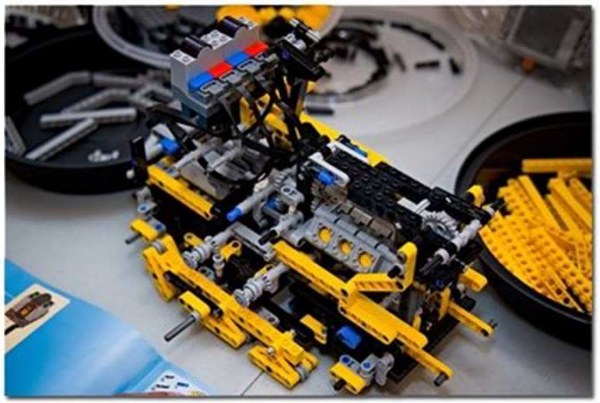
Message authentication and authenticated encryption

Security Protocols (bmevihim132)

Dr. Levente Buttyán
associate professor
BME Hálózati Rendszerek és Szolgáltatások Tanszék
Lab of Cryptography and System Security (CrySyS)
buttyan@hit.bme.hu, buttyan@crysys.hu

Outline

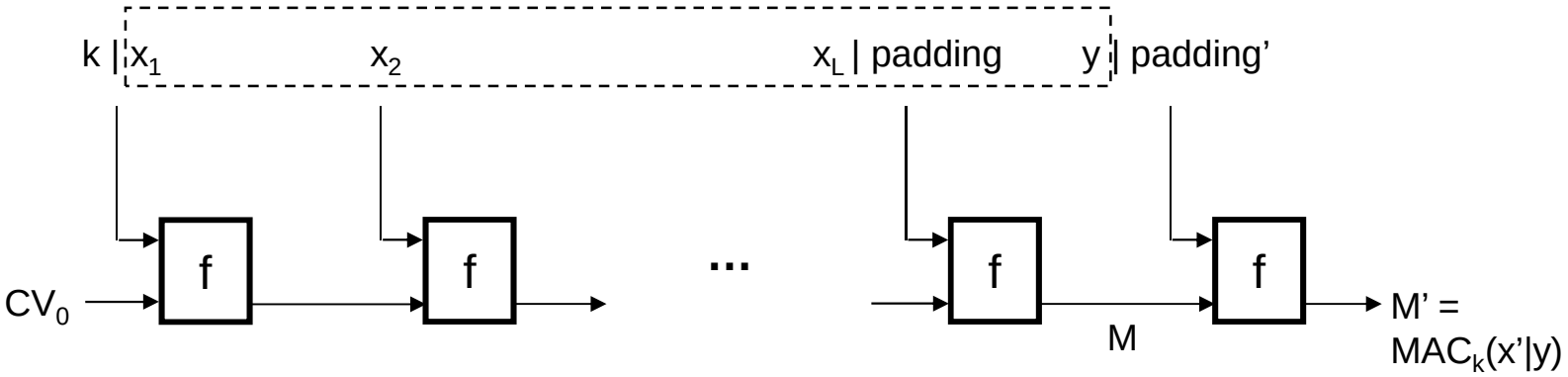
- message authentication
  - naïve constructions
  - HMAC
  - CBC-MAC
  - CMAC
- authenticated encryption
  - generic combinations
  - CCM
  - OCB
  - GCM



Secret prefix method

$MAC_k(x) = H(k|x)$

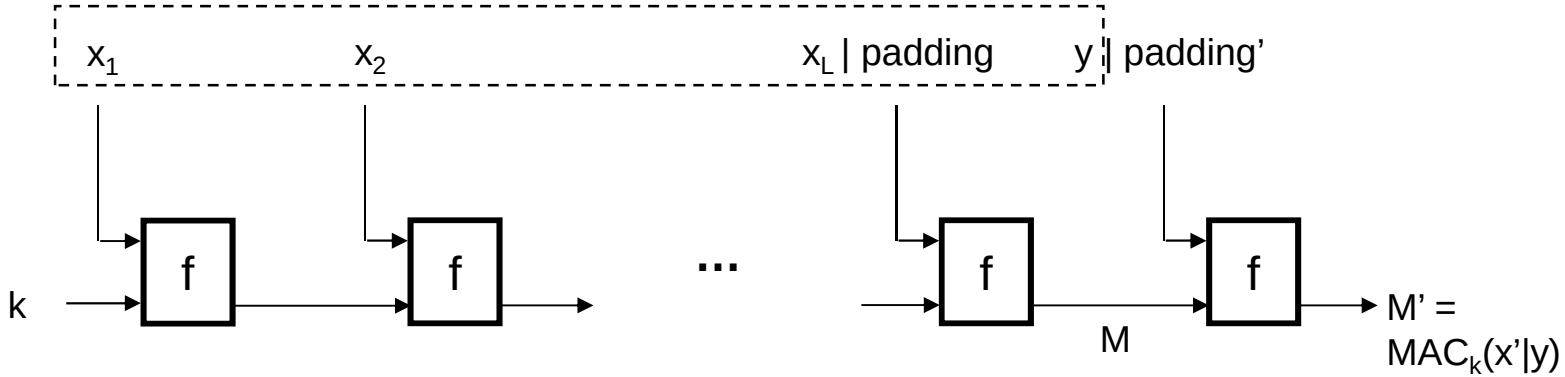
- insecure !
  - assume an attacker knows the MAC on x: M = H(k|x)
  - he can produce the MAC on x'|y as M' = f(M,y), where x' is x with padding and f is the compression function of H



