Limits to Anonymity when Using Credentials

Andreas Pashalidis** and Chris J. Mitchell

Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom,
{A.Pashalidis,C.Mitchell}@rhul.ac.uk,
WWW home page: http://www.isg.rhul.ac.uk

Abstract. This paper identifies certain privacy threats that apply to anonymous credential systems. The focus is on timing attacks that apply even if the system is cryptographically secure. The paper provides some simple heuristics that aim to mitigate the exposure to the threats and identifies directions for further research.

Keywords: anonymous credential systems, pseudonym systems, unlinkability, privacy, timing attacks

1 Introduction

Credential systems allow subjects to prove possession of attributes to interested parties. In a sound credential system subjects first need to obtain a structure termed a *credential* from an entity termed the credential *issuer*. The issuer encodes some well-defined set of attributes together with their values into the credential which is then passed on, or 'granted', to the subject. Only after having gone through this process can the subject prove possession of those (and only those) attributes that are encoded in the credential. During this latter process, the interested party is said to 'verify the credential' and is therefore called a *verifier*. Subjects are typically human users, issuers are typically well-known organisations with authority over the attributes they encode into the credentials they issue, and verifiers typically are service providers that perform attribute-based access control.

An example of a credential system is a Public Key Infrastructure (PKI). In a PKI, credentials are public key certificates that bind together subject attributes such as subject name, public key, its issue and expiry dates, and so on. The credential issuer is the Certification Authority (CA); it grants public key certificates according to some subject registration procedure. Finally, credential verifiers are the entities within the PKI that accept the certificates issued by the CA.

** The author is sponsored by the State Scholarship Foundation of Greece.

In conventional credential systems (e.g. a PKI), issuers and verifiers identify any given subject by a system-wide identifier. This has a potentially severe impact on the subject's privacy, as it enables issuers and verifiers to combine their knowledge about the subject. Indeed, they can construct individual transaction histories for all the subjects in the system, simply by correlating credential-related events using these identifiers.

Over the last 20 years, a significant amount of research has been performed on credential systems that try to address the above privacy issue (see, for example, [2–4, 6–8, 10, 11]). These systems are known as *anonymous* credential systems. In an anonymous credential system, subjects establish a different identifier with each issuer and verifier they wish to interact with, where we assume throughout that these pseudonyms cannot be connected to the subject's true identity. These identifiers, termed the subject's *pseudonyms*, are unlinkable, i.e. they do not possess any connection with one another. This means that it is infeasible, for colluding issuers and verifiers, to decide with certainty whether or not any given pair of pseudonyms belongs to the same subject[1]. While a subject obtains a credential under the pseudonym that was established with the issuer, proof of its possession[2] takes place under the pseudonym established with the verifier. Of course, in order for the system to remain sound, subjects should only be able to successfully prove possession of credentials that they were indeed issued by some legitimate issuer.

In this paper, we consider practical limits to the level of pseudonym unlinkability (and, thus, subject privacy) offered by anonymous credential systems. In particular, assuming the soundness and security of such a system, we consider how timing attacks, launched by colluding issuers and verifiers, may affect pseudonym unlinkability. Finally, we outline possible pragmatic approaches to minimising exposure to such attacks.

The paper is structured as follows. The next section outlines the assumptions we make about anonymous credential systems, section 3 discusses the issue of encoding freshness into credentials and section 4 presents the timing attacks. Section 5 provides some simple heuristics to counter the attacks and section 6 concludes, giving directions for further research.

2 A general model for anonymous credential systems

A number of anonymous credential systems have been proposed in the literature, each with its own particular set of entities, underlying problems, assumptions and properties. This section presents the model of anonymous credential systems on which the rest of the paper is based. It is intended to be as general as possible, in order to be consistent with the majority of existing schemes.

[1] Assuming that at least two subjects exist within the system.

[2] Proving possession of a credential amounts to proving possession of the attributes that are encoded within the credential. We refer to this process also as the *showing* of a credential.

