

On the Relationship of Privacy and Secure Remote Logging in Dynamic Systems

Rafael Accorsi

Department of Telematics
Albert-Ludwigs-Universität Freiburg, Germany
accorsi@iig.uni-freiburg.de

Abstract. We investigate a mechanism for secure remote logging to improve privacy guarantees in dynamic systems. Considering an extended threat model for privacy, we first describe outer and inner privacy: outer privacy expresses the traditional attacker model for privacy where identity management systems control the collection of personal, observable information; inner privacy denotes the threat posed by an attacker who attempts to get hold of private log data by tampering with a device. While privacy enhancing technologies should take outer and inner privacy into account, there is to our knowledge no approach for inner privacy, in particular for dynamic systems. To this end, we develop protocols to address inner privacy based on secure logging. Our approach accounts for the capacity limitations of resource-poor devices in dynamic systems, as it allows for the remote storage of log data, while fulfilling its security guarantees. Further, our approach can be smoothly integrated into identity management systems to combine outer and inner of privacy.

1 Introduction

“Logging” refers to a sort of prophylactic process of collecting and storing information about the events in the system. Provided that the collected data is authentic, it gives a sound basis upon which other functionalities can be deployed. Initially, log data was chiefly used by system administrators to investigate the behaviour of a system in case of malfunction. The advent of the Internet and e-commerce brought with it the need for accountability and billing mechanisms, which are also deployed upon logged data. Subsuming the previous functionalities, dynamic systems also use log data for self-management tasks in autonomous components [10] and, more recently, as “digital witnesses” in forensic disputes [13].

Log mechanisms can be seen as a privacy endangering technology, as they greedily collect information and prepare it for a proceeding analysis. This becomes a threat when the logged data includes personal attributes. This situation is a result of the underlying attacker model under consideration. Traditionally, the attacker is taken to be the current communication

partner, or the devices capturing observable attributes of an individual. To minimise the chances of a privacy violation, identity management and privacy aware systems have been devised [1, 2, 14]. These tools employ several techniques to diminish individual’s observability and give him control over the release of his personal attributes, thereby allowing the characterisation and to some extent the enforcement of his privacy policies.

However, it is questionable whether such an attacker model for privacy is realistic in dynamic systems. We consider dynamic systems, i.e. systems allowing for the omnipresent availability of computing power, communication and data, as mixed-mode infrastructures: they combine resource-rich and resource-poor components. (We consider this as a result of advances in distinct aspects of computing systems, namely autonomy [10], pervasiveness [16], and reachability [9] of devices.) The underlying computational ubiquity makes it impossible to fully control observability: data will inevitably be collected. Nevertheless, what the next privacy threat in dynamic systems is remains to be defined.

In this paper, we distinguish between two concepts of privacy: *outer* and *inner* privacy. These are distinguished in terms of the underlying threat model: while outer privacy denotes the customary privacy threats as lack of observability and control over the release of data, inner privacy is defined by an attacker who attempts to obtain private information by tampering with private log data. These concepts are complementary and cannot be considered in isolation: getting hold of private log data indeed invalidates the effort that prevented its release.

Nevertheless, while there is a plethora of mechanisms addressing outer privacy, tools to address inner privacy are lacking. To this end, we present a secure logging approach to tackling inner privacy based on techniques for tamper evident logging presented in [18]. Due to the underlying mixed-mode setting, capacity limitations should be taken into account. Our approach is tailored to dynamic systems, in particular for resource-poor devices: we equip our secure logging mechanism with features to allow for secure remote storage. This is achieved by an acknowledgement phase in which the data’s collector proves the possession of log data to the sending device. Moreover, selective access control mechanisms for each entry are built in, thereby facilitating the analysis without disclosing the whole logfile. We remark, though, that the applicability is not restricted to resource-poor devices. Resource-rich components could employ similar techniques to secure logging, while omitting the acknowledgement phase.

Overall, our work sheds a light on the relationship between security and privacy in dynamic systems on the one hand, and secure logging

mechanisms on the other. Although we focus here on privacy, we see our contribution as a wider one. Dynamic systems are characterised by a high degree of autonomy, which can only take place when the components are able to adapt themselves to the needs of a particular context. This requires an analysis of logged data, which, if inaccurate, leads to an incorrect functioning: *ex falsum quod libet*, i.e., from a falsity everything follows. By providing the necessary authenticity guarantees, we ensure that log data provides a reliable basis for this, as well as similar processes.