A similar mistake

$MAC_k(x) = H_k(x)$

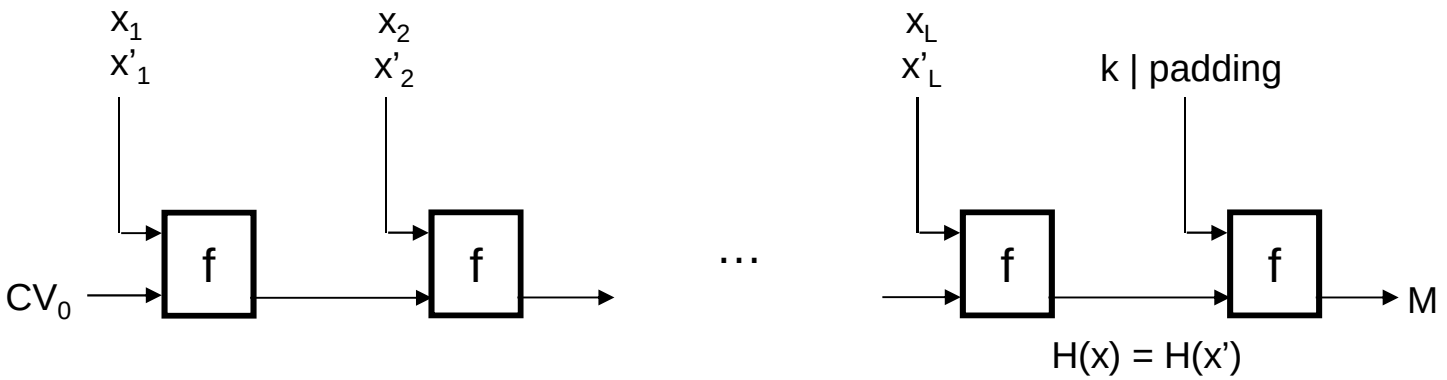
where $H_k(.)$ is $H(.)$ with $CV_0 = k$



Secret suffix method

$MAC_k(x) = H(x|k)$

- insecure if H is not collision resistant
  - using a birthday attack, the attacker finds two inputs x and x' such that H(x) = H(x') (can be done off-line without the knowledge of k)
  - then obtaining the MAC M on one of the inputs, say x, allows the attacker to forge a text-MAC pair (x', M)
- weakness
  - key is involved only in the last step



Envelop method

$MAC_K(x) = H(k|x|k)$

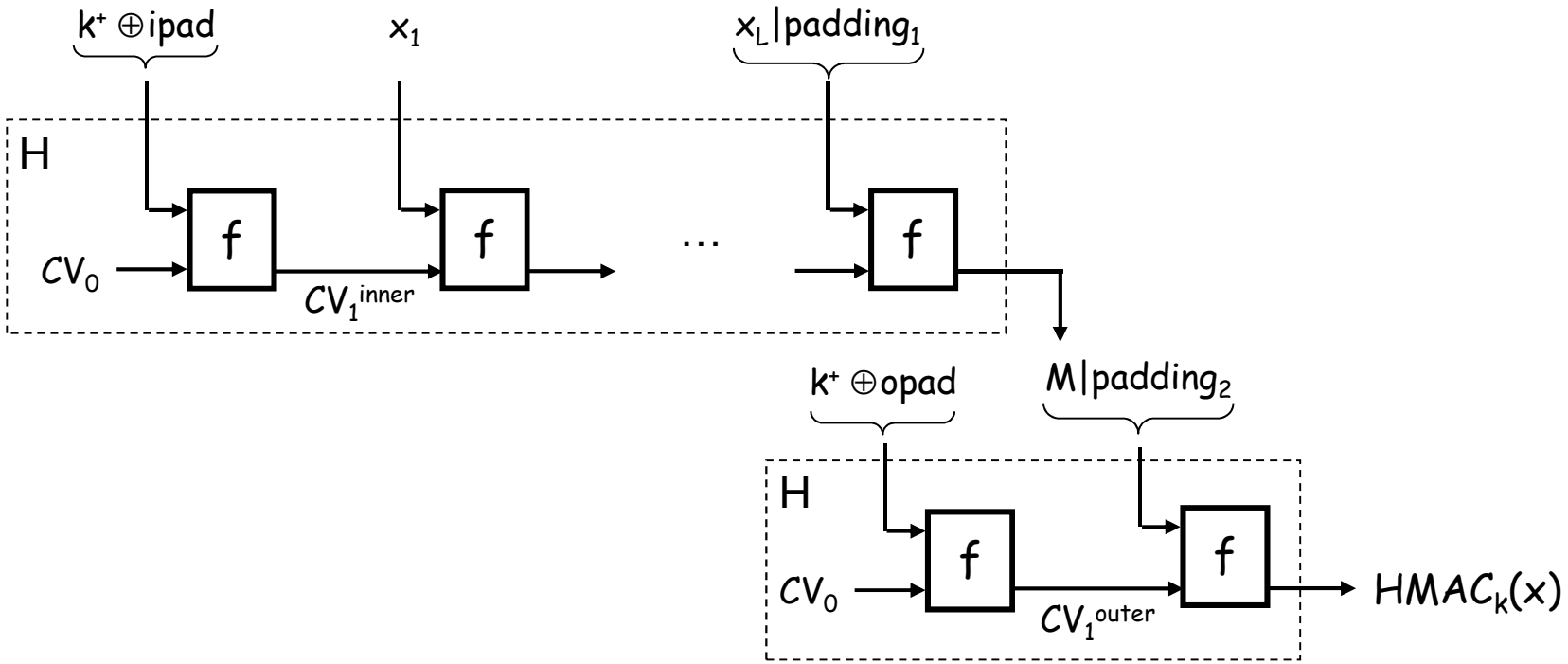
- a key recovery attack has been discovered on this scheme (requiring 2^{64} text-MAC pairs for MD5 with 128-bit key)
- although, not really practical, the attack still represents an architectural flaw

