligned} \tag{2}$$

so that the owners of $SK_1, ..., SK_n$ need to collaborate to reveal the confidential information. Note that if the parent encryption key times out and is regenerated, then the shares must also be updated.

5. Adaptations Required for Time-based Data Expiry

The fifth Privacy by Design principle requires end-to-end security with full lifecycle management of private or confidential data from inception and until destruction [2]. This can be implemented by introducing time-based data expiry, assuming that the messages can be protected using encryption until they reach the anonymiser. Time-based data expiry means that the encryption key expires after a given retention time, so that confidential information in the XML messages can not be accessed beyond this time. This ensures safe destruction of confidential data in the messages. Time-based data expiry can also be used to limit how long users will have access to the data they have analysed, for example to set up policies to avoid access to confidential data beyond the current work shift.

5.1. Implementing Time-based Data Expiry

The reversible anonymiser can with small adaptations support a time-based data expiry scheme similar to [12]. This scheme uses Smartcards to enforce the key expiration scheme (one card per user). Only the necessary adaptations will be discussed here.

The encryption key management of the reversible anonymiser needs to be adapted as shown in Fig. 11 to support the key derivation scheme in [12]. To support time-based data expiry, the XACML policy must return an obligation with the retention time $t_{retention}$ for element authorisation requests, so that the data retention time can be configured per encryption key. The retention time is then sent to the PEP in the initial XACML Response as part of the XACML Obligation. The key expiry time t_{exp} is then calculated as the current time plus $t_{retention}$.

To achieve time-based data expiry, the key expiry time t_{exp} must be cryptographically bound to the encryption key K_i. Assume that *padding* is a 64 bit number, that is initialised to zero. The time expiring encryption key K_i, that encrypts the classified information l_i, is derived from a master key K_i^{master} by using the key derivation function: $K_i = SHA512(padding||K_i^{master}||t_{exp})$. This encryption key is then used to encrypt the confidential information in l_i. The authentication map is the same as the basic scheme uses, i.e: $AM = \{(PK_{card}, EK_{card}, \Lambda_{card}, card)\}$. Λ_{card} is however modified to contain the master key K_i^{master}, the expiry time t_{exp} and the security level i. i.e: $\Lambda_{card} = \{(K_i^{master}, t_{exp}, i)|i \in 1, ..., d\}$.

The decryption algorithm must be modified to ask the Smartcard to decrypt Λ'_u as shown in Fig. 12. The figure only shows Smartcard based key retrieval. Smartcard initialisation and authorisation will be similar to [12]. The Smartcard will first initialise the generated key mapping $\Lambda_u^{gen} \leftarrow \emptyset$. Subsequently, it verifies that the key belongs to the card and decrypts the encryption key using $EK_u = Dec(SK_{card}, EK_u^{'})$. If this succeeds, then the Smartcard will decrypt Λ_u using $\Lambda_u \leftarrow Dec(EK_u, \Lambda_u^{'})$, and then iterate through all tuples $((K_i^{master}, t_{exp}), i)$ in Λ_u and verify that the current time is less than t_{exp}. If this test succeeds, then the Smartcard will generate the decryption key for security level i by using the key derivation function $K_i \leftarrow SHA512(padding|K_i^{master}|t_{exp})$ and generate a new keymap $\Lambda_u^{gen} = \Lambda_u^{gen} \cup \{(k_i, i)\}$. The Smartcard then returns Λ_u^{gen} to the Anonymisation reversal function, which reverses the anonymisation for the given security level i in the same way as shown previously.

Note that this scheme assumes trusted time sources, which can be implemented in a similar way as proposed in [12].

6. Experiments

The anonymiser and deanonymiser was tested on a server with 8 Gb RAM and a 3.3 GHz Intel Core i5 CPU. The anonymiser was connected to the deanonymiser using a SOAP web service with persistent HTTP connections, to verify the entire production pipeline. The performance can be expected to be somewhat higher if the anonymiser and deanonymiser are run separately. The experiments used the IDMEF alarm log from previous IDS experiments using PreludeIDS. These experiments are based on alarms from the 1999 KDD Cup data set (DARPA IDS test set)[5]. This is an old synthetic data set, and will therefore have less diversity that one can expect from real traffic today. The cache hit rate and performance is therefore higher than what one can expect from a production system. The KDD Cup data set was chosen despite these deficiencies, since this still is considered the gold standard for IDS measurements, and it is difficult to get access to real IDS data.

