RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function (HKDF)

- 2.2. Step 1: Extract

```
HKDF-Extract(salt, IKM) -> PRK

Options:
   Hash     a hash function; HashLen denotes the length of the
            hash function output in octets

Inputs:
   salt     optional salt value (a non-secret random value);
            if not provided, it is set to a string of HashLen zeros.
   IKM      input keying material

Output:
   PRK      a pseudorandom key (of HashLen octets)
```

```
The output PRK is calculated as follows:

PRK = HMAC-Hash(salt, IKM)
```

key

message

RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function (HKDF)

- 2.3. Step 2: Expand

```
HKDF-Expand(PRK, info, L) -> OKM

Options:
   Hash     a hash function; HashLen denotes the length of the
            hash function output in octets

Inputs:
   PRK      a pseudorandom key of at least HashLen octets
            (usually, the output from the extract step)
   info     optional context and application specific information
            (can be a zero-length string)
   L        length of output keying material in octets
            (<= 255*HashLen)

Output:
   OKM      output keying material (of L octets)
```

```
The output OKM is calculated as follows:

N = ceil(L/HashLen)
T = T(1) | T(2) | T(3) | ... | T(N)
OKM = first L octets of T

where:
T(0) = empty string (zero length)
T(1) = HMAC-Hash(PRK, T(0) | info | 0x01)
T(2) = HMAC-Hash(PRK, T(1) | info | 0x02)
T(3) = HMAC-Hash(PRK, T(2) | info | 0x03)
...
```

RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function (HKDF)

- HKDF-Extract(salt, IKM) -> PRK
- HKDF-Expand(PRK, info, L) -> OKM
- Used in TLS 1.3 for key derivations, see RFC 8446, 7.1.

```
7.1.  Key Schedule

   The key derivation process makes use of the HKDF-Extract and
   HKDF-Expand functions as defined for HKDF [RFC5869], as well as the
   functions defined below:

       HKDF-Expand-Label(Secret, Label, Context, Length) =
            HKDF-Expand(Secret, HkdfLabel, Length)

       Where HkdfLabel is specified as:

       struct {
           uint16 length = Length;
           opaque label<7..255> = "tls13 " + Label;
           opaque context<0..255> = Context;
       } HkdfLabel;

       Derive-Secret(Secret, Label, Messages) =
            HKDF-Expand-Label(Secret, Label,
                              Transcript-Hash(Messages), Hash.length)
```

TLS 1.3 – Key Schedule (RFC 8446, 7.1.)

```
                 0
                 |
                 v
   PSK ->  HKDF-Extract = Early Secret
                 |
                 +-----> Derive-Secret(., "ext binder" | "res binder", "")
                 |                     = binder_key
                 |
                 +-----> Derive-Secret(., "c e traffic", ClientHello)
                 |                     = client_early_traffic_secret
                 |
                 +-----> Derive-Secret(., "e exp master", ClientHello)
                 |                     = early_exporter_master_secret
                 v
           Derive-Secret(., "derived", "")
                 |
                 v
   (EC)DHE -> HKDF-Extract = Handshake Secret
                 |
                 +-----> Derive-Secret(., "c hs traffic",
                 |                     ClientHello...ServerHello)
                 |                     = client_handshake_traffic_secret
                 |
                 +-----> Derive-Secret(., "s hs traffic",
                 |                     ClientHello...ServerHello)
                 |                     = server_handshake_traffic_secret
                 v
           Derive-Secret(., "derived", "")
                 |
                 v
      0 -> HKDF-Extract = Master Secret
                 |
                 +-----> Derive-Secret(., "c ap traffic",
                 |                     ClientHello...server Finished)
                 |                     = client_application_traffic_secret_0
                 |
                 +-----> Derive-Secret(., "s ap traffic",
                 |                     ClientHello...server Finished)
                 |                     = server_application_traffic_secret_0
                 |
                 +-----> Derive-Secret(., "exp master",
                 |                     ClientHello...server Finished)
                 |                     = exporter_master_secret
                 |
                 +-----> Derive-Secret(., "res master",
                                       ClientHello...client Finished)
                                       = resumption_master_secret
```

Salt

IKM

0

HKDF-Extract(0,0)

handshake

application

Eg: Hash = SHA-256

0 <-> new byte[32]

HKDF-Expand

-> 32 bytes for a key (to encrypt and authenticate following messages of the handshake sent by the client.)

TH OWL TECHNISCHE HOCHSCHULE OSTWESTFALEN-LIPPE UNIVERSITY OF APPLIED SCIENCES AND ARTS

(P.9) Example. $\mathbb{F}_{2^8}$

The elements of the field $\mathbb{F}_{2^8}$ are the polynomials over $\mathbb{F}_2$ of degree ≤ 7, where addition is the ordinary addition of polynomials and multiplication can be defined to be the ordinary multiplication of polynomials followed by a reduction modulo the irreducible polynomial

$$m(x) := x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$$

Small irreducible polynomials of degree ≤ 4 :

$\deg(p) = 1$: $\boxed{x, \quad x+1}$ both do not divide $m(x)$, because $m(0) = 1 = m(1)$

$\deg(p) = 2$: ~~$x^2 + 1$~~ $= (x+1)^2$

$\boxed{p_2(x) = x^2 + x + 1}$ is irreducible, because $p_2(0) = 1 = p_2(1)$

$\deg(p) = 3$ $\boxed{\begin{aligned} p_{3,1}(x) &= x^3 + x + 1 \\ p_{3,2}(x) &= x^3 + x^2 + 1 \end{aligned}}$ $\Big\}$ are irreducible

$\deg(p) = 4$

$p_{4,1}(x) = x^4 + x + 1$

$\cancel{x^4 + x^2 + 1 = (x^2+x+1)^2}$

$p_{4,2}(x) = x^4 + x^3 + 1$

$p_{4,3}(x) = x^4 + x^3 + x^2 + x + 1$

(R.8) Definition. If r is an element of a Ring R and there exists some $n \in \mathbb{N}$ with $r^n = 1$, than the smallest such integer is called the order of r, denoted by

$$o(r) \; .$$

(R.9) Remark. Assume R is a ring and $r \in R$ has finite order $o(r) \in \mathbb{N}$. Than

(i) r is invertibel with inverse $r^{o(r)-1}$, and

(ii) the elements $1, r, r^2, ..., r^{o(r)-1}$ are pairwise distinct.