We consider an anonymous credential system to involve three types of player: subjects, issuers and verifiers. We refer to issuers and verifiers, collectively, as 'organisations'. It is assumed that subjects establish at least one pseudonym with each organisation with which they wish to interact. These pseudonyms are assumed to be indistinguishable, meaning that they do not bear any connection to the identity of the subject they belong to. We further assume that pseudonyms are unlinkable, i.e. two pseudonyms for the same subject cannot be linked to each other. Subjects may obtain credentials, i.e. structures that encode a well-defined, finite set of attributes together with their values, from issuers. They may subsequently show those credentials to verifiers, i.e. convince them that they possess (possibly a subset of) the encoded attributes. A credential is issued under a pseudonym that the subject has established with its issuer, and it is shown under the pseudonym that the subject has established with the relevant verifier.

It is assumed that the anonymous credential system is sound. This means that it offers *pseudonym owner protection*, i.e. that only the subject that established a given pseudonym can show credentials under it. Soundness also implies credential *unforgeability*; the only way that subjects may prove possession of a credential is by having obtained it previously from a legitimate issuer. In some applications, it is required that the system offers the stronger property of credential *non-transferability*. This property guarantees that no subject can prove possession of a credential that it has not been issued, even if the subject colludes with other subject(s) that may have (legitimately) obtained such a credential. In other words, a system that offers non-transferability prohibits credential sharing, whereas a system that offers only unforgeability, does not. (Of course, the degree of protection against credential sharing is always limited, since if one subject gives all its secrets to another subject then the latter subject will always be able to impersonate the former and use its credentials.) We require that credentials are bound to the subject to which they have been issued. We therefore assume that either the system offers non-transferability or that in practice subjects do not share their credentials.

It is assumed further that the system properly protects privacy in that a subject's transactions with organisations do not compromise the unlinkability of its pseudonyms. We note, however, that this unlinkability can only be guaranteed up to a certain point, as credential *types* potentially reveal links between pseudonyms. The type of a credential is defined as the collection of attribute values that are encoded into the credential. An organisation, for example, that issues demographic credentials containing the fields `sex` and `age group`, with possible values of {`male`,`female`} and {`18-`,`18-30`,`30-50`,`50+`} respectively, may actually issue up to 8 different types of credential (one for each combination of values). To see how credential types can be exploited to link subject's pseudonyms, consider the following trivial scenario. At time τ, a credential of type t is shown under the pseudonym p. However, suppose that up to time τ, only one credential of type t has been issued, and this was done under pseudonym p'. It follows, under the assumption that credentials are bound to subjects, that the

two pseudonyms p, p' belong to the same subject; the colluding organisations can successfully link those two pseudonyms.

We note that, as part of credential showing, some anonymous credential systems allow subjects to reveal only a subset of the encoded attributes; in the above example it may be possible for the subject to reveal only the value of `sex`. For these systems, it is tempting to define the type of a credential as the collection of attributes that is revealed to the verifier during showing. However, we restrict our attention to the scenario where the verifier, rather than the subject, selects the attributes to be revealed during credential showing. This is, as far as our analysis is concerned, equivalent to the case where only the required set of attributes is encoded into a credential in the first place. This scenario is also likely to be valid for the case where verifiers perform attribute-based access control.

3 Encoding freshness into credentials

In certain applications, typically those involving short-lived credentials, verifiers need to validate the freshness of credentials. Thus, some indication of freshness has to be encoded into the credentials by their issuers. However, this indication constitutes an additional attribute, and its value helps determine the type of the credential. If the indication of freshness is unique for each credential (such as a serial number, a counter value, or a nonce), then it becomes trivial for organisations to link pseudonyms, as every credential will have its own, unique, type. It is thus desirable, from a privacy perspective, that the indication of freshness is shared among as many credentials as possible. One possible freshness indication is a timestamp generated using a universal clock, with a sufficiently coarse accuracy. We thus henceforth assume that one of the attributes that is present in all credentials is such a timestamp.

A question that arises in this context is who decides whether or not a credential has expired. If it is the issuer, it seems more appropriate for the timestamp to indicate the time of expiry. If, on the other hand, it is the verifier, then it makes more sense for the timestamp to indicate the time of issue. Since the latter alternative enables verifiers to have individual policies with respect to expiry of credentials, in the sequel we assume that the timestamp indicates the time of credential issue[3]. With τ_i denoting the time interval at which the timestamp is being refreshed by the issuer (and without loss of generality), we assume that all credentials issued between time τ and $\tau + \tau_i$ will carry the timestamp τ. More generally, credentials issued between $n\tau_i$ and $(n+1)\tau_i$ are assumed to carry the timestamp $n\tau_i$ (with $n \geq 0$ being some non-negative integer indicating the current period).

