Contents

- Introduction
- Symmetric-key cryptography
  - Block ciphers
  - Symmetric-key algorithms
  - Cipher block modes
  - Stream cipher
- Public-key cryptography
  - RSA
  - Diffie-Hellman
  - ECC
  - Digital signature
  - Public key Infrastructure
- Cryptographic hash function
  - Attack complexity
  - Hash Function algorithm
- Integrity and Authentication
  - Message authentication code
  - GCM
  - Digital signature
- Key establishment
  - server-based
  - Public-key based
  - Key agreement (Diffie-Hellman)

Digital Signature

3 kinds of public key crypto

- There are 3 kinds of mathematically hard one-way functions on which the public key crypto are based.
  - Factoring integers
    - RSA
  - Discrete Logarithm
    - Diffie-Hellman
  - Elliptic curve: generalized discrete logarithm
    - ECDH, ECDSA

Uses of Public Key Crypto

- Encryption
  - Suppose we **encrypt** M with Bob's public key
  - Bob's private key can **decrypt** to recover M
- Digital Signature
  - **Sign** by encrypting with your private key
  - Anyone can **verify** signature by decrypting with sender's public key
- Key exchange

RSA: Key Exchange

Alice

Bob

Message x

1. Select large primes p, q
2. n=pq
3. φ(n)=(p − 1)(q − 1)
4. Choose e=3 (relatively prime to φ(n))
5. Compute d·e=1 mod φ(n)

$K^+ = (n, e)$ ←

$K^-=d$

K_{AB}: AES key

$Y= E_{K_{AB}}(x)$

$Y'= (K_{AB})^e \bmod n$

→ (y, y')