We proceed as follows. In §2 we describe outer and inner privacy and describe the requirements the latter puts on secure logging mechanism. We present our approach to secure remote logging in §3 and discuss its security properties and limitations in §4. In §5, we report on related work and conclude in §6.

Preliminaries. We start out by defining the terminology used in this paper. We refer to the components involved in logging services according to their role: the *device* generates a log message; the *relay* receives a log message and forwards it to another service; and the *collector* receives a message and, in one way or another, stores log messages. To simplify matters, we consider that a device and a collector communicate either without an intermediary relay, or that the relay does not misbehave. (We currently investigate an approach where we leave out these assumptions; see §5 for details.) We employ the following notation:

- d_i denotes the i th device, r_i the i th relay and c_i the i th collector.
- $\{X\}_K$ denotes the symmetric encryption of message X under the key K . $Sign(X)_K$ stands for the signature of X with K .
- X, X' stands for the concatenation of the messages X and X' .
- K_s stands for the public key of the service s and K_s^{-1} the corresponding private key of s .
- $MAC_K(X)$ denotes the message authentication code of X under K .
- $Hash(X)$ is the one way hash of X .

While we do not prescribe a particular set of algorithms to implement the cryptographic primitives above, we assume that these functions fulfil the expected properties in that, e.g., it is infeasible for an attacker to induce collisions of hash values. These primitives are described in, e.g., [17].

2 Outer and Inner Privacy

In order to make the discussion on the relationship between privacy and logging mechanisms clear, it is necessary to distinguish between two views

of privacy: *outer* and *inner* privacy. Roughly speaking, these standpoints are circumscribed in terms of the underlying attacker model used to characterise attacker’s potential threat. We first illustrate the idea behind the attacker model defining outer privacy by means of an example.

Example 1 (Outer privacy). Suppose an individual goes shopping in the city centre. There are hundreds of other individuals there and everybody can see him entering a shop, choosing a product and paying for it. Now suppose the individual simply asks for the article and receives it packed in a store’s carrier bag. Except for the seller, nobody else knows what has been bought. However, if the individual leaves the shop, it is possible to infer that he was in a particular shop, since he carries his carrier bag. In turn, if we assume that the carrier bag given by the seller has no name or logo, it can be inferred that something has been bought, however not knowing what. Finally, if the individual packs the carrier bag into his backpack before leaving the shop, it is impossible to infer whether anything has been bought at all. \dashv

The scenario depicted above can be compared to a dynamic system. Pedestrians of our example are sensors capturing the context, i.e. individual’s actions, in various ways. Individuals are devices that communicate with other devices. Privacy protecting measures, such as bundling the product or packing it into a backpack, involve the communication over secure channels and the use of identity management systems.

In computing systems, privacy enhancing technologies such as identity management and privacy aware systems attempt to deter an attacker impersonated by the communication partner itself, as it stores attributes of an individual, and surrounding context, that can observe this communication. These are represented in our example by the seller and other individuals observing the transaction. To account for privacy, systems offer a knob to control the release of data, and minimise individuals observability. Overall, they control what can be observed, while avoiding the correlation of this information with the individual behind it. Technically, mechanisms like network anonymisers, pseudonyms and partial identities are employed to control individuals’ observability and the disclosure of data. They thus protect what we refer to as outer privacy. We now look at Example 1 from a different perspective.

Example 2 (Inner privacy). Take the same situation as in Example 1. Irrespective of the level of privacy the individual aims at, the information “logged” by this individual does include the purchase of a particular product, as well as the preferences leading to this transaction. Assume that

a preventive measure, e.g. packing the purchased good into a backpack, is carried out. This expresses the individual’s unwillingness to share this information, namely the purchase of a particular good, with individuals other than the seller. Hence, if this action is somehow stored—and there is enough evidence to assume that it will be—, then it must be securely stored, for the release of this information entails loss of privacy. \dashv

Note that we consider a different attacker in this example. Namely, an attacker who tries to obtain private information by tampering with the individual’s device. This unveils a, to our knowledge novel, tight relationship between the preventive measures chosen to protect the individuals’ privacy during a transaction and the log data associated with this transaction: the preventive measures determine the degree of sensibility ascribed to log data. Inner privacy denotes the confidentiality of private log data.