The software is implemented in Jython and based on the XACML Decision-cache based anonymiser in [23], which uses SunXACML with a Java HashMap based Least Recently Used cache and virtual token descriptor based XML parser VTD-XML for increased XPath performance. The solution in this paper has been extended with the GeoXACML patches [1], to support more advanced XACML data types like pointlists which are used by the implementation. This also allows for supporting location-aware anonymisation and authorisation policies. Apache Santuario is used for XML encryption, Apache CXF as SOAP server for the deanonymiser and SUDS as SOAP client for the anonymiser. The anonymiser uses three threads - one for reading and buffering IDS alarms, one for anonymising the alarms and an output thread for buffering and sending data to the deanonymiser. This strategy decouples the anonymiser from the deanonymiser to avoid any of the threads blocking the production pipeline.

[5]KDD Cup 1999 data (DARPA IDS test set) http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Each statistical value is calculated as the average of an ensemble of 100 IDS alarms. Each experiment furthermore selects a random uniform sample of 1000 IDS alarms from a corpus of 130.000 IDS alarms from the KDD Cup'99 test set. A maximum limit of 30 anonymisation rules, 10 security levels and 10 key shares was chosen, since this is expected to be around the maximum numbers needed for IDMEF anonymisation policies. A new XACML policy with a random selection of the current number of anonymisation rules was generated for each experiment in the ensemble. It uses an XACML policy generator to generate random anonymisation policies with between 0 and 30 anonymised resources, each resource consisting of two different policies matching the relevant Target section and containing a Condition section that matches one of two different policies per anonymised resource. The policy generator furthermore supports a configurable number of users, security levels and key shares.

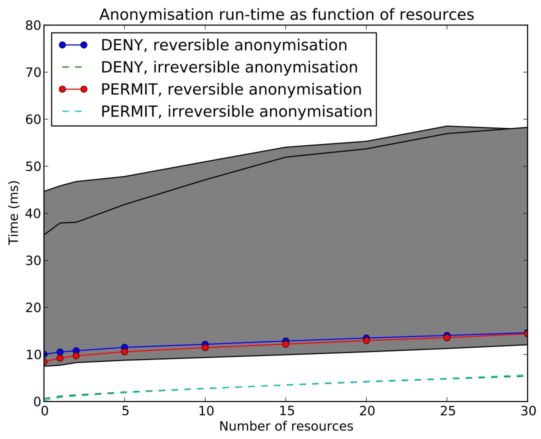
6.1. Anonymiser Performance

Anonymiser run time as a function of anonymised resources, default protocol type, number of security levels and number of key shares is shown in Fig. 13. Each subfigure shows the median and the 95% confidence band (0.025-0.975 percentile) for reversible anonymisation, as well as the median for irreversible anonymisation indicated using stapled lines.

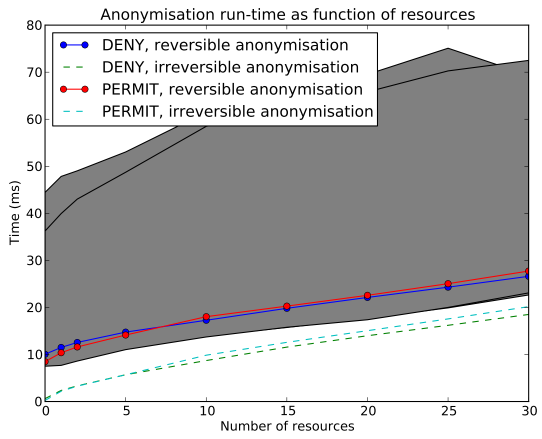
The Figures show that the distribution functions are significantly skewed towards lower run-times both for non-cached and cached results, which means that the median gives a more representative picture of the mode of the distribution than the standard deviation.

Figures 13a and 13b show that the anonymiser run-time for the default PERMIT scheme is nearly the same as the default DENY scheme as a function of number of resources in the interval between 0 and 30 resources. These experiments use one security level and no key shares. There is perhaps a trend that the default DENY scheme starts with somewhat higher run-time and scales slightly better than the default PERMIT scheme. The figures furthermore show that the decision cache reduces the average run-time from 33 ms to 18.5 ms (median from 27 to 14ms) for 30 anonymised or declassified resources when reversible anonymisation is used. This means that the reversible anonymiser performance is increased from approximately 30 to 54 IDS alarms/s for 30 anonymised resources by using decision caching.