$K_{AB} = (y')^d \bmod n$

$X= D_{K_{AB}}(y)$

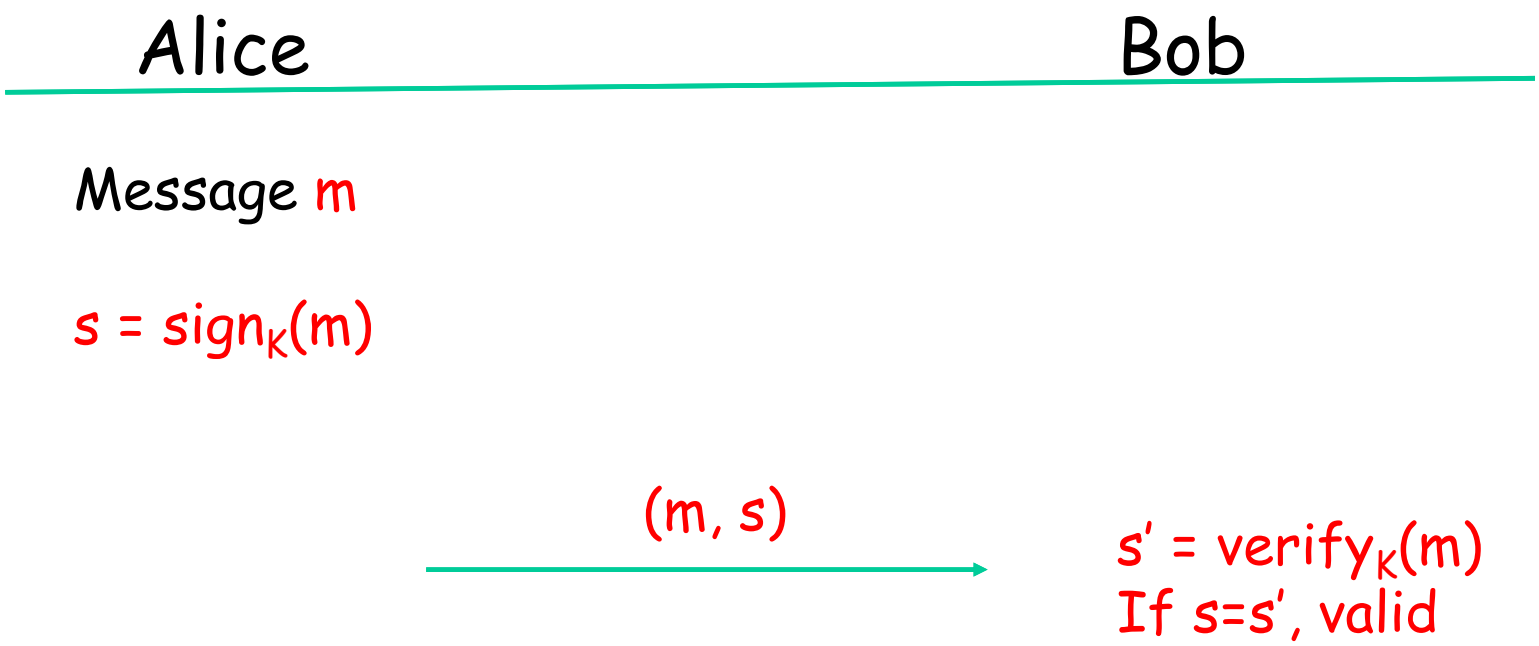
Security objectives

- Confidentiality
- Message integrity
- Message authentication
  - ensure that the sender (origin) of a message is authentic
- Entity authentication (identification)
  - verify the identity of an entity
- Non-repudiation
  - prevent the denial of previous actions

- Availability
  - system(service) is available when needed
- Access control
  - restrict access to the resource
  - restrict actions by privileges : authorization
- Auditing
  - provide evidence about security-related activities and events
- Privacy
  - Protect illegal use of identity
- Physical security
  - Prevent any physical intrusion and tampering

Digital signature

- Alice sends a message with her signature which denotes her own unique identity, similar to handwriting.
- However, to prevent forgery of its signature, She should compute the signature that is related to the message being sent.
- In this way, She can ensure that she sends the very message.


