Digital Signatures

G.P. Biswas
Prof/CSE, IIT, Dhanbad

Outline

- Principle of digital signatures
- RSA digital signature scheme
- ElGamal digital signature scheme
- One Standard: **Digital Signature Algorithm (DSA)**

Introduction

- A person is used to sign a document to confirm that the document is originated/approved by him.
- How will Bob ensure that the message comes from Alice, not from Eve when Alice sends a message to Bob?
- In practice, Bob can ask Alice to sign the message cryptographically.
- In other words, an digital signature can authenticate Alice as a valid sender of the message.

Contd…

- Suppose you want to sign a 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 normal signatures, this would require cutting the signature off the document, or photocopying it, and posting it on the check.
- This would rarely pass for an acceptable signature. However, such a forgery is quite easy and cannot be distinguished from the original.
- Therefore, we require digital signatures that cannot be separated from the message and attached to another.

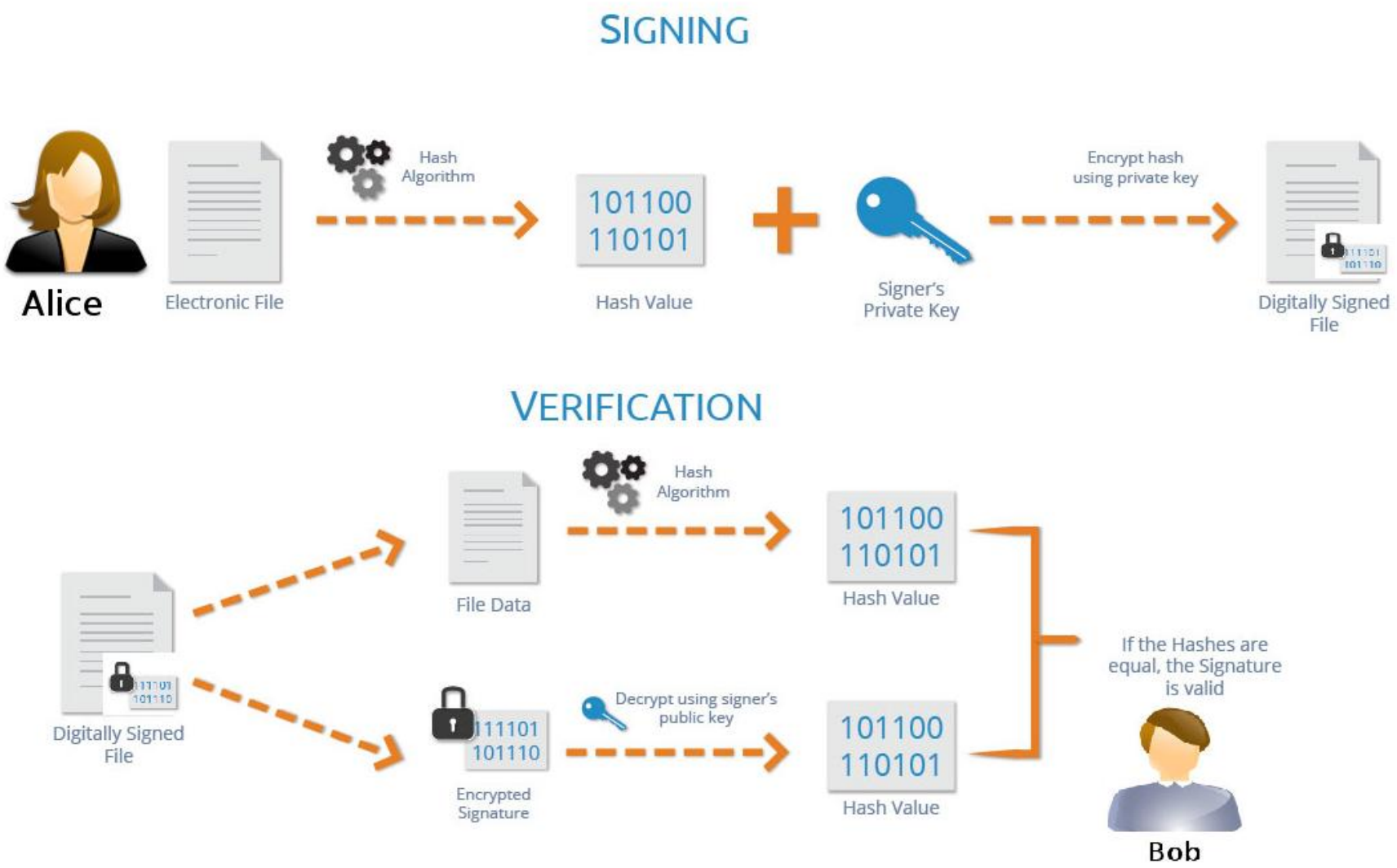
Contd…

- 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**.

