A Federal PKI with Multiple Digital Signature Algorithms

W. E. Burr, NIST
W. Timothy Polk, NIST

Abstract

Several digital signature algorithms are coming into general use. A certificate containing a key for one algorithm can be signed with a different algorithm. This paper discusses the interoperability issues where different digital signature algorithms are used in one Public Key Infrastructure. The key to interoperability is client software that can validate signatures for all the algorithms used. Some rules that will simplify certification path processing are proposed.

Introduction

NIST has recently proposed [FR 97] to increase the scope of the Digital Signature Standard [FIPS 186] to allow US Federal Government use of the present Federal standard Digital Signature Algorithm (DSA) [FIPS 186], the Rivest-Shamir-Adelman (RSA) algorithm [X9.31], or the Elliptic Curve Digital Signature Algorithm (ECDSA) [X9.62] for digital signatures. However, the question then arises, can Federal users of different digital signature algorithms interoperate with each other, or will a kind of Tower of Babel situation result, where users of different algorithms are unable to validate each other's signatures? Can the emerging Federal Public Key Infrastructure (PKI) be leveraged to promote interoperability for users of different digital signature algorithms? This paper examines several possible multi-algorithm interoperability solutions and proposes a specific approach for the Federal PKI.

The three algorithms proposed for Federal use are all used in commercial products. It is likely that there will be several digital signature algorithms in common use. If public key certificates are used mainly by closed communities (as is largely the case now), then the use of different algorithms by different communities hardly matters. But that implies that individual users may have several, perhaps dozens, of certificates and public key pairs to keep and manage, hardly a desirable state of affairs, since it limits the usefulness of digital signatures. If there is to be a broad national and international PKI that citizens and businesses can use to establish their identities, sign binding documents, and conduct business with parties they have no previous relationships with, then there should also be a systematic PKI organization to accommodate several algorithms. We believe the solutions proposed for the Federal PKI could reasonably be applied in an international PKI for citizens and businesses.

Definitions

In this paper we use the following terms:

- *certificate:* A digitally signed document that binds two or more attributes together. In this paper we are only concerned with digital signature certificates that bind a subject's digital signature public key (as opposed to his key management or encryption key) to his name.
- *Certificate Revocation List (CRL):* A signed list of certificates that have been revoked;
- *Certification Authority (CA)*: A trusted entity that issues (i.e., signs and publishes) certificates and/or CRLs;
- *certification path*: a sequence of certificates beginning with a self-signed signature certificate issued by a CA trusted by a relying party and ending with an end-entity's signature certificate, where the issuer of any certificate in the sequence is the subject of the preceding certificate;
- *consistent certificate*: a certificate is considered to be consistent when the same algorithm is used for the public key certified in the certificate and to sign the certificate;
- *end-entity*: a certificate holder that is not acting as a CA. In most cases an end user with a certificate.
- *inconsistent certificate*: A certificate where the subject's algorithm for the certified key

is different than the algorithm used by the issuing CA to sign the certificate.

- *relying party*: An entity that validates a digital signature;
- *self-signed certificate*: A certificate signed with the key it certifies. It is used by a CA to state (but not authenticate) its public key;

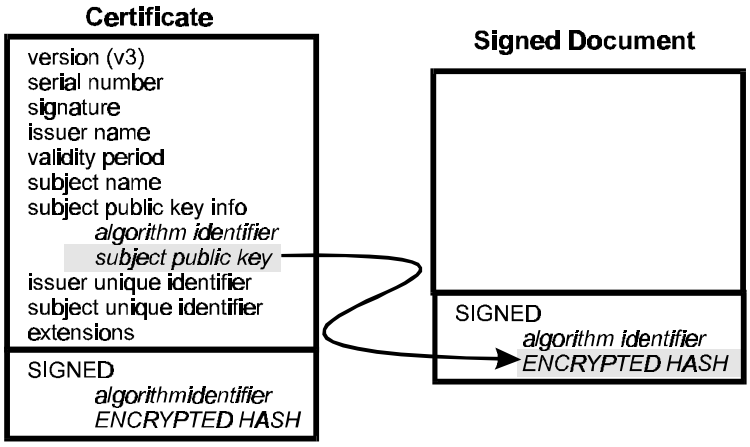

Figure 1 - Certificate and signed document

Assumptions

We assume that several digital signature algorithms will be used in the government and elsewhere and expect that different communities will standardize on different algorithms. We believe that it is relatively simple to implement signature validation for several algorithms, but more burdensome to sign with different algorithms, since this implies more keys and certificates for users to manage. Moreover, the secrecy of private keys must be strictly maintained. Most end-entities will prefer to use as few signature keys as possible and to sign with a single algorithm. Two end-entities with consistent certificates that use the same algorithm, should not ordinarily have to use any other algorithm to validate each other's certificates.

