CMAC

V. Balasubramanian

Introduction

- An alternative method is to construct MACs from block ciphers. The most popular approach in practice is to use a block cipher such as AES in cipher block chaining (CBC) mode.

CBC MAC

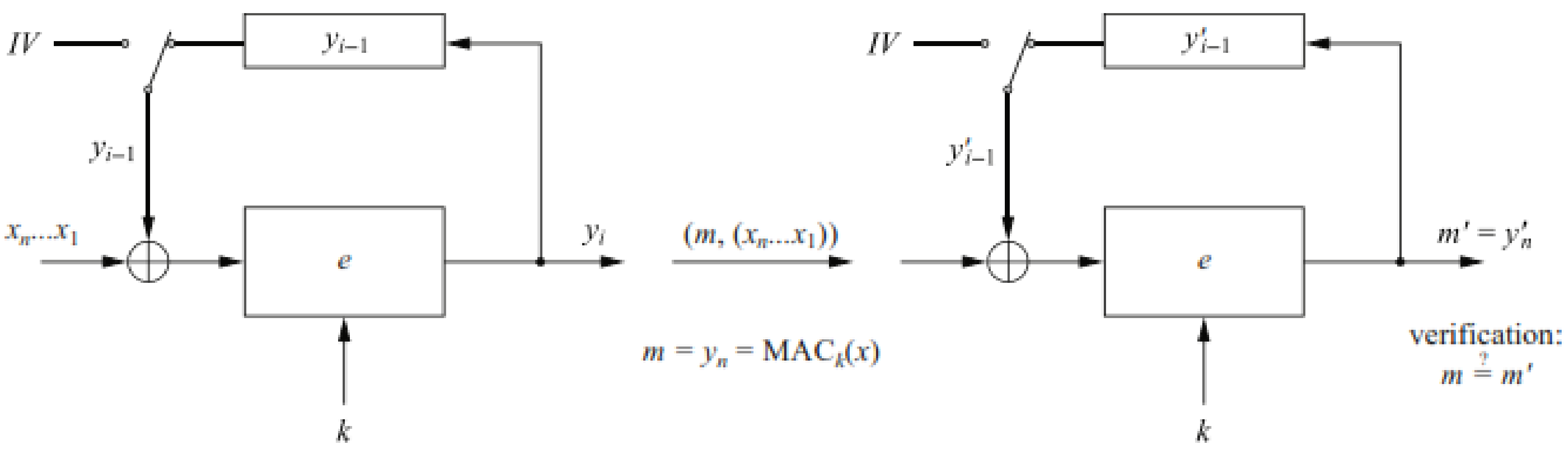
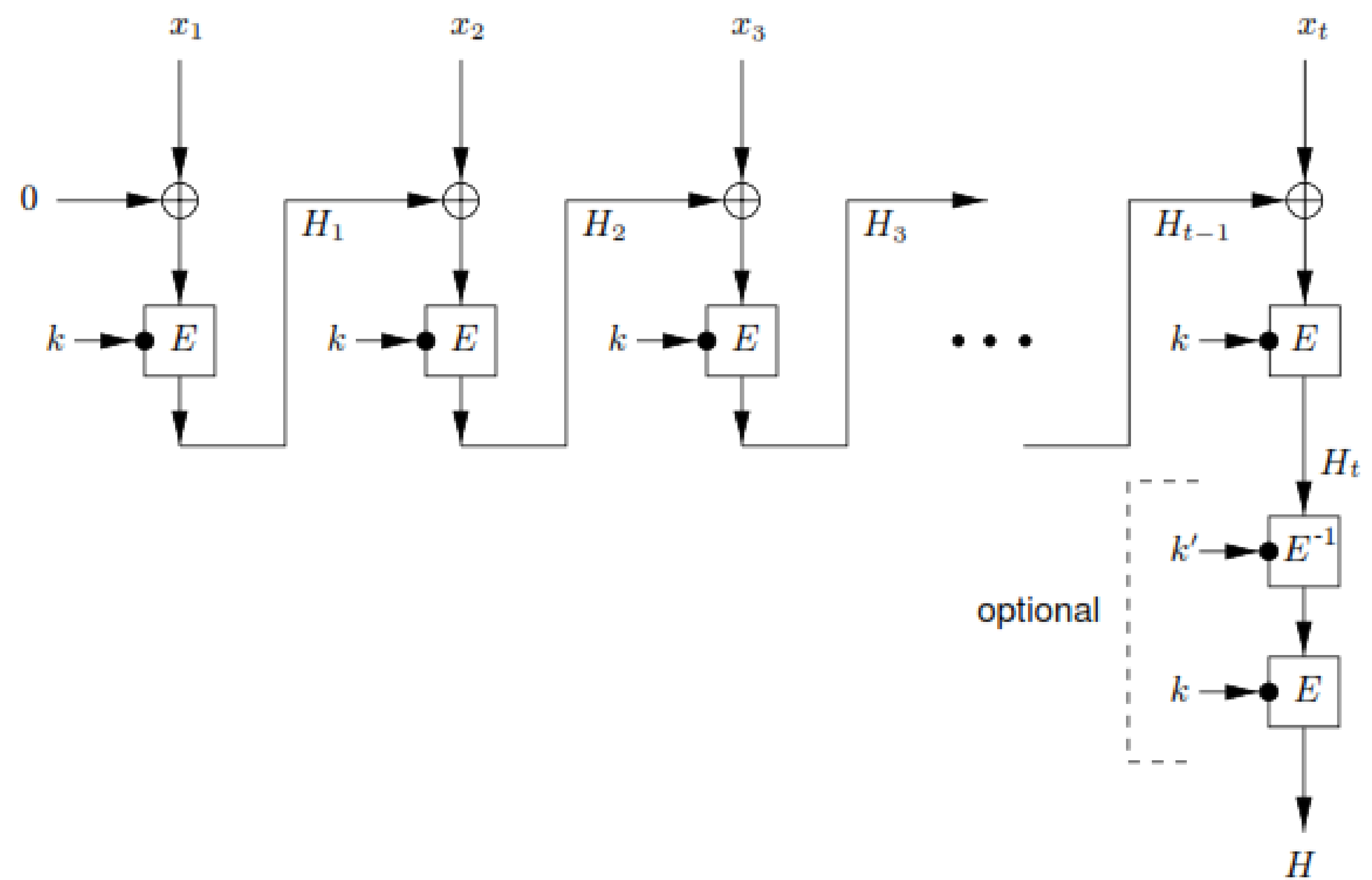