Unless strong, albeit hypothetical, assumptions regarding tamper resistance are made (or, equivalently, we assume that the underlying attacker model disregards intrusion), outer and inner privacy cannot be considered in isolation: an unbalanced distribution between them harms the overall privacy level. This balance is essential in dynamic systems. Due to an extended attack vector, tamper resistance for resource-poor devices can be hardly achieved, and improvements on tamper resistance usually come at the cost of performance. Nevertheless, research into privacy mechanisms is tailored to address outer privacy. Below, we present an approach to protecting inner privacy based on secure remote logging.

2.1 Inner Privacy by Secure Remote Logging

Three issues arise from the discussion on inner privacy. First, since complete tamper resistance is infeasible, mechanisms should be in place to guarantee that the logged data cannot be tampered with. Second, the usual amount of data logged by a device is enormous. This already causes problems to resource-poor devices and, if additional data ensuring the security of log data is added, log data grows in size and the device is bound to run out of storage. Hence, a secure log service should also allow for the secure remote storage of data. Third, if log data is remotely stored, the logging mechanisms should encode data in such a way that a third-party service only gains access to those entries it has permission to. Together, these three characteristics build the setting within which we introduce our approach to protecting inner privacy.

In our approach, log data is secured at logging the entry associated to an event and not as a separate process. Each log entry is (symmetri-

cally) encrypted with a unique key. The techniques we employ guarantee forward integrity, i.e. if an attacker succeeds in breaking in at time t , all the log data stored before t cannot be compromised. (We introduce forward integrity in §3.) Forward integrity also helps assessing the circumstances that allowed for and led to an intrusion. Moreover, we ensure that attempts to tampering with logged data do not go undetected.

Further, our logging mechanism supports secure remote storage. By addressing this issue, we consider that the collector can misbehave or, similarly, that it is not tamper resistant enough. Thus, while in our protocol the collector possesses enough information to prove the (integral) possession of log data, it cannot derive the keys necessary to decrypt the log entry. Moreover, even if the collector (or an attacker) succeeds in decrypting one entry by guessing the right key, this does not reveal enough information to derive the keys used to encrypt the previous or next entries. This stems from the secrecy of a piece of information we refer to as A_0 , which is not known by the collector.

Finally, our approach allows the selective disclosure of log data based on techniques introduced in [18]. The permission mask W works as an access control list and indexes the service (or group thereof) that can access an entry. The symmetric key used to encrypt the log entry can be recomputed and given to authorised services when needed. (Protocols to regulate the delegation and access from collector and third-party services are not in the scope of our work.) Note that this is also in-line with a common understanding of privacy that postulates that an individual should have the chance to actively determine who learns information about him.

3 An Approach to Secure Remote Logging

Log data can only provide a sound basis for further services when it is authentic. We define authenticity as the simultaneous fulfilment of data *integrity* and *uniqueness*, as illustrated in Fig. 1. *Confidentiality* of log entries is necessary for privacy and is considered as an extra protection goal. A log service is labelled *secure* when integrity, uniqueness and confidentiality properties are fulfilled.

- Integrity states that log data faithfully reflects the state of the devices, i.e., the log data is accurate (entries have not been modified), complete (entries have not been deleted), and compact (entries have not been illegally added to the logfile). Thus, log data is not modified, deleted, or appended during the transmission to, and storage at, the collector.

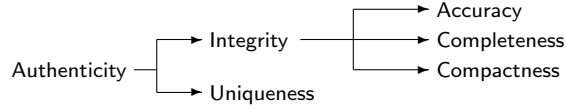


Fig. 1. Authenticity property for secure logging.

- Uniqueness states that log data shall not allow for parallel realities. Concretely, it is impossible to intercept log data sent from d_1 to c_1 and to resend it (possibly in modified form and claiming a different device identity) to c_2 . Log data must be uniquely tagged.
- Confidentiality states that log entries cannot be read by unauthorised individuals, for this would harm inner privacy. Note that confidentiality is also related to uniqueness, for log data transmitted in clear-text can be easily duplicated.

