



# Cryptography

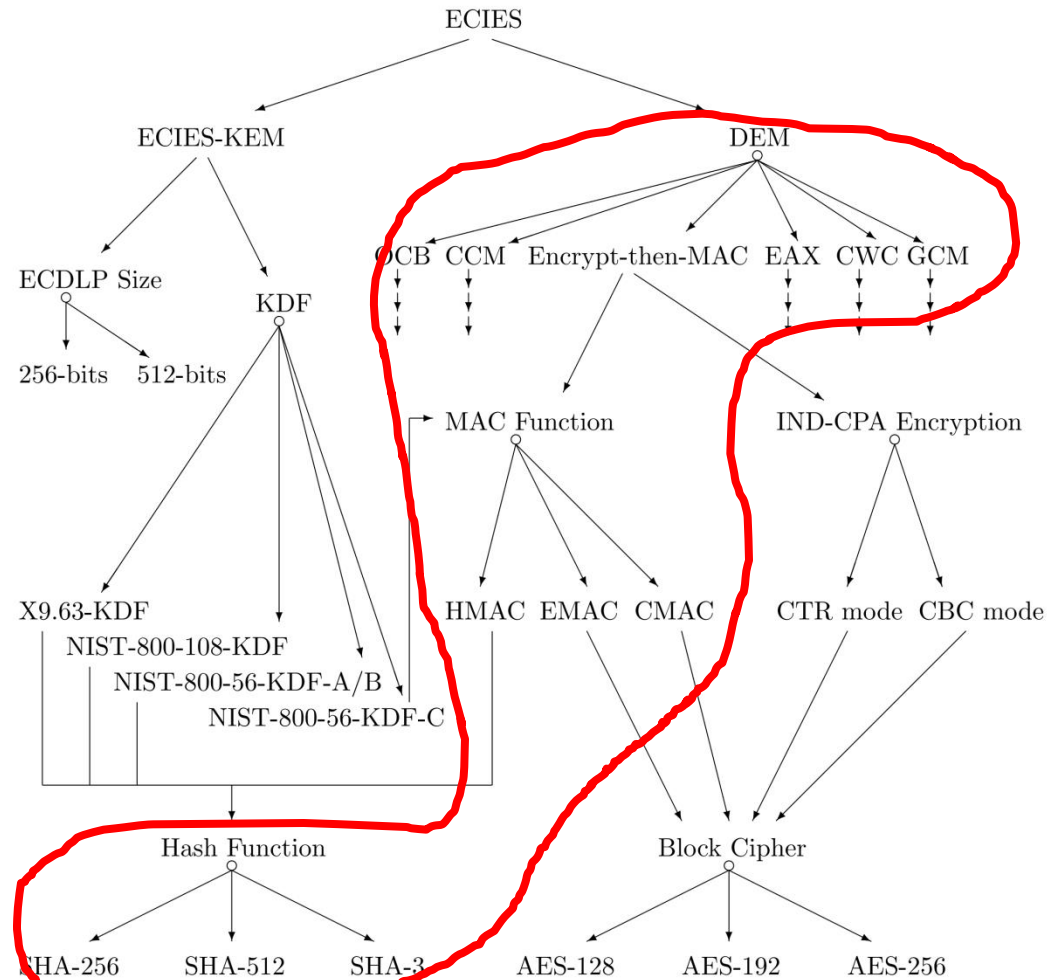
## Lecture 4

*Dr. Panagiotis Rizomiliotis*

# Agenda

- Data Integrity Symmetric key schemes
  - Message Authentication Codes
- Authenticated Encryption
- Lightweight cryptography

# OVERVIEW



\* Algorithms, key size and parameters report. ENISA– 2014

**Message  
Authentication  
Code (MAC)**



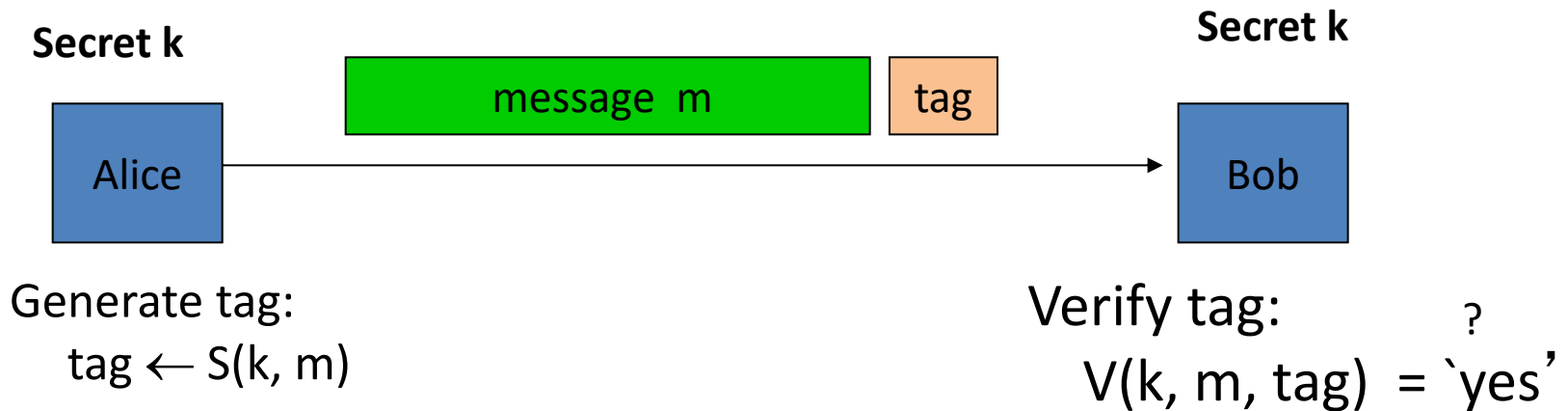
**MESSAGE AUTHENTICATE CODES**

# Protecting the integrity of a message

- Message Authentication Codes (MAC) are symmetric-key cryptosystems that aim to achieve **message integrity**
  - Attention: only integrity
  - **No** confidentiality!!
  - **No** authenticity!!
- 
- ✓ block-cipher based schemes
  - ✓ hash function based schemes
  - ✓ universal hash functions based schemes

# Message integrity: MACs

- MAC is a pair of algorithms (S,V).
  - S produces a tag
  - V verifies the integrity



Generate tag:  
 $\text{tag} \leftarrow S(k, m)$

Verify tag: ?  
 $V(k, m, \text{tag}) = \text{'yes'}$

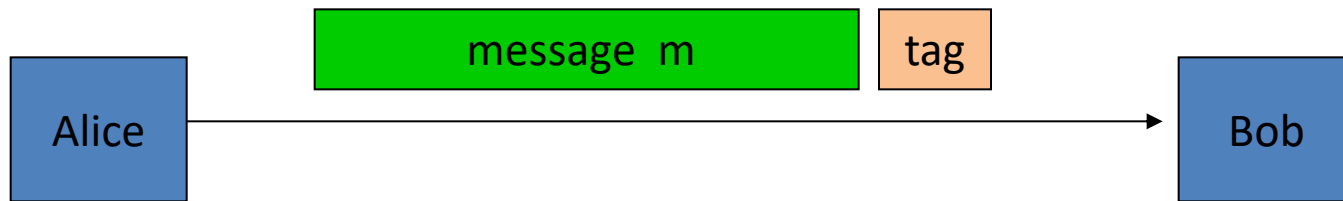
How V() works:

1. Computes  $\text{tag}' = S(k, m)$
2. If  $\text{tag} == \text{tag}'$ , data is valid

Otherwise, data has been corrupted

# Integrity requires a secret key

- ECC = error correction and detection code (well studied)



**Generate tag:**  
 $\text{tag} \leftarrow \text{ECC}(m)$

**Verify tag:**  
 $V(m, \text{tag}) = \overset{?}{\text{'yes'}}$

- Attacker can easily modify message  $m$  and re-compute ECC.
- ECC designed to detect random, not malicious errors.