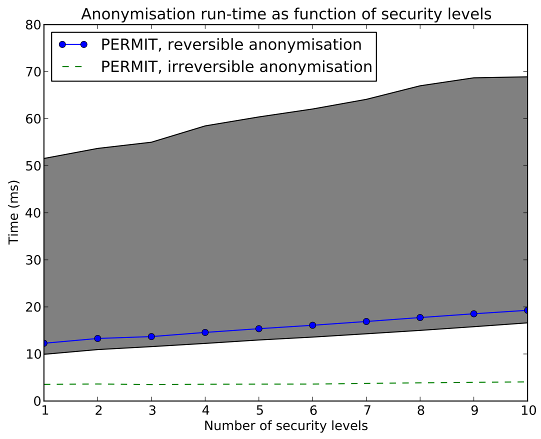
The cache hit rate for the experiments is 98%, which is higher than one can expect in a production system due to lack of entropy in the KDD Cup data set. The figures furthermore show that reversible anonymisation based on XML Encryption and XML Signatures adds an average cryptographic overhead of 11 ms (median 8ms) compared to using irreversible anonymisation for the given experiments. Irreversible anonymisation with decision caching would give a performance of around 130 anonymised IDS alarms/s on the given hardware.



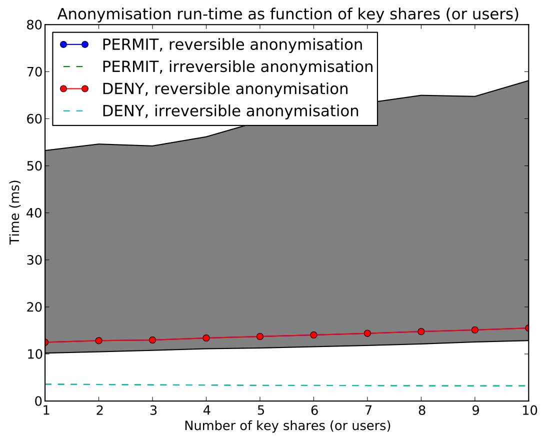
a: Run-time as function of resources with decision cache for one user and one security level.



b: Run-time as function of resources without decision cache for one user and one security level.



c: Default PERMIT run-time for 15 resources, one user per security level and no key shares as function of security levels with decision cache.



d: Default PERMIT run-time for 15 resources and one security level as function of key shares or users with decision cache.

Figure 13: Anonymiser run-time as a function of number of anonymised resources, default protocol type, number of security levels and number of key shares. The subfigures show the median and the 95% confidence band (in gray) of the measurements.

Fig. 13c shows the anonymiser run-time as a function of number of security levels for 15 resources, one user per security level and no key shares[6]. It shows that there is a linear dependency between run-time and number of security levels, where the run-time increases with 0.7ms per additional security level.

Fig. 13d shows the anonymiser run-time as a function of number of key shares or users for 15 resources and one security level. The logic for mapping access from a set users to a set of security levels or key shares is essentially the same for the anonymiser, hence using only one figure to show the performance as a function of of either users or shares. The Anonymiser run-time increases linearly and has little influence from number of shares, increasing only with 0.3ms per additional user or key share.

6.2. Deanonymiser Performance

Fig. 14a shows that the run time of the deanonymiser for the default PERMIT scheme increases linearly (0.27 ms/resource) after an initial transient part for the first 0-2 deanonymised resources, and is approximately 3 ms faster than the anonymiser for anonymisation of 1-30 elements. The average value is 14 ms/alarm (median 13 ms/alarm) for 30 resources, which means that the deanonymiser manages to deanonymise 71 IDS alarms/s for 30 resources with the default PERMIT scheme in the experiments.

Fig. 14b shows that the deanonymiser run time of the default DENY scheme scales much better with number of deanonymised resources than the default PERMIT scheme. It has an average run-time of 8.8 ms (median 7.5 ms) and decreases slightly (by -0.02 ms/resource) with number of deanonymised resources. The reason for this is that all the work on deanonymising resources for the default DENY scheme is done in the anonymiser. Only one XPATH search is required to replace the content in the default security level, and less effort is required with more declassified resources, since these resources are not copied back from the default level. The default DENY scheme is in other words very efficient for the deanonymiser. The deanonymiser manages to deanonymise up to 113 alarms/s for the default DENY scheme, which can be an advantage, for example if the deanonymiser is used as part of an alarm correlation system.