These properties are realised with cryptographic techniques, which need to ensure *tamper evidence*, i.e., attempts to illicitly manipulate log data must be detectable to a verifier [12], and *forward integrity*, i.e., log data contains sufficient information to confirm or rebuke allegations of log data modification before the moment of the compromise [5].

The goal of the attacker is to gain access to private log data and, thus, to violate its integrity, uniqueness, and confidentiality. The threats posed by an attacker are described using an attacker model. While we are aware of recent ongoing research on formally characterising attacker models for dynamic environments [6], we refrain from sticking to a particular model.

3.1 Overview of our Approach

Our approach to secure remote logging services in dynamic systems is based on and extends the techniques proposed in [18]. The idea is to devise a protocol to securely store log data. The protocol starts at the device: it is in charge of applying cryptographic techniques to ensure tamper evidence and forward integrity. When the device runs out of storage, it contacts the collector to request remote storage. The protocol ends with an irrefutable proof of possession of the collector. In detail:

1. *initialisation and construction of the logfile*: the device is in charge of applying cryptographic techniques to safeguarding the integrity and uniqueness of its logfile. To this end, it computes a secret random value pv_0 and, for each entry, a proof value Z associated to that entry

based on pv_0 . The Z -values build a chain, so that the integrity of the whole chain can be checked by analysing its last link.

2. *mutual authentication of services*: apart from authentication, the device and the collector also agree on a secret value pv_0 that will be used to ascertain authenticity of log messages.
3. *acknowledgement of receipt from collector*: by receiving the chunk of log data, the collector computes the Z -value associated to each entry and sends the last Z -value signed together with a timestamp and protocol sequence number back to the device. The device then stores this unambiguous piece of information, as it demonstrates that the collector received the chunk of log data correctly and can be thus held accountable for attacks upon this data.

We describe the initialisation, appending, and acknowledgement processes below; the authentication phase and the secure exchange of pv_0 are taken for granted.

3.2 Initializing the Logfile

Assuming that the device d successfully generates a secret value pv_0 , d creates the log file by inserting the first entry into it. Entries have the following fields and are initialised as:

- W_0 is a permission mask to regulate the access to the log entry L_0 . As in [18], the initialisation value is **LogfileInitializationType**.
- $\{D_0\}_{K_0}$ is the symmetrically encrypted log data for the entry L_0 and K_0 is a random session key. To provide the necessary security guarantees, D contains not only the event to be logged, but also a timestamp and a protocol identifier. (The former states the recentness of the entry, the latter avoids harmful protocol interactions.)
- Y_0 is the first link of a hash-chain. (In the simplest form, a hash-chain Y can be inductively defined as $Y_1 = Hash(Y_0)$ and $Y_n = Hash(Y_{n-1})$.)
- A_0 is the authentication key for the first entry. As we discuss in §4, this is the secret upon which the security of the whole protocol rests. In practice, since we do not assume that the device is tamper resistant, we suggest that this secret should be kept off-line.
- $L_0 = W_0, \{D_0\}_{K_0}, Y_0$ stands for the initial log entry.
- Z_0 is the message authentication code of L_0 defined as $MAC_{pv_0}(L_0)$. This piece of information is used to compute the proof value associated to the whole chunk of log entries and, thence, can be used as a challenge against the collector. For this, it will *not* be send along with L_0 to the collector.

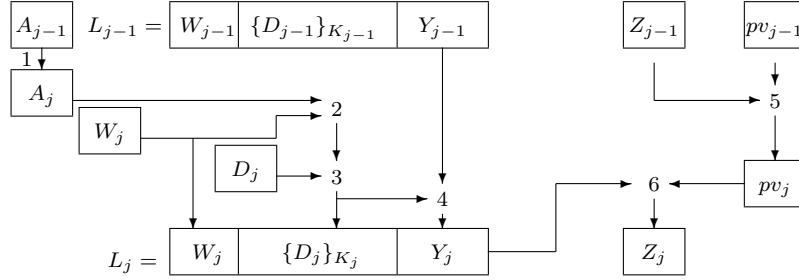


Fig. 2. Adding an entry to the logfile.