# Protecting the integrity of a message

- Main MAC designs
  - ✓ block-cipher based schemes
  - ✓ hash function based schemes
  - ✓ universal hash functions based schemes

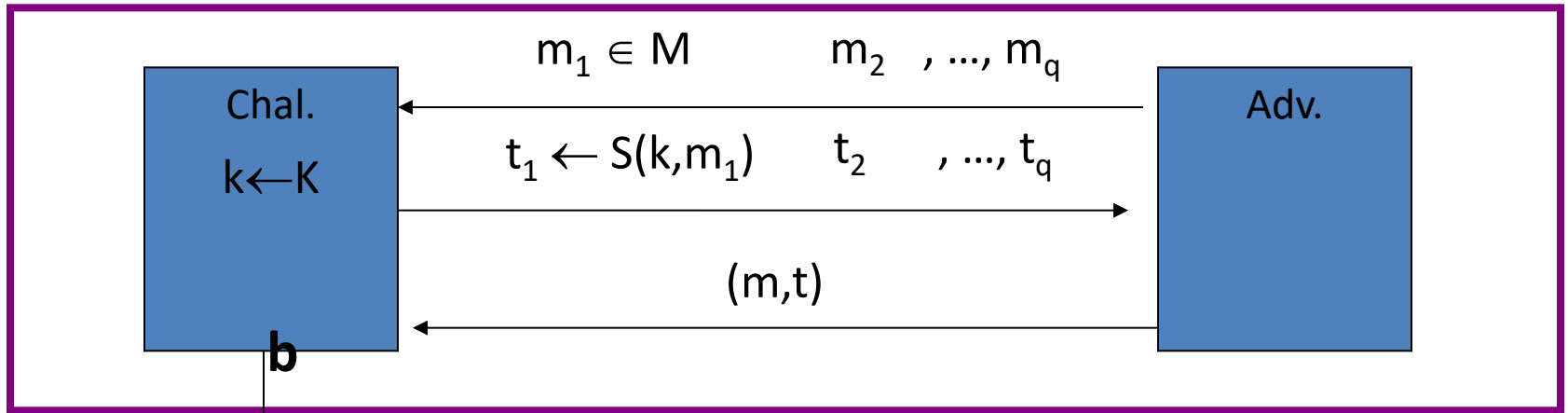
Scheme	Classification		Building Block
	Legacy	Future	
CMAC	✓	✓	Any block cipher as a PRP
HMAC	✓	✓	Any hash function as a PRF
UMAC	✓	✓	An internal universal hash function
EMAC	✓	✗	Any block cipher as a PRP
GMAC	✓	✗	Finite field operations
AMAC	✓	✗	Any block cipher



# Attack Scenario

- Attacker's power:
- chosen message attack
- The attacker can choose  $q$  messages:  $m_1, m_2, \dots, m_q$
- The key owner must produce all the tags  $t_i \leftarrow S(k, m_i)$
  
- Attacker's goal:
- existential forgery
- produce a new valid message/tag pair  $(m, t)$ , i.e.
  - $(m, t) \notin \{ (m_1, t_1), \dots, (m_q, t_q) \}$
  - **Secure MAC:**
  - The attack can produce the pair with negligible probability (attackers advantage)
    - $\text{Adv}_{\text{MAC}}[A, I] = \Pr[\text{Valid pair}]$  is "negligible."

# Secure MACs




$\begin{cases} \mathbf{b}=1, & \text{if } V(k, m, t) = \text{'yes'} \text{ and } (m, t) \notin \{(m_1, t_1), \dots, (m_q, t_q)\} \\ \mathbf{b}=0. & \text{otherwise} \end{cases}$

- $\text{Adv}_{\text{MAC}}[A, I] = \Pr[b = 1]$  is “negligible.”

# Quiz

- Let  $I = (S,V)$  be a MAC.
- Suppose an attacker is able to find  $m_0 \neq m_1$  such that
$$S(k, m_0) = S(k, m_1) \quad \text{for } \frac{1}{2} \text{ of the keys } k \text{ in } K$$
- Can this MAC be secure?
  - Yes, the attacker cannot generate a valid tag for  $m_0$  or  $m_1$
  - No, this MAC can be broken using a chosen msg attack
  - It depends on the details of the MAC

# Quiz

- Let  $I = (S,V)$  be a MAC.
- Suppose an attacker is able to find  $m_0 \neq m_1$  such that
$$S(k, m_0) = S(k, m_1) \quad \text{for } \frac{1}{2} \text{ of the keys } k \text{ in } K$$
- Can this MAC be secure?
  - Yes, the attacker cannot generate a valid tag for  $m_0$  or  $m_1$
  -  • No, this MAC can be broken using a chosen msg attack
  - It depends on the details of the MAC

# Quiz

- Let  $I = (S,V)$  be a MAC.
- Suppose  $S(k,m)$  is always 5 bits long
- Can this MAC be secure?
  - No, an attacker can simply guess the tag for messages
  - It depends on the details of the MAC
  - Yes, the attacker cannot generate a valid tag for any message

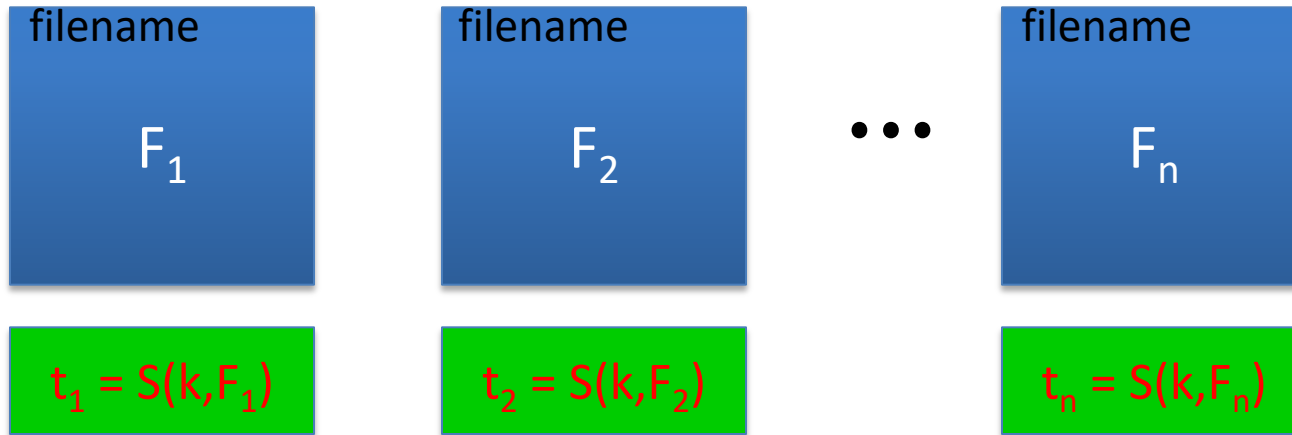
# Quiz

- Let  $I = (S,V)$  be a MAC.
- Suppose  $S(k,m)$  is always 5 bits long
- Can this MAC be secure?
  - ⇒ •No, an attacker can simply guess the tag for messages
  - It depends on the details of the MAC
  - Yes, the attacker cannot generate a valid tag for any message

# Can the tag length be small?

- Let  $I = (S,V)$  be a MAC and let the tag be 5 bits long.
- Can this MAC be secure?
  - No, an attacker can simply guess the tag for messages
  - The MAC is Weak by design

# EXAMPLE: PROTECTING SYSTEM FILES



- **Suppose at install time the system computes all the MAC tags.**
  - Later a virus infects system and modifies system files
  - User reboots into clean OS and supplies his secret key
- Then: secure MAC  $\Rightarrow$  all modified files will be detected

