Basics of Digital Signatures & PKI

This brief note provides a quick background to PKI-based digital signatures and an overview of how the signature creation and verification processes work. It also describes how the cryptographic keys used for creating and verifying digital signatures are managed.

1) Background to Digital Signatures

Digital signatures are basically "enciphered data" created using cryptographic algorithms. The algorithms define how the enciphered data is created for a particular document or message. Standard digital signature algorithms exist so that no one needs to create these from scratch. Digital signature algorithms were first invented in the 1970's and are based on a type of cryptography referred to as "Public Key Cryptography".

By far the most common digital signature algorithm is RSA (named after the inventors Rivest, Shamir and Adelman in 1978), by our estimates it is used in over 80% of the digital signatures being used around the world. This algorithm has been standardised (ISO, ANSI, IETF etc.) and been extensively analysed by the cryptographic research community and you can say with confidence that it has withstood the test of time, i.e. no one has been able to find an efficient way of cracking the RSA algorithm. Another more recent algorithm is ECDSA (Elliptic Curve Digital Signature Algorithm), which is likely to become popular over time.

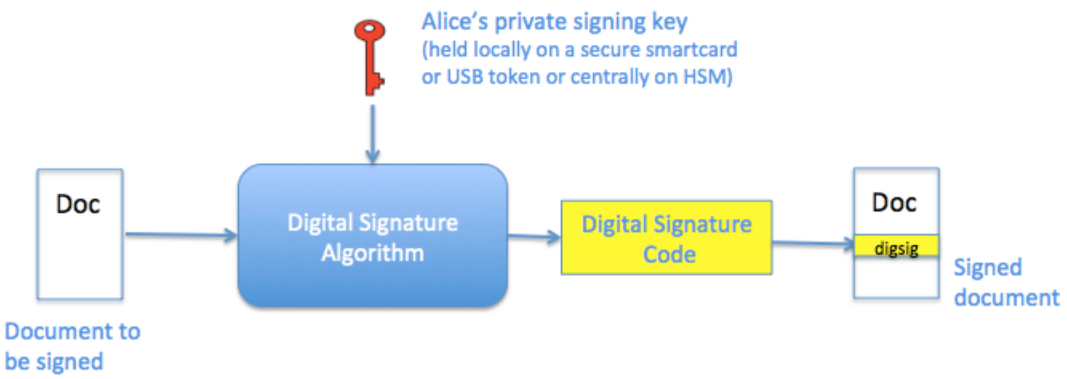
Digital signatures are used everywhere even when we are not actually aware, example uses include e.g. retail payment systems like MasterCard/Visa chip and pin, high-value interbank payment systems (CHAPS, BACS, SWIFT etc), e-passports and e-ID cards, logging on to SSL-enabled websites or connecting with corporate VPNs.

Such digital signature algorithm requires each user to have a their own private key and public key pair. These two keys are linked, i.e. they are mathematical inverses of each other, so that an operation performed with the private key can be reversed with the public key. The private key is used solely for signing purposes, whilst the public key is used for verification of signatures.

A strong digital signature algorithm like RSA, is based on the fact that knowledge of the public key does not reveal anything about the private key. This means the public key can be made public information and anyone can use it to verify digital signatures from the user. The private key remains securely held by the owning user so only they can sign with it. The following section explains the actual signing process.

2) The Digital Signature Creation Process

The following diagram illustrates the basic signing processes for user "Alice":


Alice's private signing key
(held locally on a secure smartcard
or USB token or centrally on HSM)
Doc
Document to
be signed
Digital Signature
Algorithm
Digital Signature
Code
Doc
digsig
Signed
document

At a high-level, the security of the signing process depends on ensuring only Alice has access to her private signing key and no one else. If this can be ensured to a high degree of confidence then it can be proven to an independent adjudicator that only Alice could have created the digital signature since she was the only one with access to the private signing key. This is what gives the non-repudiation property. Alice's signature on a document cannot be produced by anyone who does not have her private key. Also the digital signature cannot be copied from one document to another as its produced over a particular document's contents.

The private signing key can be protected from misuse by storing it securely within a tamper-resistant hardware device (e.g. smartcard, USB token or HSM) or keeping it in encrypted form in software. The key should only be activated for signing after successfully authenticating the owner. When using smartcards or USB tokens the user holds such a device locally and the device is PIN protected. Hence this is known as 2-factor authentication mechanism because the user must have access to the token and also know the PIN number in order to sign documents. A certain number of failed PIN entry attempts, typically 3, will lock the token from further use.

