CISC 468: CRYPTOGRAPHY

LESSON 17: MESSAGE AUTHENTICATION CODES

Furkan Alaca

READINGS

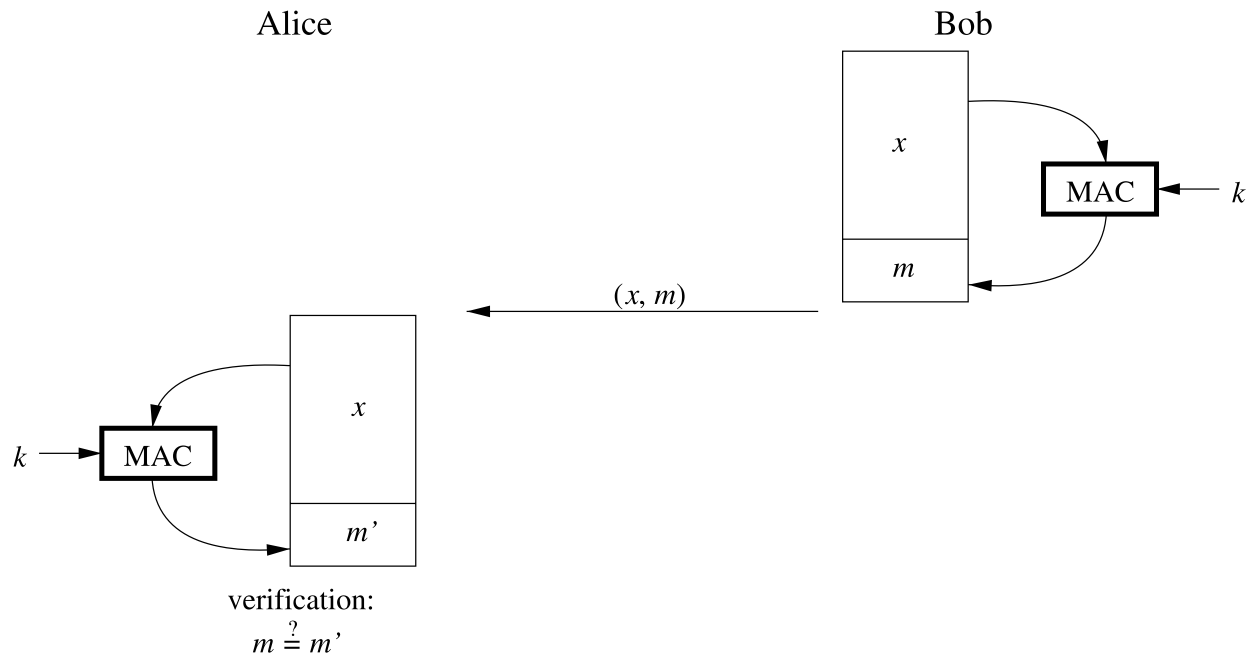
- Section 12.1: Principles of Message Authentication Codes, Paar & Pelzl
- Section 12.2: MACs from Hash Functions: HMAC, Paar & Pelzl
- Section 12.3: MACs from Block Ciphers: CBC-MAC, Paar & Pelzl

PRINCIPLES OF MESSAGE AUTHENTICATION CODES (MACS)

- MACs are the symmetric-key analog of digital signatures
  - They append an authentication tag to a message
  - The tag is generated and verified using the same key k
- We use the notation $m = MAC_k(x)$ to denote the computation of a MAC m on a message x using the key k

MOTIVATION FOR USING MACS

- A MAC allows Alice and Bob to detect any in-transit manipulation of a message x
- Since both Alice and Bob share the key k, Alice can recompute the MAC to verify m



PROPERTIES OF MACS

1. *Message integrity*: Any manipulation of the message during transit will be detected by the receiver.
2. *Message authentication*: The receiver is assured of the origin of the message.

If an attacker modifies the message in transit, they cannot recompute the MAC without possession of the secret key. For the same reason, they cannot forge a new message either.

PROPERTIES OF MACS

3. MACs are based on *symmetric keys*, so the signing and verifying parties must share a secret key.

Non-repudiation is not provided, since both the sender and receiver can compute the same MAC; thus a neutral third party cannot judge which of the two parties constructed a message.