Alice
Bob
Message m
s = sign_K(m)
(m, s)
s' = verify_K(m)
If s=s', valid

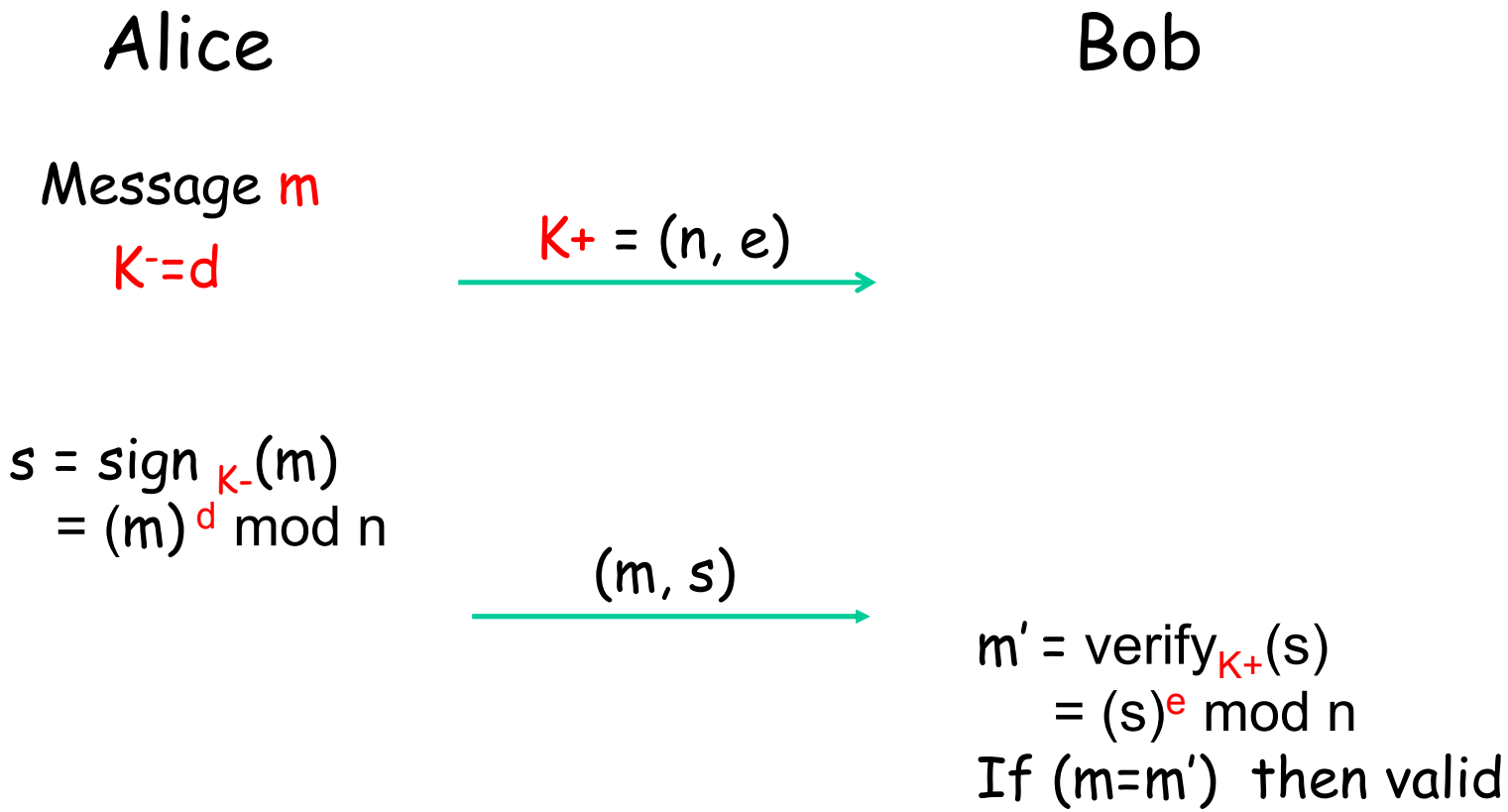
Sign with symmetric key

- The cypto keys can identify the uniqueness of the sender.
- If the symmetric key is used for signature, it verifies:
  - Sender identity
  - Message integrity

Signature with private key

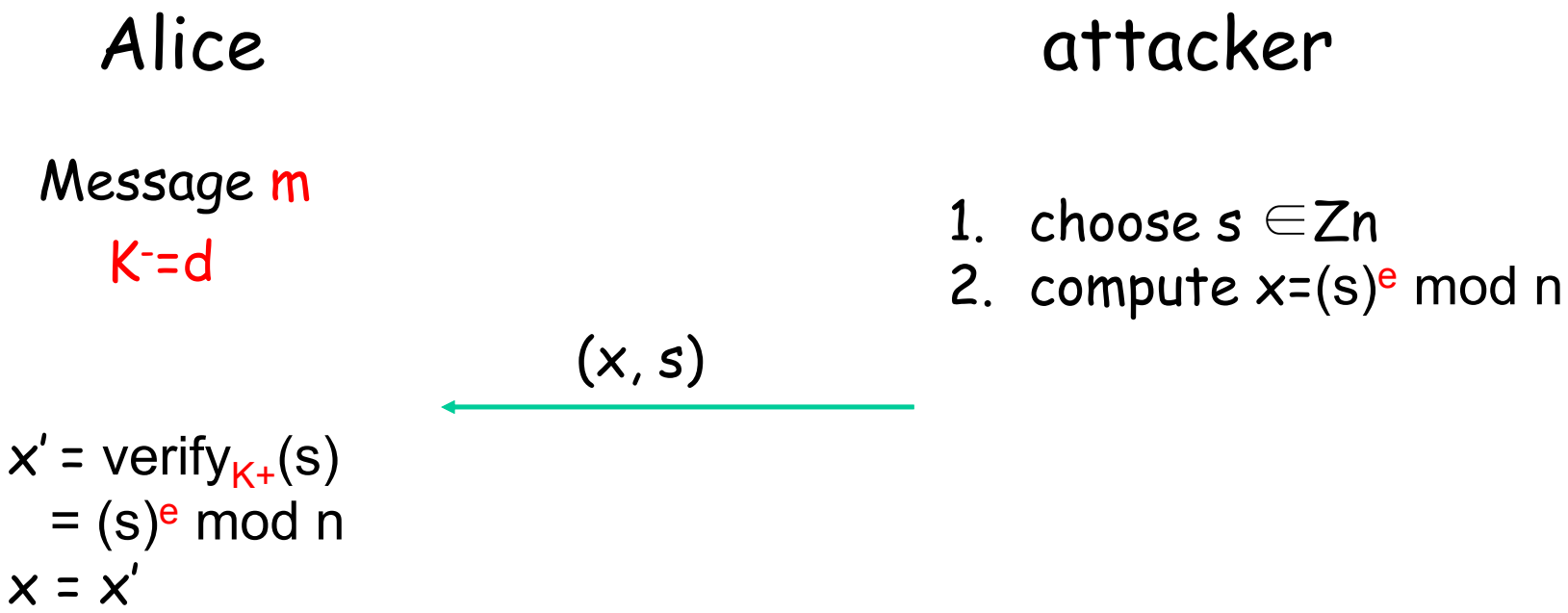
- The private key can also identify the uniqueness of the sender.
- If the private key is used for signature, it verifies
  - Sender identity
  - Message integrity
- Better yet, it also provides
  - Non-repudiation