Finally, while we accept that the Federal PKI must support several digital signature algorithms, we do not believe that the same principle necessarily applies to hashing algorithms. The only standardized hashing algorithm that is now generally accepted as secure is the SHA-1 algorithm [FIPS 180]. Therefore there is no need for Federal users to use clients that support other algorithms or for the Federal PKI to issue certificates signed using other hashing algorithms.

Background

The generally accepted standard for public key certificates is the X.509 standard [X.509 97], which seems to have been embraced by most vendors of commercial products that use certificates. The most current version of the standard, which specifies the version 3 certificate and version 2 certificate revocation list (CRL) format, is apparently being widely implemented. Each certificate includes a subject public key and is signed with the Certification Authority's private key. Figure 1 illustrates how the certificate is used to obtain the subject's public key to validate his signature.

A key concept of a PKI is a *certification path*, a chain of certificates, starting from one that is trusted by the relying party, leading to the certificate of the signer. This is illustrated in Figure 2. Starting with the certificate issued by CA1, which she trusts, Alice, the relying party, can successively validate a chain of certificates leading to Bob's certificate, and then use Bob's certificate to validate his signature.

Both the subject public key field and the signature field of the certificate contain an algorithm identifier that identifies the algorithm for the subject's public key and the algorithm used to sign the certificate, respectively. The two algorithms need not be the same. Therefore a valid certification path can include "inconsistent" certificates signed using different algorithms, or certifying keys for different algorithms.

The validity of a certification path may also reflect certificate status information. A CA may choose to revoke a certificate. This information may be provided to the relying party through an on-line status check or a CRL.

A CRL is normally signed by the CA that issued the revoked certificates. The CA can sign the CRL with a different key or algorithm than used

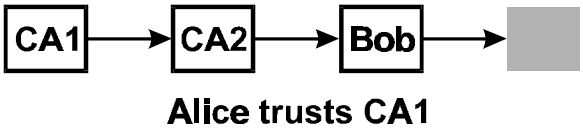

Figure 2 - Certification Path

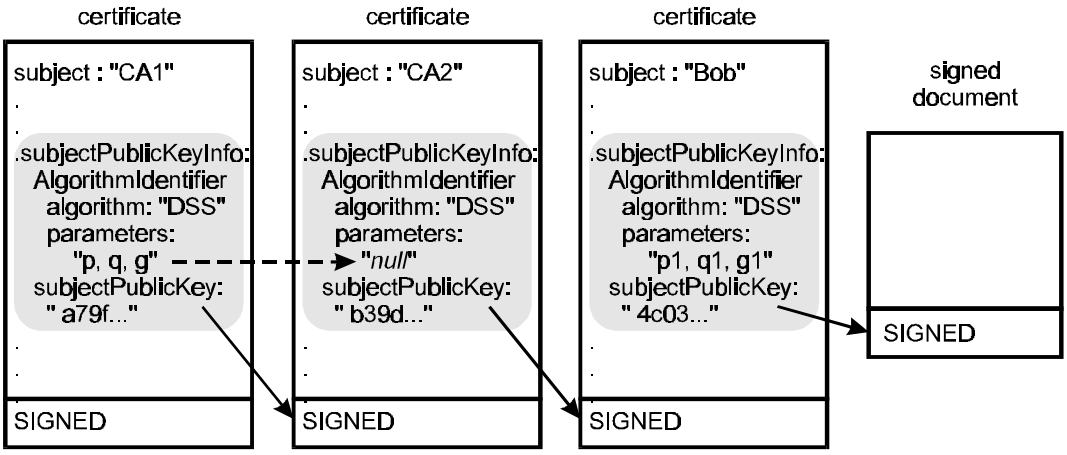

Figure 3 - Parameter Inheritance in Certification Path

to sign the certificates. So, determining the validity of a particular certificate could require use of multiple algorithms.