[3] We do not consider the case where two timestamps are encoded into the credential, indicating both the beginning and the end of its validity period.

4 Timing attacks

We consider some attacks that may be launched by colluding organisations who wish to link pseudonyms that belong to the same subject. It is sufficient for the organisations to link the events of credential issuing and showing as corresponding to the same subject; this amounts to linking the pseudonyms that correspond to those events. We distinguish between two attack strategies. As both of them exploit temporal information in the system, they both fall into the category of timing attacks.

The first strategy does not exploit the timestamps that are encoded into the credentials. Instead, it exploits the behaviour of subjects in certain scenarios, i.e. the high likelihood that a subject will show a credential to a verifier soon after it was issued to them by an issuer. That is, if a credential is issued at time τ and subsequently shown at time $\tau + \tau_\delta$, where τ_δ is small, then the issuer and verifier could collude to learn (with high probability) that the two pseudonyms involved belong to the same subject. Of course, the meaning of 'small' here will depend on the application, in particular on the rate of credential issuing. As a result, this timing attack is not equally serious in all scenarios. While in some applications (e.g. driving licences) it may not be a concern at all, in others (e.g. tickets, electronic cash, authentication tokens) the threat may be much more significant.

The second strategy is essentially the one already mentioned in section 3 above, namely the correlation of issuing and showing events based on the type of the credentials involved, and thereby linking the associated pseudonyms. However, the fact that we are now assuming that credentials encode timestamps (whose freshness is checked by verifiers), guarantees that credentials issued in different periods are of different types. Exploiting this particular fact may render the generic correlating-by-type attack much more dangerous.

5 Countermeasures

An obvious countermeasure to the first attack strategy is to require subjects to wait between obtaining and showing a credential. However, this needs to be managed carefully, since simply imposing some fixed waiting time, say τ_w, does not really reduce the exposure to the attack. This is because correlation between issuing and showing of credentials can still be performed by pairing these events if they are separated by a time difference of a little over τ_w. Thus, the delay τ_w should be randomised in some way. This, however, raises new questions: what are the minimum and maximum acceptable values for τ_w, given that it is likely (if not certain) to affect the system's usability? How does the choice of these limits affect unlinkability? How does the probability distribution according to which τ_w is drawn affect unlinkability?

The second attack scenario requires slightly different countermeasures. The objective here is to require subjects *not* to show credentials of some type until sufficiently many credentials of that type have been issued (to other subjects).

Again, an obvious countermeasure is to require some (randomised) delay between the issuing and showing of credentials. The requirement by verifiers to validate the timestamps of credentials, however, introduces an additional constraint, namely that, at the time of showing, credentials should not have expired.

We now describe a simple heuristic that, using random delays, tries to approach a reasonable compromise between usability, security and privacy in the face of the above timing attacks. It requires issuers to generate credentials in batches, containing a range of consecutive issue timestamps. More precisely, suppose a subject requests an issuer to provide a credential encoding a certain set of attributes α. Instead of issuing a single credential with type defined by the concatenation of α with the current timestamp $n\tau_i$, the issuer generates a set of k credentials with types defined by the concatenation of α with the timestamps $(n-j)\tau_i$, where $0 \leq j \leq k-1$, where k is a policy-based parameter (which may be chosen by the subject or by the issuer, depending on the system context).

This means, of course, that timestamps no longer precisely encode the time of issue; it is assumed, however, that this does not affect security since those credentials with 'old' timestamps are bound to expire sooner than those with current ones. It is also required that subjects maintain a loosely synchronised clock, such that they can distinguish time periods. Further, it is assumed that verifiers accept credentials that were issued during the k most recent periods (including the current one).

With respect credential showing, the subject must follow a two stage process. Note that we suppose that the set of k credentials were actually issued at time τ_{is}, where $n\tau_i \leq \tau_{is} < (n+1)\tau_i$.