RSA Digital Signature



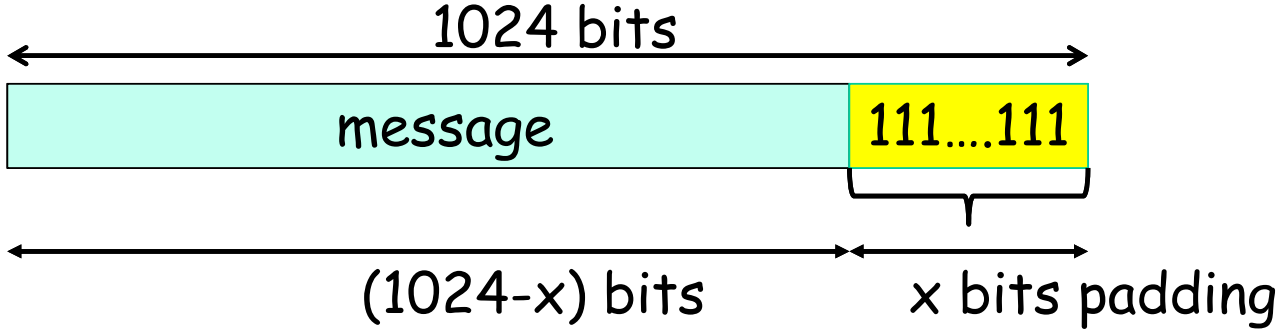
Existential Forgery Attack against RSA DS

- An attacker can forge the signature in the following way.
  - choose any arbitrary signature, s.
  - Compute the message, m, to correspond to s.
  - Send message and signature.
- But, the attacker can't control the content of the message, i.e., the semantics of the message.
  - Not serious attack, but still bothering


Alice
attacker
Message m
K-=d
1. choose s ∈Zn
2. compute x=(s)e mod n
(x, s)
x′ = verifyK+(s)
= (s)e mod n
x = x′

