Preventing Unofficial Information Propagation

Zhengyi Le[1], Yi Ouyang[1], Yurong Xu[1], James Ford[2], and Fillia Makedon[2]

[1] Computer Science Department, Dartmouth College, Hanover, NH, 03755
[2] Heracleia Lab, Computer Science and Engineering Department
University of Texas at Arlington, Arlington, TX, 76019
{zyle,ouyang,yurong}@cs.dartmouth.edu, {jcford,makedon}@uta.edu

Abstract. Digital copies are susceptible to theft and vulnerable to leakage, copying, or manipulation. When someone (or some group), who has stolen, leaked, copied, or manipulated digital documents propagates the documents over the Internet and/or distributes those through physical distribution channels many challenges arise which document holders must overcome in order to mitigate the impact to their privacy or business. This paper focuses on the propagation problem of digital credentials, which may contain sensitive information about a credential holder. Existing work such as access control policies and the Platform for Privacy Preferences (P3P) assumes that qualified or certified credential viewers are honest and reliable. The proposed approach in this paper uses short-lived credentials based on reverse forward secure signatures to remove this assumption and mitigate the damage caused by a dishonest or honest but compromised viewer.

1 Introduction

Digital credentials are widely used in the Internet, and the issue of privacy related to release of credentials is attracting increasing attention. Digital credentials may be used with various security services, including access control, data origin authentication, and trust negotiation. They are issued and signed by Certificate Authorities (CAs) or Attribute Authorities (AAs) and often contain sensitive information about credential holders, since like the drivers licenses and other multi-purpose documents they may replace, they often bundle private information of different types together. Whether in the real world or online, it is preferable that private information should only be accessible by authorized parties: for example, a bar patron providing a drivers license as an age verification document may reasonably object to having their address and other details recorded, and a newly admitted graduate student may only want to show her history of previous grades to the graduate advisor of the department, and not all department staff and faculty. However, after you reveal your credential to other parties, your information is exposed to potential leakage.

There are two specific situations of information leakage we wish to prevent from happening: first, that others get more information from credentials than they are intended to, and second, that after others view and verify the information in credentials, they pass them to others (or their computers are compromised) and information is exposed to attackers. Existing work, such as on

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 113–125, 2007.

privacy preserving credentials [6], has addressed the first problem by altering an original certificate to cover private information and still maintain its usage. However, as soon as sensitive information is revealed, neither the credential holder nor the issuer can prevent others from propagating this information. This paper focuses on this problem. The current solutions are access policies and the Platform for Privacy Preferences (P3P). Access policies can prevent unqualified parties from viewing protected credentials; they cannot help with information leakage intentionally or unintentionally by qualified parties, as for example in the cases described in [21]. P3P provides a technical mechanism for ensuring that users can be informed about privacy policies before they release sensitive information [2]. It does not provide a technical mechanism for making sure sites act according to their policies.

This paper addresses this problem from another angle. We clearly cannot prevent the propagation of information that others have already obtained. However, we can take steps to make credential contents they propagate unofficial, in the sense that they lose the credential's basic ability to prove the correctness of its contents. For example, suppose A knows B's annual income because B showed A a credential with this information in it. If A wants to leak the information of B's annual income and prove to C she does have the correct information, A can show C the credential of B or send C a copy of the credential; once C verifies the credential, C knows B's annual income for sure. We can prevent this sequence of events by revoking the credential; in other words, if the credential is time-limited, then after an interval A cannot prove to C that her copy of B's credential is authentic. If the interval is dependent on the action of B, and B can, after using the credential once, invalidate it, then B's information liability will be more limited than before. Thus, whether A distributes B's actual or past salary information or any arbitrary made-up figure, A will be unable to prove that whatever number she has is or ever was valid.