1. The subject is not permitted to show any of the issued credentials until a randomised waiting time τ_w has elapsed, where $\tau_{\texttt{min}} \leq \tau_w \leq \tau_{\texttt{max}}$, and $\tau_{\texttt{min}}$ and $\tau_{\texttt{max}}$ are domain specific parameters. Clearly $\tau_{\texttt{min}}$ will depend on the rate of credential issue and showing, and should be chosen to prevent simple correlation between credential issue and showing. Further, $\tau_{\texttt{max}}$ must satisfy $\tau_{\texttt{max}} < (n+k)\tau_i - \tau_{is}$, since otherwise all the credentials may have expired before they can be used. In addition, $\tau_{\texttt{max}}$ should be large enough to sufficiently randomise the delay between issue and showing, but not so large as to damage usability. The precise choices for both parameters will be a sensitive issue, and these parameters can be subject-specific. Similarly, the probability distribution to be used to randomly select the waiting time could be varied, but it seems reasonable to use a uniform distribution.
2. The subject should then show the credential with the oldest timestamp which is still valid (according to the policy of the verifier). That is, if we assume that the credential is to be shown at time τ_{sh} (where $\tau_{sh} \approx \tau_{is} + \tau_w$), then the subject should show the credential with timestamp $(\lfloor \tau_{sh}/\tau_i \rfloor - k + 1)\tau_i$. Observing that $n\tau_i \leq \tau_{is}$ and that $0 \leq \tau_w < (n+k)\tau_i - \tau_{is}$, we have $n \leq \tau_{sh}/\tau_i < (n+k)$ and hence $n \leq \lfloor \tau_{sh}/\tau_i \rfloor \leq n+k-1$. Thus $(n-k+1)\tau_i \leq (\lfloor \tau_{sh}/\tau_i \rfloor - k + 1)\tau_i \leq n\tau_i$, and hence such a credential exists within the set of credentials that was issued to the subject.

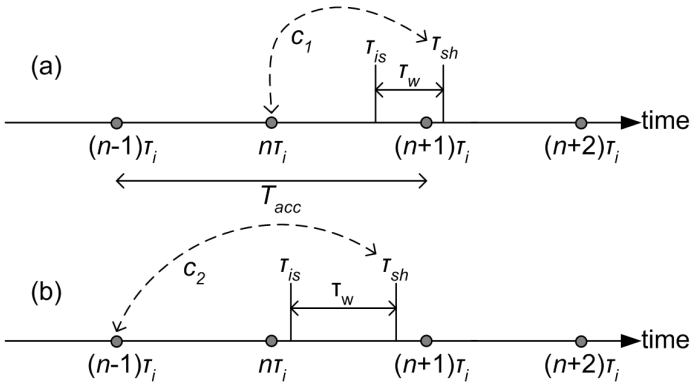

Fig. 1. Example for $k = 2$.

Figure 1 illustrates the operation of the heuristic for the case $k = 2$, i.e. in the case where the verifier accepts credentials that were issued during the last two periods (this amount of time is denoted τ_{acc} in the figure). At time τ_{is} the subject obtains two credentials: c_1 and c_2, where c_1 contains timestamp $n\tau_i$ and c_2 contains timestamp $(n-1)\tau_i$. A random waiting time t_w is then selected. In case (a) the waiting time is such that τ_{sh} falls within the next period, so credential c_1 is shown (c_2 has expired). In case (b) τ_{sh} falls within the same period, so the 'older' credential c_2 is shown.

The limit $\tau_{\mathtt{min}}$ should be selected such that, in every time period of that length, the expected number of subjects obtaining credentials of the type in question is sufficiently large. Imposing this lower limit is necessary because otherwise organisations may unambiguously link of a pair of showing and issuing events whenever some subject selects a waiting time no more than $\tau_{\mathtt{min}}$.

In some applications it may be unacceptable for a subject to obtain k credentials instead of one. A simple variation of the above heuristic does not require the subject to obtain k credentials. This involves simply choosing the waiting time τ_w first, and then only obtaining the credential that is appropriate to be shown at time $\tau_{is} + \tau_w$. This variation, however, offers significantly less protection against attacks based on correlation of credential types, since the number of issued credentials drops by a factor of k.