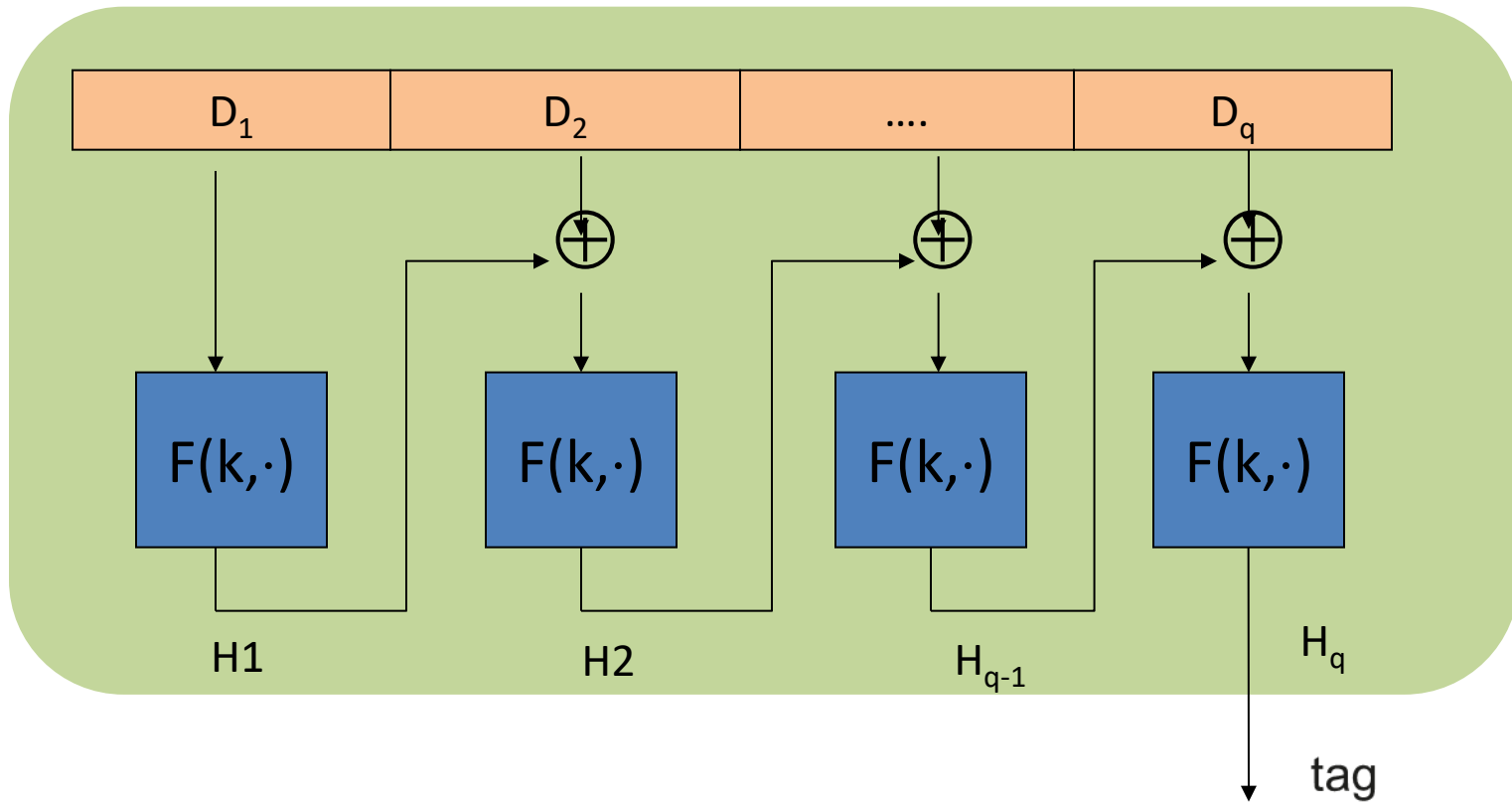


# Block cipher based mac

- In the text books it is called CBC-MAC
- One of the main constructions used in practice
- EMAC is commonly used as an AES-based MAC:CCM encryption mode (used in 802.11i)
- ISO 9797-1

Scheme	Classification		Building Block
	Legacy	Future	
CMAC	✓	✓	Any block cipher as a PRP
HMAC	✓	✓	Any hash function as a PRF
UMAC	✓	✓	An internal universal hash function
EMAC	✓	✗	Any block cipher as a PRP
GMAC	✓	✗	Finite field operations
AMAC	✓	✗	Any block cipher

# (RAW) CBC-MAC



# Why CBC-MAC is insecure?

Suppose we define a MAC  $I_{\text{RAW}} = (S, V)$  where

$$S(k, m) = \text{rawCBC}(k, m)$$

Then  $I_{\text{RAW}}$  is easily broken using a 1-chosen msg attack.

Adversary works as follows:

- Choose an arbitrary one-block message  $m \in X$
- Request tag for  $m$ . Get  $t = F(k, m)$
- Output  $t$  as MAC forgery for the 2-block message  $(m, t \oplus m)$

Indeed:  $\text{rawCBC}(k, (m, t \oplus m)) = F(k, F(k, m) \oplus (t \oplus m)) = F(k, t \oplus (t \oplus m)) = t$

# More complicated in practice

- The model for MAC generation comprises six steps:
  1. Padding of the data to a multiple of the cipher block size
  2. Splitting of the data into blocks
  3. Initial transformation of the first block of data
  4. Iteration through the remaining blocks of data
  5. Post-processing of the result of the last iteration
  6. Truncation of the result to the required length

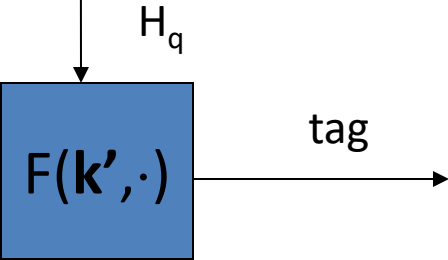
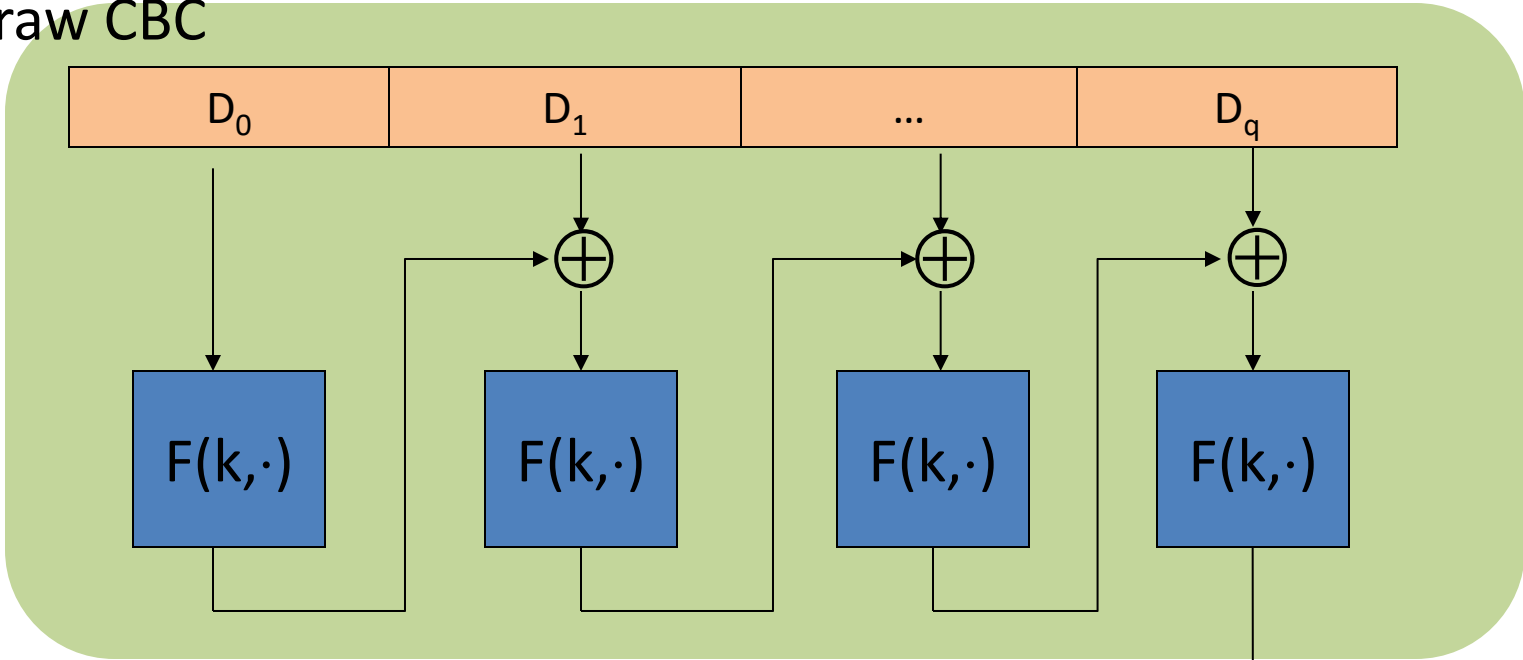
# ISO 9797-1

- ISO/IEC 9797-1 Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a

ISO 9797-1 Number	First Iteration	Final Iteration	Post Processing	a.k.a
1	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = H_q$	CBC-MAC
2	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = E_{K'}(H_q)$	EMAC
3	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = E_K(D_{K'}(H_q))$	AMAC
4	$H_1 = E_{K''}(E_K(D_1))$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = E_{K'}(H_q)$	-
5	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1} \oplus K')$	$G = H_q$	CMAC
6	$H_1 = E_K(D_1)$	$H_q = E_{K'}(D_q \oplus H_{q-1})$	$G = H_q$	LMAC

