Digital signature

A **digital signature** or **digital signature scheme** is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering.

Digital signatures are often used to implement electronic signatures, a broader term that refers to any electronic data that carries the intent of a signature, but not all electronic signatures use digital signatures. In some countries, including the United States, India, and members of the European Union, electronic signatures have legal significance. However, laws concerning electronic signatures do not always make clear whether they are digital cryptographic signatures in the sense used here, leaving the legal definition, and so their importance, somewhat confused.

Digital signatures employ a type of asymmetric cryptography. For messages sent through a nonsecure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital signatures are equivalent to traditional handwritten signatures in many respects; properly implemented digital signatures are more difficult to forge than the handwritten type. Digital signature schemes in the sense used here are cryptographically based, and must be implemented properly to be effective. Digital signatures can also provide non-repudiation, meaning that the signer cannot successfully claim they did not sign a message, while also claiming their private key remains secret; further, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is valid nonetheless. Digitally signed messages may be anything representable as a bitstring: examples include electronic mail, contracts, or a message sent via some other cryptographic protocol.

Definition

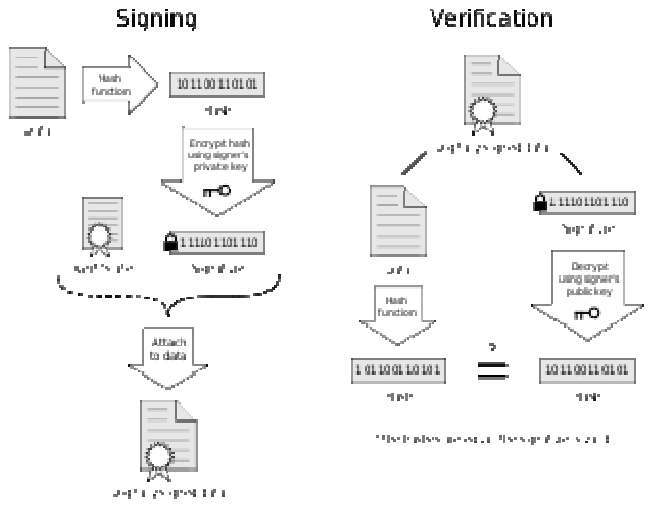

Diagram showing how a simple digital signature is applied and then verified

Main article: Public-key cryptography

A digital signature scheme typically consists of three algorithms:

- A *key generation* algorithm that selects a *private key* uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding *public key*.
- A *signing* algorithm that, given a message and a private key, produces a signature.
- A *signature verifying* algorithm that, given a message, public key and a signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, a signature generated from a fixed message and fixed private key should verify the authenticity of that message by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party who does not possess the private key.

History

In 1976, Whitfield Diffie and Martin Hellman first described the notion of a digital signature scheme, although they only conjectured that such schemes existed. Soon afterwards, Ronald Rivest, Adi Shamir, and Len Adleman invented the RSA algorithm, which could be used to produce primitive digital signatures (although only as a proof-of-concept—"plain" RSA signatures are not secure). The first widely marketed software package to offer digital signature was Lotus Notes 1.0, released in 1989, which used the RSA algorithm.

To create RSA signature keys, generate an RSA key pair containing a modulus N that is the product of two large primes, along with integers e and d such that $e\ d \equiv 1 \pmod{\varphi(N)}$, where φ is the Euler phi-function. The signer's public key consists of N and e, and the signer's secret key contains d.

To sign a message m, the signer computes $\sigma \equiv m^d \pmod{N}$. To verify, the receiver checks that $\sigma^e \equiv m \pmod{N}$.

As noted earlier, this basic scheme is not very secure. To prevent attacks, one can first apply a cryptographic hash function to the message m and then apply the RSA algorithm described above to the result. This approach can be proven secure in the so-called random oracle model.

Other digital signature schemes were soon developed after RSA, the earliest being Lamport signatures, Merkle signatures (also known as "Merkle trees" or simply "Hash trees"), and Rabin signatures.

In 1988, Shafi Goldwasser, Silvio Micali, and Ronald Rivest became the first to rigorously define the security requirements of digital signature schemes. They described a hierarchy of attack models for signature schemes, and also present the GMR signature scheme, the first that can be proven to prevent even an existential forgery against a chosen message attack.

Most early signature schemes were of a similar type: they involve the use of a trapdoor permutation, such as the RSA function, or in the case of the Rabin signature scheme, computing square modulo composite n. A trapdoor permutation family is a family of permutations, specified by a parameter that is easy to compute in the forward direction, but is difficult to compute in the reverse direction without already knowing the private key. However, for every parameter there is a "trapdoor" (private key) which when known, easily decrypts the message. Trapdoor permutations can be viewed as public-key encryption systems, where the parameter is the public key and the trapdoor is the secret key, and where encrypting corresponds to computing the forward direction of the permutation, while decrypting corresponds to the reverse direction. Trapdoor permutations can also be viewed as digital signature schemes, where computing the reverse direction with the secret key is thought of as signing, and computing the forward direction is done to verify signatures. Because of this correspondence, digital signatures are often described as based on public-key cryptosystems, where signing is equivalent to decryption and verification is equivalent to encryption, but this is not the only way digital signatures are computed.