There are trade-offs between the choices for k and τ_i. If τ_i is relatively large then k can be made smaller, saving work in credential issue and storage. However, choosing a large value for τ_i also has disadvantages; in particular it decreases the precision available for specifying lifetimes of credentials, bearing in mind that choices for k may vary across the populations of users and organisations (depending on their privacy requirements).

We note that some cryptographic constructions allow for zero-knowledge proofs of the fact that some variable lies within a certain range, without revealing its value. Anonymous credential systems that make use of such constructions (see, for example, [1]) may overcome the timing attacks introduced

by timestamps, since subjects can convince verifiers that timestamps lie within some acceptable range, without revealing any information beyond this fact. In order to maximise unlinkability in this scenario, however, this range has to be selected carefully. Assuming that timestamps encode the time of issue, the lower limit of the range should indicate 'just before expiry', i.e. the subject should prove that the issue timestamp is at least $\tau_{sh} - \tau_{acc}$, where τ_{acc} denotes the time after which credentials expire (according to the policy of the verifier).

6 Concluding remarks

Timing issues often arise in the context of privacy preserving schemes. One proposed solution to the problem of anonymous communications is mix networks [5, 9]. A mix network enables senders to send messages to recipients in a way that preserves the anonymity of the sender. A cascade of trusted parties, called 'mixes', provides the anonymising service, i.e. it hides the identities of the senders from the recipients. Each mix is susceptible to timing attacks that try to correlate incoming and outgoing messages. A typical countermeasure requires the mix to wait for a number of incoming messages before forwarding them (in shuffled order) to the next mix in the cascade (or, in the case of the final mix, to the recipient of the message).

Mix networks rely on trusted parties. Unfortunately, such entities do not exist in anonymous credential systems. In fact, in many scenarios, relying on third parties may be undesirable. In this paper we outlined some heuristics that offer protection against timing attacks, in the context of anonymous credentials. Although they do not rely on a trusted third party, they come with a cost in terms of communication and computational overhead and a potential impact on usability.

Clearly, more work is needed to devise optimal solutions which address the threat of timing attacks on anonymous credentials. These solutions should minimise three, probably contradicting, requirements: the probability of pseudonym linking, the inconvenience introduced to subjects, and the impact on system security. In other words, they should offer reasonable tradeoffs between usability, security and privacy protection – a non-trivial task indeed.

References

1. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates — Building in Privacy.* The MIT Press, Cambridge, Massachusetts, 2000.
2. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, Berlin, 2001.

3. D. Chaum. Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In J. Seberry and J. Pieprczyk, editors, *Advances in Cryptology — AUSCRYPT '90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990, Proceedings*, number 453 in Lecture Notes in Computer Science, pages 246–264. Springer Verlag, Berlin, 1990.
4. D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, number 263 in Lecture Notes in Computer Science, pages 118–167. Springer Verlag, Berlin, 1987.
5. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
6. L. Chen. Access with pseudonyms. In E. Dawson and J. D. Golic, editors, *Cryptography: Policy and Algorithms, International Conference, Brisbane, Queensland, Australia, July 3-5, 1995, Proceedings*, number 1029 in Lecture Notes in in Computer Science, pages 232–243. Springer Verlag, Berlin, 1995.
7. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. M. Heys and C. M. Adams, editors, *Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 9-10, 1999, Proceedings*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer Verlag, Berlin, 2000.
8. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity — a proposal for terminology. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, number 2009 in Lecture Notes in in Computer Science, pages 1–9. Springer-Verlag, Berlin, 2001.
9. J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, volume 2009 of *Lecture Notes in Computer Science*, pages 10–29. Springer-Verlag, Berlin, 2001.
10. A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, Berlin, 2002.
11. S. Steinbrecher and S. Köpsell. Modelling unlinkability. In R. Dingledine, editor, *Privacy Enhancing Technologies, Third International Workshop, PET 2003, Dresden, Germany, March 26-28, 2003, Revised Papers*, volume 2760 of *Lecture Notes in Computer Science*, pages 32–47. Springer-Verlag, Berlin, 2003.