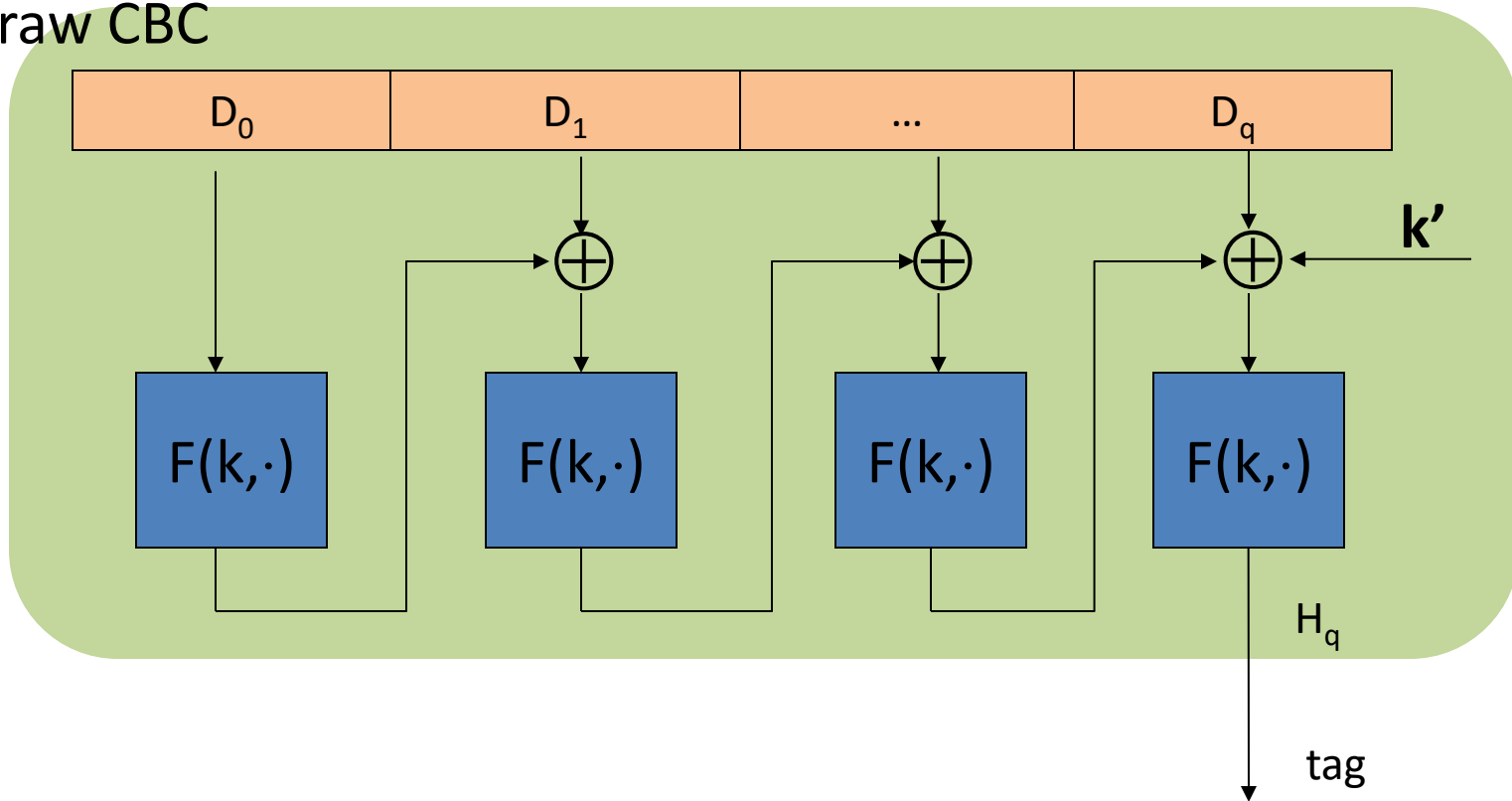
# encrypted CBC-MAC (EMAC)

raw CBC

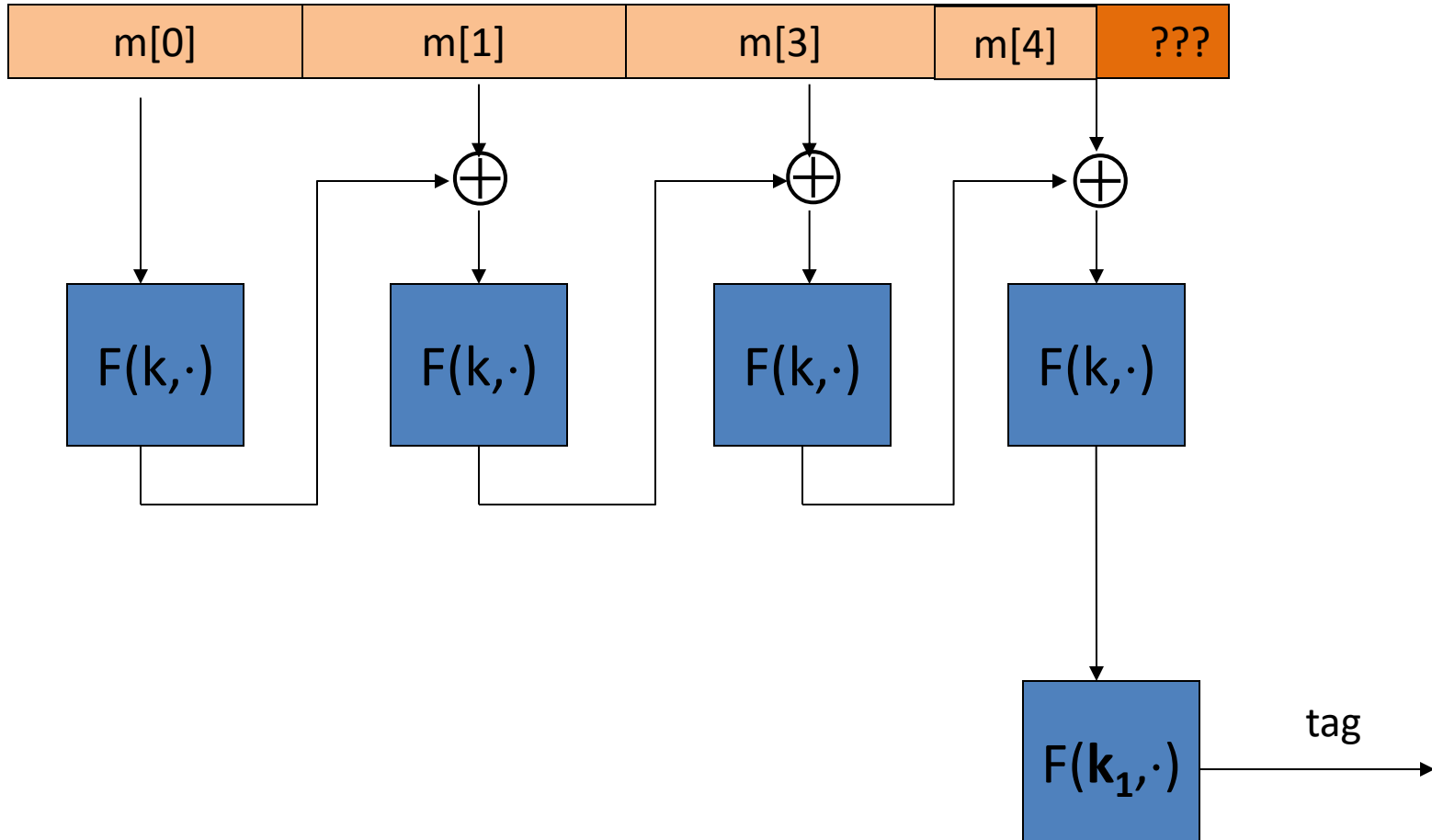


# CMAC

raw CBC



# Why padding is crucial





# Example of 'bad' padding

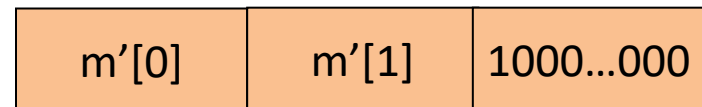
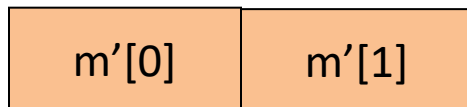
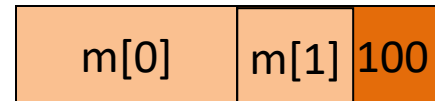
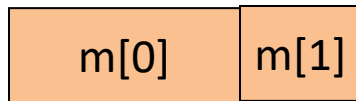
- Bad idea: pad  $m$  with 0's



- Is the resulting MAC secure?
- No, given the tag on message  $m$  the attacker obtains the tag on all the messages  $m' = m || 0$

# EMAC padding

- For security, padding must be invertible
- $m_0 \neq m_1 \Rightarrow \text{pad}(m_0) \neq \text{pad}(m_1)$
- ISO 9797-1: (Use “1” to indicate the beginning of pad)
- pad with “1000...00”.
- Add new dummy block if needed.