3.3 Appending Log Entries

After creating the logfile, the device starts adding entries to the logfile. This is schematically illustrated in Fig. 2, where the numbers mean:

1. $A_j = Hash(A_{j-1})$ denotes the authentication key of the j th log entry. The confidentiality of this information is for accomplishing the aforementioned security properties, as it is used to encrypt log entries. Thus, we assume that the computation of the new value irretrievably overwrites the previous value.
2. $K_j = Hash(W_j, A_j)$ is the cryptographic key with which the j th log entry is encrypted. This key is based on the permission mask W_j , so that only authorised services gain access to the entry.
3. $\{D_j\}_{K_j}$ is the encrypted log entry.
4. $Y_j = Hash(Y_{j-1}, \{D_j\}_{K_j}, W_j)$ is the j th value of the hash-chain. Each link of the hash-chain is based on the corresponding encrypted value of the log data. This ensures that the chain can be verified without the knowledge of the actual log entry.
5. $pv_j = Hash(Z_{j-1}, pv_{j-1})$ is the proof value associated with entry j .
6. $Z_j = MAC_{pv_j}(Hash(L_j))$ is the authenticator of the j th log entry. We compute the message authentication code for the whole entry instead of a field of it (in the case of [18], the hash-chain value Y).

The log entry L_j generated by the device consists of the permission mask W_j , the encrypted log entry $\{D_j\}_{K_j}$, and the hash-chain value Y_j ; it is denoted by $L_j = W_j, \{D_j\}_{K_j}, Y_j$.

3.4 Acknowledgement Phase

The last phase of the protocol aims to provide irrefutable evidence regarding collector's possession of the chunk of log data sent by the device, as well as the chunk's integrity. For this, the following steps are used:

- by receiving the chunk starting at L_j and ending at L_k (with $j < k$), the collector computes for each entry L_i the corresponding pv_i and, thus, Z_i values.
- after $k - j$ iterations, the collector obtains $Z_k = MAC_{pv_k}(Hash(L_k))$.
- the signed proof value is sent to the device $Sign(Z_k)_{K_c^{-1}}$. This message includes a timestamp and protocol step identifier.
- the device then checks whether the authenticator Z_k matches with the authenticator computed during the second phase. If it does, the device frees the storage by deleting the chunk.

4 Security Properties and Limitations of our Protocol

Our logging protocol ensures inner privacy by guaranteeing the integrity, uniqueness and confidentiality of log data, and its implementation should fulfil tamper evidence and forward integrity. We now report on the characteristics of the protocol responsible for ensuring these properties, as well as its underlying limitations.

Security Properties. The integrity of log messages is ensured by a combination of cryptographic operations and stems from the message authentication code Z and the hash-chain Y . The message authentication code ensures that accuracy, completeness and compactness have not been violated during the transmission. Similarly, the hash-chain provides long-lasting security guarantees regarding the stored data.

The message authentication code is essential in the acknowledgement phase, for otherwise no guarantees regarding the receipt of log data can be met. We use a one-way function to hash the proof value pv and the message authentication code Z of the previous log entry, thereby building a chain of log values. Since the value of the last link depends on the previous values, it suffices to verify the last link of the chain to ensure the integrity of the whole chunk.

The hash-chain Y plays a similar role. However, unlike Z , the hash-chain is part of the log entry and can be used to ascertain whether an entry j (as well as the chunk of entries before j) is correct, since it depends on the permission mask, the previous link on the hash-chain, and the encrypted log entry.

We ensure uniqueness by means of timestamps, as well as sequence numbers, and the confidentiality of log data. This combination makes it impossible for an attacker to reuse chunks of log data. Timestamps are used in protocols to prevent replay attacks and its implementation

is straightforward [7]. (We remark, however, that the practical use of timestamps is tricky, as it subsumes an underlying synchronised clock.)

This easiness contrasts with the subtleness associated with the mechanism to ensure confidentiality. The contents of the j th log entry is encrypted using an entry authentication key derived from A and the permission mask, as in [18]. This key is derived using an one-way function and the secrecy of the entry depends on the knowledge of the attacker about A , for if the attacker knows A_{j-1} , he can obtain K_j and, thus, D_j . (Observe that W_j is not encrypted.)