Fields

(F.1) Definition.

A *field* is a ring R, where every non–zero element $r \in R \setminus \{0\}$ is invertible.

(F.2) Examples.

(i) The fields of rational numbers $\mathbb{Q}$, real numbers $\mathbb{R}$ and complex numbers $\mathbb{C}$ are probably well known.

(ii) For every prime number p the ring $\mathbb{Z}_p$ is a field, the Galois field of p elements, also denoted by $\mathbb{F}_p$ or $GF(p)$.

(iii) It can be shown, that there exists a finite field containing exactly q elements, if and only if $q = p^n$ is a power of some prime p. Furthermore such a field is essentially unique and can therefore be denoted by:

$$\mathbb{F}_q = \mathbb{F}_{p^n} = GF(q) = GF(p^n)$$

A construction of the field $\mathbb{F}_q$ is given below, see (P.8).

(F.3) Order of elements in $\mathbb{F}_q$.

(i) For every element $r \in \mathbb{F}_q \setminus \{0\}$ the order $o(r)$ of r divides $q-1$.

(ii) For every finite field $\mathbb{F}_q$ there exist elements s with

$$o(s) = q - 1 \; .$$

Such elements are called *primitive* elements of $\mathbb{F}_q$.

A special case of the above fact is known as Fermat's little theorem:

(F.4) Fermat's little theorem. If p is a prime and a any integer, that is not divisible by p, then:

$$a^{p-1} \equiv 1 \pmod{p}$$

(F.5) Squares in $\mathbb{F}_q$ for odd q. Let p be an odd prime, $q = p^n$ for some $n \in \mathbb{N}$ and g be a primitive element of $\mathbb{F}_q$. Furthermore, let $\mathcal{S}^*$ denote the set of all squares in $\mathbb{F}_q^*$. Then

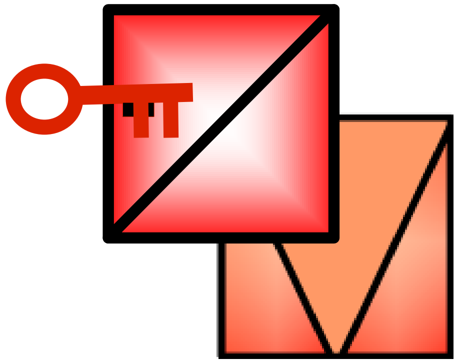
$$\mathcal{S}^* \quad := \quad \{a^2 \mid a \in \mathbb{F}_q\} \quad = \quad \left\{g^i \;\middle|\; i = 0, 2, 4, 6, \ldots, q-1\right\}$$

and for $a \in \mathbb{F}_q^*$:

$$a^{\frac{q-1}{2}} \quad = \quad \begin{cases} 1 & \text{if } \; a \in \mathcal{S}^* \\ -1 & \text{if } \; a \notin \mathcal{S}^* \end{cases}$$

AEAD Algorithms

- Authenticated Encryption with Associated Data



- **Algorithm for combined encryption/decryption and MAC calculation/verification**
- **Encryption and MAC calculation:**
  - Input: Plaintext P, Additional Data A, Key k, Nonce IV
  - Output: Ciphertext $C = E_{k,IV}(P)$, MAC $T = MAC_{k,IV}(A,C)$
- ***AEAD algorithm used with TLS:***
  - Galois/Counter Mode (GCM) of operation of the AES algorithm (NIST Special Publication 800-38D)
  - ChaCha20 and Poly1305 (RFC 8439)