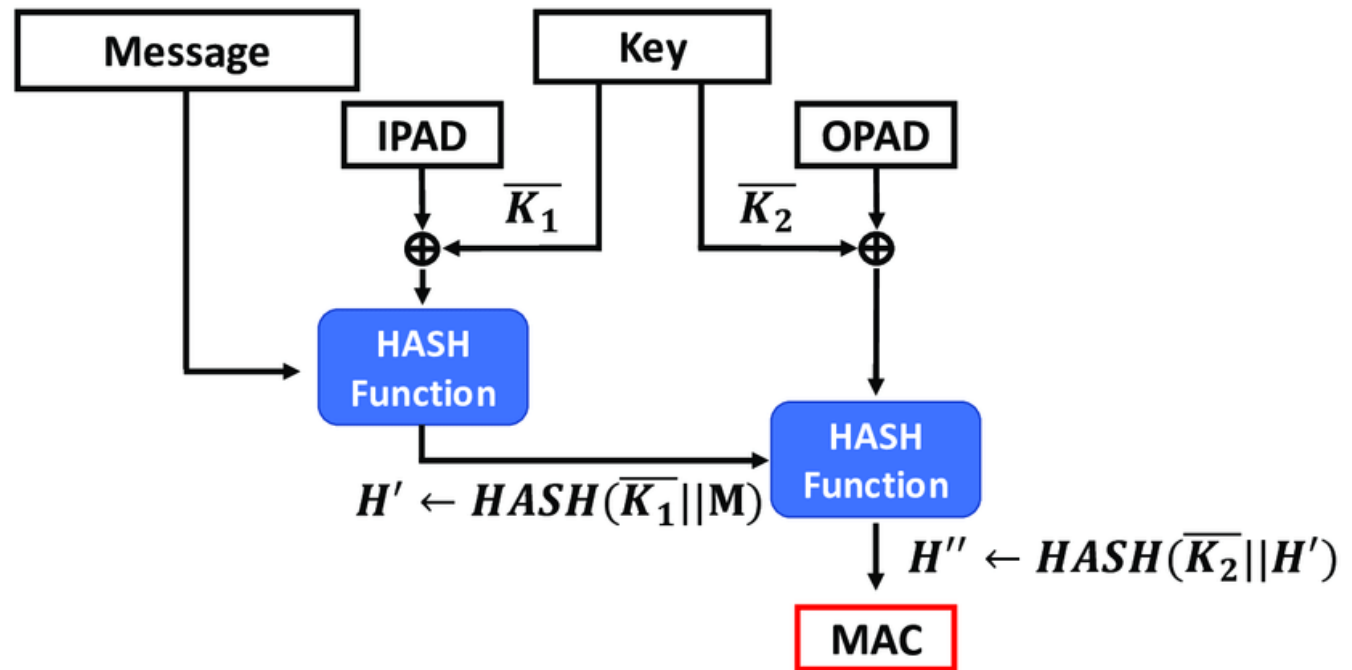


# recommendations

- When shortened MAC outputs are used, then the MAC key must be relatively short lived
  - The MAC can only be verified for a short length of time!!
- EMAC
  - ✓ Algorithm 2 in ISO-9797-1
  - ✓ frequent rekeying is required
  - ✓ Two keys
  - ✓ the security of the scheme is bounded by  $2^k$ , where  $k$  is the length of a single key
- CMAC
  - ✓ Algorithm 5 in ISO-9797-1
  - ✓ provable security guarantees
  - ✓ the scheme should be used for at most  $2^{48}$  messages
  - ✓ no party learns the enciphering of the all-0 string (there is an attack)

# Hash based mac

- 1996: HMAC construction published by M. Bellare, R. Canetti, H. Krawczyk
- 1997: RFC 2104



- Uses:
- SSL/TLS, IPsec, SSH, JSON Web Tokens

# Hash based mac

- where
  - $H()$ : is a cryptographic hash function
  - $m$ : is the message to be authenticated
  - $k$ : is the secret key
  - $k'$ : is a block-sized key derived from the secret key,  $K$ ; either by padding to the right with 0s up to the block size, or by hashing down to less than the block size first and then padding to the right with zeros
  - $\parallel$  denotes concatenation
  - $\oplus$  denotes bitwise exclusive or (XOR)
  - $opad$ : is the block-sized outer padding, consisting of repeated bytes valued 0x5c
  - $ipad$ : is the block-sized inner padding, consisting of repeated bytes valued 0x36

# HMAC properties

- HMAC is assumed to be a secure PRF
- Can be proven under certain PRF assumptions about  $H()$
- Does not require collision-resistance!!!
  
- Never use HMAC-MD4
- Can be used HMAC-SHA1, and HMAC-MD5
- Conservative instantiations: HMAC-SHA2 and HMAC-SHA3.

# ECBC-MAC and HMAC analysis

Theorem: For any  $L > 0$ ,

For every eff.  $q$ -query PRF adv.  $A$  attacking  $F_{\text{ECBC}}$  or  $F_{\text{NMAC}}$  there exists an eff. adversary  $B$  s.t.:

$$\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq \text{Adv}_{\text{PRP}}[B, F] + 2q^2 / |X|$$

$$\text{Adv}_{\text{PRF}}[A, F_{\text{HMAC}}] \leq q \cdot L \cdot \text{Adv}_{\text{PRF}}[B, F] + q^2 / 2|K|$$

CBC-MAC is secure as long as  $q \ll |X|^{1/2}$

HMAC is secure as long as  $q \ll |K|^{1/2}$  ( $2^{64}$  for AES-128)

# An example

$$\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq \text{Adv}_{\text{PRP}}[B, F] + 2q^2 / |X|$$

$q = \#$  messages MAC-ed with  $k$

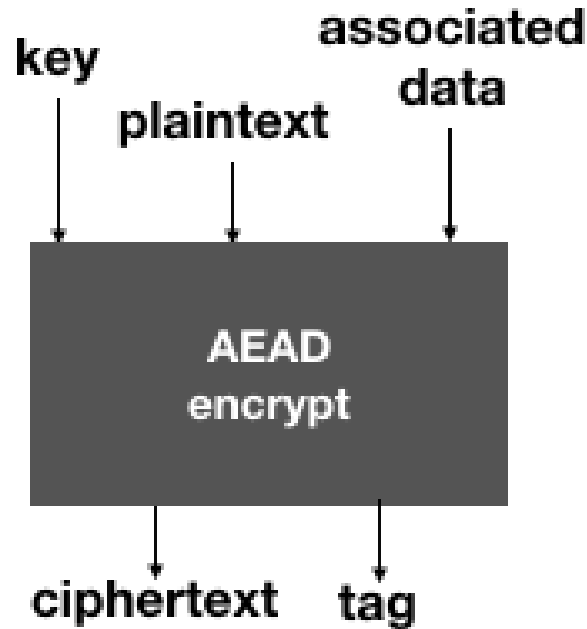
Suppose we want  $\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq 1/2^{32} \iff q^2 / |X| < 1/2^{32}$

- AES:  $|X| = 2^{128} \implies q < 2^{48}$

So, after  $2^{48}$  messages must, must change key

- 3DES:  $|X| = 2^{64} \implies q < 2^{16}$





# **AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA (AEAD)**

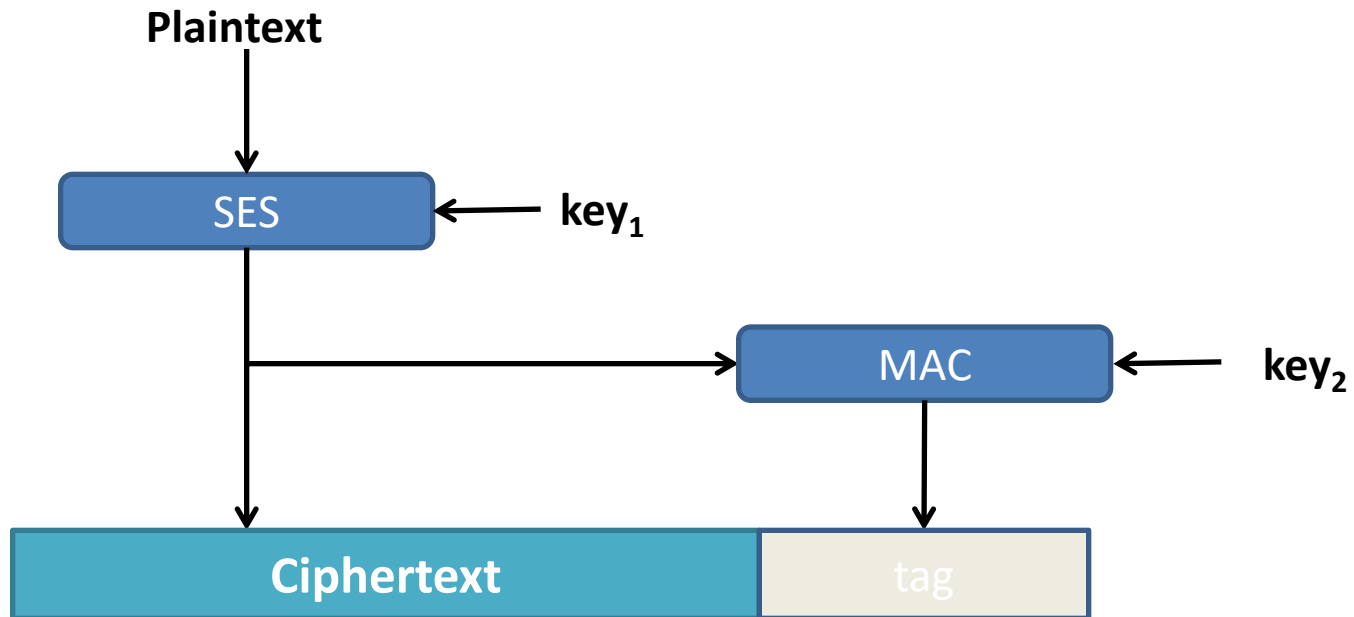
# Authenticated encryption

- So far we have symmetric encryption schemes for confidentiality and message authentication codes for data integrity.
- In practice, most of the time, we need both.
- Authentication encryption offers
  - ✓ Confidentiality of data
  - ✓ Ciphertext integrity
- Usually we are talking about nonce-based authenticated encryption with associated data (AEAD)

# Main constructions

- Generic Composition (Encrypt-then-MAC)
- Offset Codebook (OCB) mode
- CCM mode
- EAX mode
- Carter-Wegman + Counter (CWC) mode
- Galois/Counter Mode (GCM)
- CAESAR Competition

# Encrypt-then-mac



# Encrypt-then-mac

- ✓ simplest mechanism
- ✓ it is a two pass process
- ✓ Encrypt-and-MAC or MAC-then-Encrypt, in general should not be used

