Efficient and Fresh Certification

Irene Gassko[1], Peter S. Gemmell[2], and Philip MacKenzie[3]

[1] Bell Laboratories, Lucent Technologies
1600 Osgood Street, N. Andover, MA 01845
gassko@lucent.com
[2] Computer Science Department, University of New Mexico
Albuquerque, NM 87131 gemmell@cs.unm.edu
[3] Information Sciences Research Center, Bell Laboratories
600 Mountain Ave., Murray Hill, NJ 07974-0636
philmac@research.bell-labs.com

Abstract. Electronic commerce is becoming more and more commonplace, but security is still a major concern. To provide security, a good public-key infrastructure (PKI) is needed. However, PKIs have been slow in developing, with one of the major difficulties being the creation of certification authorities (CAs), and in particular, dealing with the problem of certificate revocation. We propose a new solution to this problem.
Our solution is based on the idea that individually signed certificates provide little information over any significant time period, given that they may be revoked. That is, after a certain amount of time, a certificate is not useful without some more recent knowledge that it has not been revoked. In all previous work, this has either been handled by off-line/on-line schemes, which require costly updates by the CA for every outstanding certificate for every update period, or by certificate revocation lists/trees.
We propose a system called EFECT (Easy Fast Efficient Certification Technique), which combines the best properties of individual certificates and certificate revocation trees. We show that EFECT allows CAs to be *more secure*, even while providing *more frequent freshness updates* for certificates, and making certification verification *extremely lightweight*. We compare EFECT to previously proposed systems, including traditional X.509 certificates and Certificate Revocation Lists (CRLs), SDSI/SPKI, Micali's Certificate Revocation System (CRS), Kocher's Certificate Revocation Trees (CRTs), and Naor and Nissim's 2-3 Certificate Revocation Trees (23CRTs). Finally, we discuss some novel qualities of EFECT that no previous solution possesses.

1 Introduction

There is an ever increasing need for public-key systems to provide security in network protocols, particularly in electronic commerce. One of the current bottlenecks is the development of a public-key infrastructure (PKI), namely, certification authorities (CAs). In this paper we consider how to implement a large CA, such as for a credit card company.

H. Imai, Y. Zheng (Eds.): PKC 2000, LNCS 1751, pp. 342–353, 2000.

The defining feature of all currently proposed CA implementations is that they have a digital certificate, consisting of a c-statement (i.e., a *certificate statement* which might include information like a person's name, a public key, and a birth date), along with the CA's signature, thus validating that c-statement. The problem with this is that authorization may be revoked at some future time. Then to maintain security, either certificates have to be issued with very short expiration periods (necessitating frequent reissuing), or there must be a mechanism for certificate revocation. There have been many recent proposals to solve the problem using one or both methods [Micali,Rivest,Kocher,NaorNissim]. We will review these in Section 2.

We propose a system called EFECT (Easy Fast Efficient Certification Technique), which allows us to eliminate structures containing certificate revocation information, such as Certificate Revocation Lists (CRLs) or Certificate Revocation Trees (CRTs) from certificate directories. Thus certificate search alone allows us to find both a certificate and information sufficient for its validation, including current status of a c-statement . This greatly simplifies secure communication. Our model combines the best properties of individual certificates and certificate revocation trees, and possesses some novel properties that no previously proposed solution offers. We show that EFECT allows CAs to be *more secure*, even while providing *more frequent freshness updates* for certificates, and making certification verification *extremely lightweight*.

The design of EFECT is relatively simple. As in previous models, it assumes the existence of *certificate directories*, untrusted databases that store information periodically released and authenticated by the CAs, and make this information available on-line. EFECT uses Merkle Trees [Merkle] like other previously proposed systems, such as CRS2 [Micali], [Kocher], and 23CRT [NaorNissim], but for the purpose of authenticating the certificates themselves, instead of simply certificate revocation information. In EFECT, these trees are called certificate verification trees (CVTs). The untrusted directories store the CVTs, and can thus provide information about the validity of certificates to inquiring parties.

In Section 5, we discuss some of the advantages of this new approach, along with direct comparisons to previously proposed systems.

2 Background

In this section we review previous work in the area of CAs and certificate revocation. In all of these schemes, the CA individually signs each c-statement, and another structure is stored by a directory and used to determine if the c-statement is still valid (i.e., still *fresh*). For concreteness, we assume daily freshness updates.

