11. Digital Signature

11. 1 Requirements

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible. Consider the following disputes that could arise:

1. Bob may forge a different message and claim that it came from Alice.
2. Alice can deny sending the message. Because it is possible for Bob to forge a message, there is no way to prove that Alice did in fact send the message.

Both scenarios are of legitimate concern. Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of founds transferred and claims the larger amount had arrived from the sender. An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent.

In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature.

Suppose you want to sign an electronic document. Why can't you simply digitize your signature and append it to the document? Anyone who has access to it can simply remove the signature and add it to something else, for example, a check for a large amount of money. With classical signatures, this would require cutting the signature off the document, or photocopying it, and pasting it on the check. This would rarely pass for an acceptable signature. However, such an electronic forgery is quite easy and cannot be distinguished from the original.

There, we require that digital signatures cannot be separated from the message and attached to another. That is the signature cannot be separated from the message and attached to another. That is, the signature is not only tied to the signer but also to the message that is being signed. Also, the digital signature needs to be easily verified by other parties. Digital signature schemes therefore consist of two distinct steps: the signing process, and the verification process.

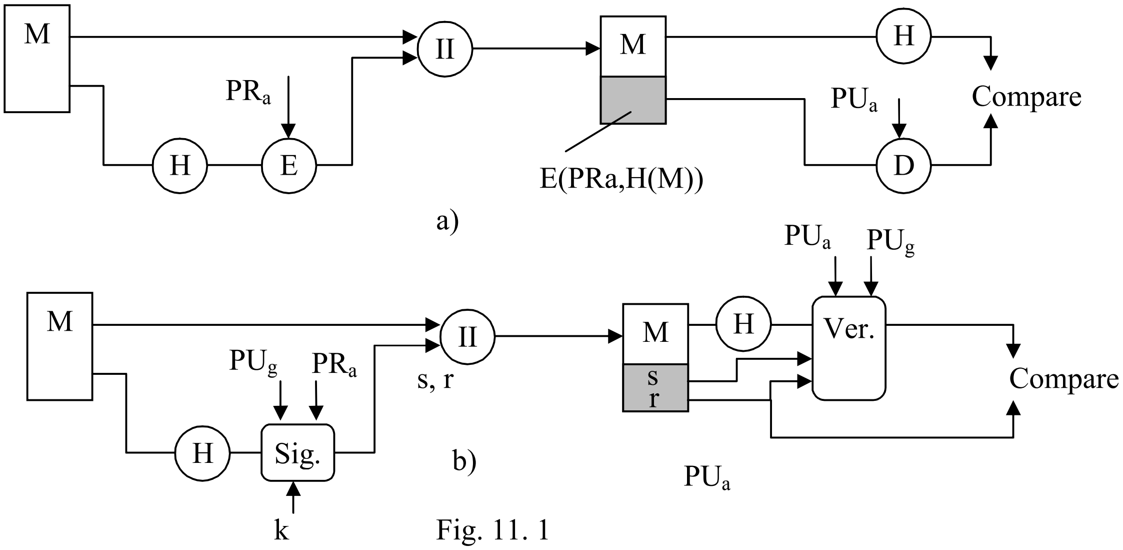
11. 2 Digital Signature Standard (DSS)

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) described in presents a new digital signature technique, the Digital Signature Algorithm (DSA). The DSS was originally proposed in 1991 and revised in 1993 was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

11. 3 The DSS Approach

The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

Figure contrasts the DSS approach for generating digital signatures to that used with RSA. In the RSA approach (Fig. 11. 1 a), the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.



a)

b)

Fig. 11. 1

The DSS approach (Fig. 11. 1 b) also makes use of a hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G). The result is a signature consisting of two components, labeled s and r.

At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key. The output of the verification function is a value that is equal to the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

We turn now to the details of the algorithms.

11. 4 RSA Signatures

Bob has a document that Alice agrees to sign. They do the following:

1. Alice generates two large prime *p, q,* and computes $n = pq$. She chooses e_A such that $1<e_A<\Phi(n)) = 1$, and calculates d_A such that $e_A\ d_A \equiv 1$ (mod $\Phi(n)$). Alice publishes ($e_{A,}$n) and keeps private d_A, p, q.

2. Alice's signature is

$$y \equiv m^{d_A} \pmod{n}.$$

3. The pair (m, y) is then made public.

Bob can then verify that Alice really signed the message by doing the following:

1. Download Alice's (e_A, n).
2. Calculate $z \equiv y^{e_A} \pmod{n}$. If z = m, then Bob accepts the signature as valid; otherwise the signature is not valid.