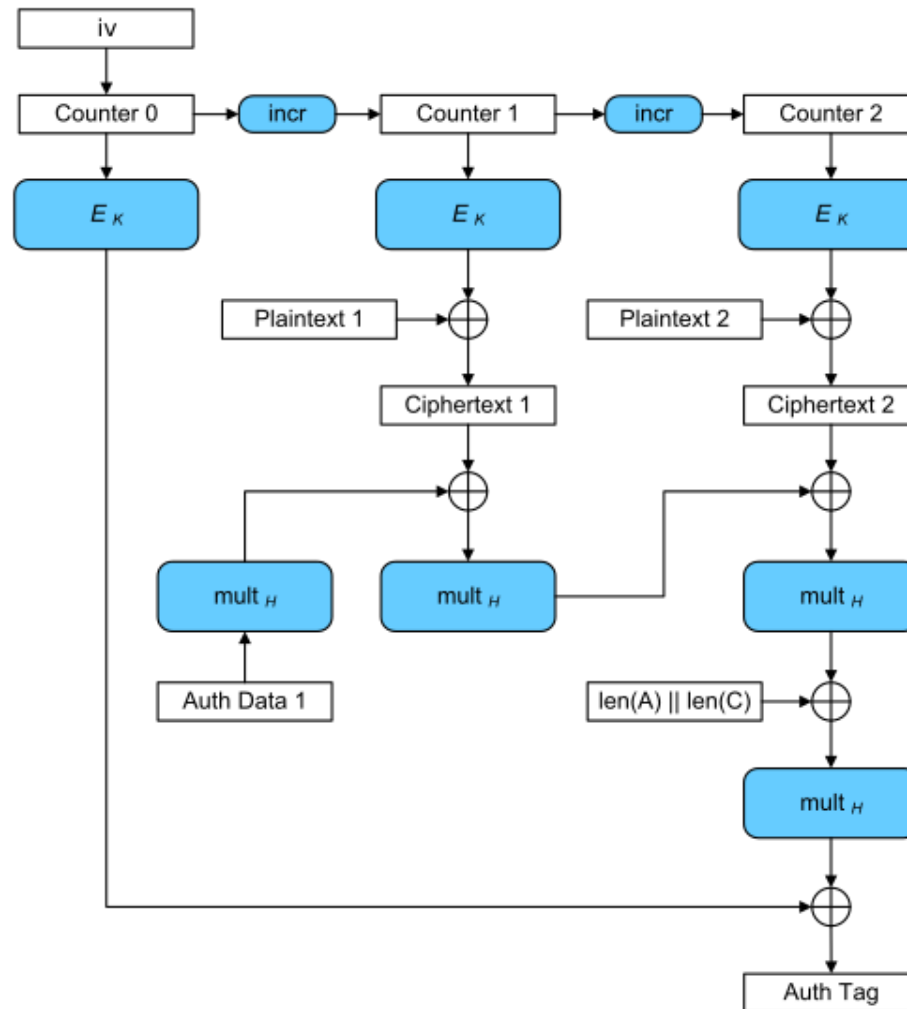
# ISO/IEC 19772:2009

- ISO/IEC 19772:2009. Information technology -- Security techniques -- Authenticated encryption
- Two main constructions
  1. CCM mode
  2. Galois/Counter Mode (GCM)

# Galois/Counter Mode (GCM)

- designed by McGrew and Viega
- combines Counter mode with a Carter-Wegman MAC (i.e. GMAC)
- recommended in the IETF RFCs for IPsec, SSH and TLS
- online scheme
- fully parallelisable
- efficient implementations in hardware
- provably secure given that
  - ✓ the IV is a nonce
  - ✓ the underlying block cipher is secure
- Authentication tags must have length at least 96 bits

# Galois/Counter Mode (GCM)



From  
Wikipedia



# Other schemes

- CCM mode
  - ✓ combines CTR mode with CBC-MAC
  - ✓ provably secure
  - ✓ main drawback comes from its inefficiency
  - ✓ It is not “on-line”
  
- OCB (Offset Codebook) mode
  - ✓ proposed by Rogaway et al
  - ✓ provably secure building on the security of the underlying block cipher
  - ✓ one-pass mode of operation making it highly efficient.
  - ✓ two U.S. patents.
  - ✓ As of January 2013, free for software usage under an GNU General Public License

# Other schemes

- CWC (Carter-Wegman + Counter ) mode
  - Proposed by Kohno, Viega and Whiting [202].  
combines a Carter-Wegman MAC, with CTR mode
  - the IV is a nonce
  - provably secure building on the security of the underlying block cipher
- EAX
  - proof of security
  - very similar to CCM mode,
  - based on CTR mode and CBC-MAC

# Caesar competition (2013-2019)

- Open competition
- Output: a portfolio of algorithms
- not a standardization project
- 57 submissions
- 7 schemes in the final portfolio
- Main goal was to identify a portfolio of authenticated ciphers that
  - offer advantages over AES-GCM
  - are suitable for widespread adoption
- <https://competitions.cr.yp.to/caesar.html>

# Caesar final portfolio

- The final CAESAR portfolio is organized into three use cases:
  1. Lightweight applications (resource constrained environments)

Cipher	Designer
Ascon	Christoph Dobraunig, Maria Eichlseder, Florian Mendel, Martin Schläffer
ACORN	Hongjun Wu

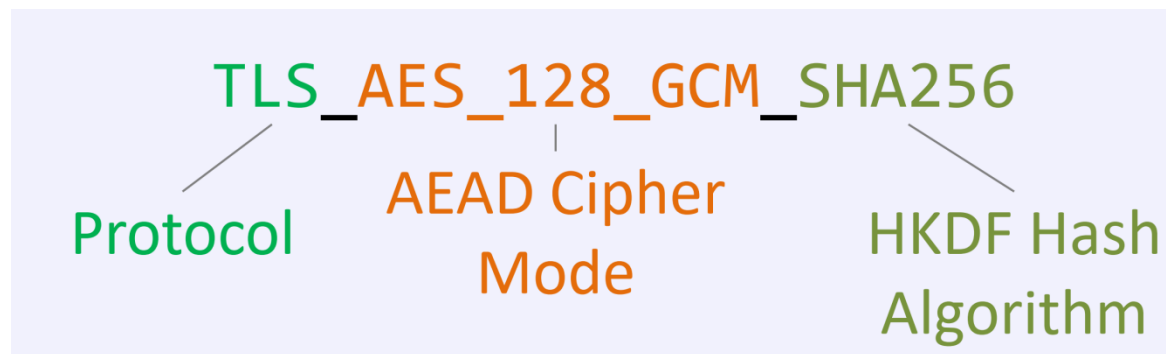
# Caesar final portfolio

Cipher	Designer
AEGIS-128	Hongjun Wu, Bart Preneel
OCB	Ted Krovetz, Phillip Rogaway

Cipher	Designer
Deoxys-II	Jérémy Jean, Ivica Nikolić, Thomas Peyrin, Yannick Seurin
COLM	Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser, Kan Yasuda

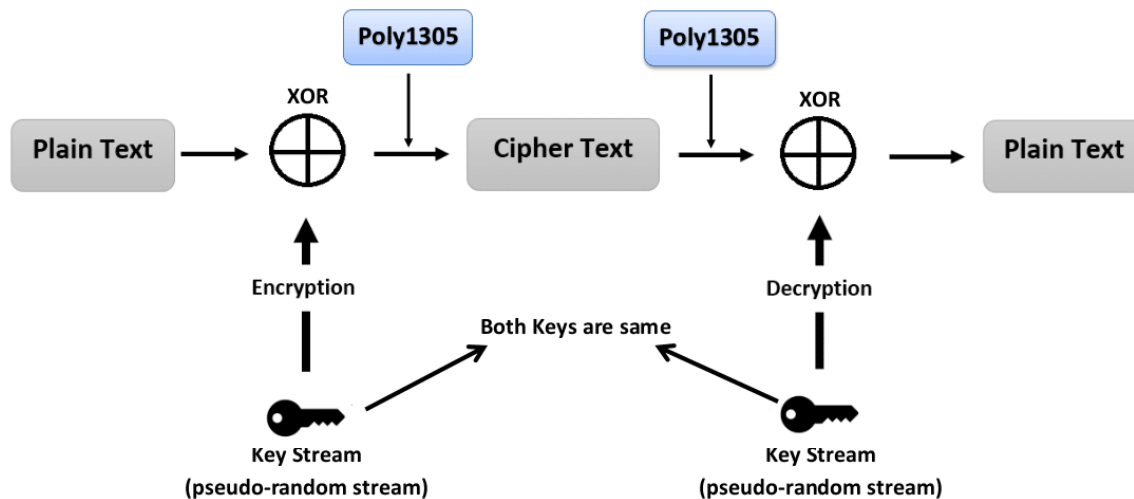
# Cipher Suite specification – TLS 1.3

- TLS v1.3 supports 5 cipher suites.
  - TLS\_AES\_128\_GCM\_SHA256
  - TLS\_AES\_256\_GCM\_SHA384
  - TLS\_CHACHA20\_POLY1305\_SHA256
  - TLS\_AES\_128\_CCM\_SHA256
  - TLS\_AES\_128\_CCM\_8\_SHA256



