Learning from data retention logging for an anonymity service

Stefan Köpsell[1] and Petr Švenda[2]

[1] TU Dresden, Germany, <*sk13@inf.tu-dresden.de*>
[2] Masaryk University, Czech Republic, <*svenda@fi.muni.cz*>

Abstract. The recently introduced legislation on data retention to aid prosecuting cyber-related crime in Europe also affects the achievable security of systems for anonymous communication on the Internet. We have analyzed the newly arising risks associated with the process of accessing and storage of the retained data and propose a secure logging system, which utilizes cryptographic smart cards, trusted timestamping servers and distributed storage. These key components will allow for controlled access to the stored log data, enforce a limited data retention period, ensure integrity of the logged data, and enable reasonably convenient response to any legitimated request of the retained data. A practical implementation of the proposed scheme was performed for the AN.ON anonymity service, but the scheme can be used for other services affected by data retention legislation.

1 Introduction

The recently introduced legislation on data retention affects—at least in some countries (e.g., Germany)— systems for anonymous communication on the Internet such as AN.ON [BeFK00] or TOR [DiMS04]. These systems alter the source IP-addresses of users and these alterations should be logged and accessible on request from legal authorities (cf., [BeBK08] for a description of the legal obligations and the usefulness of the retained data).

Standard secure logging mechanisms such as [MaTs09] protect the logged records sufficiently against unauthorised access (confidentiality), unauthorised modification (integrity) and in some cases attempt to ensure availability of records. But when applied to the needs of data retention logging on the logging entity side, newly arising risks remain unsolved as the attacker model has changed. Potentially sensitive data are present on logging entity side as a result of compliance with data retention legislation. The logging entity can be forced to reveal, delete or modify this data—threats that did not exist before as there was no need to store such data in the first place. Specifically, threats related to the data retention period must be addressed and mitigated.

Note that the risk for a user to be deanonymised, if the operators of the chosen anonymity servers behave *dishonestly*, exists before the introduction of data retention. But if the operators are *honest*, the attacker gains an additional advantage of mounting a successful attack on the anonymity of a given user

with the help of retained data. Moreover, it is now possible for the attacker to start his attack *after the fact* (i.e. after the activity, the attacker wants to deanonymise took place). This was not possible before, as the attacker had to log the anonymised and encrypted traffic at the time of this activity in order to analyze it later on.

Completely new risks arise from the fact that the logged data is used for law enforcement. One such attack results in the risk of the attacker modifying the logged data so that an innocuous user of the anonymity service becomes suspicious. For an operator of an anonymisation server, the new risk is that an attacker forces him to modify the logged data in such a way (or at least in a way which hides the criminal activities of the attacker). So, our solution should not only protect the users of the anonymity service but also its operators.

A demand for the practical implementation originates from the needs of the AN.ON anonymity service. This anonymity service has been open to public since 2000 and has to fulfil the legal obligations given by the data retention legislation. But the proposed logging scheme can be used for other services affected by the data retention legislation as well. More generally, the scheme can be used for any logging service where the logged records are accessible only for a limited time period or where knowledge of cryptographic secrets might lead to personal threats of the holder.

TBD: to be actualized TBD Our paper is organised as follows: the first section describes the requirements for data retention logging and summarizes related work. The second section describes the logging scheme and analyzes security of the scheme. This section also provides an overview of the steps involved in logging and answering requests. Selected properties of practical implementation and results of the performance analysis are given in section three, followed by conclusions in section four.

1.1 Legal and operational requirements on logging of retained data

In this section, we summarize the requirements for the retained data and the logging procedures. These are general requirements applicable to any service which needs to be compliant with the EC data retention directive. They can be derived from the legal obligations (R1–R4) and the operational needs (R5). Moreover, they can be classified as functional requirements (R1; what the system should *do*) and non-functional requirements (R2–R5; how the system should *be*).

R1: Logged data has to include all statutory categories of data. Article 5 of the data retention directive describes what types of services have to retain which data categories. National implementations of the directive could extend this. This functional requirement basically states that some meaningful data has to be logged and that logging of (e.g.) random data would not be sufficient.

R2: Logged data have to be deleted after a specific period of time. This means that logged records cannot be accessed outside a given data retention

period. In the following text we use the term "outdated" to describe a property of a given item (cryptographic key, log entry etc.) to which the access should be prevented because the related retention period already expired.

R3: Logged data need to be accessible, so that requests from law enforcement agencies can be answered without undue delay.