2.1 X.509 Certificates and Certificate Revocation Lists (CRL)

Certificate Revocation Lists (CRLs) used with X.509 certificates is the standard certificate revocation scheme that is currently being deployed. A certificate is

just a c-statement with serial number, issue date, and expiration date, signed by the CA. A CRL consists of a list of serial numbers of revoked certificates (that have not yet expired) along with a time stamp and some other information, signed by the CA.

In general, after a certificate's signature is verified, to find if the certificate (signed c-statement) is still valid, one would query a directory for a copy of the most recent CRL. To answer a query, the directory must send this CRL in its entirety. Note that once the expiration date of a revoked certificate is reached, its serial number can be removed from the CRL, since the certificate is obviously no longer valid.

2.2 Certificate Revocation Systems

CRS1 CRS1 is one of Micali's [Micali] schemes to solve the certificate revocation problem. It uses an off-line/on-line signature scheme [EGM] to update signatures daily with reduced computation and communication cost (compared to recomputing and redistributing completely new signatures daily). The scheme involves the use of a one-way function f. A random value y_0 is chosen, and the value $f^k(y_0)$ is stored in the certificate, where $f^k(y_0)$ is defined recursively as $f(f^{k-1}(y_0))$. Then the signature may be updated k times, where the ith update involves releasing the value $f^{k-i}(y_0)$. The most recent released value serves as a freshness proof for the certificate.

Once a certificate signature is verified, to find if the certificate is still valid, one would query a directory for a copy of the most recent update value, which can then be checked to see if it corresponds to the value in the certificate. The query communication cost is thus very short. However, the CA-to-directory communication costs are high, since the CA must send the new hash value for every outstanding certificate. Although each hash can be computed relatively quickly, the verification time is proportional to the number of updates since the certificate was issued, which severely limits the lifetime of the certificate and the rate of updates.

Note that most of the following schemes may be combined with CRS1 to allow some communication/freshness tradeoffs.

CRS2 CRS2 is another of Micali's schemes. It uses Merkle Trees to provide information about the revocation status of a certificate. Each serial number of an outstanding certificate is assigned two bits indicating whether it was issued and whether it was revoked. These bits are stored in leaves of the tree (64 certificates, or 128 bits, per leaf), and each parent node in the tree stores the hash of the children's values. Finally the root is signed by the CA.

Once a certificate signature is verified, to find if a certificate is still valid, one could query a directory for the hash values on the path from the certificate's leaf to the root and the signature on the root. Then one would check the appropriate bits at the leaf, verify the hash values, and then verify the root signature. The query communication is logarithmic in the number of certificates, plus the length

of a signature. CA-to-directory communication consists only of changed leaf values and the new root signature. All other hash values can be computed by the directory. (Although not stated by Micali, it is assumed that the signature on the root value also contains a date, so as to prevent replay attacks.)

One disadvantage of this scheme is that it provides extraneous information to the relying party, by informing it not only of revocation status of the certificate in question, but of the revocation status and existence of many neighboring certificates, which is a dangerous security policy and can facilitate many attacks.

2.3 Certificate Revocation Tree (CRT)

CRT is a scheme from Kocher [Kocher] which also uses Merkle trees, but each leaf basically consists of a single revoked certificate. Thus the CRT could be much smaller than the tree in CRS2 (depending on the number of revoked certificates).

Again, it has the features of CRS2, providing good query communication costs. However, it seems that any change might result in recomputation of the entire tree (since one new revoked certificate could change the whole structure of the tree).

2.4 Certificate Revocation 2-3 Tree (23CRT)

Naor and Nissim [NaorNissim] eliminate the problem in Kocher's CRT scheme by replacing the CRT with a 2-3 tree. Then insertion and deletions do not require changing the whole tree, but just a path (logarithmic in the size of the tree).

Notice, however, that the last two schemes are constrained to providing revocation information only, and do not allow a straightforward adaptation for cases when a certificate search has to be provided (e.g. when a CA certificate needs to be retrieved for the purpose of path construction).

2.5 SDSI/SPKI

SDSI/SPKI is a combination of the Simple Distributed Security Infrastructure of Rivest and Lampson [SDSI], and Simple Public Key Infrastructure of Ellison [SDSI]. These were suggested as a simpler alternative to X.509. Revocation (or as they call it, *reconfirmation*) is handled online.