However, if traditional public/private key pairs are used to implement the time-limited credential, a CA could be overwhelmed with generating key pairs, issuing public keys, signing credentials, and revoking. This paper approaches this problem with two schemes—interactive and non-interactive. The basis of our schemes is *Forward Secure Signature (FSS)* crytosystems. In FSS, there are a series of private keys with consecutive indexes and a single, fixed public key. Private keys evolve with a one-way function. Once the current private key is exposed, future private keys are exposed too. In the context of our problem, the FSS fixed public key can improve efficiency considerably since it avoids repeatedly issuing public/private key pairs, but forward private key updating will invalidate future credentials when private key exposure happens. Thus, in our schemes, a CA signs credentials with FSS private keys and invalidates each in reverse order when requested (interactive) or at regular intervals of time (non-interactive) so that even though the past private keys and corresponding signatures are invalid, its public key does not need to be re-signed and its future signatures are still valid. Moreover, since these credentials are time-limited, this eliminates the need for a revocation mechanismt, such as OCSP and CRLs, which is typically costly in PKI systems. This idea, while

simple, does not seem to have appeared previously. Yet, the result is a direct improvement in both the overall computational complexity, as well as the revocation efficiency, over previous traditional approaches.

A variety of electronic transactions, such as online pharmacy prescriptions, medicaid/medicare applications, and digital library browsing, run a risk of information propagation. Our schemes can establish credentials to help with this problem. Users can assure friends, business associates, and online services that the electronic information they receive is authentic and after a short system-defined time, the information is not official any longer.

The rest of the paper is organised as follows. Section 2 surveys related work. Section 3 describes two approaches that follow the outline given above. Section 4 discusses practical issues related to underlying cryptosystems and revocation. Some applications are suggested in Section 5. Section 6 compares performance against RSA. Section 7 presents conclusions and describes future work.

2 Related Work

2.1 Credentials

Trust negotiation is an important application of digital credentials. Winsborough *et al.* first introduced the notion of Automated Trust Negotiation (ATN) and an architecture for managing the exchange of credentials between two strangers for the purpose of determining bilateral trustworthiness [21]. As an example, suppose an AIDS patient is interested in a free online service and treatment. He sends a request to the server. The server checks its access policies and sends a counter request for required credentials, such as a signed prescription and citizenship. The patient reads the counter request and checks his local credential release policies, which say his citizenship is not sensitive but the disease information is and his diagnosis can be released only to HIPAA[1]certified hospitals. Then, he sents his citizen credential and a request for a HIPAA certificate to the server. If the server's policy says its HIPAA certificate is publicly accessble, the server will send it to the patient. After the patient successfully verified the certificate, he will send the diagnosis to the server and the trust negotiation succeeds. In this way, ATN involves digital credentials and uses access policies to allow only qualified parties to view sensitive credentials in a step-wise manner to build trust.

However, how to prevent the illegal propagation of released credentials has been one of the open problems in the ATN research. Researchers have designed ATN systems [7,22,23] and addressed related privacy and security issues in [6,19,20,24]. In order to work against unnecessary information leakage in trust negotiation, the authors proposed privacy preserving credentials to release minimal information [6]. The idea is that sensitive attributes in protected credentials are selectively disclosed. A sensitive attribute "A" is replaced with "$H(A|R)$",

[1] The Health Insurance Portability and Accountability Act (HIPAA) was enacted by the U.S. Congress in 1996. Its Privacy Rule took effect on April 14, 2003. `http://www.HIPAA.org`

in which R is a random number, "H" is a hash function, and "|" means concatenation. The new credential is signed by a CA. To verify it, "R" and "H" should be disclosed to the other party. This can prevent credential viewers from obtaining additional information. However, once the "R" and "H" are revealed to aother party, he can show them to someone else to prove the authenticity of the information he obtained.

Another approach, the Platform for Privacy Preferences Project (P3P) [2], defines the syntax and semantics of privacy policies, and the mechanisms for associating policies with Web resources. So, the privacy terms and conditions of web sites can be expressed in a standard format and users can retrieve and interpret this information automatically and then make a decision to release their personal information or not. However, whether and how the sites comply with the policies is outside the scope of the P3P specification.