R4: Logged data have to be secure, so that no access to the logged data by unauthorised person is possible. This requirement covers confidentiality as well as integrity of the logged records. Note that in our case the integrity means that the operator can detect if the logged data have been altered—it is not necessary that the operator proves something to the third party.

R5: The cost of logging has to be reasonable. It includes the monetary costs (e.g. initial necessary investments, operational costs) but also the degradation of the overall performance of the system as well as the organisational overhead.

1.2 Related work

Mechanisms for secure logging were previously described in the literature, e.g. [BeYe97, ScKe99, Acco05, Holt06, WSBL08, MaTs09]. One of the presented ideas was the use forward-secure MACs to protect the integrity of log entries. The use of hash chains ensures *forward integrity*, which means that any alterations of the log entries stored *before* the system was compromised could be detected.

The common idea of all of the mentioned schemes is to divide the timeline into several epochs. All log entries which belong to the same epoch are protected by the corresponding epoch key. Once the epoch is over, the key of that epoch is destroyed and a new one is generated for the next epoch. Usually the so-called key evolution scheme is used to derive the next key from the current one. Normally, one way function is used for key evolution. Thus, it is hard for an attacker who knows the key of the current epoch to calculate a valid key of any previous epoch. But as analysed in [MaTs09] the systems described in [BeYe97, ScKe99, Holt06] suffer from a so-called *truncation attack*—"a special kind of deletion attack, whereby the attacker deletes a contiguous subset of tail-end log entries."For others, the idea of using hash chains for log file protection (used in [ScKe99, Acco05, Holt06, WSBL08]) is patented (US patent 5978475).

The European Telecommunications Standards Institute (ETSI) has also given some advices how to implement secure logging for data retention. The proposed mechanisms are described in the ETSI document TR 102 661 „Lawful Interception (LI); Security framework in Lawful Interception and Retained Data environment" [**?**]. The document describes a general architecture for logging of retained data and describes how the retained data can be protected. But the proposed solution does not fit our needs because it is to "heavyweight" and contains some design flaws [reference needed].

TBD: related work in time limited data accessibility? TBD

Based on the reasons given, we decided to design and implement lightweight, patent-free scheme with special focus on data retention requirements.

2 The proposed scheme for secure logging

The following roles are represented in the scheme:

1. Mix **operator** – is responsible for general maintenance of Mix server(s) and logging required traffic data into protected log files. Mix operator does not need to be able to access content of the log files afterwards.
2. Law enforcement agency **officer** – will issue the data retention request backed up by court order. The usual procedure is to issue the order and receive the response in the plaintext. One cannot assume that the law enforcement agencies can easily change such procedures e.g. integrate new cryptographic mechanisms.
3. Data retention request **responder** – the entity responsible to collection and accession of protected log files, search for entries relevant to particular data retention request and responding to law enforcement officer. Serves as communication party for an officer.
4. External **storage(s)** – responsible for the keeping of the log files with redundancy required to provide the reliable backup and integrity protection.
5. Trusted **time source(s)** – responsible for providing the current date and time for the decision process about data retention period validity.

As the responsibility of Mix operator is only to keep logging software running and is usually the same person as the data retention requests responder, we will refer to both simply as an *operator* in the following text.

2.1 Data retention for anonymity services

We have developed a secure logging scheme primarily for our anonymity service called AN.ON. which is based on Mixes. A Mix [Chau81] is a server which forwards messages thereby ensuring that an outsider (e.g. an eavesdropper) cannot link incoming and outgoing messages. This is accomplished by a combination of several (cryptographic) mechanisms. In order to enhance the trustworthiness of the anonymity system, several Mixes are chained together. The sender of a given message can only be deanonymised if all Mixes along the path of his message reveal the linkage between the appropriate incoming and outgoing messages. Therefore, the use of multiple Mixes offers some protection against the dishonest Mix operators.

One can imagine our anonymisation service described below as a simple proxy which a user uses to hide its own IP-address, e.g. towards a Web-Server. Therefore, the proxy exchanges the IP-address of the user (IP_U) with its own IP-address (IP_P). This alteration of the source IP-address (together with a timestamp t) has to be retained (cf. requirement R1). For simplicity, we assume that the IP-address of the proxy will change rarely so that it is not necessary to store it with every log entry. Finally each log entry can be seen as a pair of IP-address and timestamp (in our example: (IP_U, t)). Multiple log entries are stored within one log file.