2.6 PayTree

PayTree [JY] is not actually a CA system, but a micro-payment system which involves an entity signing the root of a Merkle tree over random-valued leaves which serve as coins. The idea is that efficiency is gained by not having to sign coins individually.

3 Our System

In their seminal paper, Diffie and Hellman [DH] suggested using a trusted directory representing a modified telephone book which stores public keys (instead of telephone numbers) for secure communications. Kohnfelder [Kohnfelder] noticed that using an on-line service to provide these certificates can create a communications bottleneck. To alleviate this problem he proposed the creation of digitally-signed directory entries which he called certificates. Now certificates could be kept in untrusted directories and retrieved upon requests from users. A big problem with this approach was the possibility of premature revocation. This led to introduction of various structures for storage of revocation information, such as CRLs and CRTs. We suggest a model that eliminates the need for keeping separate revocation data by augmenting a common index structure used for search in directories: B-trees with additional values in each node. This design allows us to use Merkle trees for certificate validation and allows us to eliminate individual signing of certificates. It turns out that the latter step leads to many advantages in our scheme.

In the next subsection we describe the EFECT model in detail.

3.1 Certification Verification Tree (CVT)

Since the most resource consuming part of the certification process are digital signatures, we abolish the tradition of individually signed certificates.

For all certificates distributed by CA, a B-tree (for concreteness you can think of it as a 2-3 tree) is constructed in the following way: each leaf is a c-statement plus the hash of that c-statement. Any types of c-statements can be used with our model, including those in the popular X.509 format or SDSI/SPKI format.

The hash values of siblings in the B-tree are hashed together to get a hash value for their parents node. Hashing continues until we get to the root node of the B-tree. The hash value of the root node (we will henceforth call it RV, following Micali's notation) plus some additional information, like the date and time, is then signed by CA. We assume collision-free hash function. The Merkle tree can be constructed incrementally (for a new CA) or all at once (for an existing CA) and can be updated incrementally on a regular (e.g., daily) basis. (With 2-3 trees, each certificate update only requires an $O(\log n)$ work for the directory, where n is the total number of certificates.)

Define a *cert-path* for a c-statement to be the path from the leaf containing the c-statement to the root, along with the hash values necessary to verify that path. (This includes the hash values for all siblings of nodes along that path.) When the user requests a certificate, she is supplied additionally with the cert-path of the c-statement, along with the signature on the RV. This information is sufficient to verify that the certificate has been valid as of the latest update.

In the next two subsections, we discuss how to handle CA key change, and historical queries. Our EFECT system handles both situations much better than all previous systems. Following that, we discuss how the EFECT system allows for many more efficiency optimizations.

CA Key Change The CA key may need to be changed due to being compromised, or possibly as a simple security measure, e.g., a move from a 2048-bit key to a 4096-bit key to improve security. In any case, our system handles the change in a reasonable way. The CA simply generates a new key pair, broadcasts the new public key using some method for authentication (perhaps publishing in the NY Times in the case of key compromise, or simply signing the new key with the old one in the case of a scheduled update), signs the current root value with the new private key, and uses the new private key for all future signatures.

In the situation of the CA private key being compromised, our system also has the advantage that creating counterfeit certificates is difficult, because for that the forger has to fake the root of the CVT, and this is much easier to notice than a stand-alone bogus certificate. In previous schemes with individually-signed certificates, compromise of the CA's private key forces the CA to revoke all existing certificates, reissue those that are still valid, and somehow supply those new certificates to their owners.

Historical Queries There are two radically different types of certificate usage: ephemeral and persistent. When establishing a connection to a web server we only care if a certificate is valid for the current session. In this case the model described above is adequate. However, if we buy a house, a non-repudiation requirement comes into play. 10 years from now we have to be able to prove that our key was valid on the date of the contract.

To allow historical queries in our system, the CA should store the roots of the CVTs (one root per update period) and the corresponding signatures. For any contract that needs persistent certificates, the cert-paths of the necessary c-statements should be stored with the contract, along with the signature on the root. In case the CA key has not changed since the contract, the historical validity of the c-statements can be verified offline.

Note that in case of a CA key change, the CA should sign all old roots with a new key. Then one can verify historical certificates by checking the current signature on the old root. This is sufficient to maintain historical technical non-repudiation.

Our solution is much more efficient than previous schemes, in which the CA or another party would essentially have to store all certificates ever issued, along with complete CRLs for every update period. In the event of a key change, it would need to resign all certificates ever issued, and all the old CRLs.