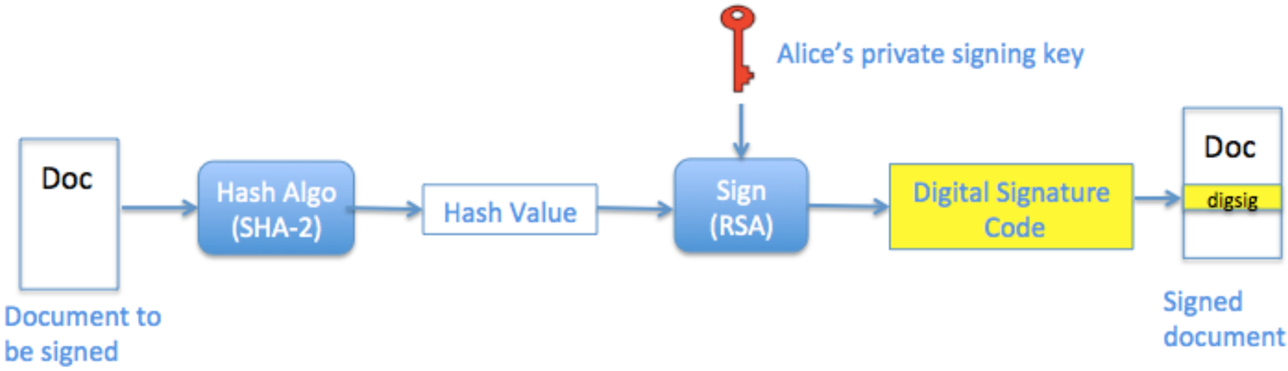
When using centrally held keys and certificates, the server application is responsible for authenticating the user before allow the signing key to be used. Again 2-factor authentication techniques can be used such as OTP (One Time Passwords) sent to user's registered mobile phone via SMS. The user must enter this OTP value back into the webpage to complete the authentication process.

As shown in the previous diagram the digital signature code is dependent not only on the signer's private key but also the input document. This means it is computationally infeasible for two different documents to produce the same digital enciphered data. In fact the algorithms used are so strong that even a 1-binary bit change to the document will produce a totally different digital signature value. Therefore digital signatures protect the document's integrity, i.e. if the document was to change accidently or intentionally even by a single character then the digital signature would no longer be valid. Also as stated the digital signature cannot be copied from one document to another.

As shown in the diagram for PDF documents the digital signature is actually embedded back into the PDF document. This is very useful for human users as it means the digital signatures do not need to be handled as separate software files.

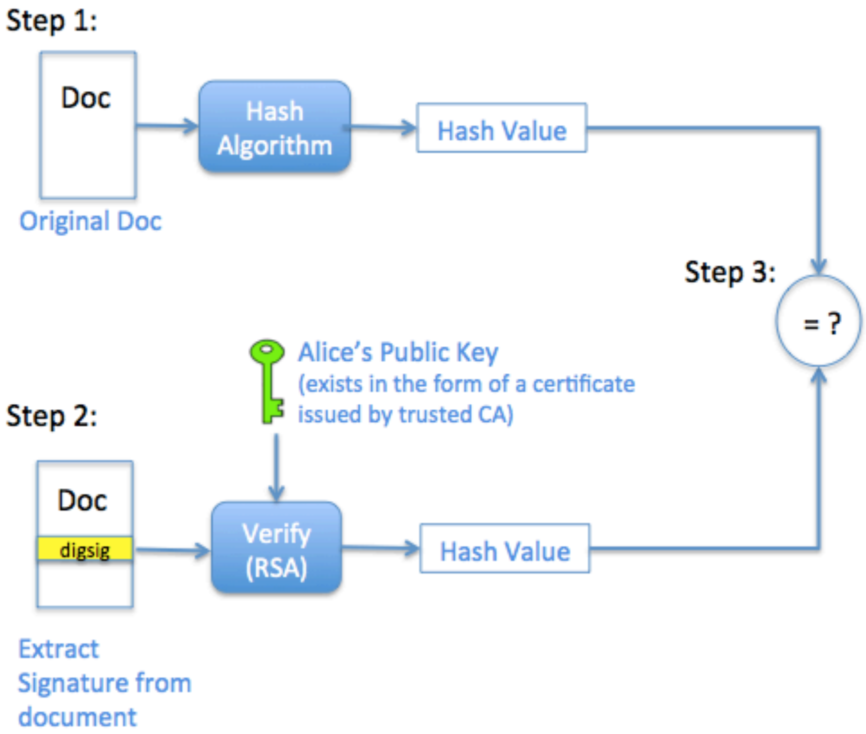
3) Digital Signature Detail

The digital signature process actually includes two steps. Firstly the document is hashed to create a small unique fingerprint of the document. The use of a strong hash algorithm (such as SHA-256 as used within Ascertia Docs) ensures that the resulting hash value is dependent on every bit of the document. This means that two documents differing by even a single bit should produce totally different hash values. It is therefore computationally infeasible to find two documents with the same hash value. The hash value is then digitally signed by encrypting the hash value with the signer's private key (a 2048 bit value in our system) using the RSA algorithm:



4) The Signature Verification Process

The following diagram illustrates the signature verification process.