# CHACHA20\_POLY1305

- IETF Protocols (RFC 8439)
  - The ChaCha20 cipher
  - The Poly1305 MAC



# CHACHA20

- ChaCha20 is a stream cipher designed by Daniel J. Bernstein, ChaCha20 is a variant of the Salsa20 family of stream ciphers
- Add-Rotate-XOR (ARX) Operations
- Faster than AES and resilient to timing attacks



# Poly1305

- Poly1305 is a cryptographic Message Authentication Code (MAC)
- Published in 2004
- Compared to the more widely used HMAC, Poly1305 is extremely faster.
- Uses AES
- 32-byte secret key

# **LIGHTWEIGHT CRYPTOGRAPHY**

# Lightweight cryptography

- Cryptographic algorithms for constrained devices
- Limited resources for cryptography
- Internet of things
  
- Standardization efforts
  - ISO
  - NIST

# ISO

- Block ciphers
  - ISO/IEC 29192-2:2012 specifies two block ciphers suitable for lightweight cryptography:
    - PRESENT: a lightweight block cipher with a block size of 64 bits and a key size of 80 or 128 bits;
    - CLEFIA: a lightweight block cipher with a block size of 128 bits and a key size of 128, 192 or 256 bits.
- Stream Ciphers
  - Enocoro: key size of 80 or 128 bits, based on a finite state machine and uses operations defined over the finite field  $GF(24)$  and  $GF(28)$ .
  - Trivium: key size of 80 bits, three nonlinear feedback registers, 288 bits of internal size.

# ISO – hash functions

- ISO/IEC 29192-5:2016 specifies three hash-functions suitable for applications requiring lightweight cryptographic implementations.
  - PHOTON: a lightweight hash-function with permutation sizes of 100, 144, 196, 256 and 288 bits computing hash-codes of length 80, 128, 160, 224, and 256 bits, respectively.
  - SPONGENT: a lightweight hash-function with permutation sizes of 88, 136, 176, 240 and 272 bits computing hash-codes of length 88, 128, 160, 224, and 256 bits, respectively.
  - Lesamnta-LW: a lightweight hash-function with permutation size 384 bits computing a hash-code of length 256 bits.
- The requirements for lightweight cryptography are given in ISO/IEC 29192-1.

# NSA – Simon and Speck

## NSA Ciphers “Simon and Speck” Are Dead – But Not Entirely Buried Says ISO

ED TARGETT EDITOR  
9TH MAY 2018

+ INCREASE / DECREASE TEXT SIZE -



# NIST lightweight project

- <https://csrc.nist.gov/Projects/Lightweight-Cryptography>
- Scope:
  - All cryptographic primitives and modes that are needed in constrained environments.
  - Initial Focus: Symmetric Cryptography.
  - Target functionality: Encryption, AE, hashing, key agreement, sensor/tag authentication.
  - Target devices: ARM Cortex-M0 processors, Intel Quark SoC X1021, Atom E3826.
  - Side channel resistance: In general, good to have.

# NIST

- Target applications: Hardware encrypted data storage device, low-cost and low-consumption sensor data transmission, RAIN RFID tags for anti-counterfeiting solutions, IoTs, wearables, low power wireless sensor networks.
  - Modifications of well-analyzed designs: e.g., DESL, DESXL.
  - Old interesting algorithms: e.g., RC5, TEA, XTEA.
  - New dedicated algorithms: e.g., Skinny, Pride, Gimli, Simon, Speck, Simeck, Present, etc.



# NIST project

- ✓ Early September 2018, NIST will publish FRN (Federal Register Notice) and the final Call for Submissions.
- ✓ December 2018, option for early submission for initial review.
- ✓ February 2019, deadline for submissions.
  
- ✓ NIST will publish the complete and proper submissions.
- ✓ Initial evaluation will be for approximately 12 months.
  
- ✓ Workshop will be held ten to twelve months after the submission deadline.
- ✓ Standardization within two to four years, after the public analysis starts

# NEXT STEPS

- NIST plans to announced **nine** candidates (finalists) to advance to **Round 3**
- Round 3 candidates will be given an opportunity to perform **minor modifications** to their projects to improve efficiency or avoid cryptographic attacks (tweaks)
- Final round will **last about one year**
- **Winner (or winners)** will be selected in **2021**

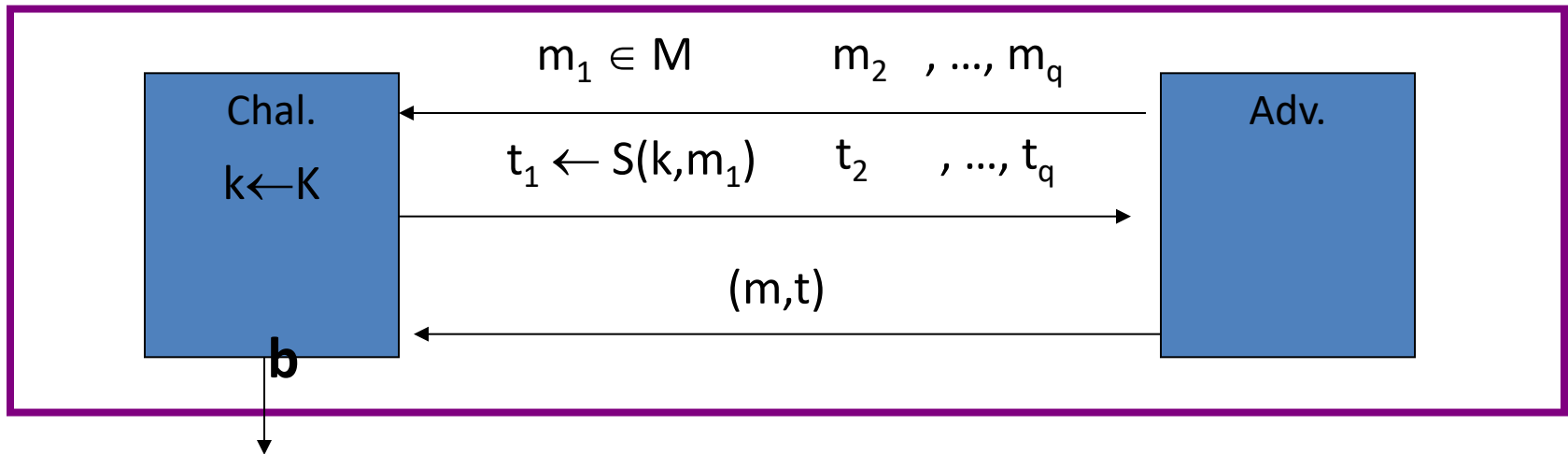
# FINALISTS

- On March 29, 2021, NIST announced the finalists as:
  1. ASCON,
  2. Elephant,
  3. GIFT-COFB,
  4. Grain128-AEAD,
  5. ISAP,
  6. Photon-Beetle,
  7. Romulus,
  8. Sparkle,
  9. TinyJambu,
  10. Xoodoo

# MAC SECURITY

# Strong Unforgeability under Chosen Message Attack (SUF-CMA)

- For a MAC  $I=(S,V)$  and adv.  $A$  define a MAC game as:



$$\begin{cases} \mathbf{b}=1 & \text{if } V(k,m,t) = \text{'yes'} \text{ and } (m,t) \notin \{(m_1,t_1), \dots, (m_q,t_q)\} \\ \mathbf{b}=0 & \text{otherwise} \end{cases}$$

Def:  $I=(S,V)$  is a secure MAC if for all “efficient”  $A$ :

