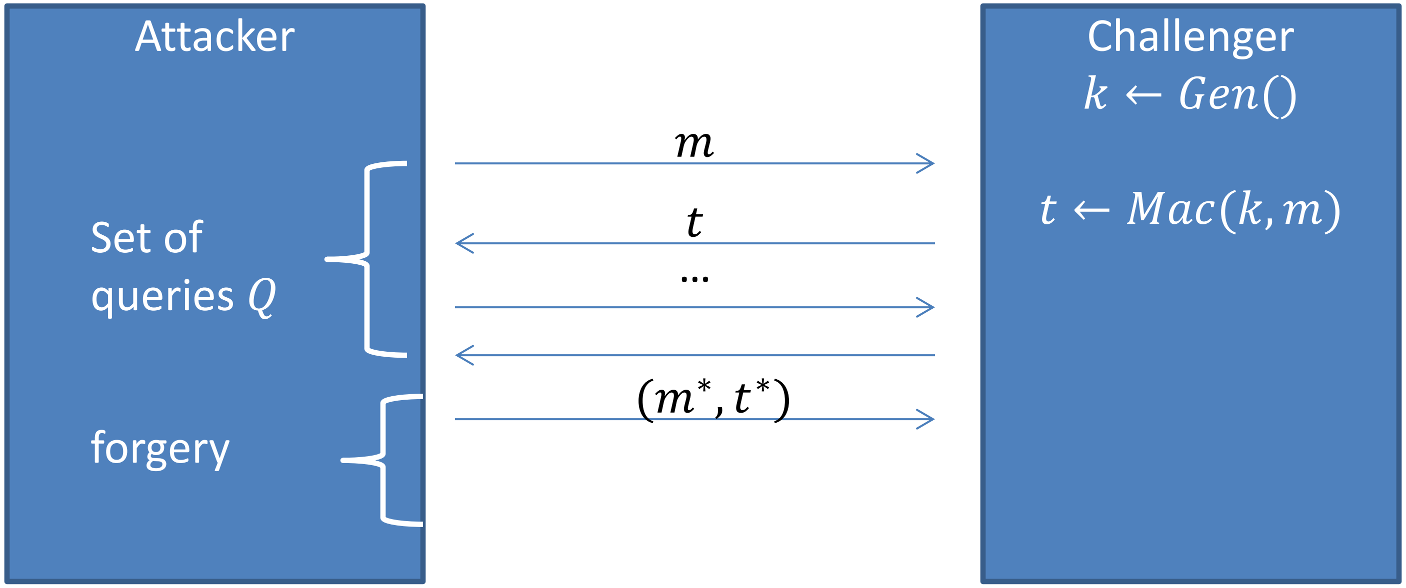
Message Authentication Codes

Definition: A message authentication code (MAC) consists of three probabilistic polynomial-time algorithms $(Gen, Mac, Vrfy)$ such that:

1. The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a key k with $|k| \geq n$.
2. The tag-generation algorithm Mac takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a tag t.
   $t \leftarrow Mac_k(m)$.
3. The deterministic verification algorithm $Vrfy$ takes as input a key k, a message m, and a tag t. It outputs a bit b with $b = 1$ meaning valid and $b = 0$ meaning invalid.
   $b \coloneqq Vrfy_k(m, t)$.

It is required that for every n, every key k output by $Gen(1^n)$, and every $m \in \{0,1\}^*$, it holds that $Vrfy_k\big(m, Mac_k(m)\big) = 1$.

Existential Unforgeability under CMA



Attacker "wins" if :

1. $m^* \notin Q$
2. $Vrfy(k, m^*, t^*) = 1$

Security Requirement: Any efficient attacker wins with probability at most *negligible*

CBC-MAC

Let F be a pseudorandom function, and fix a length function ℓ. The basic CBC-MAC construction is as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$, do the following:
  1. Parse m as $m = m_1, \ldots, m_\ell$ where each m_i is of length n.
  2. Set $t_0 \coloneqq 0^n$. Then, for $i = 1\ to\ \ell$:

     Set $t_i \coloneqq F_k(t_{i-1} \oplus m_i)$.

