PV181 Laboratory of security and applied cryptography

Asymmetric cryptography

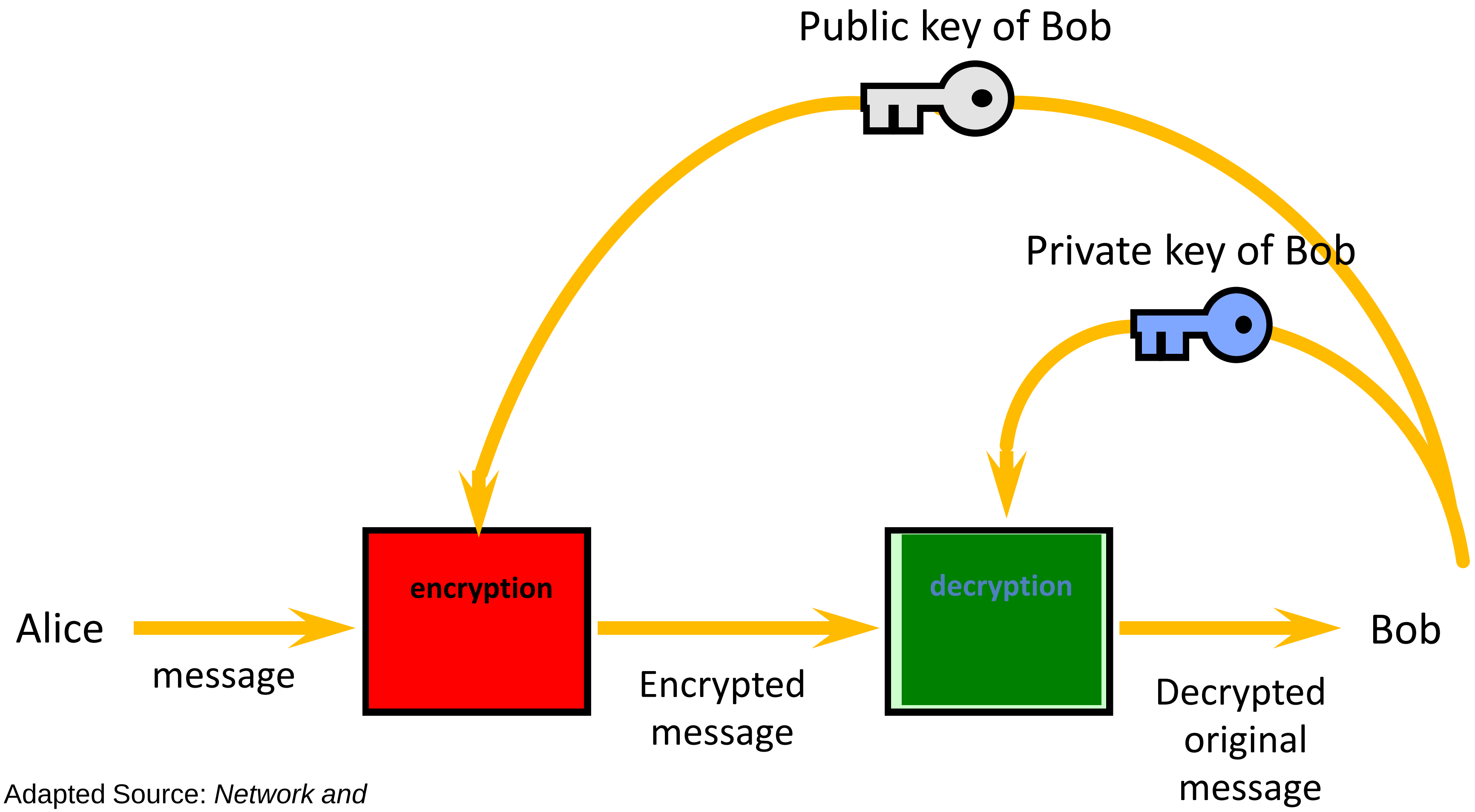
Marek Sýs, Zdeněk Říha

Centre for Research on Cryptography and Security

Asymmetric cryptography

- Confidentiality
  - **asymmetric cipher** - encryption, key agreement
- Authentication
  1. Entity – identity verification – **certificates,**
  2. Data origin - **digital signature**
- Integrity
  - **digital signature**
- Non-repudiation
  - **digital signature**