Vendor Caching for Efficiency For verifiers that check many signatures every day (like vendors or routers using certificates to establish secure communications) communication with directories can be reduced by caching the top part of the CVT. This top part together with a CA's signature can be verified once a day. Then only the lower part of the path needs to be checked to validate the certificates and only hashes need to be performed.

The savings are most prononced for OCSP. According to Visa, VisaNet, Visa's processing system, transmits more than 2,700 messages per second dur-

ing peak season. This amounts to over 64 million messages per day. One RSA signature can be computed in 1 second, which means that more than 2700 computers are needed just to verify signatures. Visa claims around 850,000,000 cards currently on the market. This means the B-tree height is at most 30. Caching the top 10 levels we obtain 20 hashes are necessary to validate the signature in EFECT. Using Rivest's figure that hashing is 10,000 faster than signing, we obtain a speedup of 500, which results in using 6 computers instead of 2700.

User Caching for Efficiency A user may reduce the need for online verification checking by storing the current cert-path for the c-statement, say on a smartcard. For instance, if a user's certificate is needed making purchases throughout a day, a user could download the cert-path in the morning and make quite a few purchases during the day without any need to contact a directory. If a vendor has not stored the RV signature for the day, then the signature would also have to be stored on the smart card for totally offline verification.

4 Advantages

Here we outline the major advantageous features of the EFECT system. (Some were discussed in the previous section.)

4.1 Efficiency

CA computation By not having to individually sign certificates, the CA performs much less computation. According to Rivest, hashing is about 10,000 times faster than signing. Since each insertion of a certificate into the Merkle tree requires about 30 new hashes, there is about a factor of 300 speedup.

Vendor computation A vendor only needs to verify one signature per day, that of the root of the CVT.

Communication Complexity Eradicating signatures from certificates significantly reduces CA-directory communication.

Suppose we deal with attribute certificates, that are short (say 100 bits or less), have short life span and are created frequently. If updates of the tree happen every minute and 100 certificates are created on average during this time providing short term access privileges that last from minutes to hours to days, say, to listen to a CD, or watch the movie, then let us see what bandwidth savings a directory will get. In an hour 100 certificates plus 1 signature will consume about 11Kbit. 100 certificates with 100 signatures will consume about 110 K bit, i.e., our scheme will provide about tenfold savings of bandwidth in this case. This means that our scheme will not be hampered even by 28Kbps connection, while any scheme that requires individual signatures will need at least 4 such links even for this simple transfer of information.

Offline shopping One can easily download the current freshness information from a directory, containing the hash values along the path from a leaf to the root and the signature of the root. This is enough information to perform certification off-line for the day, and would thus allow off-line shopping. (A similar method was pointed out in [NaorNissim].

User storage All verification information necessary is stored in the directory. This might be important in situations where a user has a device with limited available memory, such as a smart card or digital phone. This feature could also be important in preventing "chosen protocol" attacks [KSW] (see Appendix A).

4.2 Security

CA may use more secure keys Since the CA only needs to compute one signature per day, and the vendors only need to verify one signature per day, the CA can use longer (i.e., more secure) keys in its signature computation. Security is also improved since there are fewer signatures available for use in cryptanalysis.

CA may use multiple keys The root signature could be signed by multiple CA keys, stored at different locations, with a public key sharing scheme or using multiple signature schemes. These would only need to be checked if the main CA key is ever compromised. Thus it provides much stronger "backup" protection. If a signature scheme is ever broken, the resulting key compromise will be much easier to repair. With only one signature a day to compute one can apply more computationally intensive signature schemes, like Merkle's, that are sufficiently different from those currently in use so that they are not likely to be broken by advances in factoring or discrete logarithms solutions. This would be more difficult in other schemes, in which *every* certificate would need to be signed with the backup key.

Protecting CA keys We can additionally protect the CA keys by performing "the signature of the day" using a threshold scheme. Computers that participate in the scheme can be physically positioned in different locations, so that even a burglary would not expose the key. Also, some of those computers can be small and cheap and used solely for the purpose of computing the signatures. They can be offline or even turned off all the time except for the "signature" time, which will make a break in exceedingly difficult. If any of these machines is compromised, the rest can perform proactivisation [HJJKY96,FGMY] and thus make the exposure absolutely useless for the perpetrator.

Less damage from CA key compromise As discussed above, the CA only has to broadcast a new public key and sign all the old roots. In other schemes, the CA would have to resign all old certificates along with the daily certificate revocation structures.