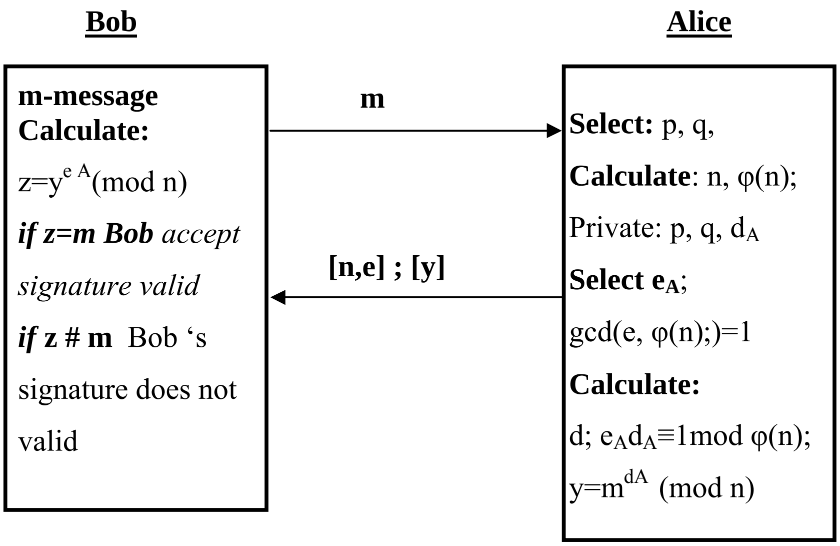


Fig. 11. 2

Suppose Eve wants to attach Alice's signature to another message m_1. She cannot simply use the pair (m_1, *y*), since $y^{e_A} \equiv m_1$ (mod n). Therefore she needs y_1 with $y_1^{e_A} \equiv m_1 \pmod{n}$. This is the same problem as decrypting an RSA "ciphertext" m_1 to obtain the "plaintext" y_1. This is believed to be hard to do.

Another possibility is that Eve choose y_1 first, then lets the message be $m_1 \equiv y_1^{e_A} \pmod{n}$. It does not appear that Alice can deny having signed the message m_1 under the present scheme. However, it is very unlikely that m_1 will be a meaningful message. It will probably be a random sequence of characters, and not a message committing her to give Eve millions of dollars. Therefore, Alice's claim that it has been forged will be believable.

There is a variation on this procedure that allows Alice to sign a document without knowing its contents. Suppose Bob has made an important discovery. He wants to record publicly what he has done (so he will have priority when it comes time to award Nobel prizes), but he does not want anyone else to know the details (so he can make a lot of money from his invention). Bob and Alice do the following. The message to be signed is m.

1. Alice chooses an RSA modulus n ($n = pq$, the product of two large primes), an encryption exponent e, and decryption exponent d. she makes n and e public while keeping p, q, d private. In fact, she can erase p, q, d from her computer's memory at the end of the signing procedure.
2. Bob chooses a random integer k (mod n) with gcd (k,n) = 1 and computes $t \equiv k^e m$ (mod n). He sends t to Alice.
3. Alice signs t by computing $s \equiv t^d$ (mod n). She returns s to Bob.
4. Bob computes s / k is the (mod n). This is the signed message m^d.

Let's show that s / k is the signed message: Note that $k^{ed} \equiv (k^e)^d \equiv k$ (mod n), since this is simply the encryption, then decryption, of k in the RSA scheme. Therefore,

$$s / k \equiv t^d / k \equiv k^{ed} m^d / k \equiv m^d \qquad (mod\ n),$$

which is the signed message.

The choice of k is random, so k^e (mod n) is the RSA encryption of a random number, and hence random. Therefore, $k^e m$ (mod n) gives essentially no information about m (however, it would not hide a message such as m = 0). In this way, Alice knows nothing about the message she is signing.

Once the signing procedure is finished, Bob has the same signed message as he would have obtained via the standard signing procedure.

There are several potential dangers with this protocol. For example, Bob could have Alice sign a promise to pay him a million dollars. Safeguards are needed to prevent such problems. We will not discuss these here.

Schemes such as these, called blind signatures, have been developed by David Chaum, who has several patents on them.

11. 5 The ElGamal Signature

The ElGamal encryption method can be modified to give a signature scheme. One feature that is different from RSA is that, with the ElGamal method, there are many different signatures that are valid for a given message.