  Output t_ℓ as the tag.
- $Vrfy$: on input a key $k \in \{0,1\}^n$, a message m, and a tag t, do: If m is not of length $\ell(n) \cdot n$ then output 0. Otherwise, output 1 if and only if $t = Mac_k(m)$.

CBC-MAC

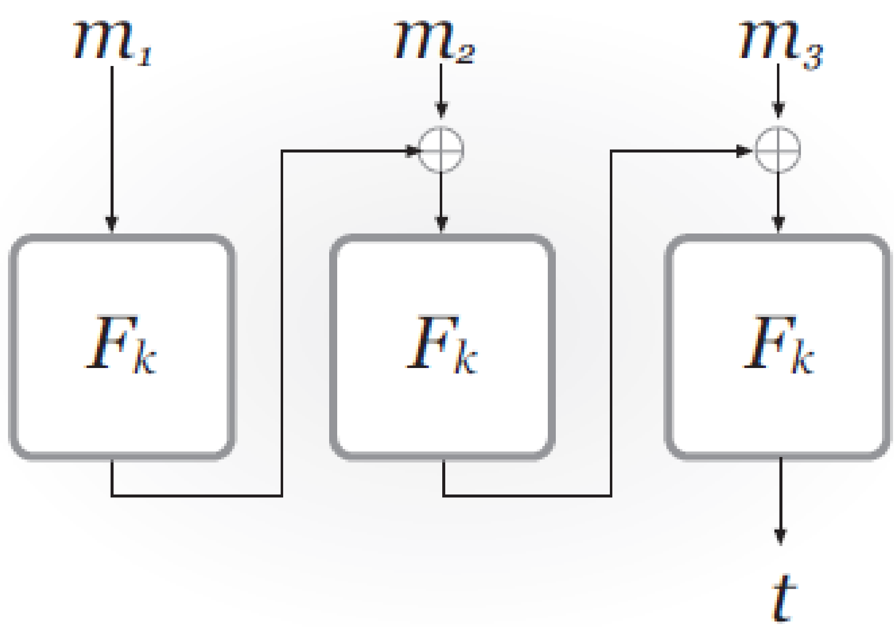
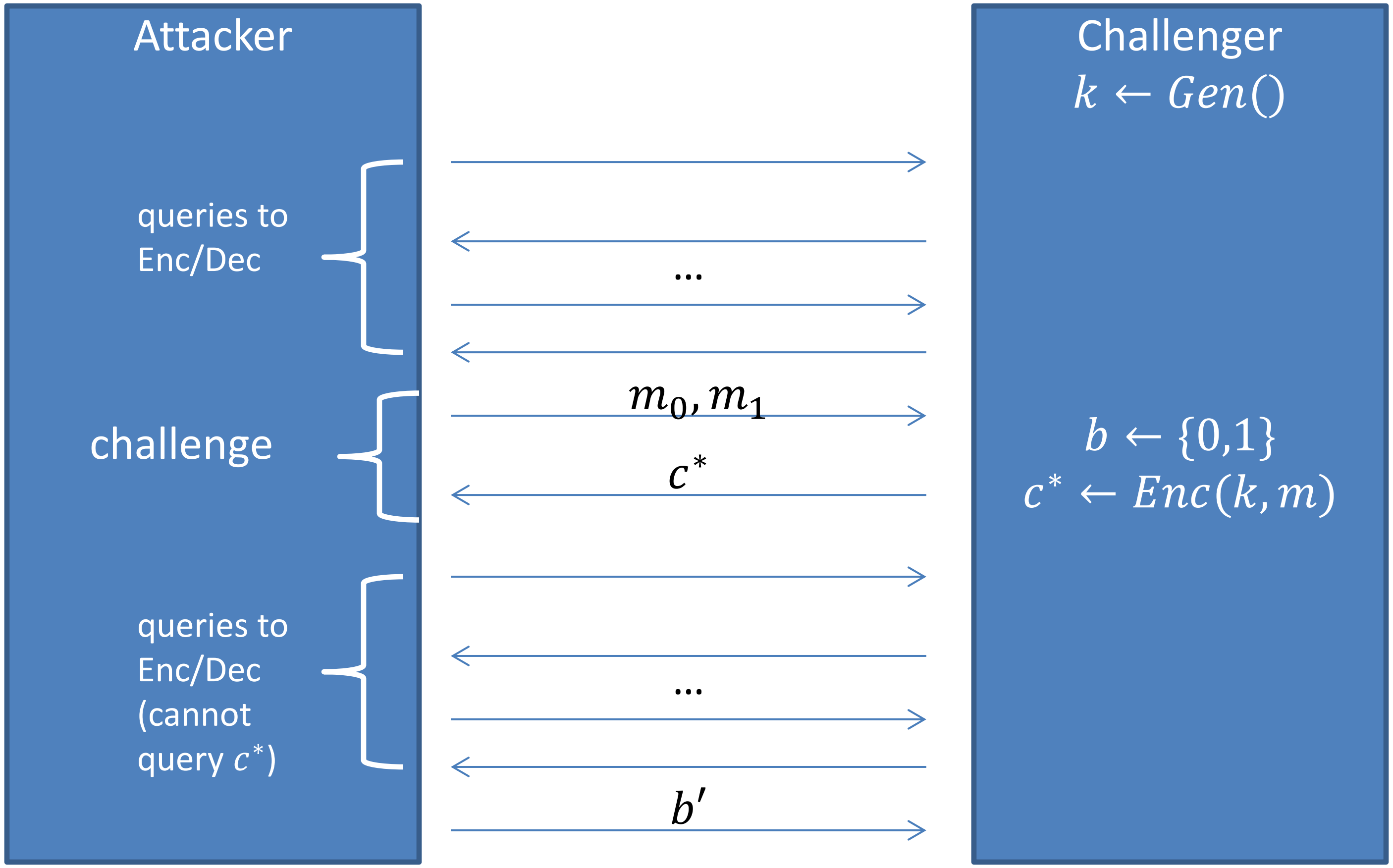