In real world, RSA with padding

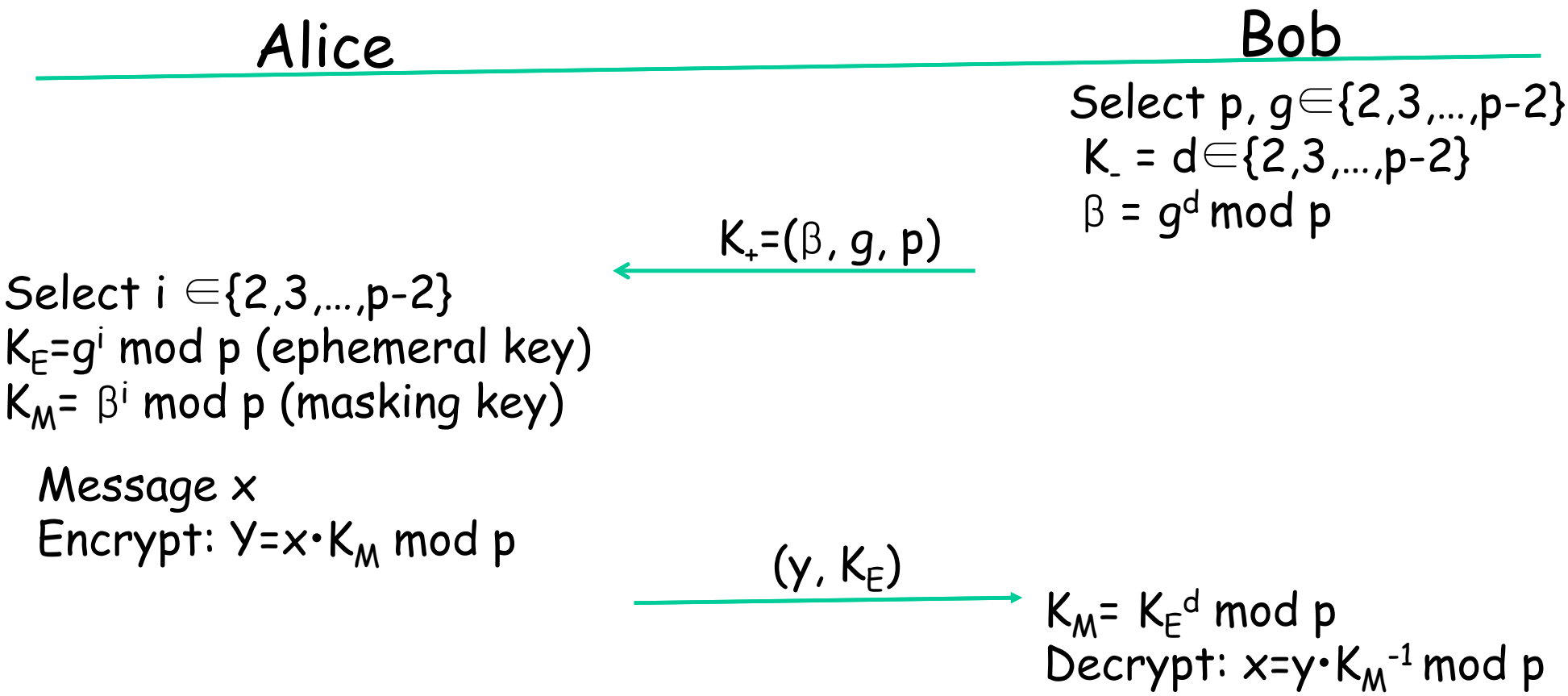
- In reality, RSA imposes the formatting rule on message which is checked by the sender.
- The padding scheme is called "Optimal Asymmetric Encryption Padding(OAEP)."
- In the example below, when an attacker computes the message corresponding to an arbitrarily chosen signature s ($x=s^e$ mode n), he has to compute 2x times to find the message with x's trailing 1s.