In addition to the Mixes, which are the logging servers generating and storing the logged data, two other parties are relevant to our setting: the Mix operators and law enforcement agencies. Mix operator is a legal or natural person responsible for the operation of a given Mix, and for the implementation of data retention, which include includes answering the requests for retained data by the law enforcement agencies.

As IP_P of the proxy will be visible in suspicious requests, the law enforcement agencies ask questions in the form of: "Who was using IP-address IP_P at time t_R". In order to answer such questions we need to search through our log files for all records with timestamps t_i for which: $t_R - \epsilon \leq t_i \leq t_R + \epsilon$. The need for the parameter ϵ reflects the fact that we cannot assume that all clocks of all servers are synchronised. The specific value of ϵ is usually given by law through technical regulations.

In order to facilitate the search process, log entries are stored and organized according to increasing timestamps t_i. This is also the natural order they were generated by the proxy.

The logged data has to be stored encrypted and integrity protected (cf. requirement R4). The encryption ensures that the content of the logged data can not be revealed without the knowledge of the secret key. Of course this is only true, if the server which logged the data was not compromised at the time of the data logging. The advantage of encrypting the logged data is that the data can be protected using available (probably insecure) backup mechanisms. Note that because of this backup, it is in generally not possible to (provably) delete the retained data. So the "deletion" has to be accomplished by cryptographic means (e.g., by destruction of a decryption key[3]).

2.2 Confidentiality of the retained records

Confidentiality can be achieved by either symmetric or asymmetric encryption[4]. Asymmetric encryption has the advantage that no secret key needs to be stored on the logging server but suffers from poor performance compared to the symmetric encryption. The use of symmetric cryptography leads to the problem where the secret key used for the encryption becomes vulnerable to attack. If the same key is used for the encryption of multiple log entries the attacker might be able to decrypt log entries generated *before* gaining control over the logging server.

As a compromise, we utilize a hybrid encryption scheme where the symmetric encryption is used for the log entries itself. The corresponding symmetric key k_r is stored within the log file using asymmetric encryption.

[3] Deletion of a single decryption key, which is not part of any backup, is much easier compared to ensuring that every backup copy of a given log file is deleted. This is especially true if the backup in place is not under full control of the operator of the anonymisation server itself. This in turn is the usually setting in dedicated hosting service scenarios.

[4] Basically we could also use tamper resistant trusted devices. But as such devices which are able to store large amount of logged data are not available for reasonable price to the operators of our anonymity servers we do not consider them here.

We decided to store the private key on a trusted device. Here, trusted device is a device able to control access to the private key. Note that we use this trusted device not only to prevent unauthorised access of third parties to the private key. Additionally, the access to outdated log files (cf. requirement R2) is prevented and the risk that the operator is forced by the attacker to decrypt outdated log files is mitigated by the use of trusted device usage. Therefore, an important property of the access control to the private key implemented by the trusted device is, that it not only depends on proper authorisation (e.g. password of the operator) but *on the current time*. The idea is, that the trusted device denies decryption of a symmetric key if the related log file is outdated. Basically any device which has a TPM and fulfils the requirements on Sealed Storage of the Trusted Computing Group could be used. But as they are not yet widely deployed, we decided to use smart cards as a possible alternative.

2.3 Integrity protection

So far we have described the mechanisms used to protect the confidentiality of the log entries. Now, we want to explain how the integrity of a log file is protected. We need to prevent/detect modification of log entry, copying of a log entry to another position, truncation of the log file and completely replacing whole log file by forged one. Integrity of a single log entry can be verified through MAC generated during the authenticated encryption (GCM) of that log entry and position of log entry used as initialization vector for MAC computation prevents entry copying. Truncation of the log file is prevented by appending specifically computed last MAC value as a log footer. These defenses are the usual ones. Harder to prevent is the deletion of legitimate log file and creation of the completely forged one.

In order to prevent this attack it is sufficient to protect the integrity of the *key binding* record containing key value for particular time period. We propose multiple mechanisms to achieve this kind of protection, namely: digital signatures, distribution and trusted timestamping. As for every single mechanism the security depends on different assumptions, we propose to use all of them for enhanced protection.

The "digital signature" mechanism means that the logging server signs the encryption of the *key binding* record. Note that the private signature key used by the logging server has to be changed frequently (in our case on a daily basis). Otherwise an attacker might generate a valid signature even for a log file generated before the logging server was compromised. In order to facilitate the key installation and management process, the digital signature key pairs can be generated in advance (e.g. one for each day of the year). The date (d) for which a given key pair is valid is encoded as the validity period of the public certificate of the signature test key.