HMAC

$HMAC_k(x) = H((k^+ \oplus opad) \mid H((k^+ \oplus ipad) \mid x))$

where

- h is a hash function with input block size b and output size n
- k^+ is k padded with 0s to obtain a length of b bits
- ipad is 00110110 repeated b/8 times
- opad is 01011100 repeated b/8 times

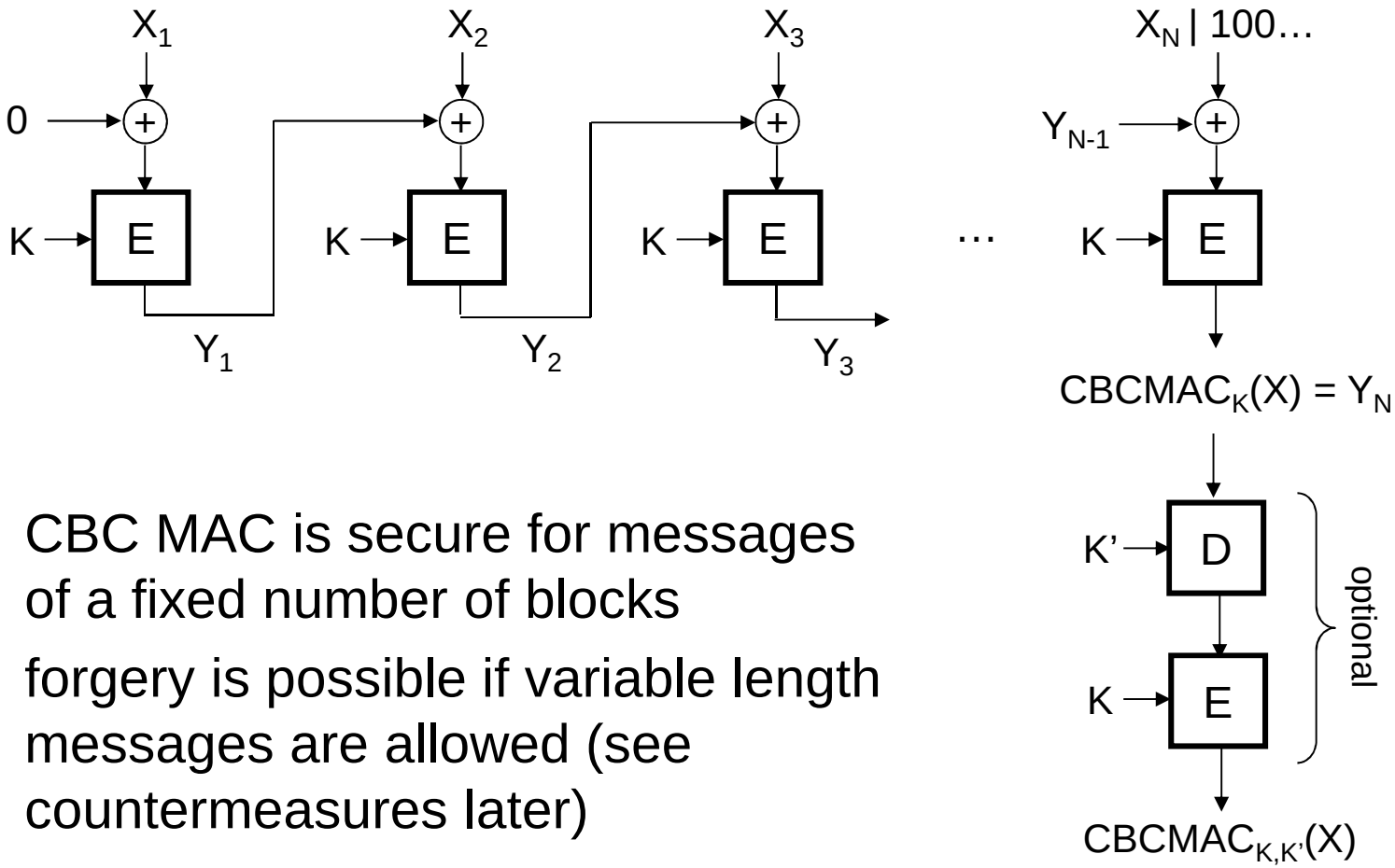


Encrypted hash

$MAC_K(x) = E_K(H(x))$

- off-line search for messages with colliding MAC values is possible here without the knowledge of k → H must be collision resistant !