Asymmetric cryptosystem



Adapted Source: *Network and Internetwork Security* (Stallings)

Asymmetric cryptography

- Two related keys – created by **one** party
  - different inverse operations (encryption - decryption, signing – signature verification)
- Properties - **hard** to compute private from public key
  - based on hard mathematical problems
- Hard problems and cryptosystems:
  - Integer factorization – RSA, Rabin, …
  - Discrete logarithm problem (DLP): ElGamal, EC, DSA, …
  - Others (DH, decoding,…) – Diffie-Helman, McElliece,…

Hard problems

- Integer factorization
  - for **N** find its nontrivial divisor
- DLP in domains:

  (parameters: define alg. structure and DLP)
  - Z_p defined by p :
    - for g, h find x such that $g^x \equiv h \: mod \: p$
  - Elliptic curve defined by triple a, b, p:
    - for points G, H find x such that $x.G = H$

RSA generation

- Two **random** primes p, q
  - $prime - 1$ must have large factor (110 bits)
- Public exponent e chosen so:
  - $\gcd(p-1, e) = \gcd(q-1, e) = 1$
  - typically set 65537 – prime number
  - very small e is insecure: e.g. e=3 sent to 3 recipients
- Private exponent d computed
  - from e so that : $e.d \equiv 1\ mod\ (p-1) * (q-1)$
  - Small d is insecure – < 0.29 bits of N (d can be computed)

RSA keys

- Public key
  - $N\ (= p.q),\ e$
- Private key
  - d but p, q must be kept secret
- Encryption: $E(p) = m^e \ mod\ N$
- Decryption: $D(c) = c^d \ mod\ N$
  - Can be sped up (4x) using CRT
    - $D(c) = c^{d_2} \ mod\ q + \mathrm{C}.(c^{d_1} \ mod\ p - c^{d_2} \ mod\ q)$, for $d_2 = d \ mod\ q, d_1 = d \ mod\ p$

RSA Padding example (PKCS#1 v1.5)

- Document
  - “00 01 02 03 04 05 06 07 07 06 05 04 03 02 01”
- Hash of the document (sha-1)
  - “b3 39 90 4c d2 a0 10 e6 19 37 eb e5 b5 83 37 8c 5d 10 51 95”
- Padded hash
  - “00 01 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 30 21 30 09 06 05 2b 0e 03 02 1a 05 00 04 14 b3 39 90 4c d2 a0 10 e6 19 37 eb e5 b5 83 37 8c 5d 10 51 95”

RSA in practice: Padding

- Non-random padding (textbook RSA) is insecure
  - Bleichenbacher attack – oracle attack (information about improper format of message) CCA

- Paddings - random value computed from message
  - ANSI X9.31
  - PKCS #1 v1.5
  - Probabilistic Signature Scheme (PSS)
  - OAEP – PKCS#1 v2 & RFC 2437- completely random - secure