The "distribution" mechanism means that every artefact involved in the integrity verification process should be distributed in a way so that it is hard for the attacker to manipulate all of the copies simultaneously. One way to achieve this is to utilise censorship resistant P2P-networks such as FreeNet [CSWH00]

or Free Haven [DiFM00]. Another possibility is to send an artefact to a number of people. In order to prevent denial of service attacks by compromising only one copy, some form of threshold voting can be introduced. The set of artefacts to be distributed should include at least a hash value of the encryption of the pair (k_r, d). If digital signatures are used, the public key certificates should be distributed immediately after their generation.

The "trusted timestamping" mechanism means that every artefact mentioned above should be timestamped. As mentioned in section 2.2 and further explained in section 2.4, trusted timestamping servers are already used to prevent access to outdated log entries. Thus, we can use the same set of servers with little reorganization.

2.4 Trusted timestamping servers as reliable time source

For the enforcement of data retention period, the smart card needs to know the current date. Smart cards usually do not have an internal clock. Therefore, the current date has to be set from the outside. An operator can set the current date, but this introduces the risk of operator being forced to set an expired date, enabling the attacker to get access to outdated log entries.

In order to mitigate this risk, we decided that the only the source of time for the smart card should be (external) trusted timestamping servers (TTS). Therefore, an additional logical step is introduced in the process of answering data retention requests, which is activated during every key recovery process. When a key recovery from the smart card is requested, the smart card creates its own unique nonce, sends it to the PC application which then creates a time stamp request according to the "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)" [RFC 3161] for every TTS with this nonce included. The TSP requests are then sent to the trusted time servers[5]. The smart card verifies the signed TTS responses, including its own challenge nonces and eventually updates the internal time according to the time stamps provided (i.e. by means of some majority decision algorithm). Note, that the public certificates of the trusted time servers can be installed immutably on the smart card during initialisation.

2.5 Fair exchange protocol for proof of released retained data

In case of a finished law trial, the defendant should able to obtain a list of these retained data which were available to police investigation, even when not used as evidence against him. If we like to provide such feature during the process of secure logging (which may potentially increase thread towards operators) with a robust proof based on digital signatures, a signed receipt for the data released to the police needs to be created. Some kind of fair exchange protocol [**?**] can be used to obtain a signed receipt in exchange for the data, but two main problems

[5] As the smart card itself has no ability to directly communicate with time servers we use the PC console as a transparent proxy, with no possibility to undetectably modify TTS response.

remains to be solved: a) place of storage of receipt and b) storage of retained data corresponding to receipt and linking meta-information.

The signed receipts should not be stored by operators (or any other single entity) only, otherwise they may become prime attacker target either by an attacker that likes to obtain information about released data or by law enforcement agency to destroy proof of this release. Both facets of attack can be solved by delinking a receipt from the released data. When such a receipt itself does not leak any information about the corresponding retained data, obtaining them makes no sense for attacker (but special attention needs to be given to side channel attacker – e.g., increased amount of receipts during large scale investigation). It can be also immediately and publicly distributed (e.g., via services providing censorship resistance) to prevent the second thread (receipt destruction).

But we have a chicken and egg problem here – the released data need to be stored for the receipt to makes sense after delinking, yet released data should be kept secret to protect ongoing investigation and privacy of (innocently) involved persons. Therefore, critical part of receipt verification – released retained data itself – cannot be publicly distributed (e.g., Eternity service) to protect against censorship. Trusted third party can be used to solve whole problem of keeping corresponding data (and optionally also receipts) and released it only to intended recipients. But such party is vulnerable to pressure and compromise.

Note that operator neither genuinely want to nor legally can store the released data itself as he is obligated to destroy all retained data after data retention period is over. Duration of this period is robustly enforced by smart card combined trusted time servers. Modification of this process for released data will negate obtained benefits of reduced attack surface, especially because of fact that released retained data are of particular interest to a potential attacker.

2.6 Overall overview

The overall process of initialisation, generation of log entries and answering law enforcement requests consists from the three groups of actions - smart card initialization, generation of retained log entries and eventually answering the law enforcement request.
The following steps are executed only once during smart card initialisation:

1. The public certificates of the signature keys of the trusted timestamping servers are immutably stored on the smart card,
2. A unique RSA-2048 key pair is generated on-card (the private key never leaves the card),

3., 4. The public key c is exported to the logging server.