It remains to be investigated as to whether tamper resistance and forward integrity are accounted for. Tamper evidence is given by the message authentication code and the hash-chain values: while the former provides tamper evidence in the acknowledgement phase, the latter ensures this property for the time that log data is kept on the collector. Note that the timestamps also play an accessory role, as they help detecting illegal entries. Forward integrity is provided by the secrecy of A . Since the previous values of A are deleted from the memory right after their use, an attacker that succeeds breaking into the device can only access the current (and the subsequent) entries. We discuss this issue below.

Limits of the Logging Protocol. The logging protocol we present in §3 has several limitations, thereby confining the class of its potential applications. Below, we briefly discuss the most significant limitations.

- Timepoint of a successful attack: the integrity of the logfile can only be ensured up to the point in time at which the attacker takes over the collector (or device). Once an attacker succeeds in compromising a machine, he can misuse log information at will. This property is closely related to forward integrity described above.
- Deletion of log entries: tamper evidence allows us to detect the deletion of log data. However, it is always possible for an attacker to completely (and irretrievably) delete the log file or entries thereof, in particular when we assume that no write-only hardware or media (e.g. WORM disc, paper print-out, optical jukebox) is at hand.
- Collector’s behaviour: unless it is compromised, the collector is assumed to follow the protocol. However, the device does not acquire any evidence about the correct behaviour of the collector. See §6 for a discussion on this topic.
- Computational cost: the computational cost associated with running the logging protocol we propose is relatively high and could be an

overkill for resource-poor devices. Two alternatives arise: either simplify our approach, or delegate some of the tasks to a (trusted) relay. We address this issue in §6.

- Denial of service (DoS): although we consider the potential for a DoS attack upon the protocol, there is too little we can do in our protocol to prevent a DoS from happening. According to [11], DoS attacks have an inherent practical characteristic, so that outsmarting these attacks requires preventive measures during the implementation.

5 Related Work

Our work lies in the intersection of two research fields: privacy mechanisms and secure logging. While this intersection is not void, it has mainly focussed on outer privacy or has ignored the challenges of dynamic systems. The approach for secure remote logging we propose above takes all these components into account.

Research into outer privacy includes the development of anonymising proxies such as JAP, identity management systems to control the disclosure of personal data, and privacy aware systems. Here we focus on a few of them closer to our work; see [15] for further approaches. JAP is a sophisticated mechanism to anonymise an individual [3]. Perfect anonymity is usually useless for practical purposes, for electronic transactions cannot be carried out without an adequate level of accountability. To this end, identity management systems use anonymising services to implement partial identities and pseudonyms. Examples of such systems are iManager [2] and idemix [1]. While the techniques employed to implement these techniques vary, the overall goal is to control the release of data and, at the same time, avoid the inference of further personal data; in particular, to deduce the real identity of the individual behind a pseudonym. *pawS* is a privacy aware system for dynamic systems [14]. In-line with our work, Langheinrich assumes that the collection of data will occur anyway, so that perfect protection of personal information will be hardly achievable. To counteract the problem, he develops a system to ensure that privacy policies are known to, and to some extent enforced by, individuals. As we point out above, research into outer privacy is complementary to ours.

Moving towards the overlap of privacy and secure logging, Flegel introduces an approach to consider privacy in unix log files [8]. Based on techniques for outer privacy, the author introduces components to pseudonymise log data generated by the syslog daemon, a standard log

mechanism in *nix systems. This addresses only the data collected from an individual, that is, it addresses only outer privacy.

Research on secure logging mechanisms is mostly of a practical nature, somehow based on extensions of syslog, and not tailored to dynamic systems. An exception is the approach proposed by Schneier and Kelsey [18]. They describe an approach to secure logging data in marginally trusted collectors. While the approach we propose employs and extends techniques of [18], we act upon a different set of assumptions, namely dynamic systems with resource-poor devices, and consider a different problem, i.e., provide log services in dynamic systems.