Some algorithms (DSA and ECDSA) require that parameters be specified. Parameters can be common to all the certificates issued by a CA, to a group of certificates, or to the entire PKI. The algorithm identifier field can (optionally) state the parameters used. Parameters are often large numbers. In the case of the DSA, two of the parameters, p and g, are the same size as the public key, between 512 and 1024 bits. If a set of parameters is shared by a community of users, it is desirable to omit parameters to reduce the size of the certificates. Therefore, the US Federal PKI Technical Working Group (TWG) has proposed a set of "parameter inheritance" rules that allows parameters to be inherited from preceding certificates in a certification path. Those rules have been incorporated in some draft standards [ISO], and are summarized as follows:

- parameters should be obtained from the same authenticated source as the public key, the subject public key field of the signer's certificate;
- if the parameters in the subject key field of the signer's certificate are null (for those algorithms requiring parameters), then the parameters are "inherited" from the preceding certificate in the certification path;
- parameter inheritance does not apply to inconsistent certificates, that is an inconsistent certificate must contain the parameters in the subject public key field, if parameters are used for the subject algorithm.

Parameter inheritance is illustrated in Figure 3. In this case CA2 inherits its parameters from the certificate of CA1, but Bob has different parameters which must be stated explicitly in his certificate.

Parallel PKI versus End-Entity Solutions

The most basic interoperability decision is, do we use inconsistent certificates at all? If not, then the only interoperability approach is parallel, independent PKIs, one for each algorithm. In this case an end-entity would need either one client that could both sign and validate every algorithm, or a separate client for each algorithm. Then the end-entity selects the appropriate algorithm, certificate, and client needed for interoperability in each case.

In this approach one party assumes the entire burden for interoperability and can sign with or validate any algorithm required. That party can, in principle, interoperate with any other party who can sign and validate signatures using any one of the algorithms for which he has a certificate.

But then which algorithm would a user use to sign any document not intended for a single specific user, whose preferred algorithm is known? Would the signer sign every document with every algorithm? Users would have multiple private keys to manage and protect. And parallel, duplicative, certification paths would be

required in the PKI itself. There may be special cases where this approach is warranted, but this solution is surely not the best general approach. It may minimize the expense for someone who never needs to interoperate with users of another algorithm, but it otherwise maximizes costs and aggravation for both the PKI and end-entity, wherever interoperation is necessary. For this reason we reject this approach.

End-Entity Solution Scenarios

For the reasons stated above, rather than a parallel-PKI approach, we recommend an "end-entity" solution where a burden is placed on all end-entities: *for interoperability we must use certification path processing software that is capable of validating all the algorithms we need to use*. However, there are simplifications for end-entities as well, because end-entities normally need use only one signing algorithm, and manage fewer private keys. Moreover, it allows considerable simplification of the PKI. At a minimum, users who wish to be broadly interoperable should use clients that can validate both RSA and DSS, and, before long, ECDSA.

Given that we adopt an end-entity solution, and will therefore have inconsistent certifications, there still remains the question of where it is best to put the inconsistent certificates needed for interoperation. There are several plausible multi-algorithm interoperability scenarios.

1. A CA signs with one algorithm, but issues end-entity certificates with subject keys for other algorithms. In our terminology, the CA issues inconsistent end-entity certificates. There might possibly be performance arguments for such a solution if we envision a CA signing algorithm that is costly to sign but inexpensive to validate, and an end-entity algorithm that is inexpensive to sign, but more expensive to validate. Since certificates are signed once, but validated many times, the cost of signing them hardly matters, but the cost of validating them may matter much more. In certain applications, end-entity signing capability may rest in devices with little computational power, so it may be important to also minimize end-entity signing computational costs. But that sort of asymmetry is hardly typical of a general purpose PKI where most end-entities have reasonably powerful PCs, workstations, and servers.

   It is, however, clear that, in a world where several signature algorithms are used, inconsistent end-entity certificates are undesirable from an interoperability point of view. Every validation of a signature signed under that inconsistent end-entity certificate will require that the relying party be able to validate signatures created using both algorithms. And the certification path created by the inconsistent end-entity is no more secure than a consistent end-entity certificate that uses the weaker of the two algorithms. Any relying party who would validate and accept the inconsistent end-entity certificate should also be able to validate and accept a consistent certificate with either of the two algorithms.

   Moreover, even a relying party who uses the same signature algorithm must be able to validate two algorithms. Something clearly is wrong when two users with certificates issued by the same CA, who use the same signature algorithm, must also validate signatures created using another algorithm to validate each other's signatures. Finally, if the end-entity algorithm uses parameters, then the parameters must be stated in the end-entity certificates, possibly making them much larger. Therefore we conclude that issuing inconsistent end-entity certificates is usually a bad idea for interoperability reasons, although there may be certain specialized applications where it is warranted for performance reasons.