Fig. 14c shows the deanonymiser run-time as function of number of security levels, assuming that only one of the security levels need to be deanonymised. The run-time only increases slightly (0.1 ms/security level) with increasing number of security levels.

Fig. 14d shows the deanonymiser run-time as a function of key shares. Adding key shares is relatively expensive, and adds 1.8 ms run-time per added share. The reason for this, is the relatively expensive RSA and ephemeral key decryptions that must be performed for each share.

[6]Note that this experiment cannot be performed for default DENY, since it by default uses one security level.

The experiments indicate that both the anonymiser and deanonymiser should have sufficient performance to be usable at least for small to medium-scale deployments of privacy-enhanced IDS. The performance should also be sufficient for several other applications where the anonymiser and deanonymiser is used as part of a service oriented architecture, and where security or privacy is prioritised above performance. It can furthermore be noted that the XACML PDP, anonymiser and deanonymiser are parallelisable on an XML message level, meaning that the capacity can be scaled up by adding more hardware, if required.

6.3. Bandwidth efficiency of the proposed solution.

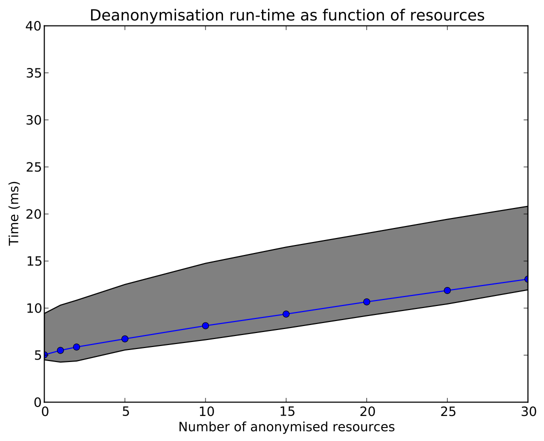
The original IDMEF message size is on average 3.8 kB. Each XML-signature user or key share adds approximately 1 kB of data to the message. There is furthermore a linear dependency where each additional anonymisation rule adds approximately 0.15 kB for the given test data. The anonymised message is 3.1 kB larger than the original message for 0 anonymised elements and 5.7 kB larger for 15 anonymised XML resources in the experiments with two signatures, one user, one security level and no key shares. For 15 resources and 10 users, each accessing an individual security level, the bandwidth usage increases by a factor of 11 to 41.6 kB per IDS alarm. This means that bandwidth usage will probably limit how complex anonymisation policies it is practical to implement with the proposed scheme. It is in particular limited how many security levels, users and key shares it is possible to implement without having too large bandwidth and performance overhead.

This means that it may not be desirable to operate with one key mapping per authorised user, since this solution scales poorly with number of authorised users. One way to mitigate this problem, at the expense of relying more on trust in the XACML authorisation, is to use a role-based authorisation scheme where roles are authorised using public keys for certain use scenarios instead of individual users. Such a scheme could for example be based on the Smartcard based encryption scheme proposed in Section 5, to securely deploy the secret role keys. It would in this case be natural to use the XACML Role-Based Access Control (RBAC) profile[7] for deploying role keys. Such a solution can be integrated with the proposed solution in a similar way as discussed in [24]. The details of this is however left as future work.

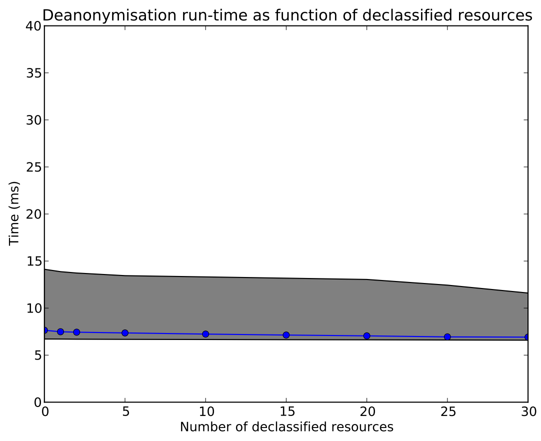
7. Discussion

Our approach has the advantage compared to existing schemes that pseudonymisation is not used, which eliminates the risk of traffic analysis attacks and known plaintext attacks on the pseudonyms. It is also an advantage that it is implemented as a proxy which allows for