Basic Requirements

- Easy for the signer to sign
- Easy for anyone to verify a message
- Hard for anyone to forge a digital signature

Working Procedure



Comparison

Digital Signature	Public Key Encryption
Only the holder of a secret can digitally sign data	**Anyone** can encrypt data
Anyone can verify that a digital signature is valid	**Only the holder** of a secret can decrypt the encrypted data

Security Services

- Digital signatures provide the following security services:
  - **Integrity:** Messages have not been modified in transit.
  - **Message Authentication:** The sender of a message is authentic.
  - **Nonrepudiation:** The sender of a message can not deny the creation of the message.

RSA Signature Scheme

Alice

$k_{pr} = d, k_{pub} = (n, e)$

(n,e) →

compute signature:
$s = \text{sig}_{k_{pr}}(x) \equiv x^d \bmod n$

(x,s) →

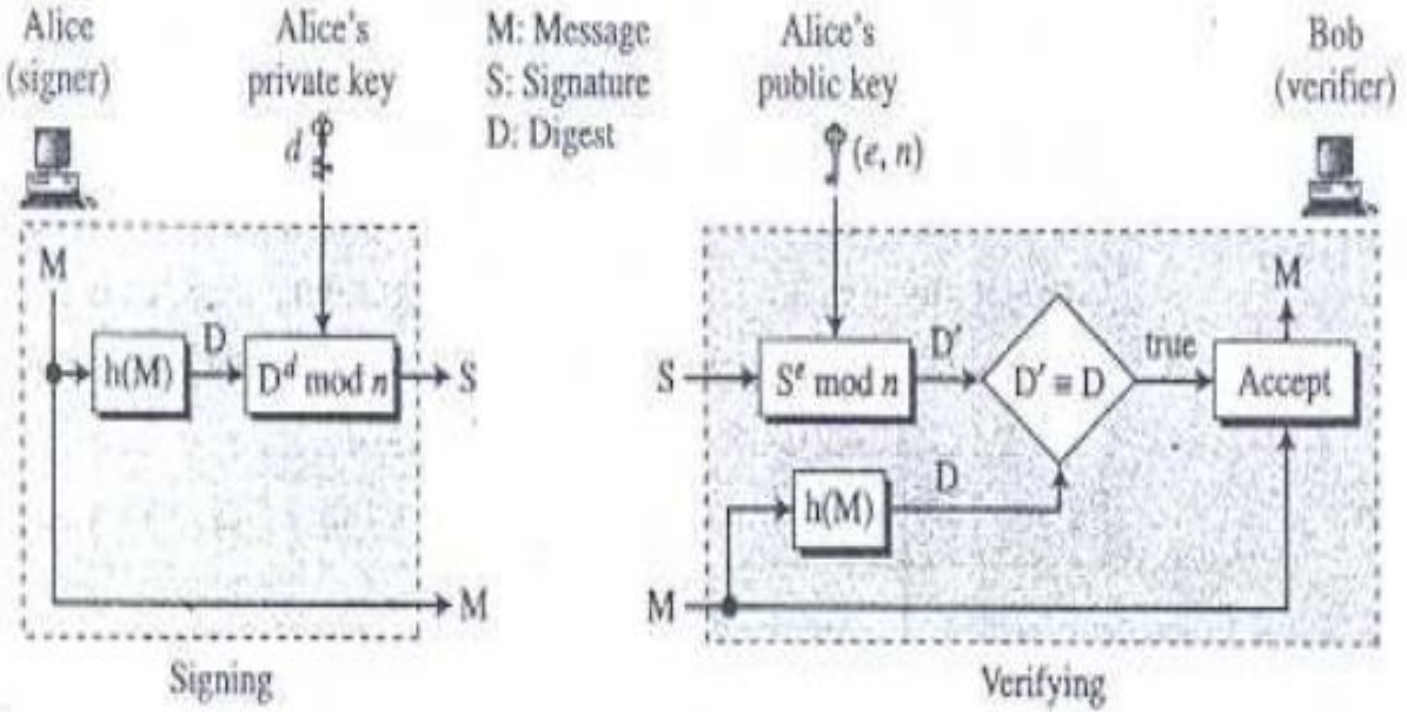
Bob

verify: $\text{ver}_{k_{pub}}(x, s)$
$x' \equiv s^e \bmod n$

$$x' \begin{cases} \equiv x \bmod n & \Longrightarrow \text{valid signature} \\ \not\equiv x \bmod n & \Longrightarrow \text{invalid signature} \end{cases}$$

RSA Signature

Figure 13.8 *The RSA signature on the message digest*



Attacks on RSA Signature

Attacks on RSA Signed Digests

How susceptible to attack is the RSA digital signature scheme when the digest is signed?