Used directly, this type of signature scheme is vulnerable to a key-only existential forgery attack. To create a forgery, the attacker picks a random signature σ and uses the verification procedure to determine the message m corresponding to that signature. In practice, however, this type of signature is not used directly, but rather, the message to be signed is first hashed to produce a

short digest that is then signed. This forgery attack, then, only produces the hash function output that corresponds to σ, but not a message that leads to that value, which does not lead to an attack. In the random oracle model, this hash-and-decrypt form of signature is existentially unforgeable, even against a chosen-message attack.

There are several reasons to sign such a hash (or message digest) instead of the whole document.

- **For efficiency:** The signature will be much shorter and thus save time since hashing is generally much faster than signing in practice.
- **For compatibility:** Messages are typically bit strings, but some signature schemes operate on other domains (such as, in the case of RSA, numbers modulo a composite number *N*). A hash function can be used to convert an arbitrary input into the proper format.
- **For integrity:** Without the hash function, the text "to be signed" may have to be split (separated) in blocks small enough for the signature scheme to act on them directly. However, the receiver of the signed blocks is not able to recognize if all the blocks are present and in the appropriate order.

Notions of security

In their foundational paper, Goldwasser, Micali, and Rivest lay out a hierarchy of attack models against digital signatures:

1. In a *key-only* attack, the attacker is only given the public verification key.
2. In a *known message* attack, the attacker is given valid signatures for a variety of messages known by the attacker but not chosen by the attacker.
3. In an *adaptive chosen message* attack, the attacker first learns signatures on arbitrary messages of the attacker's choice.

They also describe a hierarchy of attack results:

1. A *total break* results in the recovery of the signing key.
2. A universal forgery attack results in the ability to forge signatures for any message.
3. A selective forgery attack results in a signature on a message of the adversary's choice.
4. An existential forgery merely results in some valid message/signature pair not already known to the adversary.

The strongest notion of security, therefore, is security against existential forgery under an adaptive chosen message attack.

Uses of digital signatures

As organizations move away from paper documents with ink signatures or authenticity stamps, digital signatures can provide added assurances of the evidence to provenance, identity, and

status of an electronic document as well as acknowledging informed consent and approval by a signatory. The United States Government Printing Office (GPO) publishes electronic versions of the budget, public and private laws, and congressional bills with digital signatures. Universities including Penn State, University of Chicago, and Stanford are publishing electronic student transcripts with digital signatures.

Below are some common reasons for applying a digital signature to communications:

Authentication

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

Integrity

In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to *change* an encrypted message without understanding it. (Some encryption algorithms, known as nonmalleable ones, prevent this, but others do not.) However, if a message is digitally signed, any change in the message after signature will invalidate the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions .

Non-repudiation

Non-repudiation, or more specifically *non-repudiation of origin*, is an important aspect of digital signatures. By this property an entity that has signed some information cannot at a later time deny having signed it. Similarly, access to the public key only does not enable a fraudulent party to fake a valid signature. This is in contrast to symmetric systems, where both sender and receiver share the same secret key, and thus in a dispute a third party cannot determine which entity was the true source of the information.

Additional security precautions

Putting the private key on a smart card

All public key / private key cryptosystems depend entirely on keeping the private key secret. A private key can be stored on a user's computer, and protected by a local password, but this has two disadvantages:

- the user can only sign documents on that particular computer
- the security of the private key depends entirely on the security of the computer

A more secure alternative is to store the private key on a smart card. Many smart cards are designed to be tamper-resistant (although some designs have been broken, notably by Ross Anderson and his students). In a typical digital signature implementation, the hash calculated from the document is sent to the smart card, whose CPU encrypts the hash using the stored private key of the user, and then returns the encrypted hash. Typically, a user must activate his smart card by entering a personal identification number or PIN code (thus providing two-factor authentication). It can be arranged that the private key never leaves the smart card, although this is not always implemented. If the smart card is stolen, the thief will still need the PIN code to generate a digital signature. This reduces the security of the scheme to that of the PIN system, although it still requires an attacker to possess the card. A mitigating factor is that private keys, if generated and stored on smart cards, are usually regarded as difficult to copy, and are assumed to exist in exactly one copy. Thus, the loss of the smart card may be detected by the owner and the corresponding certificate can be immediately revoked. Private keys that are protected by software only may be easier to copy, and such compromises are far more difficult to detect.

Using smart card readers with a separate keyboard

Entering a PIN code to activate the smart card commonly requires a numeric keypad. Some card readers have their own numeric keypad. This is safer than using a card reader integrated into a PC, and then entering the PIN using that computer's keyboard. Readers with a numeric keypad are meant to circumvent the eavesdropping threat where the computer might be running a keystroke logger, potentially compromising the PIN code. Specialized card readers are also less vulnerable to tampering with their software or hardware and are often EAL3 certified.

Other smart card designs

Smart card design is an active field, and there are smart card schemes which are intended to avoid these particular problems, though so far with little security proofs.

Using digital signatures only with trusted applications

One of the main differences between a digital signature and a written signature is that the user does not "see" what he signs. The user application presents a hash code to be encrypted by the digital signing algorithm using the private key. An attacker who gains control of the user's PC can possibly replace the user application with a foreign substitute, in effect replacing the user's

own communications with those of the attacker. This could allow a malicious application to trick a user into signing any document by displaying the user's original on-screen, but presenting the attacker's own documents to the signing application.

To protect against this scenario, an authentication system can be set up between the user's application (word processor, email client, etc.) and the signing application. The general idea is to provide some means for both the user app and signing app to verify each other's integrity. For example, the signing application may require all requests to come from digitally-signed binaries.