GCM – Encryption and MAC (Tag) calculation

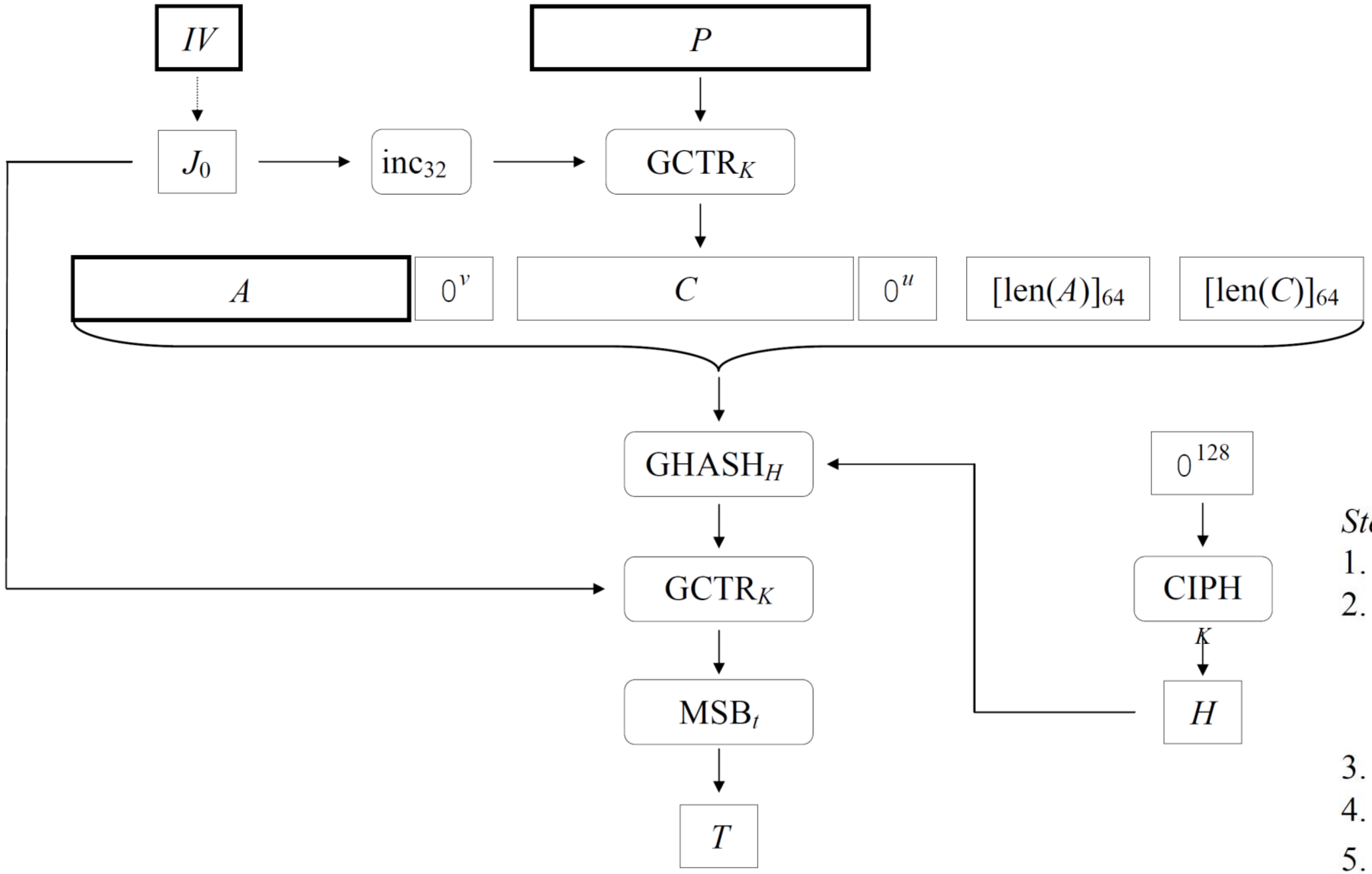


Figure 3: $GCM\text{-}AE_K(IV, P, A) = (C, T)$.

NIST Special Publication 800-38D

Steps:

1. Let $H = CIPH_K(0^{128})$.
2. Define a block, J_0, as follows:
   If $len(IV)=96$, then let $J_0 = IV \,||\, 0^{31} \,||\, 1$.
   If $len(IV) \neq 96$, then let $s = 128\lceil len(IV)/128 \rceil - len(IV)$, and let
   $J_0 = GHASH_H(IV \,||\, 0^{s+64} \,||\, [len(IV)]_{64})$.
3. Let $C = GCTR_K(inc_{32}(J_0), P)$.
4. Let $u = 128 \cdot \lceil len(C)/128 \rceil - len(C)$ and let $v = 128 \cdot \lceil len(A)/128 \rceil - len(A)$.
5. Define a block, S, as follows:
   $$S = GHASH_H(A \,||\, 0^v \,||\, C \,||\, 0^u \,||\, [len(A)]_{64} \,||\, [len(C)]_{64}).$$
6. Let $T = MSB_t(GCTR_K(J_0, S))$.
7. Return (C, T).

GCM – Encryption in CTR mode

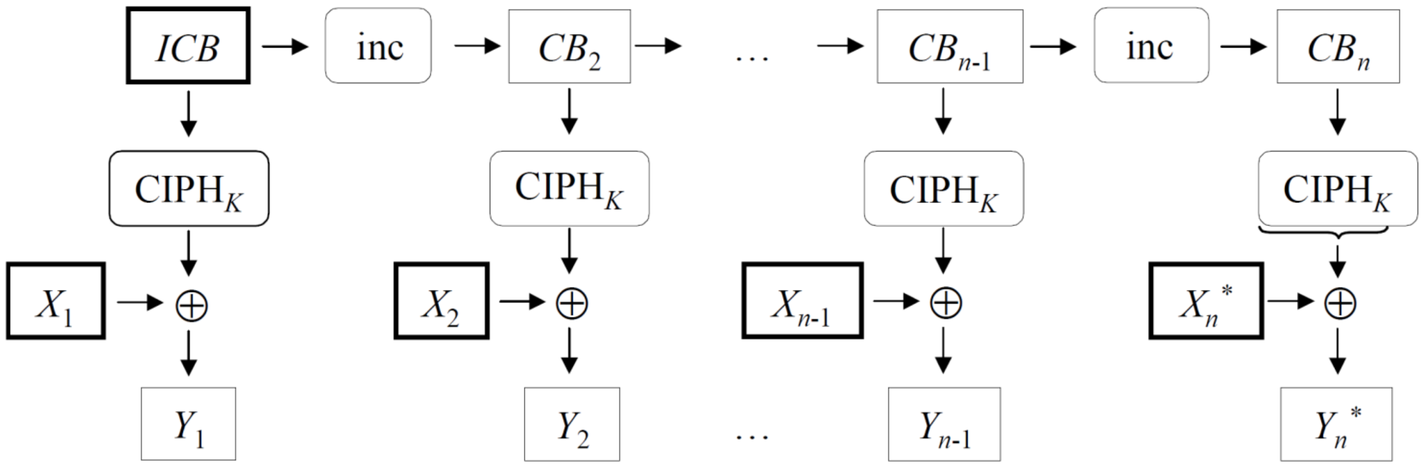


Figure 2: $\text{GCTR}_K\,(ICB, X_1 \,||\, X_2 \,||\, ... \,||\, X_n^*) = Y_1 \,||\, Y_2 \,||\, ... \,||\, Y_n^*$.