Key-Only Attack We can have three cases of this attack:

a. Eve intercepts the pair (S, M) and tries to find another message M′ that creates the same digest, h(M) = h(M′). As we learned in Chapter 11, if the hash algorithm is *second preimage resistant*, this attack is very difficult.

b. Eve finds two messages M and M′ such that h(M) = h(M′). She lures Alice to sign h(M) to find S. Now Eve has a pair (M′, S) which passes the verifying test, but it is the forgery. We learned in Chapter 11 that if the hash algorithm is *collision resistant*, this attack is very difficult.

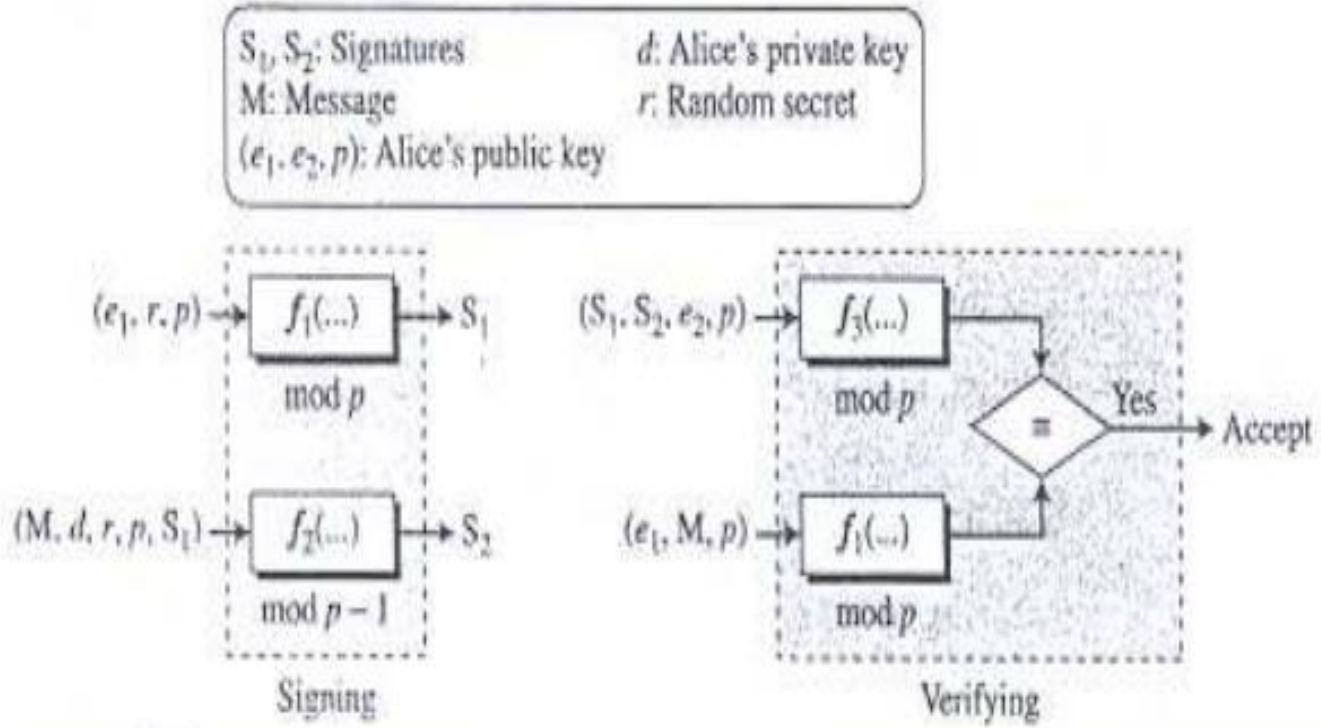
c. Eve may randomly find message digest D, which may match with a random signature S. She then finds a message M such that D = h(M). As we learned in Chapter 11, if the hash function is *preimage resistant*, this attack is very difficult to launch.

Known-Message Attack Let us assume Eve has two message-signature pairs (M_1, S_1) and (M_2, S_2) which have been created using the same private key. Eve calculates $S \equiv S_1 \times S_2$. If she can find a message M such that $h(M) \equiv h(M_1) \times h(M_2)$, she has forged a new message. However, finding M given h(M) is very difficult if the hash algorithm is *preimage resistant*.

Chosen-Message Attack Eve can ask Alice to sign two legitimate messages M_1 and M_2 for her. Eve then creates a new signature $S \equiv S_1 \times S_2$. Since Eve can calculate $h(M) \equiv h(M_1) \times h(M_2)$, if she can find a message M given h(M), the new message is a forgery. However, finding M given h(M) is very difficult if the hash algorithm is *preimage resistant*.