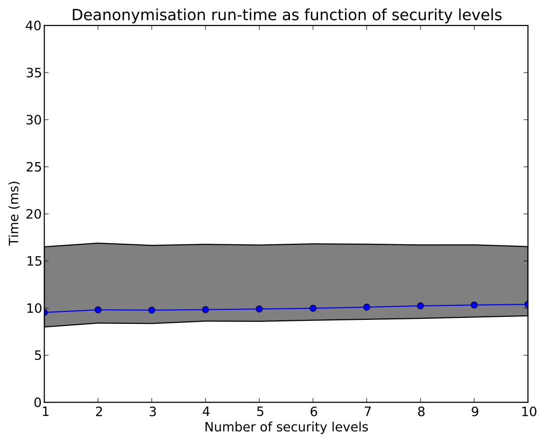
[7]XACML RBAC profile: docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cd-03-en.html



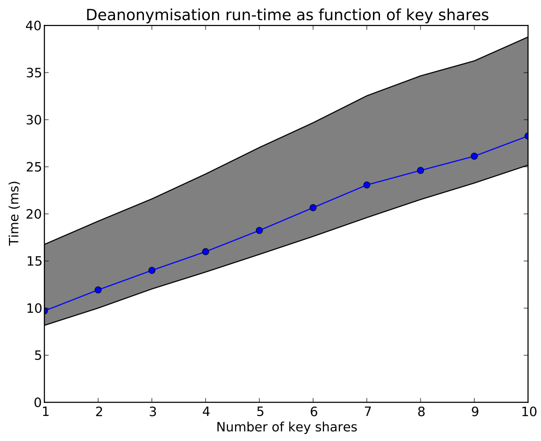
a: Default PERMIT run-time for one user, one security level and no key shares as function of anonymised resources.



b: Default DENY run-time for one user, one security level and no key shares as function of declassified resources.



c: Default PERMIT run-time for 15 anonymised resources and no key sharing as function of security levels where one level is deanonymised by one user.



d: Default PERMIT run-time for 15 anonymised resources and one security level as function of key shares (with same amount of users).

Figure 14: Deanonymiser run-time as a function of number of anonymised resources, default protocol type, number of security levels and number of key shares. The subfigures show the median and the 95% confidence band of the measurements.

anonymising any XML protocol that can be sent via the proxy service. It is able to deal with IDS technologies that use IDMEF, which is a standardised XML-based alarm format. This allows anonymised IDS alarms to be compatible with existing Security Information and Event Management systems that support IDMEF [2]. It must however be noted that anonymisation reversal may not be possible without altering the SIEM database, since this requires preserving the structure of the XML.

It is important to mention that successful reversal of the anonymisation requires that the anonymisation function does not semantically change the structure of the XML document, since this causes the XPath expression for reversing the anonymisation to fail. One example of this, is if one or more text nodes are removed, since this alters the DOM tree. Furthermore, if XML data is used to represent the anonymised data, then these data must be quoted.

A more subtle limitation is that the default DENY protocol does not support replacing anonymised data by whitespace, since this causes the function *normalize-space()* to ignore the anonymised nodes. The latter problem can be worked around by defining an XPath extension function that recursively iterates through the DOM tree and identifies all text nodes enclosed by an element node. This has been implemented for the VTD-XML based parser. Note however that the problem persists if *normalize-space()* is used in other resource identifying expressions in the XACML policies. All in all, these are relatively minor limitations that can be detected and mitigated during regression tests of the XACML policies, since the inner XML signature calculated over the original message will fail if the anonymisation reversal is not done correctly.

8. Related Works

There are some examples of prior work that describes reversible anonymisation schemes that are not based on pseudonymisation. A reversible anonymisation scheme for anonymisation of DICOM images using automatically generated policies was proposed in [14]. The policy definition consists of a list of attribute rules that describe how the document shall be anonymised. Anonymised information is stored in a separate difference file, in order to later reverse the anonymisation by merging in this information. The solution in our paper is more general and can anonymise any XML-based format using XACML-based policies, which is a standardised policy language. Our solution is furthermore different by embedding the information required to reverse the anonymisation in the messages, as well as supporting a multi-level security based scheme where different stakeholders can be granted access to information based on need. Our scheme furthermore supports both a default PERMIT, default DENY policy and key sharing, whereas this scheme only supports default PERMIT.