[NIST Special Publication 800-38D]

GCM – GHASH function

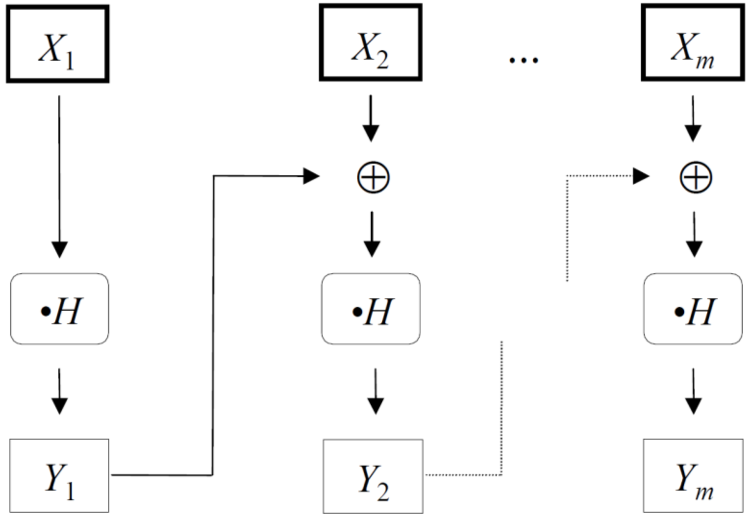


Figure 1: $\mathrm{GHASH}_H(X_1 \,||\, X_2 \,||\, \ldots \,||\, X_m) = Y_m$.

[NIST Special Publication 800-38D]

Let R be the bit string 11100001 || 0^{120}.

The • operation on (pairs of) the 2^{128} possible blocks corresponds to the multiplication operation for the binary Galois (finite) field of 2^{128} elements. The fixed block, R, determines a representation of this field as the modular multiplication of binary polynomials of degree less than 128.

<u>Algorithm 1: $X \bullet Y$</u>

Input:
blocks X, Y.

Output:
block $X \bullet Y$.

Steps:
1. Let $x_0x_1 \ldots x_{127}$ denote the sequence of bits in X.
2. Let $Z_0 = 0^{128}$ and $V_0 = Y$.
3. For $i = 0$ to 127, calculate blocks Z_{i+1} and V_{i+1} as follows:

$$Z_{i+1} = \begin{cases} Z_i & \text{if } x_i = 0; \\ Z_i \oplus V_i & \text{if } x_i = 1. \end{cases}$$

$$V_{i+1} = \begin{cases} V_i >> 1 & \text{if } \mathrm{LSB}_1(V_i) = 0; \\ (V_i >> 1) \oplus R & \text{if } \mathrm{LSB}_1(V_i) = 1. \end{cases}$$

4. Return Z_{128}.

GCM – Decryption and MAC (Tag) verification

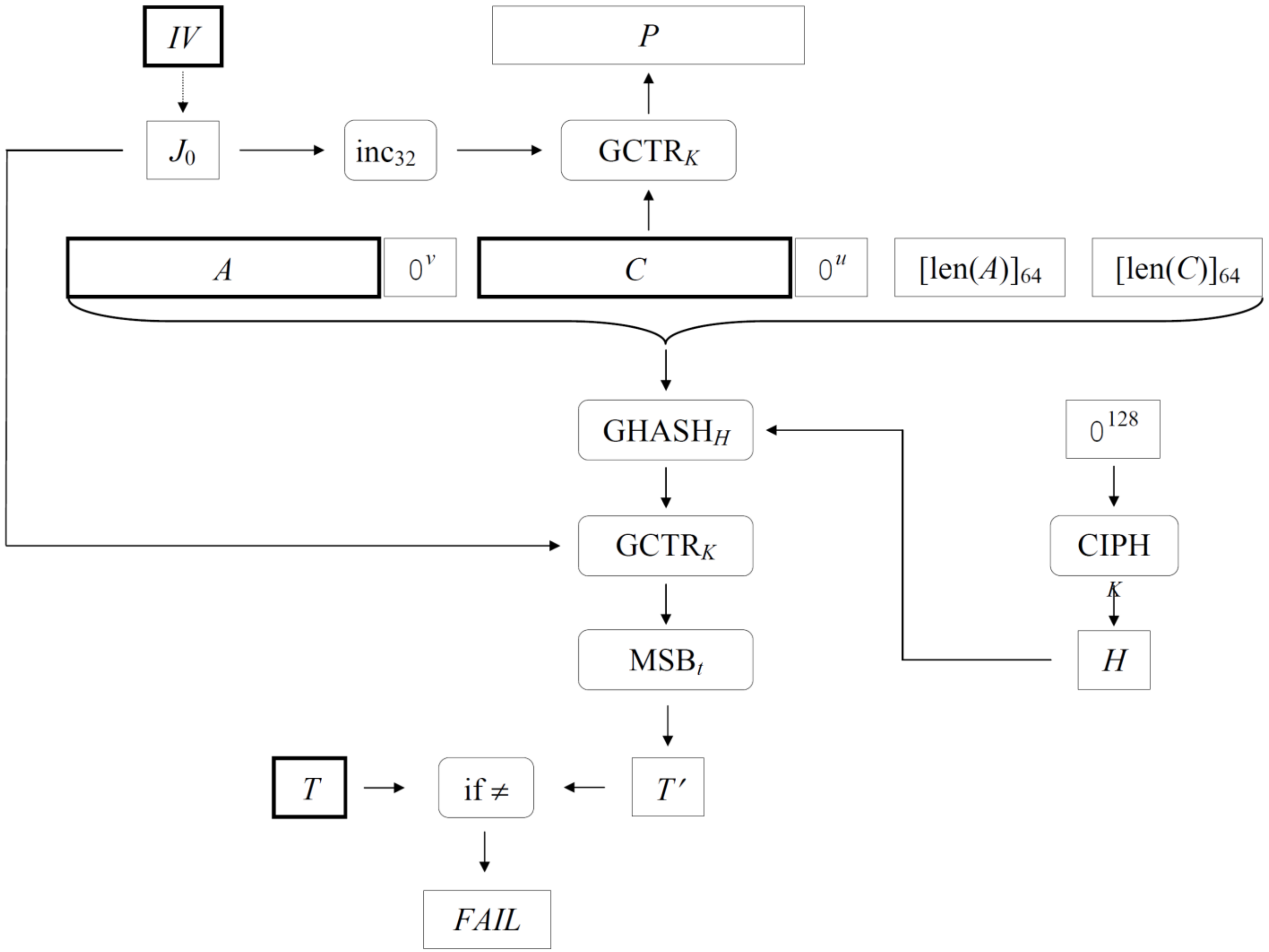


Figure 4: $GCM\text{-}AD_K(IV, C, A, T) = P$ or $FAIL$.