2.2 Forward Security

The first Forward Secure Signature (FSS) scheme was designed by Bellare and Miner [4]. Their idea is based on dividing time into periods: $0, 1, 2, \ldots, T$. The public key PK is fixed, and the corresponding private key evolves every period through a one-way function: SK_0, SK_1, SK_2,..., SK_T. In period i, $(0 \leq i \leq T)$, a message M is signed by the current private key SK_i and the current time period index i. To verify the signature SIG of M, a receiver needs to use the fixed PK and the time period index i in which the message was signed. In case a private key, SK_i, is compromised, the previous signatures signed by $SK_j (0 \leq j < i)$ are still valid (see Figure 1) though the future signatures are disabled. This is because the key evolving algorithm is one-way and public, so that once an attacker successfully compromises SK_i he can easily derive the future keys but it is computationally hard to reverse the process to obtain the previous keys. So, this scheme mitigates the damage caused by the private key exposure. Following that work, many improvements and similar forward secure schemes have been published (e.g., [3,13,15,16,8,5]).

Here we give an overview of a general FSS that is the basis of our proposed schemes. FSS is a 4-tuple of poly-time algorithms (Gen, Upd, Sgn, Vrf):

- *Gen*: the key generation algorithm.

$$Gen(k, l) \rightarrow (PK, SK_0),$$

  where k and l are two security parameters, PK is the fixed public key, and SK_0 is the initial private key.
- *Upd*: the user private key update algorithm.

$$Upd(SK_i) \rightarrow SK_{i+1}.$$

  The user uses his current private key for period i to generate a new private key for the next period.
- *Sgn*: the signing algorithm.

$$Sgn(M, i, SK_i) \rightarrow \langle i, SIG \rangle,$$

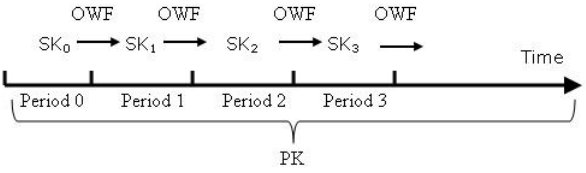

Fig. 1. Forward Secure Signature Scheme

where SIG is the signature, and i indicates the time period in which M is signed.

– Vrf: the verification algorithm.

$$Vrf(PK, M, \langle i, SIG \rangle) \rightarrow true/false.$$

If verification succeeds, Vrf outputs $true$. Otherwise, $false$.

In addition, key-insulated cryptosystems, such as [9] and [14], can provide both forward and backward security. The idea is similar to FSS, but it uses a physically secure device to store a master key to help with evolving the private key. When a user sends a request to the device to update his private key, the device generates a partial secret from its master key and then sends the partial secret to the user. The user generates the next private key from the partial secret and his current private key. Thus, the key update cannot be performed without the interaction of the secure device. However, this technology is not ready for practical use because of its efficiency.

3 Preventing Unofficial Information Propagation

Even though access policies and P3P can prevent unqualified or uncertified parties from viewing sensitive credentials, qualified or certified parties might become corrupt or be compromised by attackers so that the sensitive credentials they have viewed are at risk of being exposed. The challenge is how to disallow credential propagation by viewers. However, digital documents are well-known for being susceptible to copying, so our approach is to make credentials invalid shortly after being viewed. This mitigates the impact caused by corrupted parties or attackers. If a viewer becomes corrupt or attackers succeed after an invalidation, they will have an authentic but invalid copy of a credential. They cannot prove to other parties that the content in that credential is true since the signing key has been disabled and the signature can be forged by anyone.