Fig. 12.3 MAC built from a block cipher in CBC mode



x_1
x_2
x_3
x_t
0
H_1
H_2
H_3
H_{t-1}
k
E
k'
E^{-1}
H_t
optional
H

- When DES is used as the block cipher E, n =64in what follows, and the MAC key is a 56-bit DES key.

Algorithm CBC-MAC

INPUT: data x; specification of block cipher E; secret MAC key k for E.
OUTPUT: n-bit MAC on x (n is the blocklength of E).

1. *Padding and blocking.* Pad x if necessary (e.g., using Algorithm 9.30). Divide the padded text into n-bit blocks denoted $x_1, \ldots, x_t$.
2. *CBC processing.* Letting E_k denote encryption using E with key k, compute the block H_t as follows: $H_1 \leftarrow E_k(x_1)$; $H_i \leftarrow E_k(H_{i-1} \oplus x_i)$, $2 \leq i \leq t$. (This is standard cipher-block-chaining, $IV = 0$, discarding ciphertext blocks $C_i = H_i$.)
3. *Optional process to increase strength of MAC.* Using a second secret key $k' \neq k$, optionally compute: $H'_t \leftarrow E^{-1}_{k'}(H_t)$, $H_t \leftarrow E_k(H'_t)$. (This amounts to using two-key triple-encryption on the last block; see Remark 9.59.)
4. *Completion.* The MAC is the n-bit block H_t.

MAC Generation

For the generation of a MAC, we have to divide the message x into blocks $x_i, i = 1,...,n$. With the secret key k and an initial value IV, we can compute the first iteration of the MAC algorithm as

$$y_1 = e_k(x_1 \oplus IV),$$

where the IV can be a public but random value. For subsequent message blocks we use the XOR of the block x_i and the previous output y_{i-1} as input to the encryption algorithm:

$$y_i = e_k(x_i \oplus y_{i-1}).$$

Finally, the MAC of the message $x = x_1 x_2 x_3 ... x_n$ is the output y_n of the last round:

$$m = \mathrm{MAC}_k(x) = y_n$$

In contrast to CBC encryption, the values $y_1, y_2, y_3, \ldots, y_{n-1}$ are *not* transmitted. They are merely internal values which are used for computing the final MAC value $m = y_n$.

MAC Verification

- As with every MAC, verification involves simply repeating the operation that were used for the MAC generation. For the actual verification decision we have to compare the computed MAC m’= *m.*

Schnorr Digital Signature

- Scheme is based on discrete logarithms
- Minimizes the message-dependent amount of computation required to generate a signature
  - Multiplying a 2*n*-bit integer with an *n*-bit integer
- Main work can be done during the idle time of the processor
- Based on using a prime modulus *p*, with *p* – 1 having a prime factor *q* of appropriate size
  - Typically *p* is a 1024-bit number, and *q* is a 160-bit number

Key Generation

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $a^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

Signature Generation

A user with private key s and public key v generates a signature as follows.

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$$e = \mathrm{H}(M \| x)$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Verification

Any other user can verify the signature as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = \mathrm{H}(M \| x')$.

To see that the verification works, observe that

$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod{p}$$

Hence, $\mathrm{H}(M \| x') = \mathrm{H}(M \| x)$.