ElGamal Digital Signature Scheme (Idea)

Figure 13.9 *General idea behind the ElGamal digital signature scheme*



ElGamal Signature Scheme

- Alice's public key: (p, α, β) where $\beta \equiv \alpha^a \bmod p$ and private key: a
- 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 $r \equiv \alpha^k \pmod{p}$
3. Computes $s \equiv k^{-1}(m - ar) \pmod{p-1}$

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

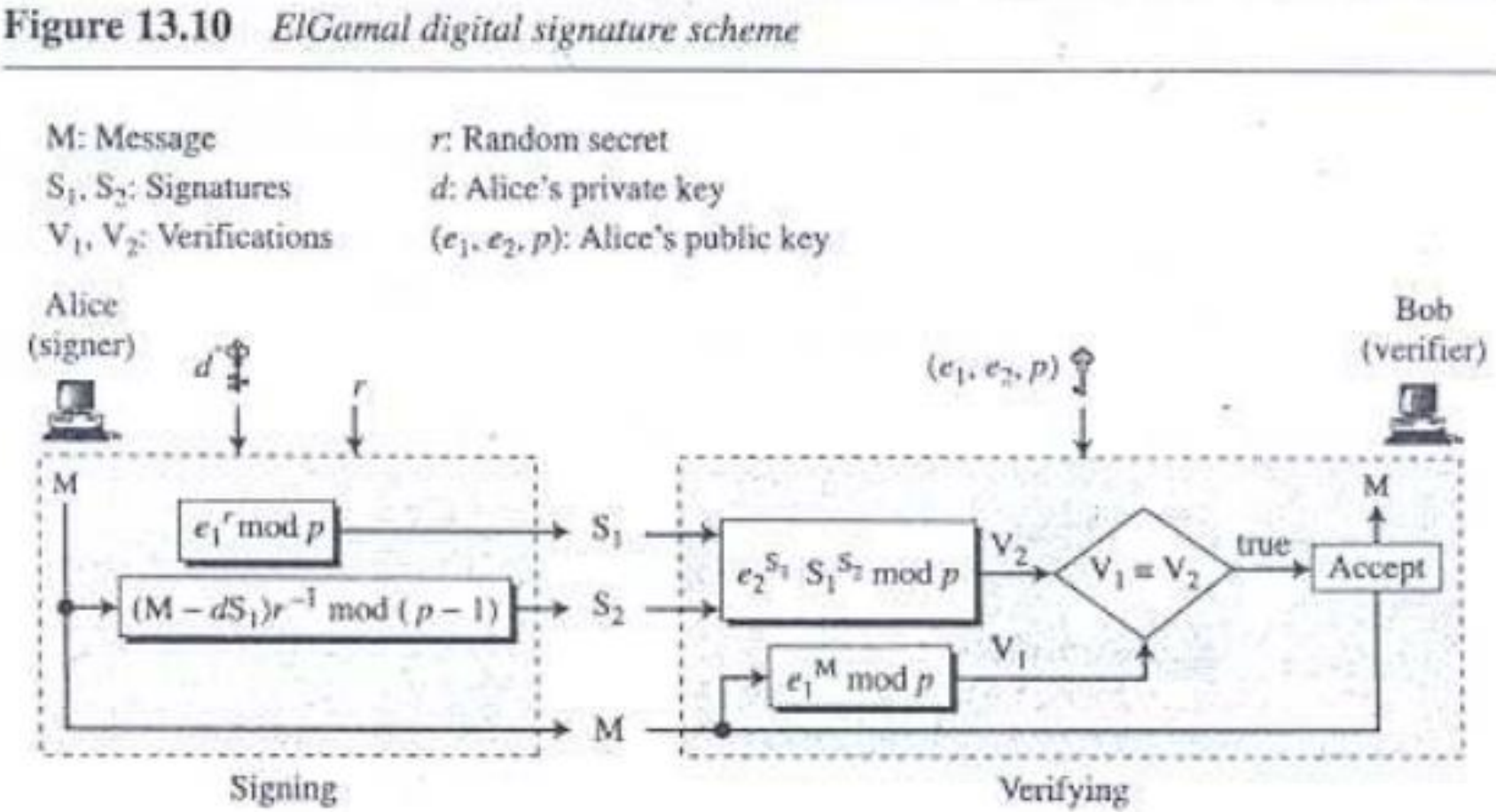
Bob can verify the signature as follows:

1. Download Alice's public key (p, α, β).
2. Compute $v_1 \equiv \beta^r r^s \pmod{p}$, and $v_2 \equiv \alpha^m \pmod{p}$.
3. The signature is declared valid if and only if $v_1 \equiv v_2 \pmod{p}$.