There are two naïve ways to prevent this kind of unofficial information propagation: CAs can make credential expiration dates occur very soon, or credential holders can request the revocation of a credential after each usage. However, when we step further into these naïve solutions, problems arise. In the first

method, if the credential owner wants to show a credential to someone else after the expiration date, he needs to ask the CA to issue a new credential. In the second method, the CA needs to put the credential on the revocation list (or list it through OCSP responders) on request and issue a new credential for each usage. In addition, in either method, as long as the CA's signing key is still valid after the expiration date or the revocation, the invalid credential cannot be forged. This is a problem because it creates a special status for obsolete credentials that makes them distinguishable from arbitrary information. So, the problem of information leakage still occurs. In order to avoid this, the CA must change its signing key frequently. If so, either its parent CA needs to be involved in signing new public keys for the CA or the CA needs to adopt a primary key pair which is used to sign a series ephemeral key pairs for itself. When the CA uses a new key to sign a credential, PKI clients will search for a new certificate for that key pair in known repositories such as LDAP directories. This introduces a tremendous cost.

This paper gives two schemes to achieve the same goal but avoid the problems caused by these two methods. The first is an interactive approach. Credential holders request of a CA that it invalidate the credentials after each usage. The second is a non-interactive one designed for credential holders who prefer to invalidate the credentials at regular intervals of time. In this approach, the CA invalidates credentials periodically on its own. Both of them use FSS in reverse order as the cryptosystem for the CA to sign the credentials. In this way, the

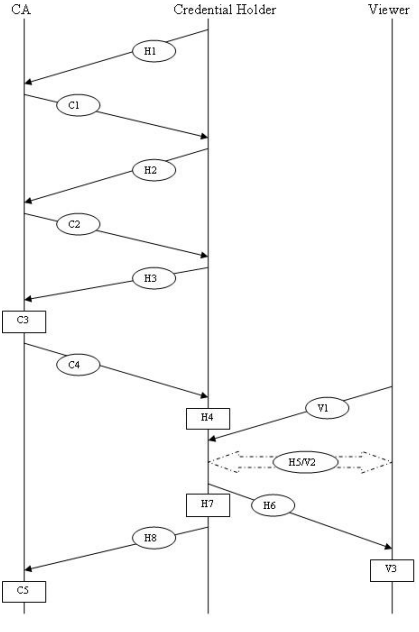

Fig. 2. Interactive Scheme

CA only has its FSS public key issued once, and each private key is invalidated before a revocation becomes necessary, so there is no need to revoke and re-issue its key pairs frequently. Further details about revocation are in Section 4.2.

The interactive scheme is given in Figure 2 and Table 1. In Figure 2, the timeline of each entity is represented by a vertical line. The communication responses and messages between each entity are represented by arrows and ellipses. Each action is defined by a rectangle. It works as follows. A user sends a request to a CA for a credential. After they agree to the type of the service, the user sends the CA the document to be signed. Then the CA verifies the contents through some online or offline methods. If they are correct, the CA signs it with multiple FSS private keys and sends the multiple FSS signatures back to the user. When the user wants to show his credential to someone else, he presents his document and the signature signed with the last FSS private key of the CA. After a grace period, he notifies the CA to disable the last FSS private key. When the CA receives the notification, it just posts the last FSS private key in a public repository. Then the key and the released signature is disabled and nobody is liable to visit or download anything from that public repository. When the user want to show his credential to another party, he sends the document and the next to last signature. Afterwards the CA disables the next to last private key.