2. A single CA issues consistent end-entity certificates for several algorithms, that is signs certificates with different algorithms as required to generate consistent end-entity certificates. Thus, needlessly inconsistent certification paths are avoided. The CA certifies each of its keys, with each of its other keys, with inconsistent certificates, so that certification paths exist between end-entities holding certificates with different algorithms. End-entities are typically issued a single consistent certificate and encouraged to be able to validate all the algorithms supported by the CA. This avoids inconsistent end-entity certificates, but introduces complications of its own. The

principle objection has to do with CRLs. With what algorithm does the CA sign its CRLs? Presumably it must issue separate versions of the same CRLs, signed with each of the algorithms it supports, or some relying parties may not be able to validate the signature on a CRL.

3. A single CA always signs with the same algorithm, and issues consistent end-entity certificates. The CA may, however issue inconsistent CA certificates when it certifies or cross-certifies other CAs, as needed for interoperability. If there is a need to issue end-entity certificates with different algorithms, separate CA's are created. In this case, a separate CA simply implies a different name for the CA for each algorithm, not necessarily a separate CA workstation. The consequence of this is that there is a separate CRL for each algorithm.

Proposed solution

Inconsistent end-entity certificates are generally a bad idea. Therefore we seek a solution that does not use inconsistent end-entity certificates. Parallel, entirely consistent PKI's are too expensive, and require too many end-entity certificates, leaving users with the problem of deciding which key to use to sign which documents. Solutions where a single CA signs certificates with different algorithms leads to complications with CRLs.

For the proposed solution we introduce the concept of a "logical CA." A logical CA is distinct from the hardware that implements it and the management entity that operates it. A single hardware platform could support several logical CAs, that is the platform would issue certificates with different algorithms under distinct issuer names. And a single management entity could operate any number of logical CAs.

We propose the following rules:

- End-entity certificates will be consistent;
- A logical CA will sign certificates with only one algorithm;
- Where an organization needs to issue certificates with different algorithms to its certificate holders, it will use different logical CAs to issue those certificates. This will allow CRL to be confined to certificates with a single algorithm;
- Where one management entity operates more than one logical CA for different algorithms, it will cross-certify those CAs with inconsistent certificates;
- Independent CAs will attempt to certify each other consistently, but may issue inconsistent certificates to each other as required to support the needs of their certificate holders.
- All self-signed certificates for algorithms that use parameters will include the parameters in the subject public key field;
- Other certificates will include parameters only if:
  - ◊ the certificate is an inconsistent certificate, or;
  - ◊ the parameters are different from the parameters of the issuing CA.
- Federal users will be encouraged to use client systems that can validate all Federally approved digital signature algorithms.

A Federal PKI that follows these rules will have certain desirable properties. Two end-entities certified by the same CA who use the same signature algorithm will not need to use additional algorithms to validate certification paths. In most cases, two end entities certified by different CAs who use the same signature algorithm will not need to use other algorithms to validate certification paths.[1] Each certificate will be consistent with its CRL. Finally, the number of inconsistent certificates, which may require that parameters be included in the certificates, will be minimized.

Conclusion

The use of several different digital signature algorithms appears to be a fact of life for the Federal PKI. The key to interoperability is clients that support signature validation for all the

[1] There is a special case where certification paths involving CA1 and CA3 "go through" CA2, and CA2 does not support the algorithm used by CA1 and CA3. In this case, validation of the certification path will require use of two algorithms.

Federally approved digital signature algorithms. Assuming that clients can validate all Federally approved algorithms, this paper proposes relatively simple rules for inconsistent cross-certificates between Federal CAs that will allow users who sign with one algorithm to validate signatures with other algorithms.

This paper has discussed only signature certificates. Note that other classes of end-entity certificates (e.g., key management certificates) may be inconsistent certificates. The ability to process certification paths with multiple algorithms may still be required, but other conclusions regarding the end-entity's signature certificates cannot be applied to other types of certificates.

References

[FIPS180] FIPS PUB 180-1, *Secure Hash Standard*, NIST, April 1995.

[FIPS186] FIPS PUB 186, *Digital Signature Standard*, NIST, May 1994.

[FR 97] NIST, "Announcing Plans to Revise Federal Information Processing Standard 186, Digital Signature Standard," *Federal Register*, May 13, 1997 pp. 26293-26294.

[X.509 97] ITU-T Recommendation X.509, *The Directory: Authentication Framework, June 1997.*

[X9.31] Working Draft American National Standard X9.31-199x, *Public Key Cryptography for the Financial Services Industry: The Reversible Digital Signature Algorithm*,

[X9.62] Working Draft American National Standard X9.62-199x, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm, June 21, 1996