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

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

Figure 13.10 *ElGamal digital signature scheme*



ElGamal Signature Example

Here is a trivial example. Alice chooses $p = 3119$, $e_1 = 2$, $d = 127$ and calculates $e_2 = 2^{127} \bmod 3119 = 1702$. She also chooses r to be 307. She announces e_1, e_2, and p publicly; she keeps d secret. The following shows how Alice can sign a message.

$$M = 320$$

$$S_1 = e_1^{\,r} = 2^{307} = 2083 \bmod 3119$$

$$S_2 = (M - d \times S_1) \times r^{-1} = (320 - 127 \times 2083) \times 307^{-1} = 2105 \bmod 3118$$

Alice sends M, S_1, and S_2 to Bob. Bob uses the public key to calculate V_1 and V_2.

$$V_1 = e_1^{\,M} = 2^{320} = 3006 \bmod 3119$$

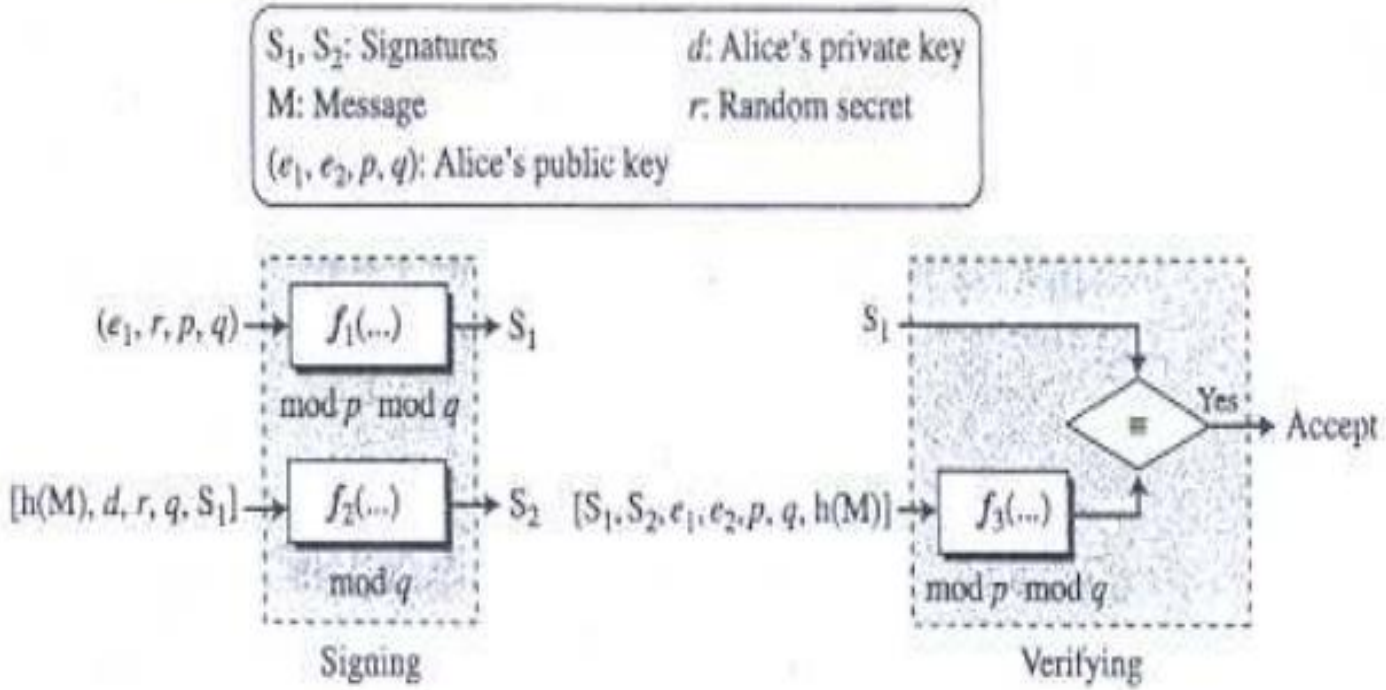
$$V_2 = d^{S_1} \times S_1^{\,S_2} = 1702^{2083} \times 2083^{2105} = 3006 \bmod 3119$$

Because V_1 and V_2 are congruent, Bob accepts the message and he assumes that the message has been signed by Alice because no one else has Alice's private key, d.

Digital Signature Algorithm (DSA)

- The National Institute of Standards and Technology (NIST) proposed the Digital Signature Algorithm (DSA) in 1991 and adopted it as a standard in 1994.
- A message digest is signed to perform the signing process faster.

Figure 13.13 *General idea behind DSS scheme*



Initialization Phase

1. Alice finds a prime q that is 160 bits long and chooses a prime p that satisfies $q|p-1$ (see Exercise 9). The discrete log problem should be hard for this choice of p. (In the initial version, p had 512 bits. Later versions of the algorithm allow for longer primes.)