Table 1. Interactive Scheme

CA:
$C1$: Answer the request for credential issuance with "YES" or "NO".
$C2$: If the on-request service is available and T is acceptable, send back "OK" to the credential holder. Otherwise send "DENY".
$C3$: Verify the content sent by the credential requester by on-line or off-line methods. If successful, use $SK_1, \ldots, SK_T$ to generate $SIG_1, \ldots, SIG_T$ on the document M.
$C4$: Send $SIG_1, \ldots, SIG_T$ to the credential requester.
$C5$: Release SK_{T-i} when requested for i times.
Credential Holder:
$H1$: Send the request for credential insurance to CA.
$H2$: Check with CA whether the on-request service with T is available.
$H3$: If accepted, send M, the content to be signed, to CA.
$H4$: Receive the credentials and review them.
$H5$: If there are local policies to protect its credentials, ask the viewer to provide documents to satisfy the policies.
$H6$: Send the credential and SIG_{T-i} to a viewer when he is qualified.
$H7$: Wait for a grace period of time.
$H8$: Send invalidation request to CA.
Viewer:
$V1$: Ask for the credential before providing some services or releasing sensitive credentials.
$V2$: If required, send a qualification proof.
$V3$: Receive the credential, check its validity and verify it.

Table 2. Non-Interactive Scheme

$C2'$: If the on-period service is available and $\langle t, T\rangle$ is available, send back "OK" to the credential holder. Otherwise send "DENY".
$C5'$: Release the signing keys at regular intervals of time.
$H2'$: Check with CA whether the on-period service with $\langle t, T\rangle$ is available.

In this way, the user consumes the FSS signatures in reverse order. For example, in $C3$, the CA runs $Gen(k, l) \rightarrow (PK, SK_0)$ to generate the fixed public key and the initial private key, and $Upd(SK_{j-1}) \rightarrow SK_j$ n times to generate $SK_1, \ldots, SK_n$. Then the CA runs $Sign(M, i, SK_i) \rightarrow \langle i, SIG_i\rangle$ to generate n signatures, $SIG_1, \ldots, SIG_n$, for M, the contents to be certified. T is a mutually agreed on total number of FSS signatures when the user requests this kind of service from the CA.

In the non-interactive scheme, the CA automatically releases a private key periodically as agreed. So, there is no step $H7$ or $H8$ to notify the CA, and $H2$, $C2$, and $C5$ are replaced by $H2'$, $C2'$ and $C5'$ in Table 2. T becomes the total number of time periods and t is the interval length.

Note that in either scheme, when the CA is ready to release a private key, it just posts it in a public repository and no seperate revocation mechanisms are needed. The private key is disabled even though nobody is required to visit or download that public repository. This is because making past private keys publicly accessible already invalidates the proof of its creator. Thus, verifiers do not need to be involved with any credential status checking procedures. So, this rules out the bandwidth and scability issue of PKI revocation.

4 Practical Issues

Here, we discuss practical issues that arise when applying our schemes. We will explain why we chose FSS instead of other new public key algorithms (see also Section 6, where we show that FSS is more practical than an RSA-based solution with respect to performance). Then the benefits of the time-limited certificates over traditional CRLs and OCSP will follow.

4.1 Underlying Cryptosystems

FSS was motivated by the key exposure problem, and they addressed this problem by mitigating the damage caused by key exposure. The first practical FSS scheme was constructed by Bellare and Miner and it is based on the ideas underlying the Fiat-Shamir [11] and Ong-Schnorr [18] identification and signature schemes. In order to solve our problem, frequently changing the CA's public key is not desired. So, directly applying the Fiat-Shamir scheme or the related GQ scheme [12] does not help ease the CA of its burden. However, FSS has a fixed public key, which meets our requirement. In our schemes, a CA releases FSS private keys in reverse order on purpose to make past credentials unofficial. Our

experiments in Section 6 also show that FSS with reasonable T is practical and efficient with regard to solving our problem.

Generally speaking, most FSS algorithms require the input of the total time periods, T, during the key generation phase. So, once the T is fixed, the credential holder cannot ask for more than T credentials without changing the public key when needs increase (changing the public key necessarily involves the CA). In contrast to typical FSS algorithms, KIS has no such limitation — it supports an unbounded number of periods. We therefore considered KIS as the basis for our approach. However, its computational time for a single signature is more than hours when k is greater than 16 on a representative Intel® Pentium® 4 machine with CPU 2.00GHz and 512KB cache, where 1024-bit RSA takes microseconds.