[NIST Special Publication 800-38D]

Asymmetric Cryptography

- **Public Key Cryptography**

- **Problem**

  Usage of symmetric ciphers require the exchange of secret keys over some secure channel.

- **Basic Idea**

  Usage of a mathematical operation, whose inversion is not computational feasible without the knowledge of a key value (trap door function).

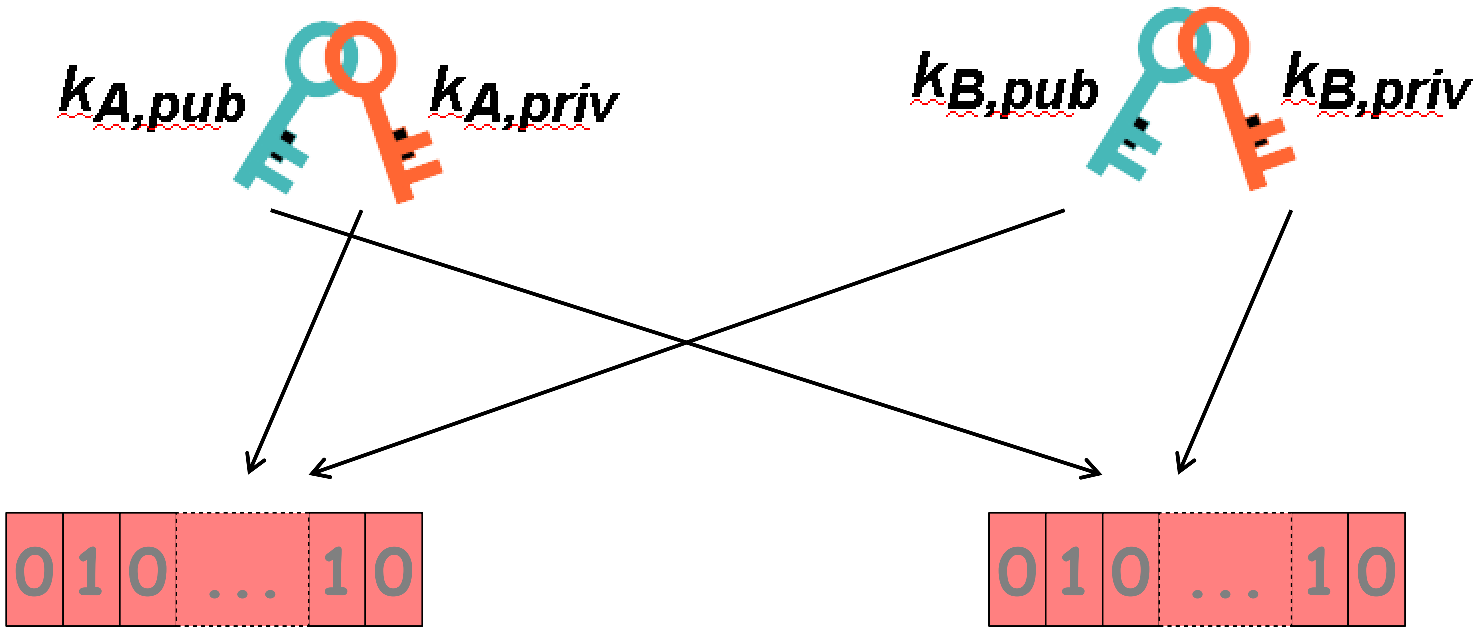
  - Factorization of integers
  - Calculation of discrete logarithms in $\mathbb{Z}_p$
  - Calculation of discrete logarithms in groups defined by elliptic curves over finite fields

First published solutions:

- W. Diffie, M.E. Hellman, *New Directions in Cryptography*, 1976
- R.C. Merkle, *Secure Communication over Insecure Channels*, 1978
- R.L. Rivest, A. Shamir, L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, 1978

- **Algorithms from public key cryptography:**
  - Key derivation algorithms / schemes
  - Asymmetric Ciphers (encryption without a shared secret key)
  - Digital Signatures

Key Derivation



$k = F(k_{A,priv}, k_{B,pub})$

$k = F(k_{B,priv}, k_{A,pub})$

- Key derivation scheme proposed by **W. Diffie** and **M.E. Hellman** in ***New Directions in Cryptography*** (1976).
- Based on the mathematical (computational) problem of finding **discrete logarithms**. (Multiplicative order of an element in $\langle g \rangle$ for some fixed $g \in \mathbb{Z}_n$.)
- **ECDH** (Elliptic Curve Diffie-Hellman): Based on the problem of determining the order of a point of an elliptic curve defined over a finite field.
  - Applying elliptic curves in cryptography was suggested by **N. Koblitz** and **V. S. Miller** in 1985.
  - Widely used since ~2005.

Let $b \in \mathbb{N}$, $n \in \mathbb{N}$.