PROPERTIES OF MACS

4. A MAC function should accept *input messages of arbitrary length*.
5. The MAC function should *generate a fixed-length authentication tag*.

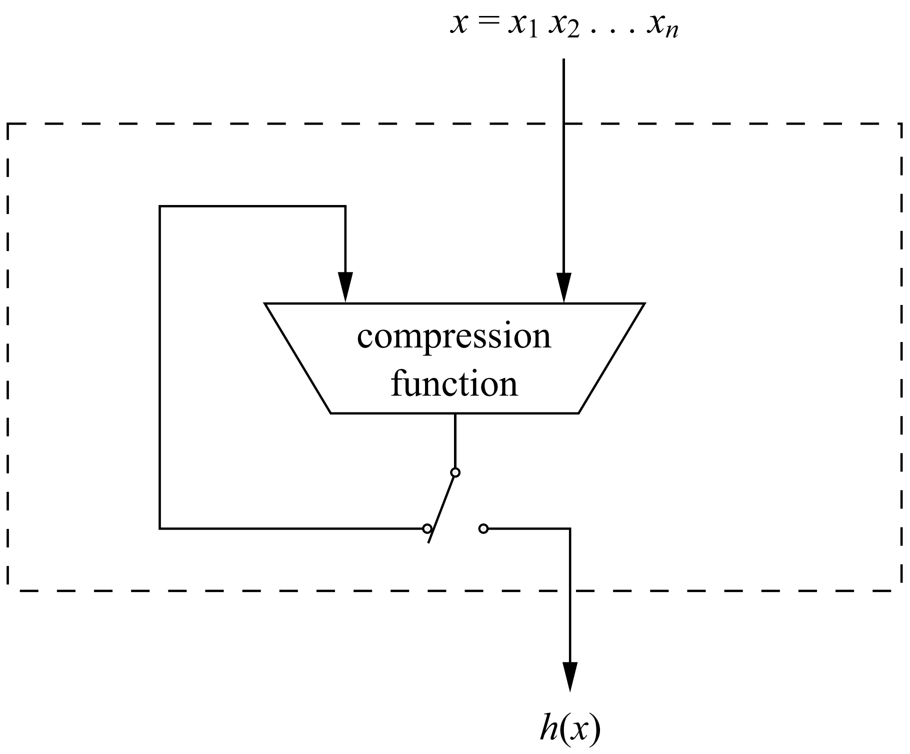
MACs are also much faster than digital signatures, since they are based on either hash functions or block ciphers.

MACS FROM HASH FUNCTIONS

- A MAC can be constructed by pairing a secret key with a hash function
- Not all hash-based MAC constructions are secure—we will see three constructions that are considered cryptographically weak, and a third construction, called HMAC, that is secure