In addition, since there are variants of FSS, such as [3,13,15], we limit ourselves to general methods based on FSS so as to make our work independent of FSS progress. This should ensure that any improvement to the underlying FSS cryptography will benefit our scheme.

4.2 Revocation and Expiration

In PKI, certificate revocation is one of the hardest problems and consequently one of the major costs. Muñoz *et al.* evaluated the main revocation policies: OCSP and Overissued-CRL in their paper [17]. In general, for a high request rate, a high bandwidth is needed. Certificate Revocation Lists (CRL) ask users to download the entire list but almost 99% of revoked certificates may be irrelavant to users. In practice, many applications do not use CRLs at all since downloading a large CRL takes too much time and makes users uncomfortable when opening a file. In addition, in order to conserve bandwidth, it is common that CRLs have a validity period of one week. Such a delay can be critical.

In contrast, the Online Certificate Status Protocol (OCSP) requires verifiers to request an online responder certificate status only for specific certificates. This solves the scalability problem of CRLs. However, the responses provided by many OCSP responders are based on CRLs. OCSP itself does not provide a timely update so that the OCSP responses is as outdated as the underlying CRLs.

Short-lived certificates are proposed in order to avoid revocation issues [1,10]. In the former paper, the idea is to make certificates expire, in weeks to hours, before a revocation becomes necessary, thereby ruling out the need for a seperate revocation mechanism. The resulting increased cost of renewing certificates by a CA is not negligible. In the latter paper, in order to decrease the expense of updating short-lived certificates, the idea is that a batch of certificates can be stored in a Certificate Verification Tree (CVT) and signed ahead of time with a single CA signature. However, it is problematic when the CA uses one signing key for all subsequent short-lived certificates in the scenario of unofficial propagation. Assuming that a previous certificate was signed by the same and currently valid signing key, this certificate will remain unforgeable even when it is expired. In order to work against this unintended information leaking, the CA has to change its signing key very frequently. This requires the involvement of its parent CA or that it adopt a long-term key pair and use them to issue

ephemeral key pairs (sign public keys) for the CA itself. In contrast, in our schemes both credentials and the CA's signing keys are short-lived but there is one fixed public key for the CA and it needs to be signed only once. This releases the CA completely from the burden of signing ephemeral public keys frequently. So it solves the problem when short-lived certificates are introduced to work against unofficial information propagation. In addition, according to the results of our experiments in Section 6, our approach can also benefit the CVT-based micropayment scheme in [10] with improved efficiency.

5 Applications

Our schemes can be applied wherever digital credentials are used. As an example from the research domain, digital credentials play a key role in Automated Trust Negotiation (ATN) and the issue of how to protect privacy and deal with sensitive credentials has called for technical solutions since ATN was introduced. In industry, on-line prescription and drug purchasing systems may have stronger requirements for privacy issues. As mentioned in the previous section, a qualified credential viewer may leak the information by saving a copy of the credential he viewed and showing it to others — the viewer becomes malicious later in this case, or he remains honest but is compromised by an attacker. In either case, the effect is that locally stored copies of credentials are no longer secret. If our schemes are adapted, this kind of information leakage can be prevented: the corrupted viewer cannot prove the authenticity of the copy to others after a specified amount of time has passed; and the attacker cannot get a valid copy of the credential unless he hacks the viewer's system in time. The trade-off is that CAs and credential holders need to pay extra computational and communication costs for it. The following section will compare the performance between our schemes and the traditional approach.