5. After this initialisation, the logging server can generate encrypted and integrity-protected log files as described in section 2.

Finally, a request for the retained data is answered by executing the following steps:

6. The log files in question (according to the date) are transferred to a dedicated machine used for a processing of data retention requests.

7. After initialization of the log file processing tool, the basic log file integrity is verified according to V1 and V2 of section 2.3.

8., 9. The smart card is inserted into the reader connected to the dedicated machine. The user authenticates himself with his user PIN. Note, that in the case that the smart card reader has its own display and keypad, the PIN is entered directly on the smart card reader and not on the dedicated machine.

10. The block with encrypted key k_r necessary to access log file is sent to the smart card.

11.,12. The smart card generates the nonces used for the requests to the remote trusted timestamping servers. These requests are generated by the log processing tool and sent to the trusted timestamping servers (step 12). The responses from the trusted timestamping servers are received and relayed by the log processing tool to the smart card. The smart card verifies the validity of the received timestamps.

13. The smart card decrypts the encrypted value of k_r. If the enclosed value d is still valid (inside data retention period), the smart card calculates k and returns k to the log processing tool.

14. The log processing tool searches for the requested records and generates a report which can be sent to the law enforcement agency.

2.7 Remarks on the practical implementation

We used the smart card with JavaCard platform for our implementation. Aside from the mechanisms explained above, we implemented separate roles – privileged and user level access with different set of functions accessible only after PIN authentication. Note that smart card readers with integrated keypads should be used so that "phishing" of PINs, e.g. through a compromised PC console, is not possible.

Logging performance was demonstrated to be very good, as in case of our AN.ON system a single log entry requires less than 20 bytes with the cryptographic time overhead being less than 0.25 μs on AN.ON server.

Searching performance via our dedicated search tool for locating entries relevant to data retention request is also sufficient. The mostly used servers of our AN.ON system generated log files with speed of roughly 85000 blocks per day. Because each block contained 128 log entries, the whole log file contains more than 10 million log entries. Altogether the processing time needed by our tool was less than 30 seconds for the search leading to roughly 2800 log entries (reasonable number w.r.t. typical data retention request)) is negligible compared to the overall time need for answering a request by the law enforcement agency (i.e. checking the validity of the request itself, transferring the right log files to the dedicated machine, obtaining the decryption key from the smart card etc.). In summary, we conclude that our logging scheme fulfills the requirement R3.

The algorithms chosen for the implementation (RSA 2048 bits, GCM-AES, OAEP mode with SHA2-512) were selected to provide adequate protection for the present and future years according to the currently recommended standards.

We were not limiting ourself in selection by the technical possibilities of current smart cards (which are usually supporting SHA-1 algorithm, even when this algorithm is not recommended for the new applications – for example). Such a selection in turn required to implement selected algorithms in software, when smart card hardware support was presently missing. Therefore, the time required for retrieval of one key was approximately 90 seconds with the current setup. Nevertheless this time period is still practically useful, provided that the law enforcement agencies do not request hundreds of files per day. Smart cards with hardware support for SHA-512 algorithm will provide key recovery process within few seconds.

Further details of the practical implementation are provided in [**?**, KS09]

3 Discussion of practical usability of retained data

TBD: some thoughts similar to paper from summer school in Brno and reference

4 Experience with data retention requests

TBD: real world experience with retention requests so far

5 Conclusions

The compliance with the new data retention directive introduces not only benefits for the law enforcement agencies, but also additional risks for the users and operators of the communication service need to be mitigated. We have proposed, implemented and start into the practical usage a secure logging service based on a combination of log file encryption, key recovery with smart cards and data retention period enforcement via trusted timestamping servers. Several categories of attackers with different capabilities and levels of access to the system were analyzed.

The main contribution lies in the design and implementation of a practical system that allows logging required data with only modest impact on performance of our anonymity service, which complies with the legal requirements and does provide additional protection for the holder of cryptographic secrets necessary to access the logged records. The records can be accessed only if a cryptographic smart card and its owner are present and the retained data is not outdated. An operator cannot be forced to reveal logged records outside the data retention period, because the period is enforced directly on the smart card with the help of trusted timestamping servers.

The log data of selected German AN.ON servers are protected with the proposed mechanism since 1st January 2009.