DL-Problem: Determine for a given

power $$c = b^e \ mod \ n$$

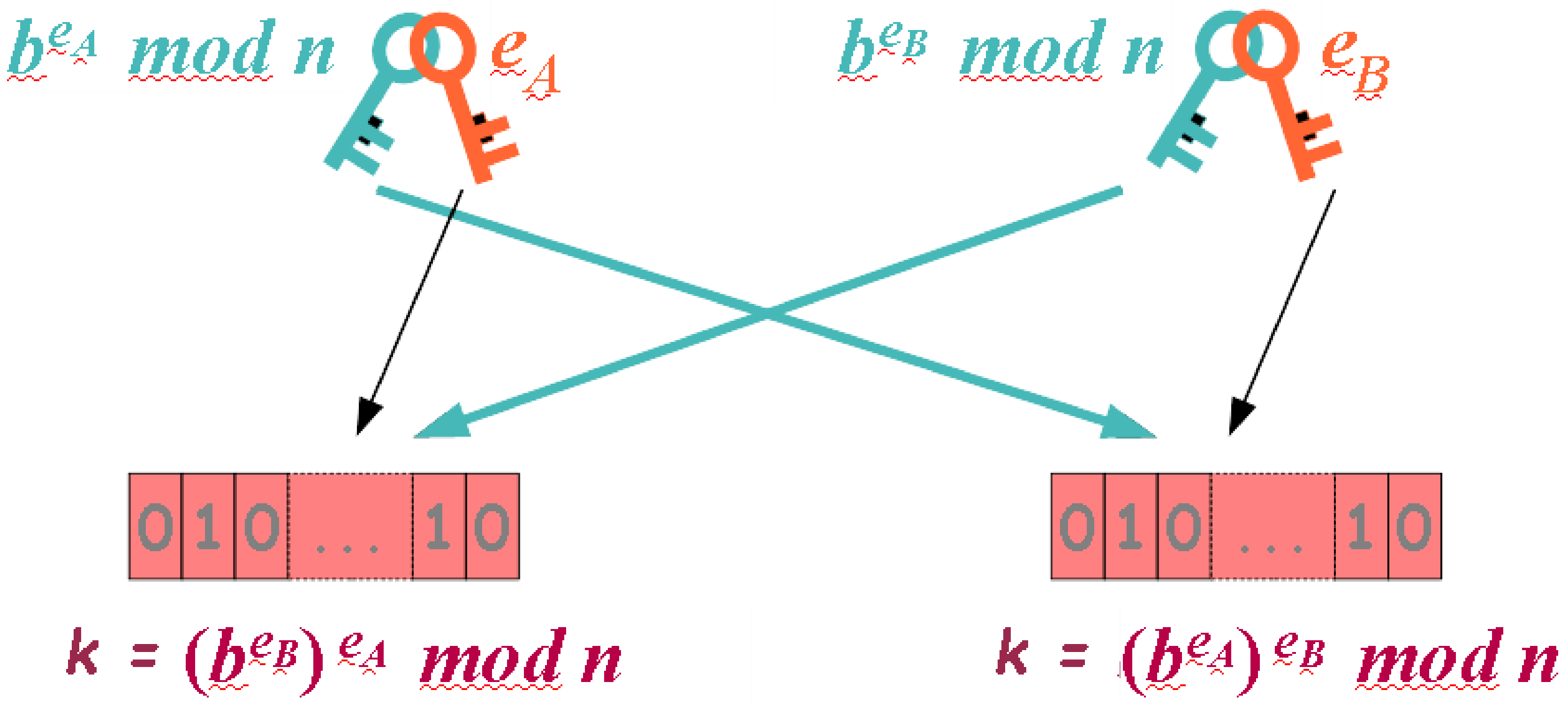
the exponent

$$e.$$

Diffie-Hellman (DH) key derivation

$$n \in \mathbb{N} \quad \textit{prime number}$$

$$b \in \{2, 3, \ldots, (n-2)\}$$



Diffie-Hellman (DH) key derivation

$$n \in \mathbb{N} \quad \textit{prime number}$$
$$b \in \{2, 3, \ldots, (n-2)\}$$

- ***n* -1** should have a big prime factor ***q***, such that ***q*** divides the order of ***b***.
- The order of ***b*** should be large.

- ECC (Elliptic Curve Cryptography) is based on the group structure on the sets of points of an Elliptic Curve defined over F_p or F_{2^n} .

(EC.1) Definition. Let $p > 3$ be a prime and $a, b \in \mathbb{F}_p$ with:

$$4a^3 + 27b^2 \not\equiv 0 \pmod p$$

Then

$$y^2 = x^3 + ax + b$$

defines an *elliptic curve* over $\mathbb{F}_p$ with set of points:

$$E(\mathbb{F}_p) := \{(x, y) \mid x, y \in \mathbb{F}_p, y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

$\mathcal{O}$ is called the *point at infinity*.

(EC.2) Additive group structure of $E(\mathbb{F}_p)$. The elements of $E(\mathbb{F}_p)$ form a commutative group with respect to the following addition:

(i)

$$P + \mathcal{O} \;:=\; \mathcal{O} + P \;:=\; P \qquad \text{for all } P \in E(\mathbb{F}_p)$$

(ii)

$$-P \;:=\; \begin{cases} (x, -y) & \text{if } P = (x, y) \\ \mathcal{O} & \text{if } P = \mathcal{O} \end{cases}$$

Consequently:

$$(x, y) + (x, -y) \;=\; \mathcal{O} \qquad \text{for all } (x, y) \in E(\mathbb{F}_p)$$

(iii) *Point addition*: Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ two points of $E(\mathbb{F}_p)$ with $P \neq \pm Q$. Then

$$P_1 + P_2 \;:=\; (x, y)$$

with:

$$s \;:=\; \frac{y_2 - y_1}{x_2 - x_1}$$

$$x \;:=\; s^2 - x_1 - x_2$$

$$y \;:=\; s(x_1 - x) - y_1$$

(EC.2) Additive group structure of $E(\mathbb{F}_p)$. The elements of $E(\mathbb{F}_p)$ form a commutative group with respect to the following addition:

(iv) *Point doubling*: Let $P = (x_1, y_1)$ be a point of $E(\mathbb{F}_p)$ with $P \neq -P$. Then

$$2 \cdot P \; := \; P + P \; := \; (x, y)$$

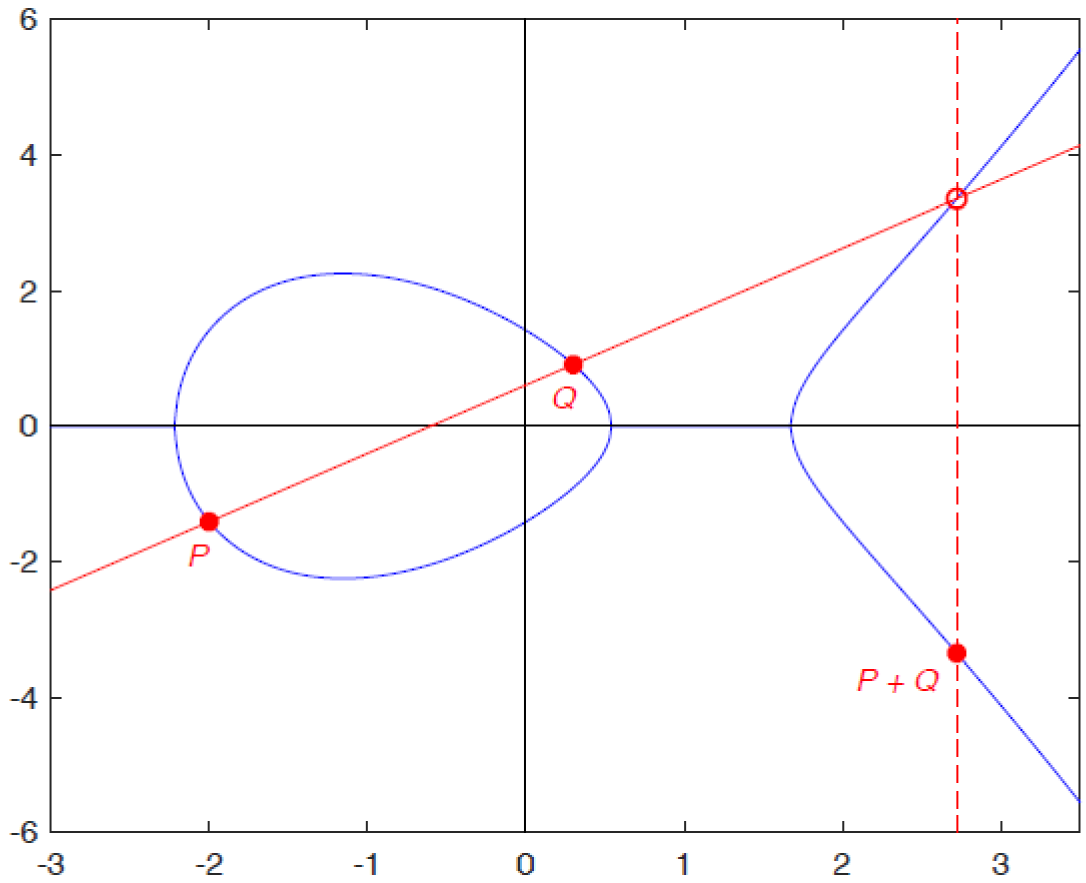
with:

$$\begin{aligned} s &:= \frac{3{x_1}^2 + a}{2y_1} \\ x &:= s^2 - 2x_1 \\ y &:= s(x_1 - x) - y_1 \end{aligned}$$

(Note: If $P = -P$, then $2 \cdot P = P + P = P + (-P) = \mathcal{O}$.)

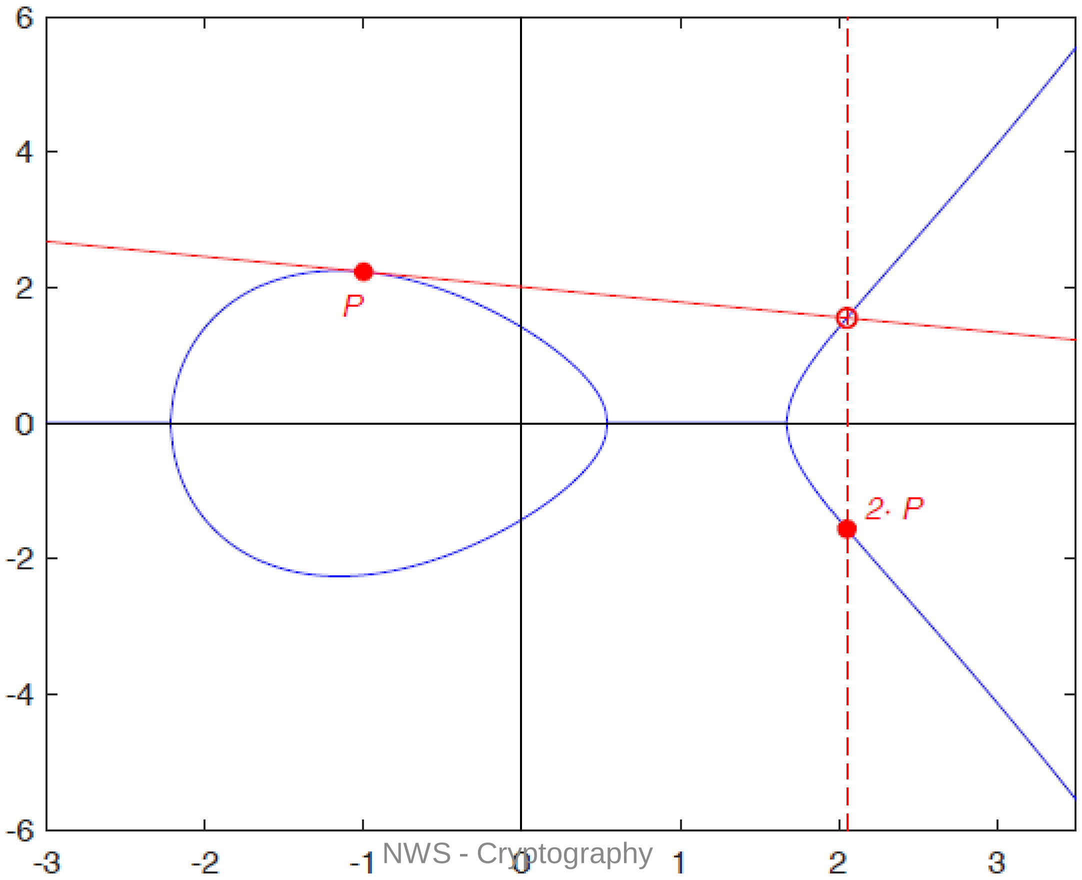
(EC.3) Remark. If $E = E(\mathbb{R})$ is the set of points defined by an elliptic curve over the real numbers, the addition of two points $P, Q \in E \setminus \{\mathcal{O}\}$ has a nice geometric interpretation:

- If $P \neq \pm Q$, the line through P and Q intersects the curve at exactly one more point. The negative (reflection at the x-axis) of this point is $P + Q$. An example of the addition of two points of the elliptic curve defined by $y^2 = x^3 - 4x + 2$ over $\mathbb{R}$ is shown in the following figure.