6 Experiments and Performance

From Section 3 and 4, we have shown that using traditional public/private key pairs need more CA involvement to issue and invalidate short-lived certificates. This section focuses on the comparison of the computational cost: the run time

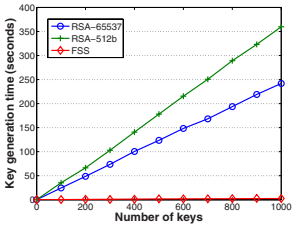

Fig. 3. Key Generation

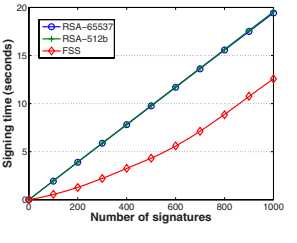

Fig. 4. Signature

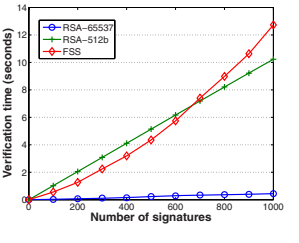

Fig. 5. Verification

of generating keys, signing documents, and verifying signatures (Figures 3 – 5). The first practical forward secure signature algorithm [4] is used for comparison. Other FSS improvements, such as [3,13,15,16], may have better performance.

The experiments are running on the same Pentium 4 platform described in Section 4.1. We set $k = 1024$ in both RSA and FSS. The performance of RSA depends not only on the length of its module but also on it length of its public key. So, we test two settings of RSA with 1024-bit module: the first is that the public key is always set to 65537, which is the default setting in OpenSSL since a common public key greater than 3 is generally considered safe; the second is that the public key is a 512-bit random prime (in the legend these cases are referred to as RSA-65537 and RSA-512b). Assuming one private key for one day, in the largest case of our experiments, the life time of FSS public key is around 3 years.

The data show RSA-65537 takes 241.84 seconds to generate 1000 public/private key pairs and RSA-512b takes 359.77 seconds while FSS takes only 2.47 seconds to generate a fixed public key and 1000 private keys. So, FSS is around 100 times faster in key generation. Figure 4 shows FSS is 1.54 times faster than RSA when signing a document. Note that RSA-65535 and RSA-512b merge together in this figure. This is because even though their public keys have different lengh, their private keys remain 1024 bits long. In Figure 5, FSS is 28.93 times slower than RSA-55537 when verifying a signature, but is not significantly different from RSA-512b. This is because verification involves a public exponent, and 65537 ($=2^{16}+1$, 17 bits) and a 512-bit prime as an exponent have a significant impact. Another effect of this is that RSA verification is much faster than its signature. If we manually set public keys longer than 512 bits, the RSA key generation and verification will be slower.

In addition, FSS takes 12.58 seconds to sign 1000 documents on a Pentium 4 machine. Thus, signing is practical for a CA (who may have a faster machine than the Pentium 4 used in our experiments). One thousand FSS verifications take 12.73 seconds, but they are calculated by different verifiers. A single FSS verification takes only 12.73 microseconds on a Pentium 4 on average. This is also practical for common users.

7 Conclusion and Future Work

Unofficial information propagation has been a prominent concern with the growing use of digital credentials. This paper presents two schemes to address this problem: an interactive scheme and a non-interactive one. In our schemes, forward secure signature cryptosystems are used in reverse order for the purpose of validating signatures so that a credential will lose its ability to prove its authenticity in a short time after it is viewed. This new approach to short-lived certificates avoids significant costs caused by traditional pair-wise signature algorithms, which are not practical because of their performance and public key issuance considerations. In order to compare, we use the first FSS algorithm for experiments and note that further performance improvements benefit our

schemes when using newer FSS versions. Our experiments show that this version of FSS is practical with a reasonable number of keys and comparable security parameters.