- collision resistant hash functions usually have larger output size than the block size of the block cipher → which mode to use to encrypt the hash?
- two messages having the same hash value will have the same MAC value under all keys

CBC-MAC

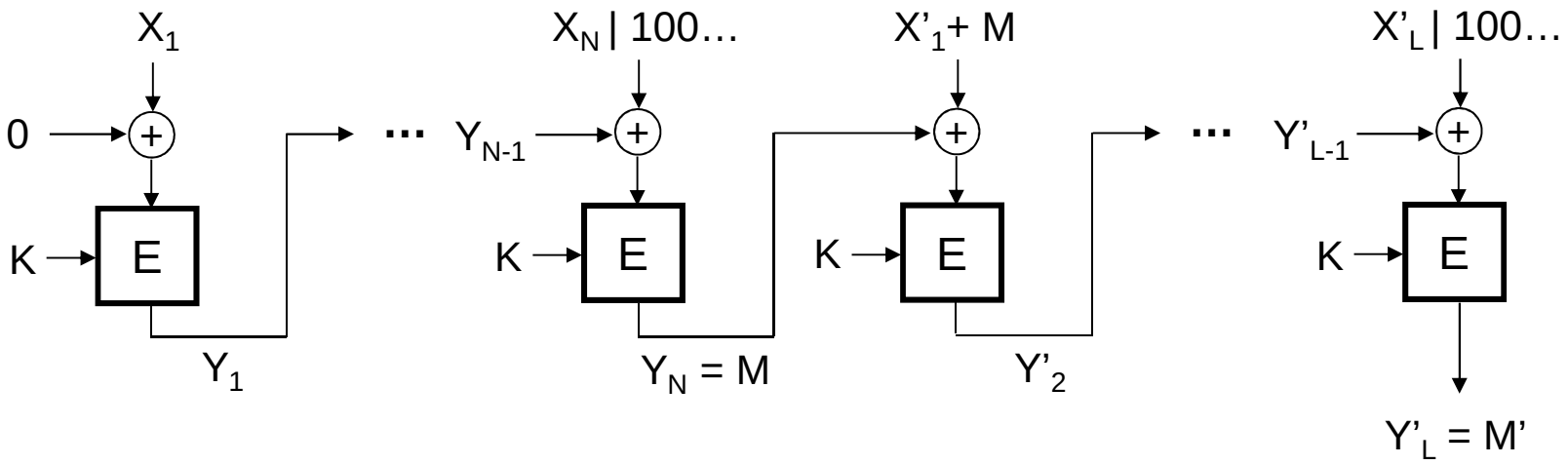


- CBC MAC is secure for messages of a fixed number of blocks
- forgery is possible if variable length messages are allowed (see countermeasures later)

A known-text forgery

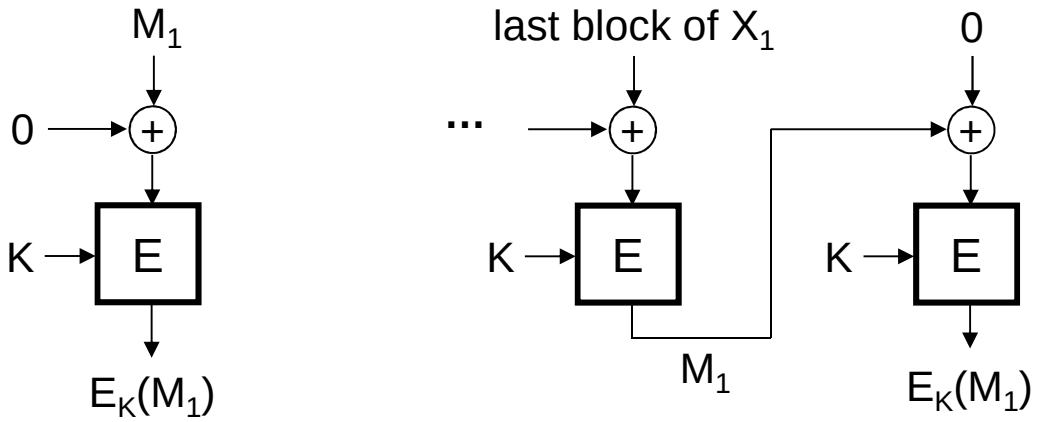
- given two text-MAC pairs (X, M) and (X', M'), a third valid text-MAC pair can be computed as follows:

  $(X \mid 100\ldots \mid X'_1 + M \mid X'_2 \mid \ldots \mid X'_L, M')$



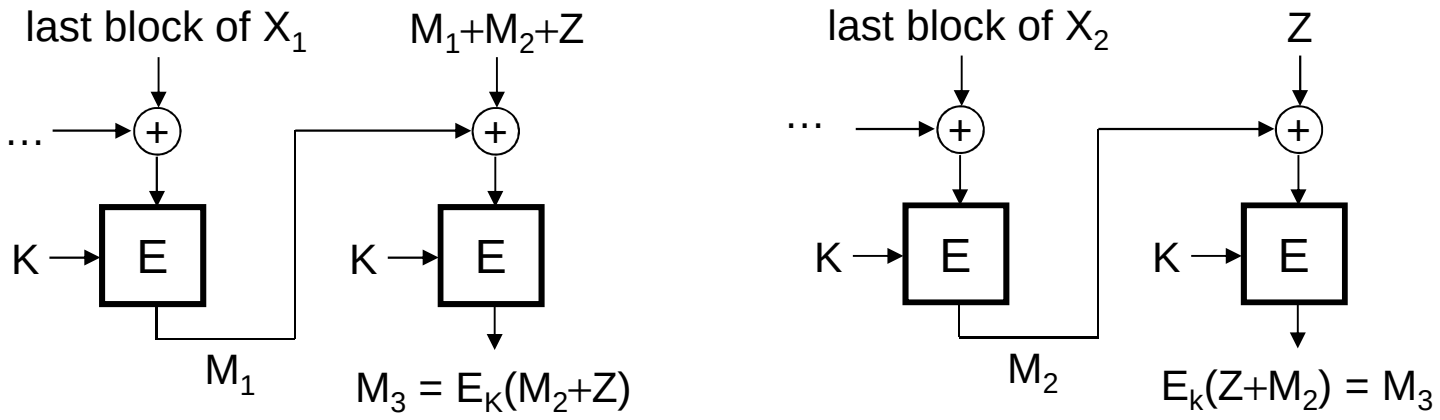
A chosen-text forgery

- given a known text-MAC pair (X_1, M_1)
- request MAC for M_1, receive $M_2 = E_k(M_1 + 0) = E_k(M_1)$
- M_2 is the MAC of the message ($X_1|0$)



Another chosen-text forgery

- given two known text-MAC pairs: (X_1, M_1), (X_2, M_2)
- request a MAC for message $X_1|M_1+M_2+Z$, where Z is an arbitrary block
- receive $M_3 = E_K(M_1+M_2+Z+M_1) = E_K(M_2 + Z)$
- M_3 is also the MAC for message $X_2|Z$



How to use CBC-MAC in practice?

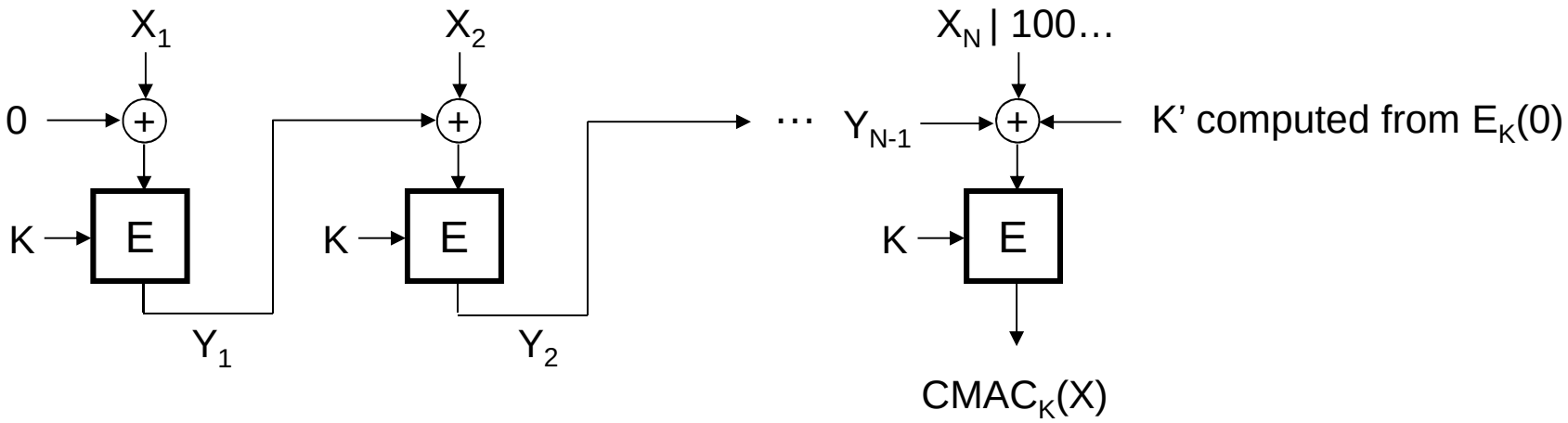
- use the optional final encryption
  - reduces the threat of exhaustive key search (key is (K, K') → key length is doubled)
  - prevents the previously presented existential forgeries
  - has marginal overhead (only last block is encrypted multiple times)