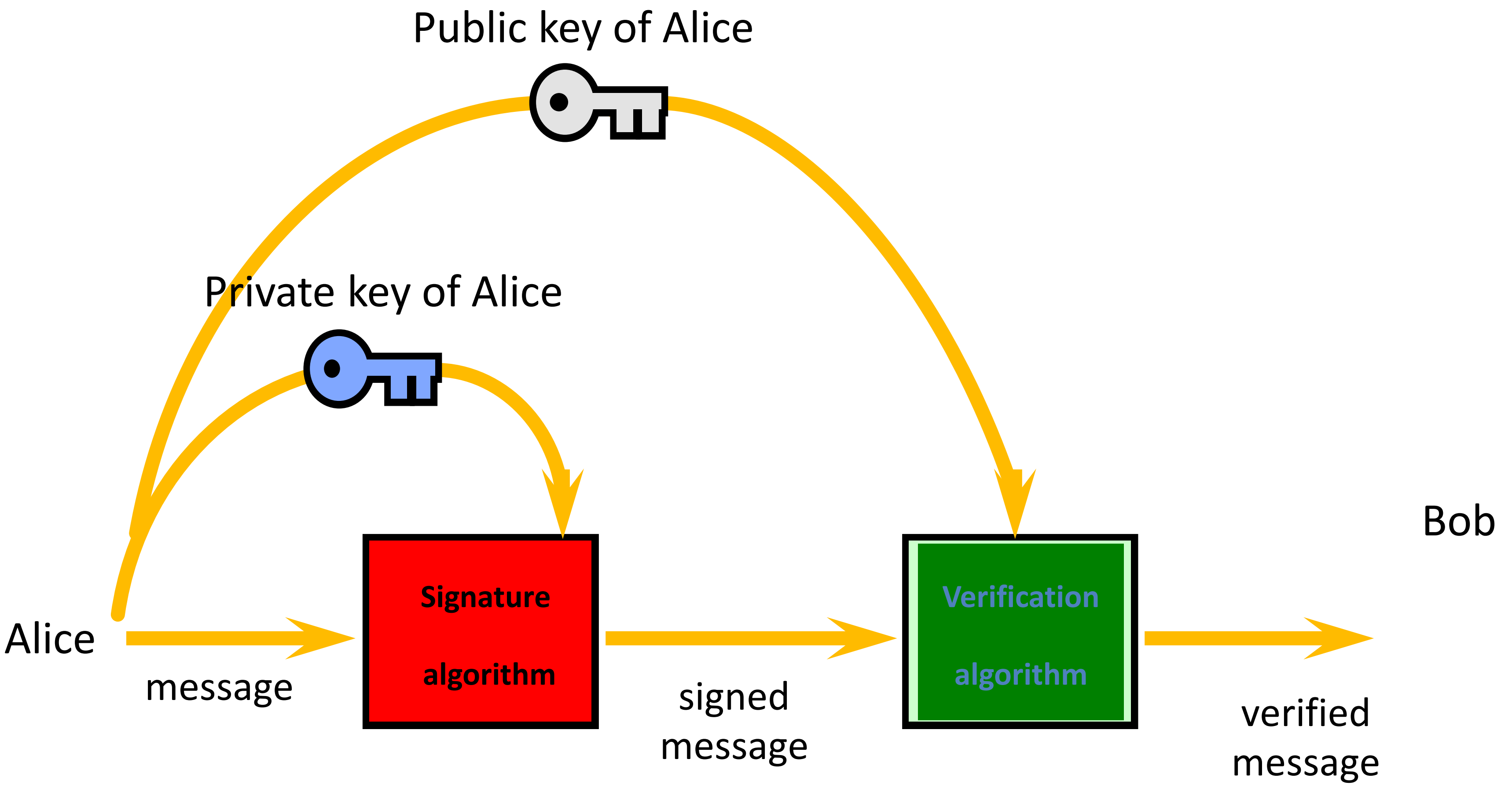
Diffie-Hellman (DH) algorithm

- Described in PKCS#1
- Params:
  - p, g prime, generator of subgroup in Z_p, define DLP $g^x \equiv h \bmod p$
- Private key: x
- Public key: g, p, h $\quad (h \equiv g^x \bmod p)$
- Key agreement - both participants affect the key
  - Alice sends: $g^a \bmod p$
  - Bob sends: $g^b \bmod p$
  - Shared key $g^{ab} \bmod p$ computed as:
    - $(g^a \bmod p)^b \bmod p$ or $(g^b \bmod p)^a \bmod p$

Digital signature

- Asymmetric cryptography
  - Private key – signature generation (usually only **hash** of data is typically signed **not** data itself)
  - Public key – verification procedure
- Data integrity + data origin + non-repudiation:
  - Non-repudiation - correct signatures can be generated only by those having the private key
- The digital signature itself does not give any guarantees with respect to signing time.

Digital signature scheme



Source: *Network and Internetwork Security* (Stallings)

Digital Signature Algorithm (DSA)

- In 1994 the selection procedure for Digital Signature Standard (DSS) was concluded – DSA (Digital Signature Algorithm) was selected.
- Modified version of ElGamal algorithm, based on discrete logarithm in Z_p.
- Became FIPS standard FIPS 186 in 1993.
  - Standards – 186-1, 186-2, 186-3, 186-4
  - Now NIST FIPS 186-4 supports RSA & DSA & ECDSA.