The basic steps are as follows:

1) The verifier first hashes the document to get a hash value
2) The verifier then verifies the signature by using Alice's public key. This reverses the signing process and also produces a hash value upon which the digital signature was created.
3) The hash values from the above two operations are then compared. If the hash values match then it means this digital signature corresponds to that document and the digital signature could only have been produced by the person with access to Alice's private signing key – if this was adequately protected as explained in the previous section, then only Alice could have created this digital signature on this document. If the two hash values do not match then signature doesn't correspond to this document and it's considered invalid. Hence if anyone changes the document the two hashes will not be the same and this fact will be easily detected.

The security of the verification process depends on the verifier knowing with certainty that the public key indeed belongs to Alice, and not some imposter masquerading as her. To provide the trust that a particular public key belongs to a particular person is the job of Certificate Authorities (CAs).

A CA is responsible for issuing a digital certificate which binds a public key with the owner's identity. A digital certificate may contain a lot of other information as explained in the next section.

5) Digital Certificates & CAs

As explained previously a CA is an entity which vouches for that fact that a particular public key belongs to a particular person. It issues digital certificates which are data structures containing following information:

a) The name of the owner
b) The owner's public key
c) The name of the issuing CA
d) Certificate Validity Dates (Valid from and Valid to)

e) Additional optional information (e.g. what the certificate can be used for, where to check the revocation status of the certificate etc.).

The CA will digitally sign the above data structure to prevent someone from modifying it. This means the CA also has its own public and private key pair. It uses its private key to sign digital certificates and anyone with the CA's public key can verify the signature on a digital certificate and then trust the information inside the certificate. Because digital certificates are signed by the CA and therefore protected from unauthorised modification, then can be distributed via any public channel, e.g. downloaded from online repositories or actually embedded inside the signed document.

It's important the verifier has the CA's public key via a trusted means so that it can verify the signatures on the certificates issued by the CA. The CA's public key is usually embedded inside software in the form of self-signed certificates and can also be downloaded from the CA's website and other channels.

Before issuing a certificate to someone the CA must be sure of the owner's identity. E.g. it must not be possible to issue digital certificates to someone masquerading as someone else. Therefore when requesting a certificate from a CA, the end-user will be required to prove their identity. This may mean a face-to-face meeting with the CA's registration representatives and providing official government issued documentation (e.g. like passports, driving licenses etc.). Alternative the registration process may be conducted online and the CA may perform various background checks with third party databases. There are many different models for how the certificate registration could work, their purpose is always to ensure the CA has confidence in the identity of the public key owner before issuing the digital certificate which states this. Note Public CAs may offer liability for their certificates, e.g. that they have followed their stated procedures when registering users for digital certificates.

Different CAs operate at different levels of security. Within Europe the EU has defined minimum standards that the CA's must achieve in order to be recognised as Qualified CAs. These minimum standards cover the technology used by the CAs and also the physical, procedural and personnel countermeasures it puts into place to ensure its secure operations. Each EU Member State is required to maintain a list of recognised Qualified CAs in its region.

6) Certificate Status Checking

Certificates are correct when first issued by a CA, but may over time require revocation. For example, a person may lose their smartcard or USB token and therefore request their certificate to be revoked. Alternatively they may have changed employers and are no longer authorised to sign on behalf of that company.

To overcome such situations the CA has an ability to revoke certificates that it has issued. It is therefore important for the verifier at the time of verifying a digital signature to check whether the certificate is on the revoked list.

Another issue is that certificates have a finite lifetime and will eventually expire; typically this is within 1 to 3 years from the issuance date. However the issue is that digitally signed documents may need to be verified beyond the certificate's lifetime. To achieve this requires the signature to be a "long-term" digital signature.

Long-term digital signatures contain a trusted timestamp from a Time Stamp Authority (TSA) to prove the time of signing. This timestamp is digitally signed so cannot be modified by anyone. Such signatures are verified according to the time of signing (and not current time), so as long as the certificate was valid i.e. not expired or revoked at the time of signing, it doesn't matter what happens later to the certificate.

Liaquat