2. Let g be a primitive root mod p and let $\alpha \equiv g^{(p-1)/q} \pmod{p}$. Then $\alpha^q \equiv 1 \pmod{p}$.

3. Alice chooses a secret a such that $1 \leq a < q-1$ and calculates $\beta \equiv \alpha^a \pmod{p}$.

4. Alice publishes (p, q, α, β) and keeps a secret.

Signing Phase

Alice signs a message m by the following procedure:

1. Select a random, secret integer k such that $0 < k < q - 1$.
2. Compute $r = (\alpha^k \pmod p) \pmod q$.
3. Compute $s \equiv k^{-1}(m + ar) \pmod q$.
4. Alice's signature for m is (r, s), which she sends to Bob along with m.

Verifying Phase

For Bob to verify, he must

1. Download Alice's public information (p, q, α, β).
2. Compute $u_1 \equiv s^{-1}m \pmod{q}$, and $u_2 \equiv s^{-1}r \pmod{q}$.
3. Compute $v = (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$.
4. Accept the signature if and only if $v = r$.

Verification Process

$$m \equiv (-ar + ks) \pmod{q},$$

which implies

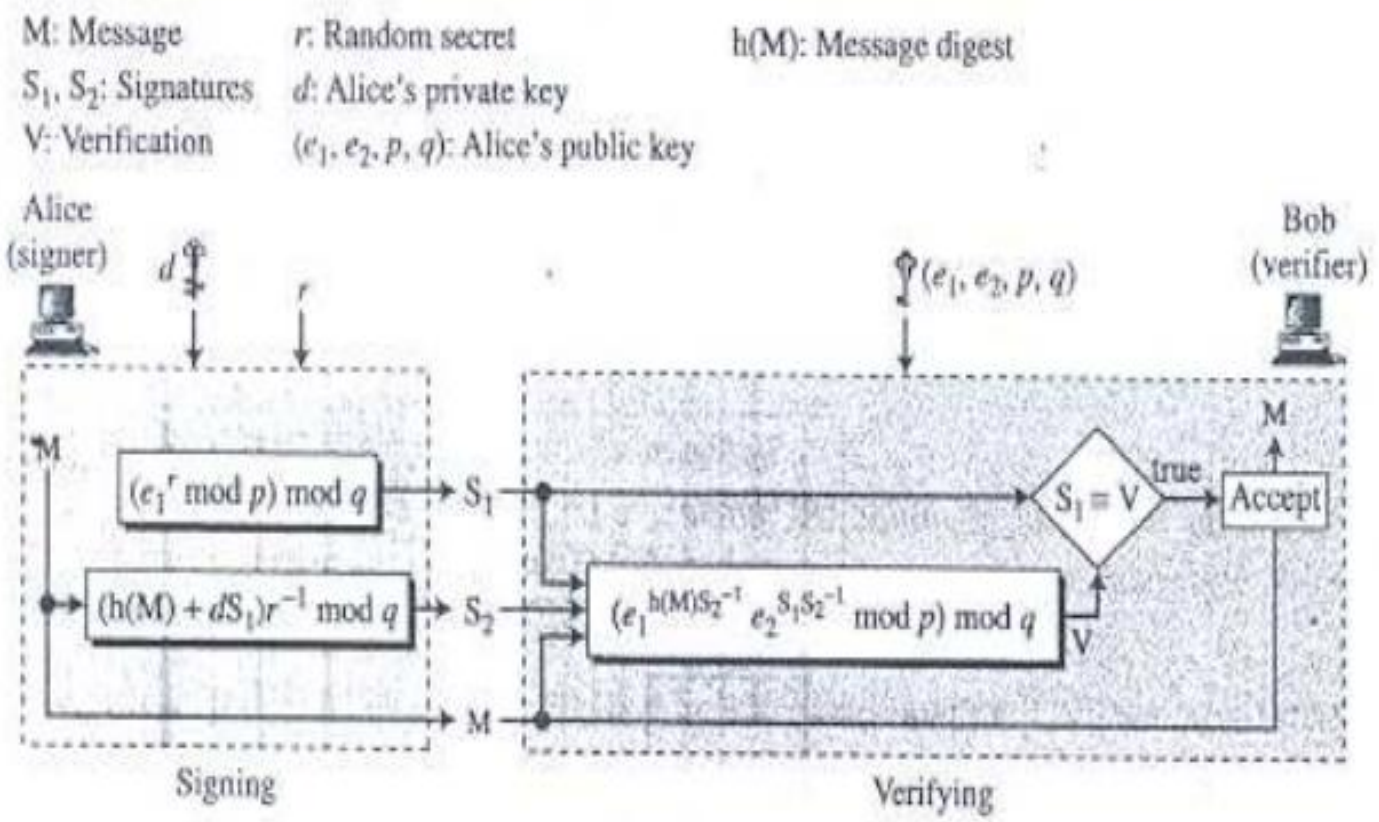
$$s^{-1}m \equiv (-ars^{-1} + k) \pmod{q}.$$

Therefore,

$$\begin{aligned} k &\equiv s^{-1}m + ars^{-1} \pmod{q} \\ &\equiv u_1 + au_2 \pmod{q}. \end{aligned}$$

So $\alpha^k = \alpha^{u_1 + au_2} = (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$. Thus $v = r$.