Digital Signature Standard (DSS)

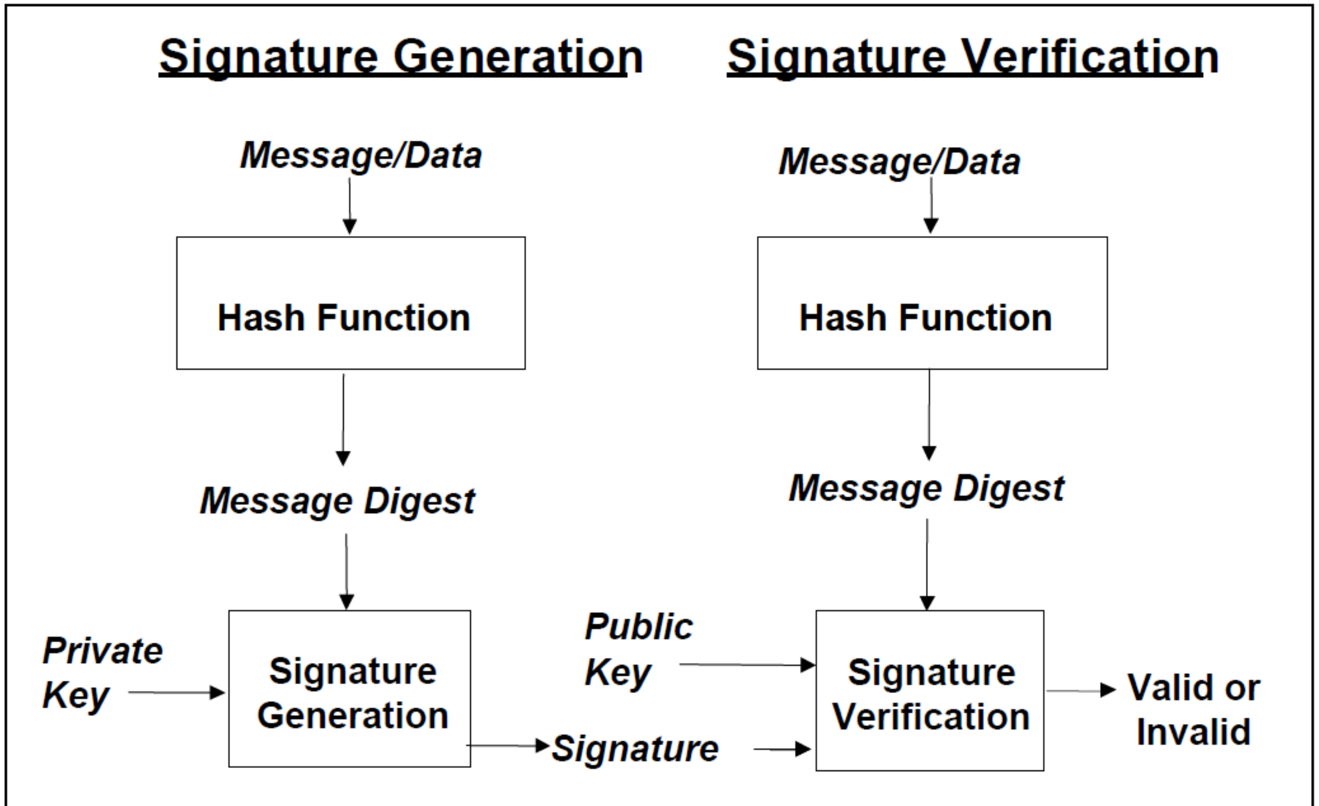


Figure 1: Digital Signature Processes

DSS

1. Selection of Parameter Sizes and Hash Functions
2. Domain Parameter Generation
   - only for DSA, ECDSA
3. Signature Generation
4. Signature Verification and Validation

DSA: Key generation

- Parameter generation
  - Key length: N, L bits
  1. Choose prime q with N bits
  2. Choose prime modulus p with L bits such that: $q|p$
  3. Choose g with order q in Z_p

Keys:

- Private key: x
- Public key: p, g, q, h $(h \equiv g^x mod\, p)$

DSA signature generation & verification

- Signature generation
  - Generate a random per-message value **k** such that $0 < k < q$.
  - Calculate **r** = $(g^k \bmod p) \bmod q$
  - Calculate **s** = $(k^{-1}(H(m) + x*r)) \bmod q$
  - The signature is (r, s).
- Signature verification
  - **w** = $(\mathbf{s})^{-1} \bmod q$
  - **u1** = $(\mathbf{H(m)}*w) \bmod q$
  - **u2** = $(\mathbf{r}*w) \bmod q$
  - **v** = $((g^{u1}*y^{u2}) \bmod p) \bmod q$
  - The signature is valid if **v = r**
- For DSA (1024,160) the signature size will be 2x160 bits.