FIGURE 4.1: Basic CBC-MAC (for fixed-length messages).

Authenticated Encryption

- Definition: A private-key encryption scheme is an authenticated encryption scheme if it is CCA-secure and unforgeable.

CCA Security



Attacker "wins" if $b' = b$.

CCA Security: Any efficient attacker wins with probability at most $\frac{1}{2} + negligible$

Generic Constructions

Encrypt-and-authenticate

Encryption and message authentication are computed independently in parallel.

$$c \leftarrow Enc_{k_E}(m) \qquad t \leftarrow Mac_{k_M}(m)$$

$$\langle c, t \rangle$$

Is this secure? NO!

Authenticate-then-encrypt

Here a MAC tag t is first computed, and then the message and tag are encrypted together.

$$t \leftarrow Mac_{k_M}(m) \qquad c \leftarrow Enc_{k_E}(m||t)$$

c is sent

Is this secure? NO! Encryption scheme may not be CCA-secure.

Encrypt-then-authenticate

The message m is first encrypted and then a MAC tag is computed over the result

$$c \leftarrow Enc_{k_E}(m) \quad t \leftarrow Mac_{k_M}(c)$$

$$\langle c, t \rangle$$

Is this secure? YES! As long as the MAC is strongly secure.

Examples

Consider multiplication modulo 23.

23 is a "safe prime" since 23 = 2*11 +1, where 11 is a prime.