- start the message with an extra block containing the length of the message

- use K to encrypt the length and obtain K' = E_K(length), and use K' as the MAC key (i.e., use message dependent MAC keys)

CMAC

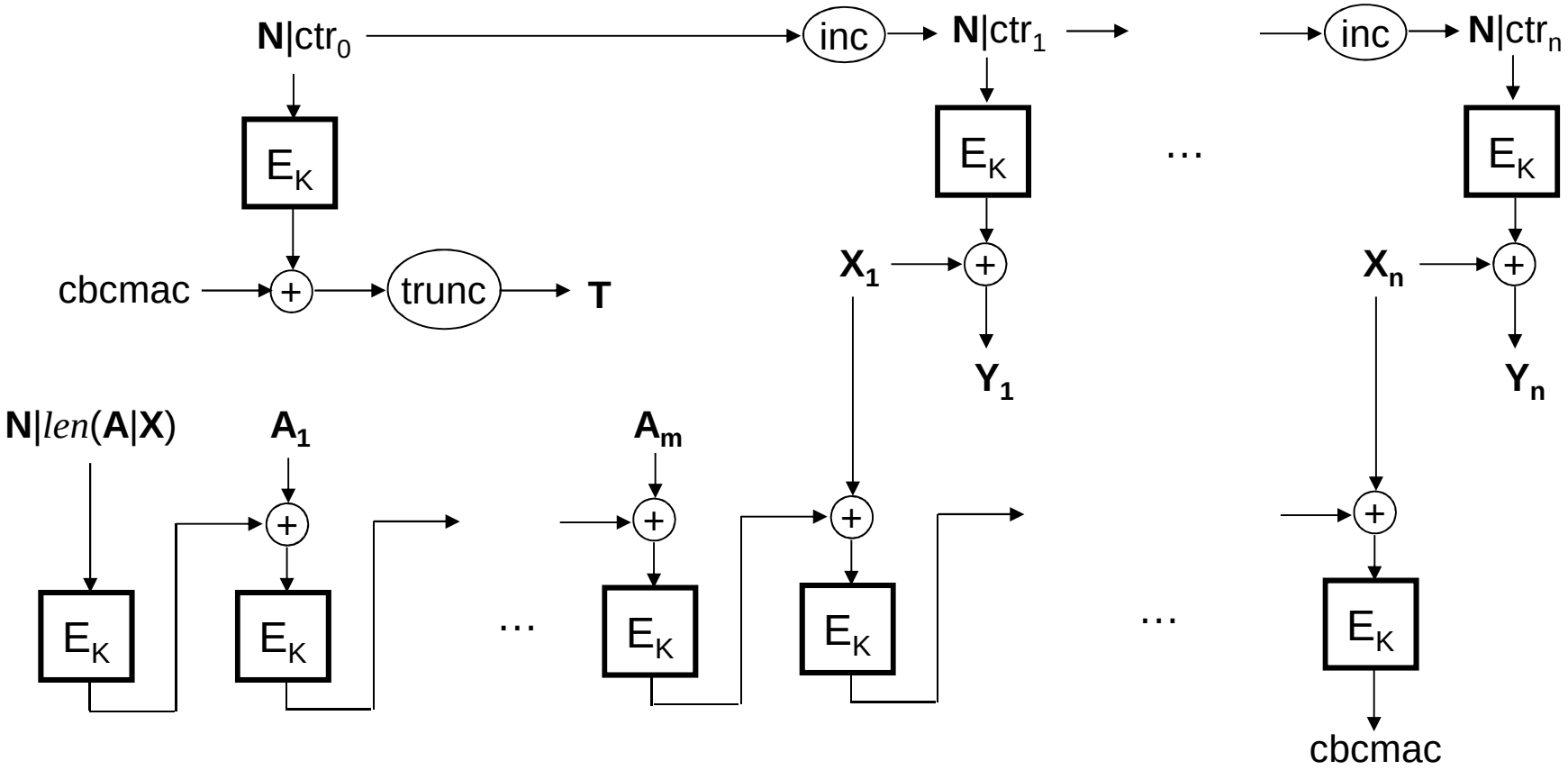
- proposed to fix problems with CBC-MAC



Authenticated encryption schemes

- simultaneously protect the confidentiality and the integrity of a message
- motivations:
  - to prevent chosen-ciphertext attacks (such as the Vaudenay attack)
    - the decryption oracle immediately recognizes improperly constructed ciphertexts and refuses to decrypt them
    - attacker can construct a correct ciphertext only if he already knows the plaintext → decryption oracle becomes useless