TBD: still true? - So far, we did not receive any valid request for retained data from the law enforcement agencies. Therefore, at present, we can not evaluate how efficiently will the large number of log entries be handled, and we hope

to provide further practical details in the near future. TBD: example of data retention request

TBD: what is the state? The separate and quite complicated problem discussed is the problem of receipt creation. These receipts will contain provable information on all of the retained data that were released to the law enforcement agencies and serve as a official record (e.g., based on digital signatures and fair exchange protocols). While a seemingly straightforward task, the solution to this problem will have to avoid introduction of new risks for an operator (caused by possession of additional sensitive data on his side). Additional requirement that complicates the problem further is a need for a protection of the AN.ON users' privacy. The official record itself must not reveal any sensitive information (e.g. content of the retained data) to an outsider.

References

[Acco05] Rafael Accorsi: *Towards a secure logging mechanism for dynamic systems*; in Proc. of the 7th IT Security Symposium, São José dos Campos, Brasilien, November 2005.

[BeBK08] Stefan Berthold, Rainer Böhme, Stefan Köpsell: *Data Retention and Anonymity Services*; Proc. The Future of Identity in the Information Society - Challenges for Privacy and Security, FIDIS/IFIP Internet Security & Privacy Fourth International Summer School, Springer, Boston, IFIP Advances in Information and Communication Technology, volume 298, 2009, 92–106.

[BeFK00] Oliver Berthold, Hannes Federrath, Stefan Köpsell: *Web MIXes: A System for Anonymous and Unobservable Internet Access*; Proc. of Privacy Enhancing Technologies Workshop (PET 2000), Springer, Berlin / Heidelberg, LNCS 2009, July 2000, 115–129.

[BeYe97] Mihir Bellare, Bennet S. Yee: *Forward integrity for secure audit logs*; Technical Report, University of California at San Diego, Dept. of Computer Science & Engineering, 1997.

[Chau81] David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24/2, 1981, 84–88.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, Theodore W. Hong: *Freenet: A Distributed Anonymous Information Storage and Retrieval System*; Proc. of the Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, Springer, Berlin / Heidelberg, LNCS 2009, July 2000.

[DiFM00] Roger Dingledine, Michael J. Freedman, David Molnar: *The Free Haven Project: Distributed Anonymous Storage Service*; Proc. of the Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, Springer, Berlin / Heidelberg, LNCS 2009, July 2000.

[DiMS04] Roger Dingledine, Nick Mathewson, Paul F. Syverson: *Tor: The Second-Generation Onion Router*; Proc. of the 13th USENIX Security Symposium, August 2004, 303–320.

[GrVi05] David A. McGrew, John Viega: *The Security and Performance of the Galois/Counter Mode (GCM) of Operation*; Proc. of Progress in Cryptology – INDOCRYPT 2004, Springer, Berlin / Heidelberg, LNCS 3348, 2005, 343–355.

[Guer09] Shay Gueron: *Intel's New AES Instructions for Enhanced Performance and Security*; Proc. Fast Software Encryption, Springer Berlin / Heidelberg, LNCS 5665, 2009, 51–66.

[Holt06] Jason E. Holt: *Logcrypt: forward security and public verification for secure audit logs*; Proc. of the 2006 Australasian Workshops on Grid Computing and E-Research, January 2006, 203–211.

[MaTs09] Di Ma, Gene Tsudik: *A new approach to secure logging*; ACM Transactions on Storage (TOS), vol. 5, issue 1, ACM, New York, March 2009.

[RFC 3161] C. Adams, P. Cain, D. Pinkas, R. Zuccherato: *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*; August 2001, Proposed Standard, http://www.rfc-editor.org/rfc/rfc3161.txt.

[ScKe99] Bruce Schneier, John Kelsey: *Secure Audit Logs to Support Computer Forensics*; ACM Transactions on Information and System Security (TISSEC), vol. 2, Nr. 2, 1999, 159–176.

[Shou01] Victor Shoup: *A proposal for an ISO standard for public key encryption*; Version 2.1, 20th December 2001, http://www.shoup.net/papers/iso-2_1.pdf, last accessed Juli, 28th, 2009.

[SP 800-38D] Morris Dworkin: *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*; U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL), November 2007, http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf, last access on December, 1st 2008.

[WSBL08] Karel Wouters, Koen Simoens, Danny Lathouwers, Bart Preneel: *Secure and Privacy-Friendly Logging for eGovernment Services*; in Proc. of the 2008 Third International Conference on Availability, Reliability and Security, IEEE Computer Society, 2008, 1091–1096.