(EC.3) Remark. If $E = E(\mathbb{R})$ is the set of points defined by an elliptic curve over the real numbers, the addition of two points $P, Q \in E \setminus \{\mathcal{O}\}$ has a nice geometric interpretation:

- If $P = Q$, the tangent line through P can be considered instead of the line through P and Q. An example of the doubling of a point on the elliptic curve defined by $y^2 = x^3 - 4x + 2$ over $\mathbb{R}$ is shown in the following figure.



(EC.4) Order of $E(\mathbb{F}_q)$ (Hasse's theorem). If $E(\mathbb{F}_q)$ is any elliptic curve defined over some field $E(\mathbb{F}_q)$, then:

$$\big||E(\mathbb{F}_q)| - (q+1)\big| \;\leq\; 2\sqrt{q}$$

(EC.5) Group structure of $E(\mathbb{F}_q)$. Either $E(\mathbb{F}_q)$ is cyclic or $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ with $n_2 \,|\, n_1$ and $n_2 \,|\, (q-1)$.

(EC.6) Elliptic Curve Diffie-Hellman (ECDH) key agreement scheme.

With respect to a fixed elliptic curve $E(\mathbb{F}_q)$ and some fixed point $G \in E(\mathbb{F}_q)$ of big order $o(G)$ Alice generates a key pair by choosing some positive random integer $k_{A,priv} < o(G)$ and calculates the corresponding public key $k_{A,pub} = k_{A,priv} \cdot G$.

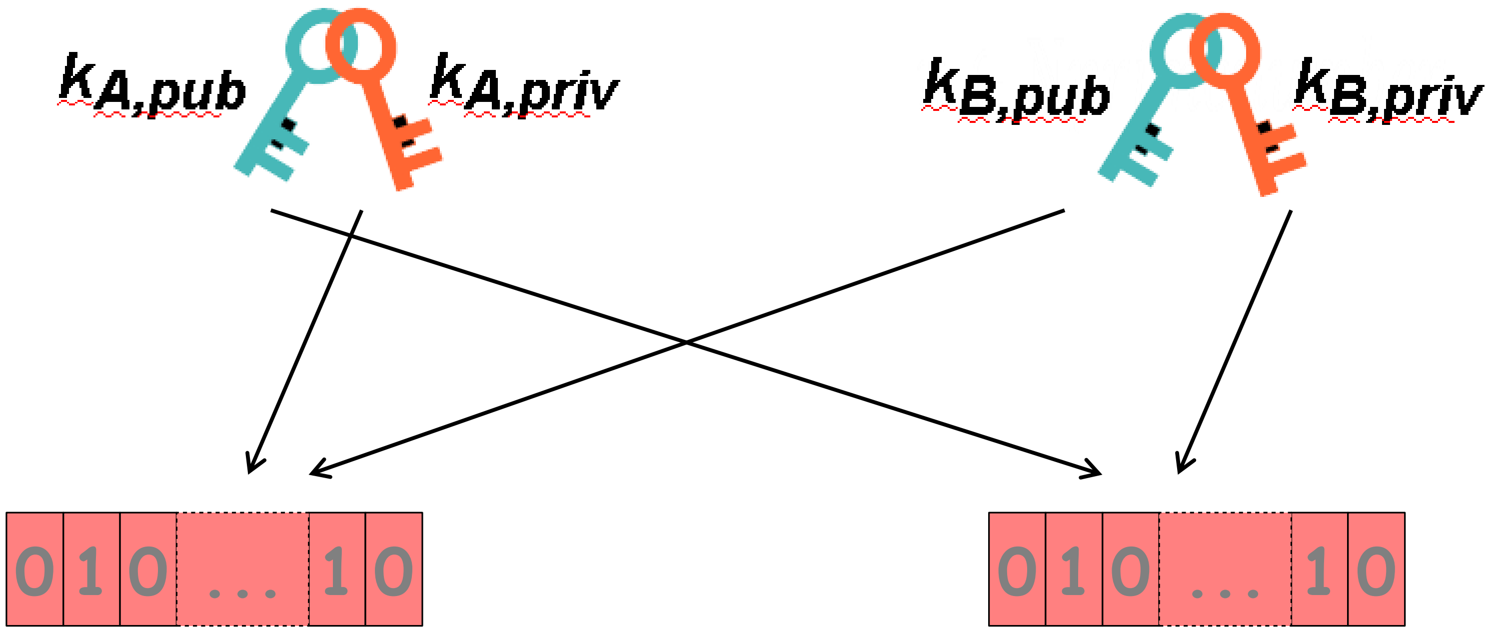
Similarly, Bob generates his key pair $(k_{B,priv}, k_{B,pub})$ by randomly choosing a positive integer $k_{B,priv} < o(G)$ and calculating $k_{B,pub} = k_{B,priv} \cdot G$.

After exchanging their public keys, Alice and Bob can both compute the point

$$S \;=\; (x_S, y_S) \;:=\; k_{A,priv} \cdot k_{B,pub} \;=\; k_{A,priv} \cdot k_{B,priv} \cdot G \;=\; k_{B,priv} \cdot k_{A,pub}$$

and use x_S as their shared secret.

Elliptic Curve Diffie-Hellman (EC-DH) key derivation



$k = F(k_{A,priv}, k_{B,pub})$ $k = F(k_{B,priv}, k_{A,pub})$