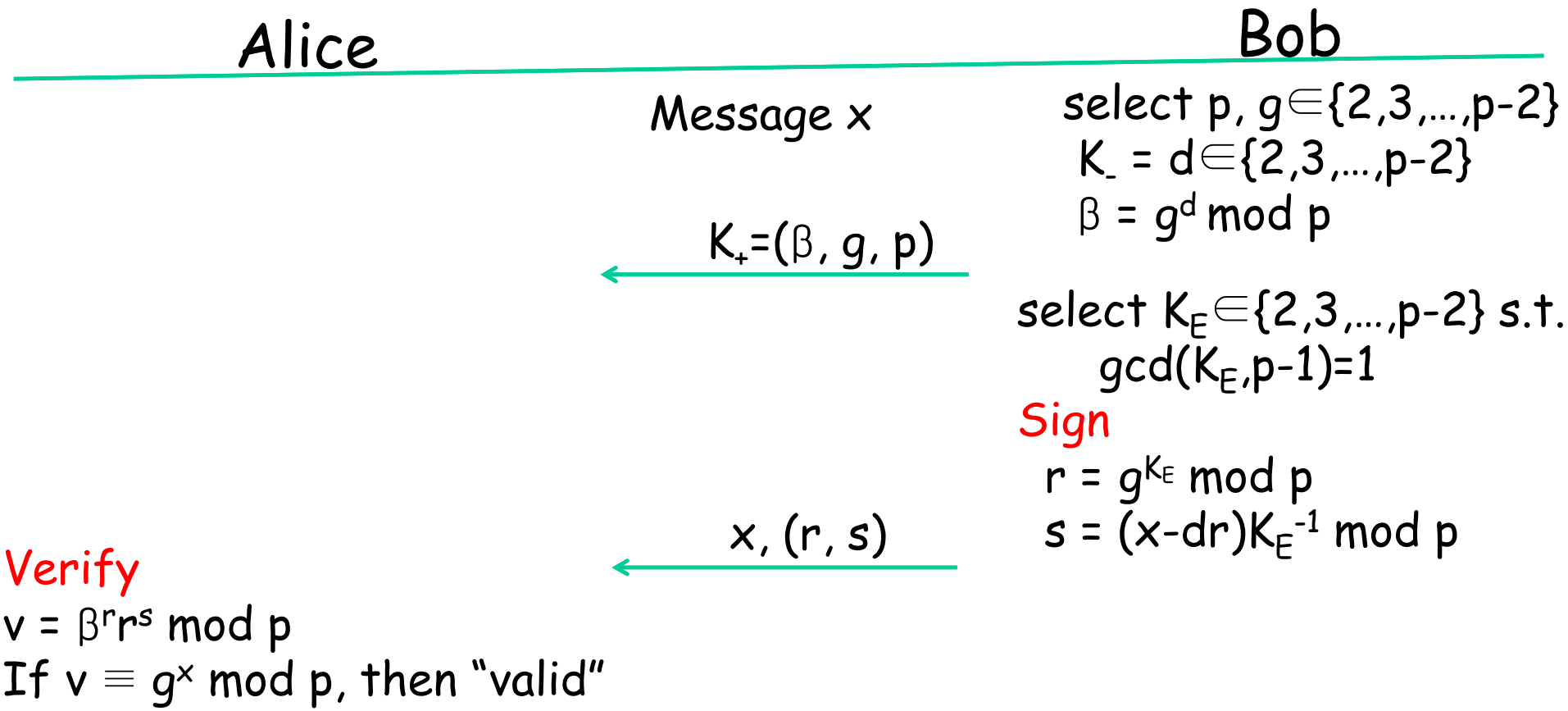
Elgamal Digital Signature algorithm

- was published around 1985
- based on the D-H algorithm
- But, the steps of the D-H algorithm are reordered.