DSA: Padding

- Decide on lengths **L** and **N**, e.g. (1024,160).
  - N must be less than or equal to the hash output length
    - E.g. for (1024,160) sha-1 is typically used, sha-256 would be ok as well and only first 160 bits would be used
  - $s = (k^{-1}(\mathbf{H(m)} + x*r)) \bmod q$
- “It is recommended that the security strength of the (L, N) pair and the security strength of the hash function used for the generation of digital signatures be the same unless an agreement has been made between participating entities to use a stronger hash function. When the length of the output of the hash function is greater than N (i.e., the bit length of q), then the leftmost N bits of the hash function output block shall be used in any calculation using the hash function output during the generation or verification of a digital signature. A hash function that provides a lower security strength than the (L, N) pair ordinarily should not be used, since this would reduce the security strength of the digital signature process to a level no greater than that provided by the hash function.” [FIPS 186-3]

Elliptic curve DSA (ECDSA)

- Elliptic curves invented by Koblitz & Miller in 1985.
- ECDSA proposed in 1992 by Vanstone
- Became ISO standard (ISO 14888-3) in 1998
- Became ANSI standard (ANSI X9.62) in 1999

- ECDSA, ECDH is a version of DSA, DH based on elliptic curves.

Elliptic curve (EC) domain parameters

- **(field,a,b,G,n,h)**
  - Finite field
    - **p** for F_p
    - **m, bases (trinomial, pentanomial)** for $\mathbf{F_2}^{\mathbf{m}}$
  - Coefficients **a, b**: $y^2 = x^3 + ax + b$
  - Group generator: **G**
  - Order of the G: **n**
  - Optional cofactor: **h**
    - (h = number of elements in field / order n)
  - The base point G generates a cyclic subgroup of order n in the field.

Elliptic curve DH (ECDH)

- Described in PKCS#1
- Params:
  - p, a, b define EC over Z_p
  - generator of EC - point $\boldsymbol{G}$, definess DLP: $x\boldsymbol{G} = H$
- Private key: x
- Public key: G, EC, H $(H = x.G)$
- Key agreement - both participants affect the key
  - Alice sends: $a.G$
  - Bob sends: $b.G$
  - Shared key $a.b.G$ computed as:
    - $b.(a.G)$ or $a.(b.G)$

ECDSA: Keys

- Generating key pair
  - Select a random integer **d** from [1,n − 1]
  - Compute **P** = d*G;
- Private key: **d**
- Public key: **P**

- For 256-bit curve
  - the private key **d** will be approx. 256-bit long
  - the public key **P** is a point on the curve – will be approx 512-bit long

ECDSA: Signatures

- Generate signature
  - Select a random integer **k** from [1,n − 1]
  - **(x_1,y_1)** = k*G
  - Calculate **r** = x_1 (mod n)
  - Calculate **s** = k^{-1}(M + r*d) (mod n)
  - Signature is **(r,s)**.
- Signature verification
  - Calculate **w** = s^{-1} (mod n)
  - Calculate **u_1** = z*w (mod n) & **u_2** = r*w (mod n)
  - Calculate **(x_1,y_1)** = u_1*G + u_2*P
  - The signature is valid if **r = x_1 (mod n)**.
- For 256-bit curve the signature length will be approx. 512 bits

ECDSA: Padding

- Rules are same as for DSA
- “It is recommended that the security strength associated with the bit length of n and the security strength of the hash function be the same unless an agreement has been made between participating entities to use a stronger hash function. When the length of the output of the hash function is greater than the bit length of n, then the leftmost n bits of the hash function output block shall be used in any calculation using the hash function output during the generation or verification of a digital signature. A hash function that provides a lower security strength than the security strength associated with the bit length of n ordinarily should not be used, since this would reduce the security strength of the digital signature process to a level no greater than that provided by the hash function.” [FIPS 186-3]