4.3 Miscellaneous

Easy changes To change the information in a c-statement, there does not need to be a revoked certificate, and a new certificate. Instead, the information is simply changed, and the change is noted in the next day's CVT. This does not give any extra power to a CA, because under current rules CA can change data in a certificate compared to a certificate request, and is otherwise more or less omnipotent in regards to the certificates it issues.

Unidirectional communication and prescheduled updates Suppose that you go on a business trip and get a special certificate that allows you to access your computer from the laptop you take on that trip. Since a lot of laptops get stolen, you want to set expiration date in a week. However, it happens that you need to stay longer and thus need to extend the validity period of your certificate. In any scheme that requires a certificate to be signed, you will have to request the changes and to get the signature back. In our scheme it is sufficient to leave a request, say, on the answering machine of a responsible person and have your certificate extended without any need for anybody to get in touch with you. On a long trip you can schedule weekly extensions in advance, so that communication is required only in case of emergency.

Atomic Certificates Atomic certificates had been introduced recently in [atomic] for the purpose of better privacy protection. Instead of creating one "catch-all" certificate, many small pieces are certified by a CA. Then their owner can combine those pieces and create a certificate that will only contain fields corresponding to a particular task at hand. Thus, a certificate for one's bank will most probably contain the Social Security Number, while a certificate for merchant - won't. The main problem with this approach in the traditional model is that each "atomic certificate" must be signed by a CA. Our solution eliminates this bottleneck.

One-time Certificates The author of the "Atomic Certificates" draft proposes that after atomic certs are combined into one cert, the owner of that cert signs it with his secret key. We suggest that this idea can be taken one step further by adding date and time to the compendium. This will allow to use them as one-time certificates to prevent replay attacks. For example, if an agent submits queries to a secure database on behalf of its users, and this database checks user's certificates to verify their authorization to access data, a rogue agent may try to replay a certificate of a user with the top secret clearance, to obtain data for another, untrusted user. One-time certificates can be useful in preventing this sort of attack.

5 Comparison to Previous Systems

5.1 X.509 Certificates and Certificate Revocation Lists (CRL)

Our system has advantages over CRLs in that the full CRL (which could be quite lengthy with a standard revocation rate of 10%) must be sent to the verifier to

assure a certificate is still valid on a certain date. Another disadvantage is that it is difficult to support non-repudiation because expired certificates are removed from the CRL. Finally, a CRL leaks information about all revoked certificates, which may not be acceptable.

In fact, our model can be used in tandem with X.509 certificates. In this case the issuing CA signature on the certificate, which is now mandatory, can be made optional.

5.2 SDSI/SPKI

The SDSI/SPKI model requires certificates to be validated online, while our scheme allows mostly offline verification.

5.3 Certificate Revocation Trees

The 23CRT scheme of Naor and Nissim [NaorNissim] improves upon the CRT scheme of Kocher [Kocher]. In terms of communication. Our scheme is very similar to the 23CRT scheme. As shown in Naor and Nissim [NaorNissim], that scheme is much improved in terms of CA-to-directory communication over previous schemes, and better than all schemes but Micali's CRS1 scheme in terms of communication for individual queries.

Our model provides the full range of services: it allows to find certificates, prevent their forgery, supports archiving and mobile devices with limited resources, such as smartcards and mobile phones.

CRT does none of these things. CRT and 23CRT deal exclusively with revocation. However, in cases when other means are employed to obtain certificates and guarantee their validity, and only revocation information is needed, Naor and Nissim's scheme can be used, since its communication for CA-directory interaction is about 10 times less then in our scheme (assuming a 10 percent revocation rate).

5.4 Other Differences

Except for CRS2, none of the previous systems can prove certificate non-existence. This is because CRLs and CRTs provide no indication at all about existence of non-revoked certificates. In the CRS2 scheme, if the CA key is quietly compromised, a forger can create fake certificates with the same certificate numbers as real certificates and this will be almost impossible to detect. This activity is even assisted by the CRS2's propensity to notify the relying party of the status of neighboring certificates. Only our model makes forging certificates very difficult to hide.

Other new features in our model are unidirectional and prescheduled certificate modification (e.g. validity extension).

Note that only CSR1 and EFECT do not provide any information about the status of unrelated certificates.

6 Conclusion

We have described a technique whereby a company like VISA, endorsing roughly 2^{30} certificates, can provide near-current information to merchants as to which of its customers is certified and whose credit privileges have been revoked in a near-offline manner.