Reminder: Elgamal Encryption



Elgamal Digital Signature



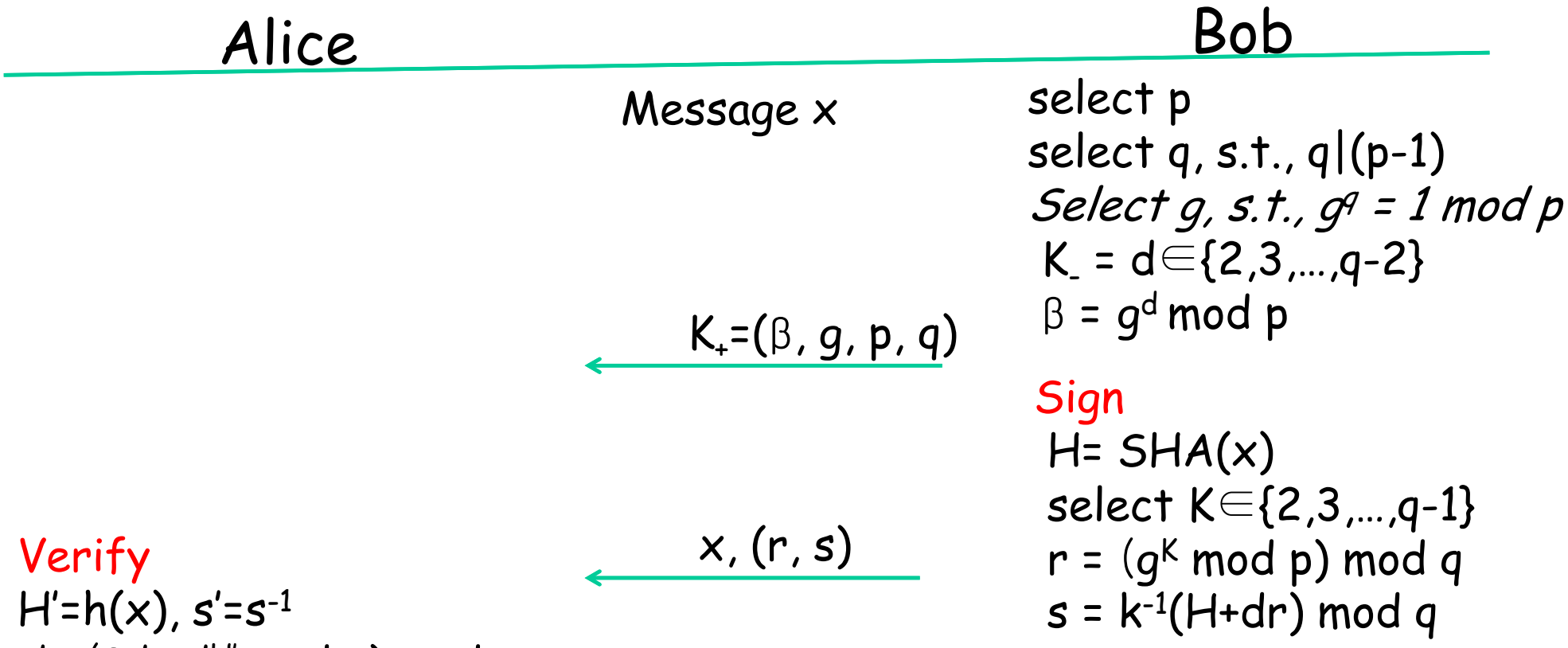
Digital Signature Algorithm(DSA)

- US federal government standard for digital signature(DSS)
- DSA is a modification of ElGamal digital signature scheme. It was proposed by NIST in August 1991 and adopted in December 1994.
- The ElGamal DS would lead to signatures with at least 1024bits which is too much for such applications as smart cards.
- In DSA a 160 bit message is signed using only 320-bit signature, but computation for verification is done modulo with 512-1024 bits, slower than RSA DS.

DSA key generation

- Generate a prime p with $2^{512} < p < 2^{1024}$ (multiple of 64 bits)
- Find a prime divisor q of $p-1$ with 160 bits
- Find an integer g with ord(g)=q ($g^q = 1 \bmod p$)
- Choose a random integer d with $0 < d < q$
- Compute $\beta \equiv g^d \bmod p$
- *Public = (p, q, g, β)*
- *Private = (d)*

DSA


Alice
Bob
Message x
select p
select q, s.t., q|(p-1)
Select g, s.t., g^q = 1 mod p
K- = d∈{2,3,…,q-2}
β = g^d mod p
K+=(β, g, p, q)
Sign
H= SHA(x)
select K∈{2,3,…,q-1}
r = (g^K mod p) mod q
s = k^-1(H+dr) mod q
x, (r, s)
Verify
H'=h(x), s'=s^-1
r'= (β^s'r g^s'H' mod p) mod q
If r' ≡ r, then "valid"