Another paper that suggests a reversible anonymisation scheme for protecting organisational data confidentiality in cloud-based services is [25]. Reversible anonymisation is however not yet implemented in this paper, so the performance measurements only show traditional irreversible anonymisation.

The paper is also related to the field of privacy enhanced intrusion detection systems. Most previously proposed privacy-enhanced IDS schemes use some kind of pseudonymisation scheme, where sensitive information in the IDS alarms is replaced by pseudonyms, to later be able to reverse the pseudonymisation process on a needs basis.

Use of pseudonyms in audit logs was first suggested by Fischer-Hübner [20, 6]. Another early example of a privacy-enhanced IDS that uses pseudonymisation is the Adaptive Intrusion Detection system (AID) [22]. Both schemes use symmetric key encryption as pseudonym mapping, and focus mainly on encrypting subject identifying data. AID in addition contains a higher order IDS (expert system) that correlates the alarms, and discloses the pseudonymised data if suspicious sequences of events are detected. The pseudonyms of these early schemes are susceptible to traffic analysis attacks during the lifetime of the session key used for encrypting the pseudonym mapping.

An example of a privacy enhanced IDS that uses anomaly detection on pseudonymised data is [13]. The pseudonymisation strategy consists of a very simple static mapping between sensitive data and pseudonyms, however this paper also reflects over the need to pseudonymise other fields than directly user/subject identifying fields.

A Kerberos based pseudonymisation scheme is proposed in [18]. The scheme implements a hierarchical IDS solution where pseudonyms only are revealed if the higher-order anomaly-based IDS detects suspicious traffic. This scheme operates with group reference pseudonyms which correspond to UNIX user groups instead of pseudonymising users directly. The scheme uses public key encryption and relies on a trusted third party for initial pseudonym creation. An extended version of the protocol uses Mixes to avoid linkability towards the original data sessions. A similarity with our solution is that both are based on authorisation schemes (Kerberos and XACML), however our scheme is more flexible when it comes to policy-based fine-grained authorisation and anonymisation of XML data. A pseudonymisation scheme based on homomorphic encryption is proposed in [16]. This allows for performing certain equality or inequality tests on encrypted information without revealing the underlying information.

A privacy enhanced intrusion detection scheme for UNIX audit records is proposed in [7, 10]. This scheme is based on Shamir's threshold cryptography, and the general idea is that pseudonymised information shall only be disclosed when an attack scenario has been identified. An attack scenario here means that a sufficient amount of shares have been recovered from IDS alarms to recover the secret key used by the pseudonymiser. This approach proposes to use transaction pseudonyms to avoid linkability between pseudonyms, however the proposed implementation has some weaknesses that cause the authors to reintroduce linkability between transactions. This scenario has the same weakness as the other pseudonymisers, since it may be vulnerable to traffic analysis attacks in the intervals between rekeying of the pseudonymiser. Some of the ideas in this scheme have been extended to support multilaterally secure ubiquitous auditing in [27, 26]. It combines transaction pseudonyms based on threshold cryptography with secure multiparty computations to support secure and privacy enhanced tracking of mobile rescue units. The solution also supports verifiability via log attestation. This solution implements transaction pseudonyms in a semantically secure way, to mitigate the risk of traffic analysis attacks against the pseudonyms.

A secure logging scheme for retained data of an anonymity service (AN.ON) is described in [12]. This solution is based on smart cards in order to provide time restricted access to system logs from the anonymity service according to the requirements in the Data Retention Directive [4]. This scheme uses a similar hybrid encryption scheme to ours in that symmetric encryption is used for the log entries and asymmetric encryption is used for access to the keys. However our scheme is different by supporting reversible anonymisation with several security levels and not only encryption. This scheme is therefore complementary to the scheme proposed here.

Our solution is also somewhat related to anonymisation of network logs. A NetFlow anonymiser which supports multiple anonymisation strategies is proposed in [21]. The intrusion detection system BRO has support for anonymisation of packet traces [15]. However neither of these solutions support reversible anonymisation of XML messages.