We have demonstrated how computation time, memory requirements, and communication complexity can be cut simultaneously by using a better data structure and better protocols. We do not need to put any additional trust into directories to achieve these dramatic savings, nor do we need to put any additional limitations on any of the parties.

Our approach also significantly reduces data available for cryptanalysis of the CA's keys, allows for better protection of those keys, as well as provides gracious CA key change in cases of the regular update, of key compromise, and even when one of the signature algorithms used is broken.

We introduce additional enhancements such as mostly offline revocation check and uni-directional certificate update.

7 Acknowledgements

The authors would like to thank unknown referees for thoughtful comments which helped to improve the paper.

References

[Anderson] Ross Anderson *Why Cryptosystems Fail*, Fairfax 93

[CRL] *Internet X.509 Public Key Infrastructure, Certificate and CRL Profile*, Internet Draft, PKIX Working Group http://www.ietf.org/internet-drafts/draft-ietf-pkix-ipki-part1-10.txt

[DH] Diffie and Hellman *New Directions in Cryptography*, IEEE Transactions on Information Theory IT-22, 6 (Nov. 1976), 644-654 346

[Kohnfelder] Loren Kohnfelder, *Towards a Practical Public-key Cryptosystem*, Bachelor's thesis, MIT, May, 1978 346

[EGM] S. Even, O. Goldreich and S. Micali, *On-Line/Off-Line Digital Signatures*, Journal of Cryptology 1996, pp. 35–67. 344

[FGMY] Y. Frankel, P. Gemmell, P. MacKenzie and M. Yung. *Proactive RSA*, Crypto 97. 349

[HJJKY96] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, *Proactive public key and signature systems*, The 4-th ACM Symp. on Comp. and Comm. Security. April 1997. 349

[JY] C. Jutla and M. Yung, *PayTree: "Amortized-Signature" for Flexible MicroPayments* Second Usenix Workshop on Electronic Commerce, 1996. 345

[KSW] J. Kelsey, B. Schneier, and D. Wagner, *Protocol Interactions and the Chosen Protocol Attack* Advances in Cryptology: CRYPTO '98. 349, 353

[Kocher] Paul Kocher, *A Quick Introduction to Certificate Revocation Trees (CRTs)* http://www.valicert.com/technology/ 343, 345, 351

[Merkle] R. Merkle, *A Certified Digital Signature,* Advances in Cryptology: CRYPTO '89, pp. 218-238. 343

[Micali] S. Micali, *Efficient Certificate Revocation*, RSA Data Security Conference, San Francisco, California, January, 1997. 343, 344

[NaorNissim] M. Naor, K. Nissim, *Certificate Revocation and Certificate Update* Proceedings of Usenix'98. 343, 345, 349, 351

[PKIX] *Internet X.509 Public Key Infrastructure, PKIX Roadmap*, Internet draft, PKIX Working Group http://www.ietf.org/internet-drafts/draft-ietf-pkix-roadmap-00.txt

[atomic] *Internet X.509 Public Key Infrastructure, ATOMIC CERTIFICATES*, Internet draft, Narayan Raghu, IBM Global Services India ltd. http://www.ietf.org/internet-drafts/draft-raghu-atomic-certificates-00.txt 350

[Rivest] *Can We Eliminate Revocation Lists?*, Proceedings of Financial Cryptography 1998. A link to this paper can be found at http://theory.lcs.mit.edu/ rivest/publications.html 343

[SDSI] links to SDSI and SPKI materials can be found at http://theory.lcs.mit.edu/ cis/sdsi.html 345

A Chosen-protocol attacks

The recent result on "chosen protocol" attacks [KSW] shows how dangerous it is to use the same certificate in two apparently unrelated applications. Thus, an owner of a smartcard (or other certificate holding device with a limited memory) may want to put on it as many certificates as can possibly fit. If the available memory is 2K and the desirable key length for CA is 2048 bits, then for every scheme that requires each certificate to be signed at least 2148 bits will be needed for a 100 bit certificates. Then no more than 7 certificates can fit on a card. By removing the signatures, 327 certificates can fit on the same card. For offline shopping, using our scheme with a CVT of height 20 and a hash of length 128 bits, we can fit more than 30 certificates on the same card: more than a 4-fold improvement in the number of certificates (compared to the standard scheme with individual signatures) plus better functionality.