  - efficiency (in some cases)
    - needs fewer operations if the message is encrypted and the authentication tag is computed in a single pass
- approaches:
  - specialized schemes (e.g., CCM, OCB, GCM)
  - generic combination of encryption and MAC (but be careful !)
    - $E_{K1}(X|MAC_{K2}(X))$ (check for padding oracle attack !)
    - $E_{K1}(X)|MAC_{K2}(X)$ (check for padding oracle attack !)
    - $E_{K1}(X)|MAC_{K2}(E_{K1}(X))$ (considered to be the most secure approach)

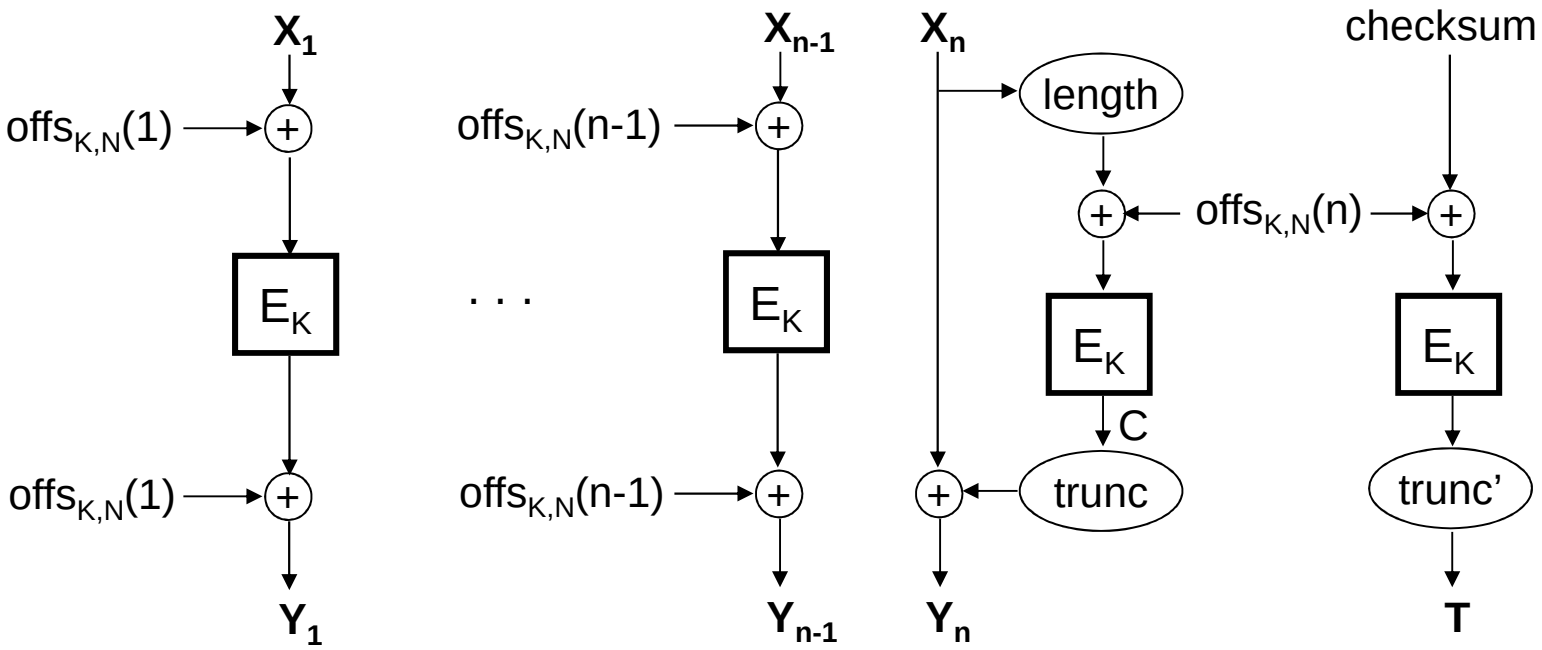
CCM – CTR mode with CBC-MAC



- input: message X, key K, associated data A, nonce N
- output: encrypted message Y, authentication tag T (encrypted CBC MAC value)
- single key
- two passes (2n+m+2 invocation of the block cipher)