Figure 13.14 *DSS scheme*



DSS Example

Example 13.5

Alice chooses $q = 101$ and $p = 8081$. Alice selects $e_0 = 3$ and calculates $e_1 = e_0^{(p-1)/q} \bmod p = 6968$. Alice chooses $d = 61$ as the private key and calculates $e_2 = e_1^{\,d} \bmod p = 2038$. Now Alice can send a message to Bob. Assume that h(M) = 5000 and Alice chooses $r = 61$:

$$h(M) = 5000 \quad r = 61$$
$$S_1 = (e_1^{\,r} \bmod p) \bmod q = 54$$
$$S_2 = ((h(M) + d\,S_1)\,r^{-1}) \bmod q = 40$$

Alice sends M, S_1, and S_2 to Bob. Bob uses the public keys to calculate V.

$$S_2^{-1} = 48 \bmod 101$$
$$V = [(6968^{5000 \times 48} \times 2038^{54 \times 48}) \bmod 8081] \bmod 101 = 54$$

Because S_1 and V are congruent, Bob accepts the message.

DSS Versus RSA

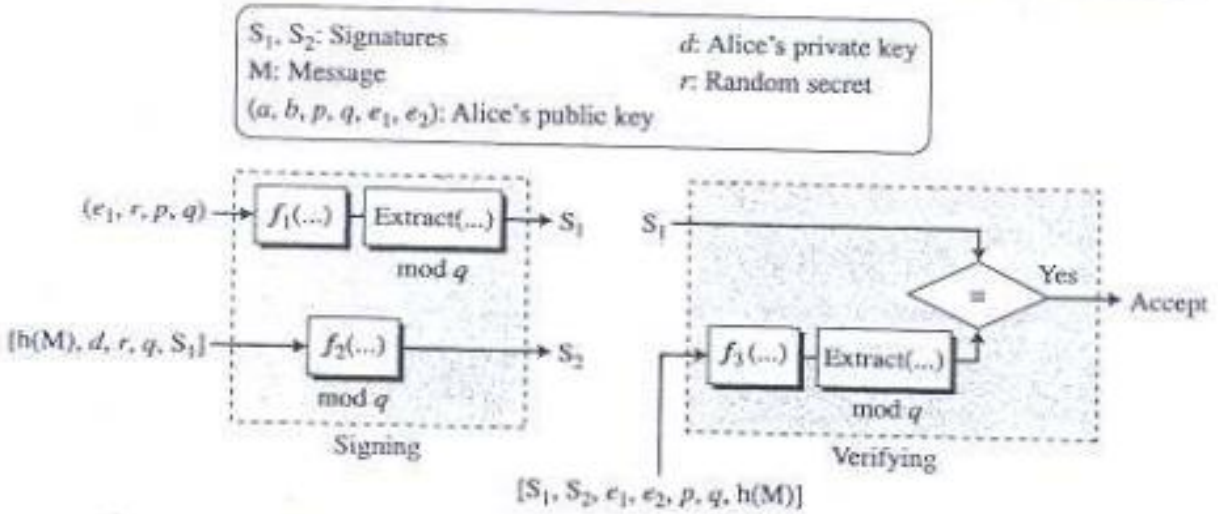
Computation of DSS signatures is faster than computation of RSA signatures when using the same p.

DSS Versus ElGamal

DSS signatures are smaller than ElGamal signatures because q is smaller than p.

Elliptic Curve Digital Signature Scheme

Figure 13.15 *General idea behind the ECDSS scheme*



Key Generation

Key generation follows these steps:

1. Alice chooses an elliptic curve $E_p(a, b)$ with p a prime number.
2. Alice chooses another prime number q to be used in the calculation.
3. Alice chooses the private key d, an integer.
4. Alice chooses $e_1(\ldots, \ldots)$, a point on the curve.
5. Alice calculates $e_2(\ldots, \ldots) = d \times e_1(\ldots, \ldots)$, another point on the curve.
6. Alice's public key is (a, b, p, q, e_1, e_2); her private key is d.

Signing and Verifying

Figure 13.16 shows the elliptic curve digital signature scheme.

Signing The signing process consists mainly of choosing a secret random number, creating a third point on the curve, calculating two signatures, and sending the message and signatures.

1. Alice chooses a secret random number r, between 1 and $q - 1$.
2. Alice selects a third point on the curve, $P(u, v) = r \times e_1 (\ldots, \ldots)$.
3. Alice uses the first coordinates of $P(u, v)$ to calculate the first signature S_1. This means $S_1 = u \bmod q$.
4. Alice uses the digest of the message, her private key, and the secret random number r, and the S_1 to calculate the second signature $S_2 = (h(M) + d \times S_1)\, r^{-1} \bmod q$.
5. Alice sends M, S_1, and S_2.