DSA example

Alice

Bob

Message x

select p=23, g=6$\in${2,3,…,p-2}
K_- = d=7$\in${2,3,…,p-2}
$\beta = 6^7 \bmod 23 = 3 \bmod 23$
Select q=11, s.t, 11|22
select K=2$\in${2,3,…,q-1}

K_+=(3, 6, 23, 11) ←

Signature
$r = 6^2 \bmod 23 \bmod 11$
$= 13 \bmod 11 = 2 \bmod 11$
H= 5
$s = 2^{-1}(5+7x2) \bmod 11 = 4$

5, (2, 4) ←

Verify
H'=5, $s' = 4^{-1} \bmod 11 = 3$
$r' = (3^{3X2}6^{3x5} \bmod 23) \bmod 11 = 2$
If r' ≡ r, then "valid"

Elliptic Curve DSA

- Bit length of 160-256 can provide the same level of security as 1024-3072 bits RSA.
- The signature is twice the used bit length. (320-512 bits)

Public Key Infrastructure

Question in Public key

- How can Bob believe that Alice's public key is real?
- Bob receives Alice's public key from any other source besides Alice herself. Then how can he trust that it is really her public key?

Public Key Certificate

- **Certificate** contains name of user and user's public key (and possibly other info)
- Any trusted issuer, called a ***Certificate Authority*** (CA) issues the Alice's certificate

M = (Alice, Alice's public key)

CA guarantees the certificate by signing the certificate using its private key:

$s = sign_{K\text{-}CA}(M)$

Alice's Certificate = (M, S)

Certificate Authority

- Certificate authority (CA) is a trusted 3rd party (TTP) — creates and signs certificates
- Verify signature to verify integrity & identity of **owner of corresponding private key**
  - Does **not** verify the identity of the **sender** of certificate — certificates are public keys!
- Big problem if CA makes a mistake (a CA once issued Microsoft certificate to someone else)
- A common format for certificates is X.509

X.509 certificate example(1)

- Next slide is a certificate to verify the public key of www.freesoft.org.
- CA is Thawte.
- Thawte signed at the bottom of the certificate to verify the certificate. (signature)
- The recipient can verify this certificate to confirm the signature by using Thawte's public key.

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 7829 (0x1e95)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/emailAddress=server-certs@thawte.com
        Validity
            Not Before: Jul  9 16:04:02 1998 GMT
            Not After : Jul  9 16:04:02 1999 GMT
        Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                 OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                    33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                    66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                    70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                    16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                    c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                    8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                    d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                    e8:35:1c:9e:27:52:7e:41:8f
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
        92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
        ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
        d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
        0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
        5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
        8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
        68:9f
```

X.509 certificate example(2)

- Then, how can the recipient know the Thawte's public key?
- Thawte lets the recipient know its public key through another certificate which is signed by its private key.
- Next slide is the certificate through which Thawte releases its public key.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/emailAddress=server-certs@thawte.com
        Validity
            Not Before: Aug  1 00:00:00 1996 GMT
            Not After : Dec 31 23:59:59 2020 GMT
        Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                 OU=Certification Services Division,
                 CN=Thawte Server CA/emailAddress=server-certs@thawte.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                    68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
                    85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
                    6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
                    6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
                    29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
                    6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
                    5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
                    3a:c2:b5:66:22:12:d6:87:0d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
        a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
        3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
        4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
        8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
        e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
        b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
        70:47
```

X.509 certificate example(3)

- Then, how can the recipient trust this certificate? In other words, how can they believe that Thawte is a trusted CA?
- There should be a root CA (or CAs) at the top of all CAs.

PKI

- Public Key Infrastructure (PKI) is needed to securely use public key crypto by specifying the following:
  - Key generation and management
  - Certificate authority (CA) or authorities
  - Certificate revocation lists (CRLs), etc.
- No general standard for PKI
- For instance, multiple trusted CAs are used in web browsers today.