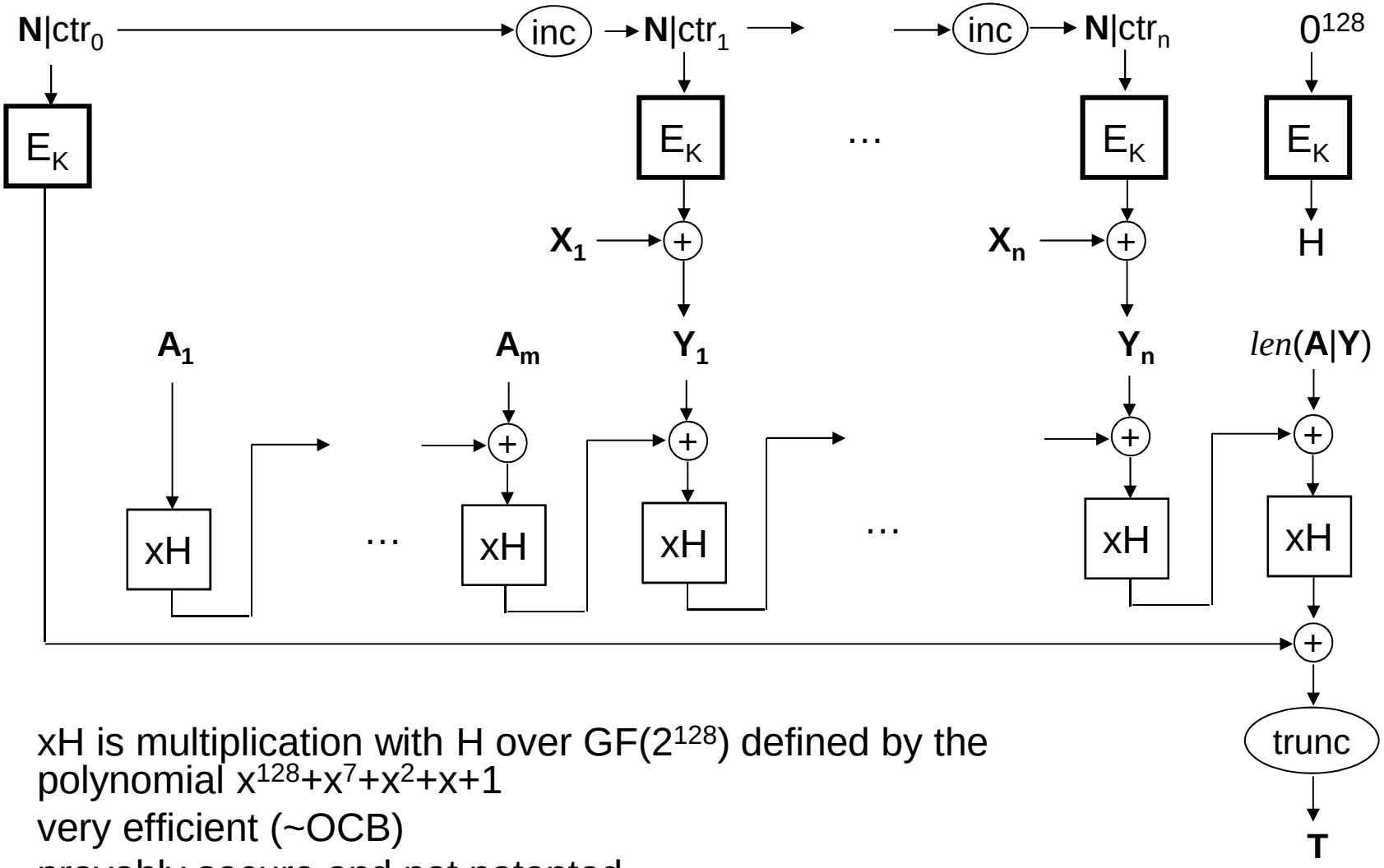
OCB – Offset Code Book

checksum = $X_1 \oplus \ldots \oplus X_{n-1} \oplus Y_n 0^* \oplus C$



- single key, single pass
- no associated data
- preserves plaintext length
- provably secure
- patented

GCM – Galois/Counter Mode



- xH is multiplication with H over GF(2^{128}) defined by the polynomial $x^{128}+x^7+x^2+x+1$
- very efficient (~OCB)
- provably secure and not patented

Summary

- naïve hash based MAC constructions are usually not secure
- better to use standard, well-studied constructions, e.g., HMAC
- CBC-MAC
  - interesting, because it does not need a hash function, but it can use the same block cipher that is used for encryption, anyway
  - existential forgeries against CBC-MAC exist, but there are countermeasures (e.g., adding context data such as message length to the message, multiple encryption of the last block, etc.)
- be careful with generic constructions for authenticated encryption
- specific authenticated encryption modes have some advantages
  - efficiency: the two goals may be achieved in a single pass
  - security: no information is leaked through a padding oracle
- CCM
  - single key, two pass, associated data, not patented
- OCB
  - single key, single pass, no associated data, patented
- GCM
  - single key, single pass, associated data, not patented
  - performance is comparable to OCB