However, even though the proposed schemes can help with unofficial information propagation, the cooperation of the CA is required to generate multiple signatures as needed and send them to users. We are interested in the possibility of a better solution, ideally one where the CA is only required to sign once, and users can evolve a signature by themselves but cannot change the signed content. In this way, users could disable signatures as needed exclusively by themselves.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Science Foundation under award number ITR 0312629/0733673 and IDM 0308229. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Revocation Made Simpler (January 2006), `http://www.pgp.com/downloads/whitepapers`
2. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification (November 2006), `http://www.w3.org/TR/P3P11/`
3. Abdalla, M., Reyzin, L.: A New Forward-Secure Digital Signature Scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 116–129. Springer, Heidelberg (2000)
4. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
5. Bellare, M., Yee, B.: Forward-Security in Private-Key Cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003)
6. Bertino, E., Ferrari, E., Squicciarini, A.C.: Privacy-Preserving Trust Negotiation. In: Proc. of the 4th Workshop on Privacy Enhancing Technologies, pp. 283–301 (2004)
7. Bertino, E., Ferrari, E., Squicciarini, A.C.: Trust-X: A Peer-to-Peer Framework for Trust Establishment. IEEE Trans. Knowl. Data Eng. 16(7), 827–842 (2004)
8. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRPYT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
9. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Signature Schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (2002)
10. Domingo-Ferrer, J.: On the Synergy Between Certificate Verification Trees and PayTree-like Micropayments. In: Katsikas, S.K., Gritzalis, S., Lopez, J. (eds.) EuroPKI 2004. LNCS, vol. 3093, pp. 180–190. Springer, Heidelberg (2004)

11. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1986)
12. Guillou, L.C., Quisquater, J.-J.: A "Paradoxical" Indentity-Based Signature Scheme Resulting from Zero-Knowledge. In: Proc. of Advances in Cryptology - Advances in Cryptology - CRYPTO 88, 8th Annual International Cryptology Conference, pp. 216–231 (1988)
13. Itkis, G., Reyzin, L.: Forward-Secure Signatures with Optimal Signing and Verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 332–354. Springer, Heidelberg (2001)
14. Itkis, G., Reyzin, L.: SiBIR: Signer-Base Intrusion-Resilient Signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Heidelberg (2002)
15. Kozlov, A., Reyzin, L.: Forward-Secure Signatures with Fast Key Update. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 241–256. Springer, Heidelberg (2003)
16. Krawczyk, H.: Simple Forward-Secure Signatures From Any Signature Scheme. In: Proc. of the 7th ACM Conference on Computer and Communication Security, CCS 2000, pp. 108–115 (2000)
17. Muñoz, J.L., Forné, J., Castro, J.C.: Evaluation of Certificate Revocation Policies: OCSP vs. Overissued-CRL. In: Hameurlain, A., Cicchetti, R., Traunmüller, R. (eds.) DEXA 2002, pp. 511–518. IEEE Computer Society, Los Alamitos (2002)
18. Ong, H., Schnorr, C.: Fast Signature Generation with a Fiat Shamir—Like Scheme. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 432–440. Springer, Heidelberg (1991)
19. Seamons, K.E., Winslett, M., Yu, T., Yu, L., Jarvis, R.: Protecting Privacy during On-Line Trust Negotiation. In: Proceedings of the 4th Workshop on Privacy Enhancing Technologies, pp. 129–143 (2002)
20. Winsborough, W.H., Li, N.: Protecting sensitive attributes in automated trust negotiation. In: WPES 2002. Proc. of the 2002 ACM Workshop on Privacy in the Electronic Society, pp. 41–51 (2002)
21. Winsborough, W.H., Seamons, K., Jones, V.: Automated Trust Negotiation. In: DARPA Information Survivability Conference and Exposition (DISCEX 2000), 1st edn, pp. 64–73 (2000)
22. Yu, T., Winslett, M.: A Unified Scheme for Resource Protection in Automated Trust Negotiation. In: IEEE Symposium on Security and Privacy, pp. 110–122 (2003)
23. Yu, T., Winslett, M., Seamons, K.E.: Interoperable strategies in automated trust negotiation. In: Proc. of the 8th ACM Conference on Computer and Communications Security, CCS 2001, pp. 146–155 (2001)
24. Yu, T., Winslett, M., Seamons, K.E.: Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans. Inf. Syst. Secur. 6(1), 1–42 (2003)