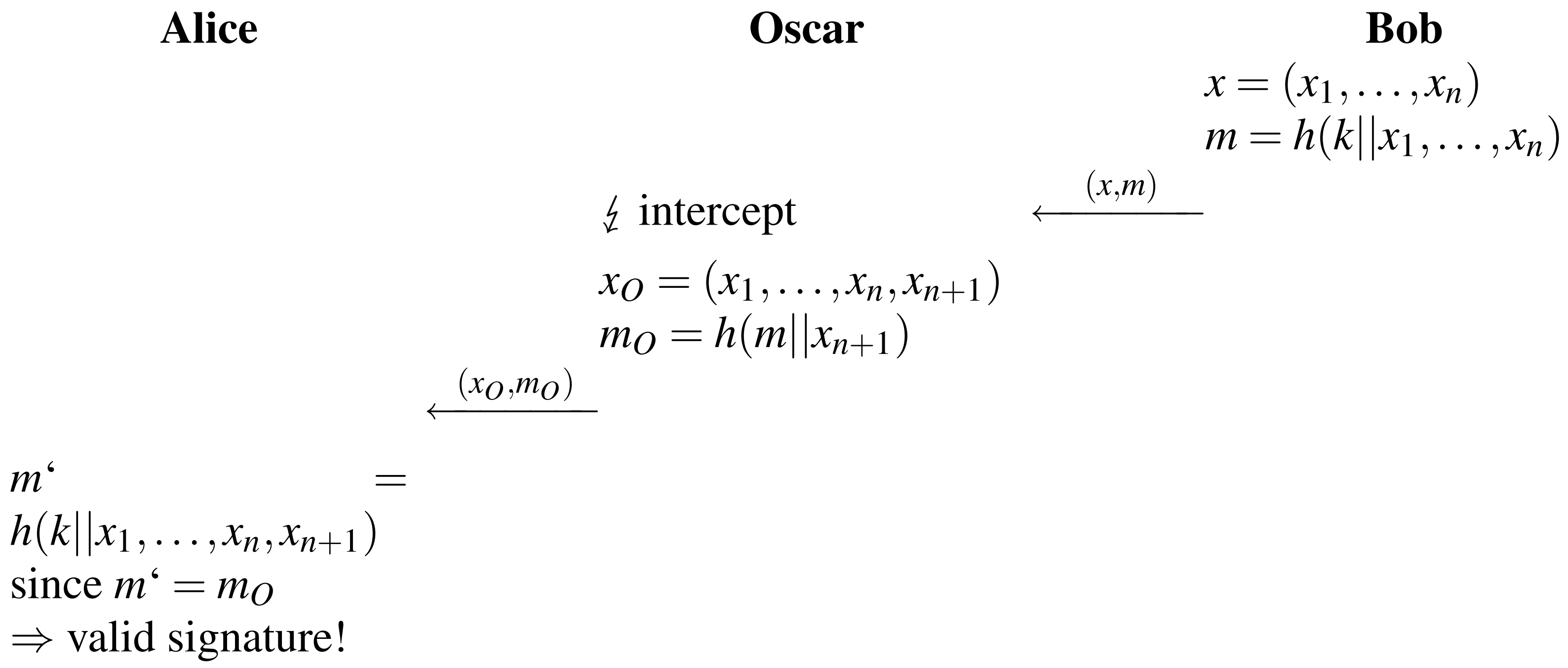
HASH-BASED MACS: SECRET PREFIX

- Consider MACs realized as $m = MAC_k(x) = h(k\|x)$
- Assume that h uses a Merkle-Damgard construction



HASH-BASED MACS: ATTACK ON SECRET PREFIX

This construction can be susceptible to *length-extension attacks* where an attacker can append a new block x_{n+1} to the message and re-compute the MAC without knowledge of the key



Alice
Oscar
Bob
$x = (x_1, \dots, x_n)$
$m = h(k||x_1, \dots, x_n)$
(x,m)
↯ intercept
$x_O = (x_1, \dots, x_n, x_{n+1})$
$m_O = h(m||x_{n+1})$
(x_O,m_O)
$m‘ = h(k||x_1, \dots, x_n, x_{n+1})$
since $m‘ = m_O$
$\Rightarrow$ valid signature!

HASH-BASED MACS: SECRET SUFFIX

- Consider MACs realized as $m = MAC_k(x) = h(x||k)$
- Assume that h uses a Merkle-Damgard construction
- Assume that the hash function is preimage resistant and second-preimage resistant, but not collision resistant

HASH-BASED MACS: ATTACK ON SECRET SUFFIX

- Offline (i.e., before initiating any communication), an attacker may perform a birthday attack to find two messages x and x_O such that $h(x) = h(x_O)$
- Then, if a communicating party can be legitimately made to send a message-MAC pair (x, m) the attacker can replace it with (x_O, m) without being detected

ENVELOPE MACS

- A more secure construction is to use a prefix and a suffix, i.e., $m = MAC_k(x) = h(k||x||k)$
- But this method is subject to an attack that can break its security with effort similar to that of breaking the secret suffix method
  - The attack is considered only *certificational* if the underlying hash function is collision-resistant

HMAC

HMAC is a more secure method, standardized in RFC 2104

- HMAC is widely used
- It does not require the underlying hash function to be collision resistant
- Thus, while SHA-1 is unsuitable (offering <80-bit security) for use with digital signatures, it is still considered acceptable (offering 128-bit security) for HMAC
  - Refer to NIST SP 800-57 Part 1 Rev. 5, Table 3: Maximum security strengths for hash and hash-based functions

HMAC: DETERMINE k^+

Let k be the secret key and b the input block size of the hash function. We must first determine k^+:

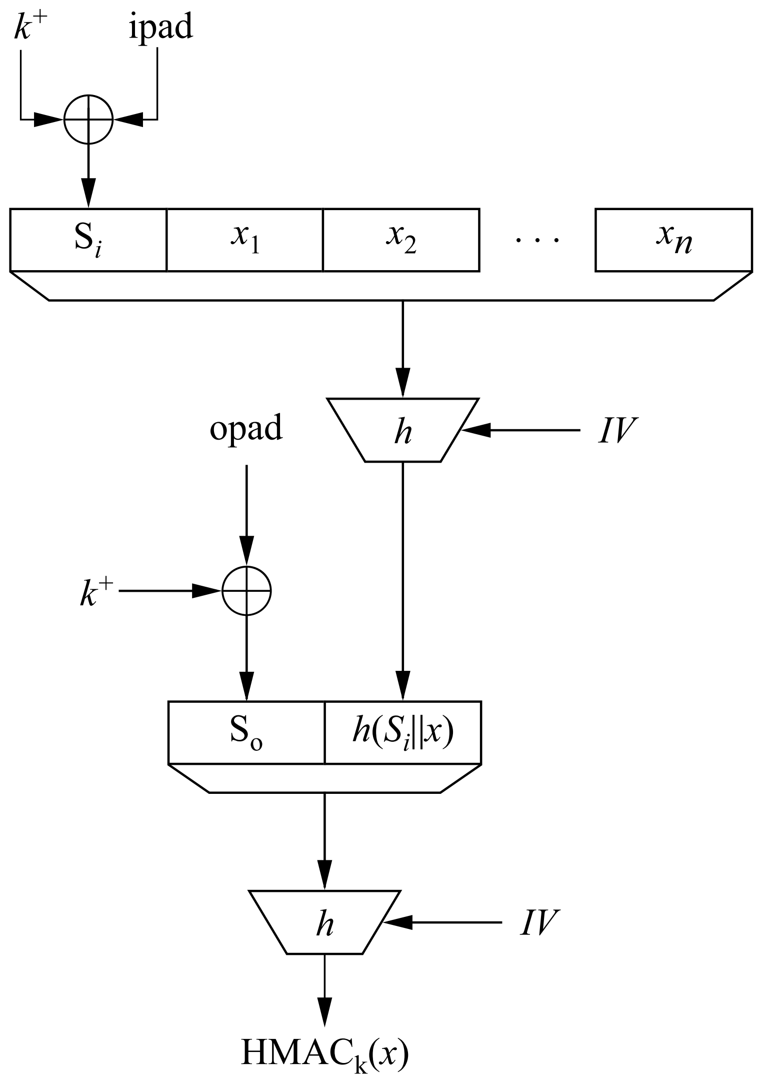
1. If k is b bits, then $k^+ = k$. Then, skip to step 4.
2. If k is longer than b, hash k and pad the result with enough zeros to obtain a k^+ that is b bits. Then, skip to step 4.
3. If k is shorter than b, pad it with enough zeros to obtain a k^+ that is b bits.