Consider the following cyclic group generated by 2:

$2^0 \ mod\ 23$	1
$2^1 \ mod\ 23$	2
$2^2 \ mod\ 23$	4
$2^3 \ mod\ 23$	8
$2^4 \ mod\ 23$	16
$2^5 \ mod\ 23$	$32 \rightarrow 9$
$2^6 \ mod\ 23$	18
$2^7 \ mod\ 23$	$36 \rightarrow 13$
$2^8 \ mod\ 23$	$26 \rightarrow 3$
$2^9 \ mod\ 23$	6
$2^{10} \ mod\ 23$	12
$2^{11} \ mod\ 23$	$24 \rightarrow 1$

Actually, all of 2, 4, 8, 16, 9, 18, 13, 3, 6, 12 are generators and each of them raised to the 11 will be equal to 1 modulo 23.

Key Agreement

The key-exchange experiment ${KE^{eav}}_{A,\Pi}(n)$:

1. Two parties holding 1^n execute protocol Π. This results in a transcript $trans$ containing all the messages sent by the parties, and a key k output by each of the parties.
2. A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$ set $\hat{k} \coloneqq k$, and if $b = 1$ then choose $\hat{k} \in \{0,1\}^n$ uniformly at random.
3. A is given $trans$ and $\hat{k}$, and outputs a bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise.

Definition: A key-exchange protocol Π is secure in the presence of an eavesdropper if for all ppt adversaries A there is a negligible function neg such that

$$\Pr\left[{KE^{eav}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + neg(n).$$

Diffie-Hellman Key Exchange

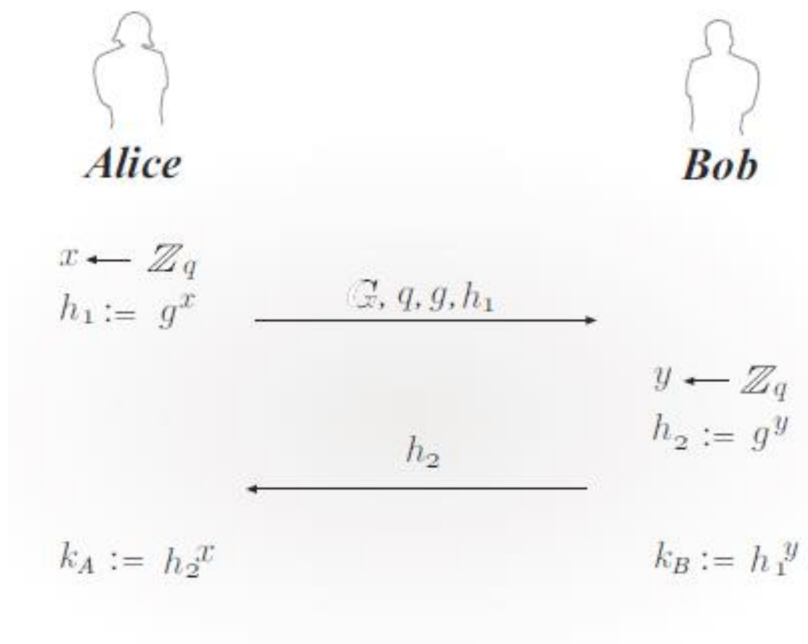


FIGURE 10.2: The Diffie-Hellman key-exchange protocol.

Example for the group we saw above with generator $g = 2$:

Alice:

$x \leftarrow \{0, \dots, 10\}$

Say $x = 8$

$2^8 \ mod \ 23 = 3$

Bob:

$y \leftarrow \{0, \dots, 10\}$

Say $y = 5$

Alice → Bob: 3

$2^5 \ mod \ 23 = 9$

Bob → Alice: 9

Alice:

Output: $9^8 \ mod \ 23$

$= 3^{16} \ mod \ 23$

$= 3^{11} \cdot 3^5 \ mod \ 23$

$= 1 \cdot 3^5 \ mod \ 23$

$= 27 \cdot 9 \ mod \ 23$

$= 4 \cdot 9 \ mod \ 23$

$= 36 \ mod \ 23 = \mathbf{13}$

Bob:

Output: $3^5 \ mod \ 23$

$= 27 \cdot 9 \ mod \ 23$

$= 4 \cdot 9 \ mod \ 23$

$= 36 \ mod \ 23 = \mathbf{13}$