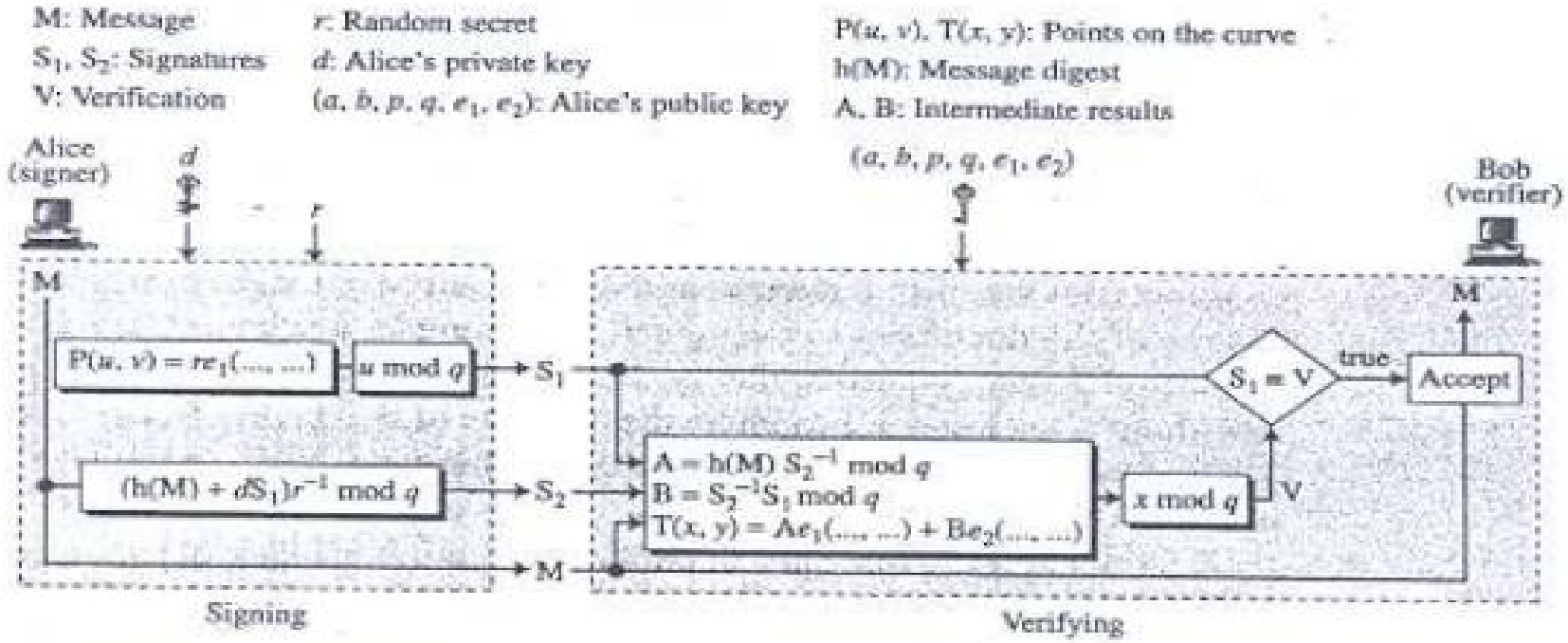
Verifying The verification process consists mainly of reconstructing the third point and verifying that the first coordinate is equivalent to S_1 in modulo q. Note that the third point was created by the signer using the secret random number r. The verifier does not have this value. He needs to make the third point from the message digest, S_1 and S_2:

1. Bob uses M, S_1, and S_2 to create two intermediate results, A and B:

$$A = h(M)\, S_2^{-1} \bmod q \qquad \text{and} \qquad B = S_2^{-1}\, S_1 \bmod q$$

   Bob then reconstructs the third point $T(x, y) = A \times e_1 (\ldots, \ldots) + B \times e_2(\ldots, \ldots)$.
2. Bob uses the first coordinate of $T(x, y)$ to verify the message. If $x = S_1 \bmod q$, the signature is verified; otherwise, it is rejected.

Figure 13.16 *The ECDSS scheme*



does not have this value. He needs to make the third point from the message digest, S_1 and S_2:

1. Bob uses M, S_1, and S_2 to create two intermediate results, A and B:

$$A = h(M)\, S_2^{-1} \bmod q \qquad \text{and} \qquad B = S_2^{-1}\, S_1 \bmod q$$

   Bob then reconstructs the third point $T(x, y) = A \times e_1 (\ldots, \ldots) + B \times e_2(\ldots, \ldots)$.

2. Bob uses the first coordinate of $T(x, y)$ to verify the message. If $x = S_1 \bmod q$, the signature is verified; otherwise, it is rejected.

Thank You