9. Conclusions

This paper proposes a reversible anonymisation scheme for protecting sensitive information in XML messages. The scheme has been applied to IDMEF-based intrusion detection system alarms, and we expect the reversible anonymisation protocol to be useful for policy based confidentiality and integrity protection of sensitive information for a range of services in a service oriented architecture.

The solution is based on existing standards like XML, IDMEF, XACML, XML-Encryption and XML-Signature, and uses a proxy-based reversible anonymiser based on an earlier proposed XACML decision cache for authorisation and anonymisation of XML documents [23]. The solution furthermore supports location-based anonymisation policies via the GeoXACML framework [1].

Using XACML gives flexibility when it comes to defining privacy or security policies for controlling access to sensitive information. It also solves deployment of encryption keys in an efficient way as part of the privacy policy. The scheme allows for defining parties that by default are authorised for access to sensitive information, but it can also support on-demand time-restricted access to sensitive data for authorised users.

A secret sharing scheme is supported, to enforce separation of duties constraints where more than one stakeholder need to agree to reveal the sensitive data. The scheme allows for policy-based control of rekeying intervals, data authorisation and anonymisation schemes. Furthermore, time-based data expiry is outlined, based on the scheme in [12], to support secure deletion of sensitive data after a configurable retention time.

This approach provides a method to improve the privacy of certain types of big data implementations for problems that scale horizontally, assuming that a large number of smaller individual data sources can be anonymised before they are aggregated and stored in a big XML database.

The proposed approach has been integrated into the existing Security Information and Event Management systems (SIEM) PreludeIDS[8], which supports IDMEF. Anonymised IDS alarms can be stored in the SIEM database using the proposed approach without any modifications, since the alarms follow the standard IDMEF extension schema. However implementing support for anonymisation reversal may require some modifications of the SIEM tools, since the structure of the IDMEF XML needs to be maintained unmodified for successful anonymisation reversal. One way to mitigate this limitation, is to store the anonymised data in an XML database.

The performance of the proposed approach has been tested and should be sufficient for small to medium scale IDS deployments. However, larger data rates can be managed by running several anonymisers or deanonymisers in parallel. A useful feature for alarm correlation systems is that the deanonymiser is fast for default DENY policies, which allows for correlating alarms between several privacy-enhanced IDS sensors in business cases where this is acceptable from a privacy and confidentiality perspective.

10. Future Work

Implementing and testing time-based data expiry using the Smartcard-based solution is left for future work. More research is also needed on how to protect the XACML policies themselves, for example using XML encryption as proposed in [9]. Implementing support for role-based instead of user-based authorisation is also left as future work. An interesting idea is to extend the multi-level security based scheme proposed here to also cover XACML policies and policy handling. Details of logging procedures to ensure transparency of the operation is also left as future work. This can for example be implemented in a similar way as the AN.ON secure logging service [12]. It is also envisaged that the proposed scheme in the future can be extended to

[8]PreludeIDS: http://www.prelude-ids.org

support operations on encrypted data, for example by using homomorphic encryption of the sensitive data elements as pseudonyms for the anonymised data. This could make the reversible anonymisation scheme more useful for XML databases, since it would allow defining certain standardised query operators (e.g. equality tests) on encrypted data, in a similar way as CryptDB does for relational databases [17]. Both anonymisation and deanonymisation are horizontally scalable, which make them suitable for performing data analysis and deanonymisation using tools like Apache Hadoop based clusters.

Acknowledgements

This work has been partially supported by the project "PRECYSE - Protection, prevention and reaction to cyber-attacks to critical infrastructures", funded by the European Commission under the FP7 frame programme with contract number FP7-SEC-2012-1-285181 (www.precyse.eu), and partially by Telenor Research & Innovation under the contract DR-2009-1.

[1] Andreas Matheus (ed). OGC 07-026r2 Geospatial eXtensible Access Control Markup Language (GeoXACML) version 1.0. `http://portal.opengeospatial.org/files/?artifact_id=25218`, 2007.

[2] A. Cavoukian. Privacy by design. `http://privacybydesign.ca/about/principles/`, 2009.

[3] J. Greene E. Karnin and M. Hellman. On secret sharing system. *IEEE Trans. on Info. Theory*, IT-29:35–41, January 1983.

[4] European Commission. Directive 2006/24/EC of the European Parliament and of the Council of 15 march 2006 on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC. `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:105:0054:01:EN:HTML`, 2006.