Suppose Alice wants to sign a message (Fig. 11.2). To start, she chooses a large prime p and a primitive roo α .Alice next chooses a secret integer a such that $1 \le a \le p-2.$ and calculates $\beta \equiv \alpha^a$(mod p) .The values of p, α , and β are made public. The security of the system will be in the fact that a is kept private. It is difficult for an adversary to determine a from $(\mathrm{p}, \alpha, \beta)$ since the discrete log problem is considered difficult.

In order for Alice to sign a message m, she does the following:

1.Selects a secret random k such that gcd(k,p-1)=1

2.Computes r$\equiv \alpha^k$(mod p)

3.Computes s$\equiv k^{-1}$(m-ar) (mod p-1)

The signed message is the triple (m,r,s).

Bob can verify the signature as follows:

1.Download Alice's public key $(\mathrm{p},\alpha,\beta)$

2.Compute $v_1 \equiv \beta^r r^s$(mod p), and $v_2 \equiv \alpha^m$(mod p).

3.The signature $v_1 \equiv \beta^r r^s$(mod p), and $v_2 \equiv \alpha^m$(mod p).

4.The signature is declared valid if and only if $v_1 \equiv v_2 (\mathrm{mod}\, p)$.

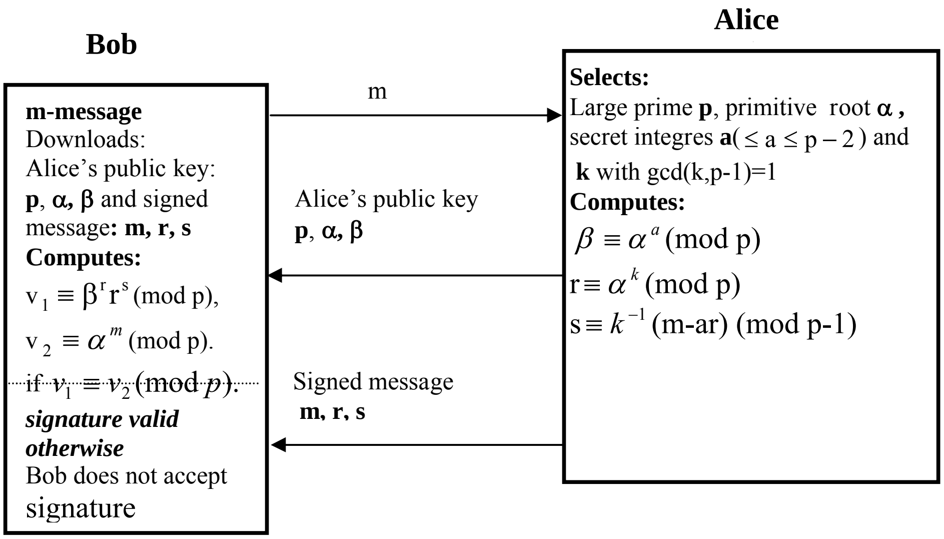

Bob
Alice
m-message
Downloads:
Alice's public key:
p, α, β and signed
message: m, r, s
Computes:
$v_1 \equiv \beta^r r^s$(mod p),
$v_2 \equiv \alpha^m$ (mod p).
if $v_1 \equiv v_2 (\mathrm{mod}\, p)$.
signature valid
otherwise
Bob does not accept
signature
m
Alice's public key
p, α, β
Signed message
m, r, s
Selects:
Large prime p, primitive root α ,
secret integres a($\le a \le p-2$) and
k with gcd(k,p-1)=1
Computes:
$\beta \equiv \alpha^a$(mod p)
r$\equiv \alpha^k$(mod p)
s$\equiv k^{-1}$(m-ar) (mod p-1)

Fig. 11.2

$v_2 \equiv \alpha^m \equiv \alpha^{sk+ar} \equiv (\alpha^a)^r(\alpha^k)^s \equiv \beta^r r^s \equiv v_1$ (mod p)

We now show that how the verification procedure works.

Assume the signature is valid. Since s$\equiv$k^{-1}(m-ar) (mod p-1),we have sk$\equiv$m-ar(mod p-1), so m$\equiv sk + ar(\mathrm{mod}\, p-1)$. Therefore (recall that a congruence mod p-1 in the exponent yields an overall congruence mod p),

$$v_2 \equiv \alpha^m \equiv \alpha^{sk+ar} \equiv (\alpha^a)^r(\alpha^k)^s \equiv \beta^r r^s \equiv v_1 \text{ (mod p).}$$

Suppose Eve discovers the value of *a*. Then she can perform the signing procedure and produce Alice's signature on any desired document. Therefore, it is very important that *a* remain secret.