HMAC: THE INNER HASH

4. Compute the XOR $S_i = k^+ \oplus ipad$, where $ipad$ is a b-bit repetition of the 8-bit sequence 00110110, i.e.,:

$$ipad = 00110110 \| \ldots \| 00110110$$

5. Prefix S_i to the message blocks $x_1, x_2, \ldots, x_n$ to be authenticated.
6. Compute $h(S_i \| x)$.

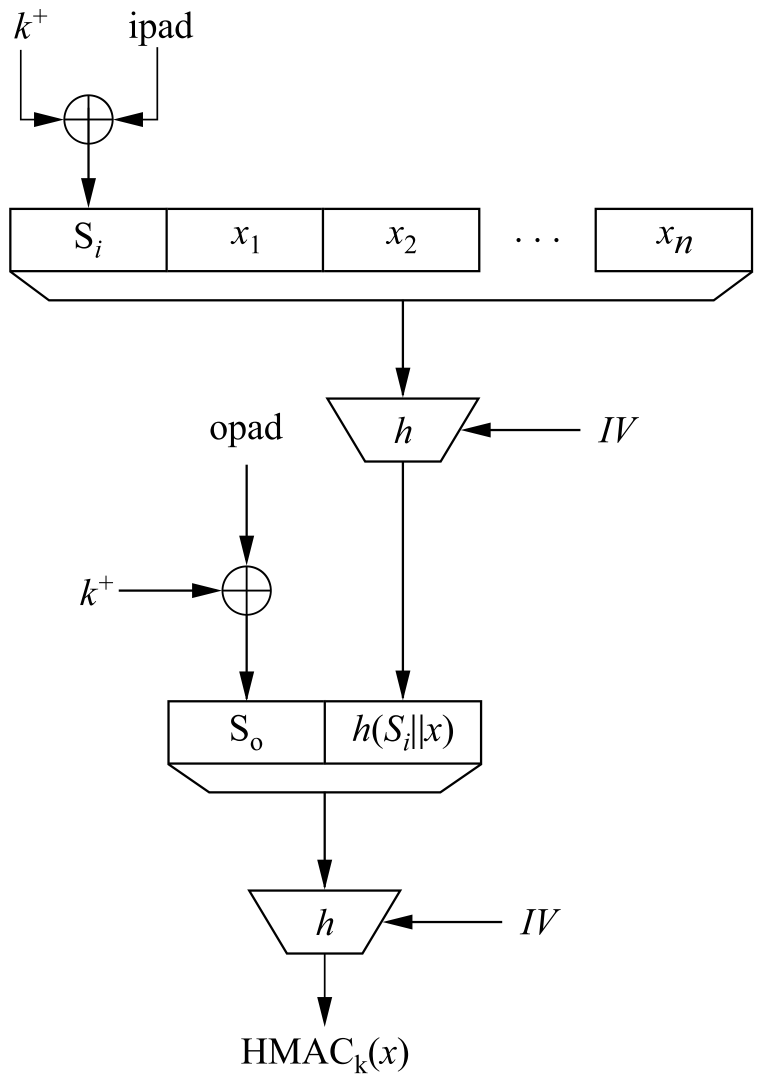


HMAC: THE OUTER HASH

7. Compute the XOR $S_o = k^+ \oplus opad$, where $opad$ is a b-bit repetition of the 8-bit sequence 01011100, i.e.,:

$$ipad = 01011100 \| \dots \| 01011100$$

8. Prefix S_o to the output of the inner hash, $h(S_i \| x)$.
9. Compute $h(S_o \| h[S_i \| x])$.

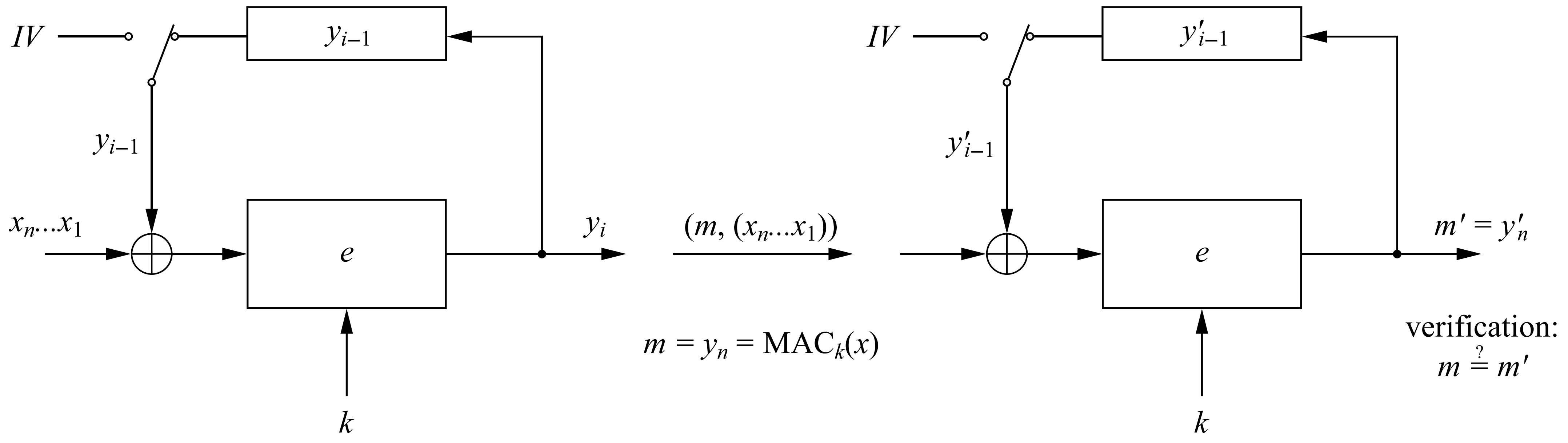


MACS FROM BLOCK CIPHERS: CBC-MAC

- MACs can also be constructed from block ciphers such as AES
- A popular approach is to use the CBC mode of operation
  - In practice, a more secure variant called CMAC is preferred
- To generate the MAC, the sender provides the message x and a randomly-generated IV (which can be public) as input to the block cipher in CBC mode

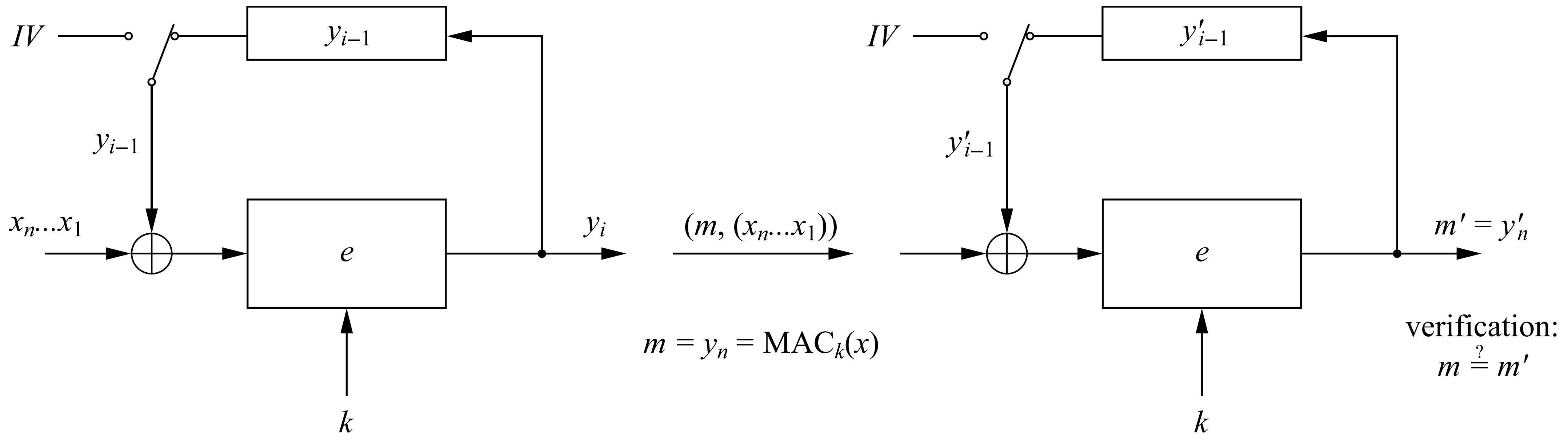
CBC-MAC: MAC GENERATION

Unlike encryption, the MAC m is the output of the last round, y_n



CBC-MAC: MAC VERIFICATION

As with any other MAC, the receiver recomputes the MAC and compares it with the received MAC value m for verification



TRUNCATED OUTPUT

- HMAC tag is typically truncated to $\frac{n}{2}$ bits, where n is the hash length (see RFC 2104)
- Rationale/advantages:
  - Since attacker cannot construct message-tag pairs without knowledge of key, strong collision resistance is unnecessary, so the tag can be truncated to the "birthday attack bound"
  - Less information on the hash result available to attacker
  - Reduces bandwidth requirements
- Disadvantage: Less bits for the attacker to predict, i.e., higher success probability of replacing message without invalidating tag

SECURITY REQUIREMENTS FOR MAC VS. ENCRYPTION

- If an encryption algorithm is broken today, data that was previously encrypted may be exposed
- If a MAC algorithm is broken, there will be no impact on data authenticated in the past
  - Even short (64- or 32-bit) tags may sometimes be considered acceptable for some applications if the forgery of some fraction of the data may be tolerable (see NIST 800-38D, Appendix C)