[5] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, volume 773, page 480–491. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[6] Simone Fischer-Hübner. *IDA - An Intrusion Detection and Avoidance System (in German)*. Aachen, Shaker, 1997.

[7] Ulrich Flegel. *Privacy-Respecting Intrusion Detection*. Springer, 1 edition, October 2007.

[8] B. Feinstein H. Debar, D. Curry. *The Intrusion Detection Message Exchange Format (IDMEF)*. 2007.

[9] G. Hsieh, R. Meeks, and L. Marvel. Supporting secure embedded access control policy with XACML+XML security. In *2010 5th International Conference on Future Information Technology (FutureTech)*, page 1 –6, May 2010.

[10] Yücel Karabulut Joachim Biskup. A hybrid PKI model with an application for secure mediation. *In 16th Annual IFIP WG 11.3 Working Conference on Data and Application Security*, pages 271–282, 2002. Published: Journal.

[11] Neil M. Richards & Jonathan H. King. Three paradoxes of big data. *Stanford Law Review Online*, 66:41, September 2013.

[12] Stefan Köpsell and Petr Švenda. Secure logging of retained data for an anonymity service. In Michele Bezzi, Penny Duquenoy, Simone Fischer-Hübner, Marit Hansen, and Ge Zhang, editors, *Privacy and Identity Management for Life*, volume 320, pages 284–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

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

[14] Michael Onken, Jörg Riesmeier, Marcel Engel, Adem Yabanci, Bernhard Zabel, and Stefan Després. Reversible anonymization of DICOM images using automatically generated policies. *Studies in health technology and informatics*, 150:861–865, 2009. PMID: 19745435.

[15] Ruoming Pang and Vern Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 339–351, Karlsruhe, Germany, 2003. ACM.

[16] Hyun-A Park, Dong Hoon Lee, Jongin Lim, and Sang Hyun Cho. PPIDS: privacy preserving intrusion detection system. In Christopher C. Yang, Daniel Zeng, Michael Chau, Kuiyu Chang, Qing Yang, Xueqi Cheng, Jue Wang, Fei-Yue Wang, and Hsinchun Chen, editors, *Intelligence and Security Informatics*, volume 4430, page 269–274. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[17] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, page 85–100, New York, NY, USA, 2011. ACM.

[18] R. Büschkes, D. Kesdogan. Privacy enhanced intrusion detection. In G. Müller and K. Rannenberg, editors, *Multilateral Security in Communications, Information Security*, pages 187–204. Addison Wesley, 1999.

[19] S. Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 2005.

[20] Klaus Brunnstein Simone Fischer-Hübner. Combining verified and adaptive system components towards more secure computer architectures security and persistence. 1990.

[21] A. Slagell and W. Yurcik. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. In *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on*, page 80–89, 2005.

[22] Michael Sobirey, Birk Richter, and Hartmut König. The intrusion detection system AID - architecture and experiences in automated audit trail analysis. In *Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security*, pages 278–290, 1996.

[23] Nils Ulltveit-Moe and Vladimir Oleshchuk. Decision-cache based XACML authorisation and anonymisation for XML documents. *Comput. Stand. Interfaces*, 34(6):527–534, November 2012.

[24] Nils Ulltveit-Moe and Vladimir Oleshchuk. Mobile security with location-aware role-based access control. In Ramjee Prasad, Károly Farkas, Andreas U. Schmidt, Antonio Lioy, Giovanni Russello, Flaminia L. Luccio, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson, editors, *Security and Privacy in Mobile Information and Communication Systems*, volume 94 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, page 172–183. Springer Berlin Heidelberg, 2012.

[25] Dawid Nowak Vanessa Ayala-Rivera. Protecting organizational data confidentiality in the cloud using a high-performance anonymization engine. 2013.

[26] Stefan Weber and Max Mühlhäuser. Multilaterally secure ubiquitous auditing. In Santi Caballé, Fatos Xhafa, and Ajith Abraham, editors, *Intelligent Networking, Collaborative Systems and Applications*, volume 329 of *Studies in Computational Intelligence*, page 207–233. Springer Berlin / Heidelberg, 2011.

[27] Stefan G. Weber. Harnessing pseudonyms with implicit attributes for privacy-respecting mission log analysis. page 119–126. IEEE, November 2009.