If Eve has another message *m*, she can not compute the corresponding s since she doesn't know *a*. Suppose she tries to bypass this step by choosing an s that satisfies the verification equation. This means she needs s to satisfy

$$\beta^r r^s \equiv \alpha^m \text{ (mod p)}$$

This can be rearranged to $r^s \equiv \beta^{-r}\alpha^m$ (mod p), which is a discrete logarithm problem. Therefore, it should be hard to find an appropriate s. If *s* is chosen first, the equation for *r* is similar to a discrete log problem, but more complicated. It is generally assumed that it is also difficult to solve. It is not known whether there is a way to choose *r* and *s* simultaneously, through this seems to be unlikely. Therefore, the signature schemes appears to be secure, as long as discrete logs *mod p* are difficult to compute (for example, *p-1* should not be a product of small primes).

Suppose Alice wants to sign a second document. She must choose a new random value of *k*. Suppose instead that she uses the same *k* for messages m_1 and m_2. Then the same value of *r* is used in both signatures, so Eve will see that *k* has been used twice. The *s* values are different, call them s_1 and s_2.Eve knows that s_1k-m_1 ≡-ar≡s_2k-m_2 (mod p-1). Therefore,

$$(s_1\text{-}s_2)k \equiv m_1\text{-}m_2 \text{ (mod p-1)}.$$

Let d=gcd(s_1-s_2, p-1).There are d solutions to congruence, and they can be found. Usually *d* is small, so there are not very many possible values of *k*. Eve computes α^k for each possible *k* until she gets the value *r*. She now knows *k*. Eve now solves

$$ar \equiv m_1 - ks_1 \text{ (mod p-1)}$$

for a. There are gcd(r,p-1) possibilities. Eve computes α^a for each one until she obtains β, at which point she has found *a*. She now has completely broken the system and can reproduce Alice's signature at will.

Example. Alice wants to sign the message m_1=151405 (which corresponds to one, if we let 01=a,02=b,....).She chooses p=225119.Then $\alpha = 11$ is a primitive root. She has a secret number a. She computes $\beta \equiv \alpha^a \equiv 18191$ (mod p).To sign the message, she chooses a random number k and keeps it secret. She computes r≡ α^k ≡164130(mod p). Then she computes.

$$s_1 \equiv k^{-1}(m_1 - ar) \equiv 130777$$

The signed message is the triple (151405, 164130,130777).

Now suppose Alice also signs the message m_2 *=202315* (which is two) and produces the signed message (202315,164130,164899). Immediately, Eve recognizes that Alice used the same value of *k*, since the value of r is the same in both signatures. She therefore writes the congruence

$$-34122k \equiv (s_1 - s_2)k \equiv m_1 - m_2 \equiv -50910(\text{mod p-1}).$$

Since gcd(-34122,p-1)=2, there are two solutions, which can be found by the method described in Section 3.3.Divide the congruence by 2:

$$-17061k \equiv -25455 \ (\text{mod (p-1)/2}).$$

This has the solution $k \equiv 239$*(mod (p-1)/2),* so there are two values of *k (mod p)* namely 239 and 239+p(p-1)/2=112798.Calculate

$$\alpha^{239} \equiv 164130, \quad \alpha^{112798} \equiv 59924 \ (\text{mod p}).$$

Since the first is the correct value of *r,* Eve concludes that k=239. She now rewrites $s_1 k \equiv m_1 - ar$ (mod p-1) to obtain 164130a$\equiv ra \equiv m_1 - s_1 k \equiv 187104$ (mod p-1).

Since *gcd(164130,p-1)=2*, there are two solutions, namely *a=28862* an *a=141421*, which cab be found by the method of Section 3.3 .Eve computes

$$\alpha^{28862} \equiv 206928, \quad \alpha^{141421} \equiv 18191 \ (\text{mod p}).$$

Since the second value is β, she has found that a=141421.

Now that Eve knows *a*, she can forge Alice's signature on any document.

The Elgamal signature scheme is an example of a **signature with appendix.** The message is not easily recovered from the signature*(r,s).*The message m must be included in the verification procedure. This is in contrast to the **RSA** signature scheme, which is a **message recovery scheme.** In this case, the message is readily obtained from the signature *y*. Therefore, only y needs to be sent since anyone can deduce m as y^{eA} (mod n).It is unlikely that a random y will yield a meaningful message m, so there is little danger that someone can successfully replace a valid with a forged message by changing y.