6 Conclusion and Ongoing Work

We distinguish between outer and inner privacy, and present a mechanism to address inner privacy in dynamic systems. Our characterisation of inner privacy is based on an extended threat model where the attacker attempts to access private log data, instead of collecting data about a individual's behaviour. To cope with inner privacy problems, we propose a secure logging mechanism that allows for tamper evident remote storage of data, being thus in-line with dynamic systems and their resource poor-devices.

This work is part of an effort to bring basic security mechanisms into dynamic systems, and several interesting issues remain open. First, although we characterise outer and inner privacy, we do so by means of examples. Defining these terms is essential to clarify our contribution.

Second, to-date outer privacy prevails and there is no mechanism that combines outer and inner privacy. We are adapting the iManager, an identity manager system for outer privacy [2], with the technique for inner privacy presented above. This allows a practical insight on the complexity involved in dealing with inner privacy by testing it in a relatively resource-poor device. (We currently employ a Compaq iPAQ.)

Third, secure logging is a demanding process and only selected, sensitive events should be securely logged; i.e. log data should not be chosen arbitrarily. For this, corresponding privacy policies should be at hand, by means of which log data could be selected. Current investigation includes the definition of policy language and its interplay with the secure logging mechanism we propose. This could also facilitate the further analysis of log data: log data could be audited to detect misbehavior.

Fourth, there is no guarantee with respect to the behaviour of the collector that receives log data and proves its possession to the device. It is possible that, although it performs the protocol as expected, it may mis-

use log data by sending it to other services, or delete log data completely. To make statements regarding the behaviour of the collector, we envisage the use of trusted computing platforms and remote attestation techniques. Our goal is not only to assert collectors' behaviour, but, chiefly, to delegate part of the logging protocol to the collector, thereby saving the computing power needed to perform the necessary cryptographic operations. Theoretical comparisons exhibit an extraordinary alleviation of devices' load [4] and current practical work investigates the issue further.

Finally, while our protocol allows for a selective disclosure of log data, we have neither investigated how this could be practically performed, nor the overheads thereby involved. Since the generation of the symmetric keys used to encrypt the log data depends on A_0 , the generation of the key space associated with the release of data should be carried out by the device. We plan to develop further protocols to address this issue.

References

1. idemix. <http://www.zurich.ibm.com/security/idemix/>, 2005.
2. iManager. <http://www.iig.uni-freiburg.de/telematik/atus/idm.html>, 2005.
3. JAP anonymity and privacy. <http://anon.inf.tu-dresden.de/>, 2005.
4. R. Accorsi and A. Hohl. Delegating secure logging in pervasive computing systems. To appear in the *3rd Conf. Security in Pervasive Computing*, 2006.
5. M. Bellare and B. Yee. Forward integrity for secure audit logs. Tech. report, Univ. of California at San Diego, Dept. of Computer Science & Engineering, 1997.
6. S. Creese, M. Goldsmith, R. Harrison, B. Roscoe, P. Whittaker, and I. Zakiuddin. Exploiting empirical engagement in authentication protocol design. In *2nd Conf. Security in Pervasive Computing*, vol. 3450 of *LNCS*, pages 119–133, 2005.
7. D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.
8. U. Flegel. Pseudonymizing unix log files. In *Infrastructure Security Conference*, vol. 2437 of *LNCS*, pages 162–179, 2002.
9. G. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, 1994.
10. W. Gibbs. Autonomic computing. *Scientific American*, 2002.
11. M. Graff and K. van Wyk. *Secure Coding: Principles & Practices*. O'Reilly, 2003.
12. G. Itkis. Cryptographic tamper evidence. In *Conf. on Computer and Communication Security*, pages 355–364, 2003.
13. E. Kenneally. Evidence enhancing technology. *login*, 28(6):62–66, 2003.
14. M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In *4th Conf. on UbiComp*, vol. 2498 of *LNCS*, pages 237–245, 2002.
15. G. Müller and S. Wohlgemuth. Study on mobile identity management. Deliverable for Fidis Project, Institute for Computer Science and Social Studies, 2005.
16. M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, pages 10–17, 2001.
17. B. Schneier. *Applied Cryptography*. John Wiley and Sons, Inc, 1996.
18. B. Schneier and J. Kelsey. Security audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, May 1999.