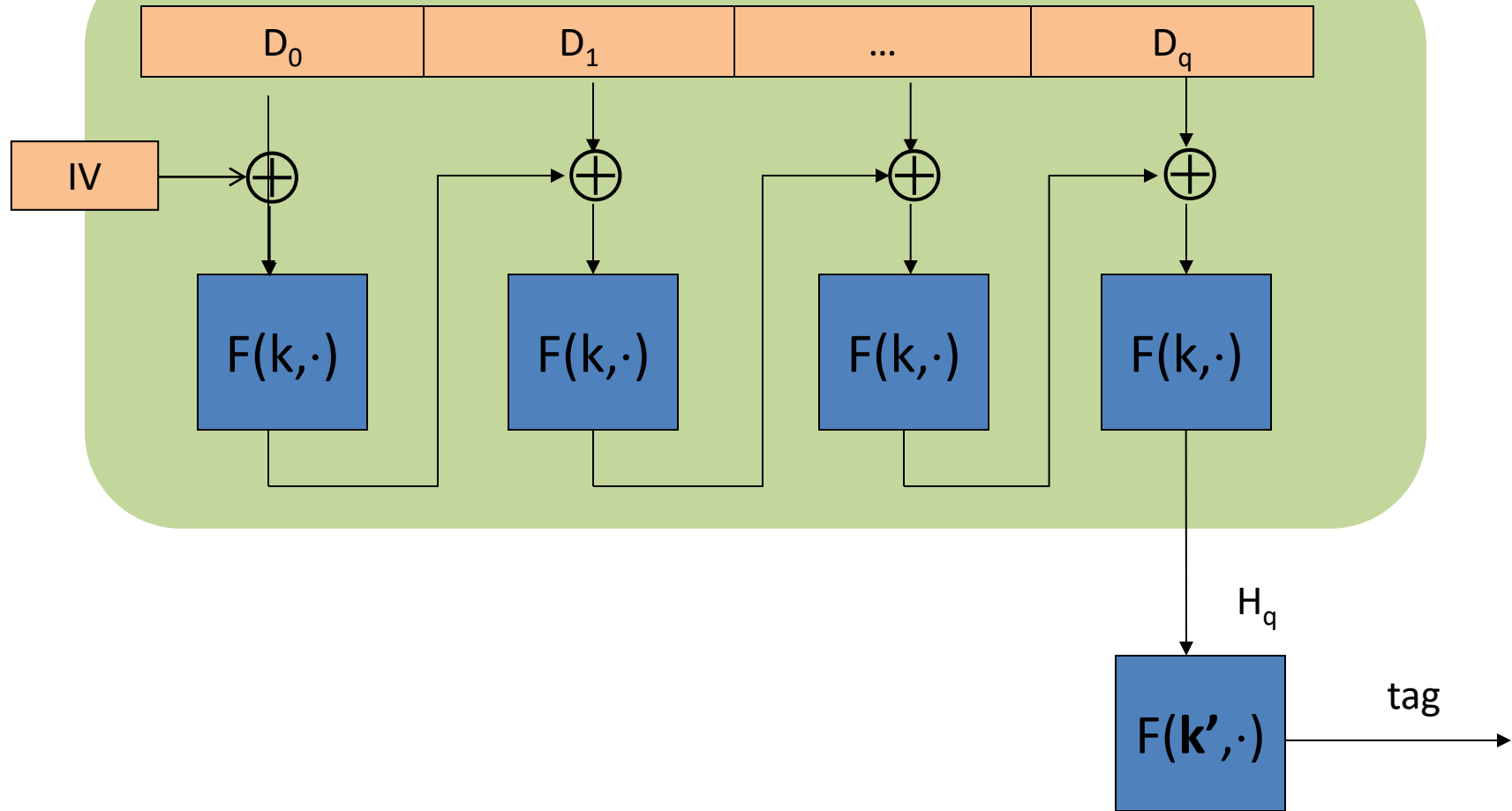
$$\text{Adv}_{\text{MAC}}[A,I] = \Pr[\text{Chal. outputs } 1] \text{ is “negligible.”}$$

# Άσκηση

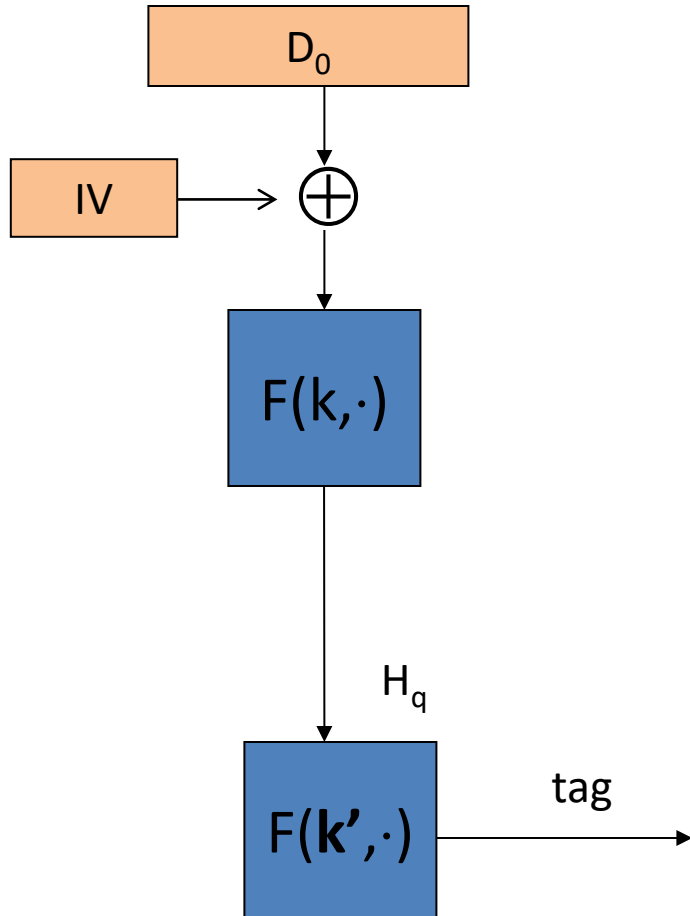
- Έστω ότι το ECBC-MAC επιλέγει ένα τυχαίο IV για κάθε μήνυμα που προστατεύεται και περιλαμβάνει το IV στο tag. Δηλαδή,  $S(k,m):=(r, ECBC_r(k,m))$  όπου το  $ECBC_r(k,m)$  αναφέρεται στο ECBC χρησιμοποιώντας το  $r$  ως IV. Ο αλγόριθμος επιβεβαίωσης  $V$  με το κλειδί  $k$ , το μήνυμα  $m$ , και το tag  $(r,t)$  επιστρέφει ``1'', όταν  $t=ECBC_r(k,m)$  και ``0'', διαφορετικά. Ο αλγόριθμος MAC δεν είναι ασφαλής. Γιατί;

# Modified encrypted CBC-MAC (EMAC)

raw CBC



# Modified encrypted CBC-MAC (EMAC)





# Proof (sketch)

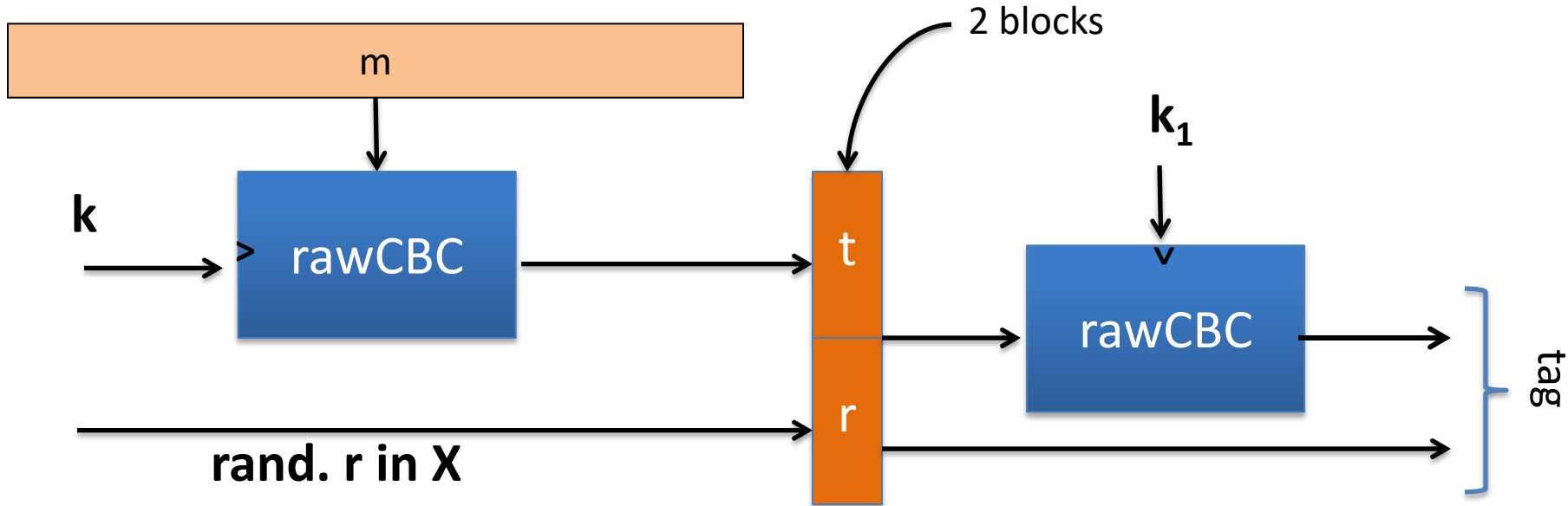
- Let  $m_1 = D_0 || m$  and let  $\text{tag} = (r, \text{ECBC}_r(k, m_1))$
- Then, it is easy to verify that for any message
  - $m_2 = D || m$ , and
  - $r' = r \oplus D \oplus D_0$

it holds that  $\text{tag} = (r', \text{ECBC}_{r'}(k, m_2))$ .

Questions?



# Better security: a rand. construction



Let  $F: K \times X \rightarrow X$  be a PRF. Result: MAC with tags in  $X^2$ .

Security:  $Adv_{MAC}[A, I_{RCBC}] \leq Adv_{PRP}[B, F] \cdot (1 + 2q^2 / |X|)$

$\Rightarrow$  For 3DES: can sign  $q=2^{32}$  msgs with one key