Cryptography and Network Security Chapter 12

Fifth Edition

by William Stallings

Lecture slides by Lawrie Brown

Chapter 12 – Message Authentication Codes

- *At cats' green on the Sunday he took the message from the inside of the pillar and added Peter Moran's name to the two names already printed there in the "Brontosaur" code. The message now read: "Leviathan to Dragon: Martin Hillman, Trevor Allan, Peter Moran: observe and tail." What was the good of it John hardly knew. He felt better, he felt that at last he had made an attack on Peter Moran instead of waiting passively and effecting no retaliation. Besides, what was the use of being in possession of the key to the codes if he never took advantage of it?*
- ***—Talking to Strange Men,* Ruth Rendell**

Road Map

- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

Message Authentication

- message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- will consider the security requirements
- then three alternative functions used:
  - hash function (see Ch 11)
  - message encryption
  - message authentication code (MAC)

Message Security Requirements

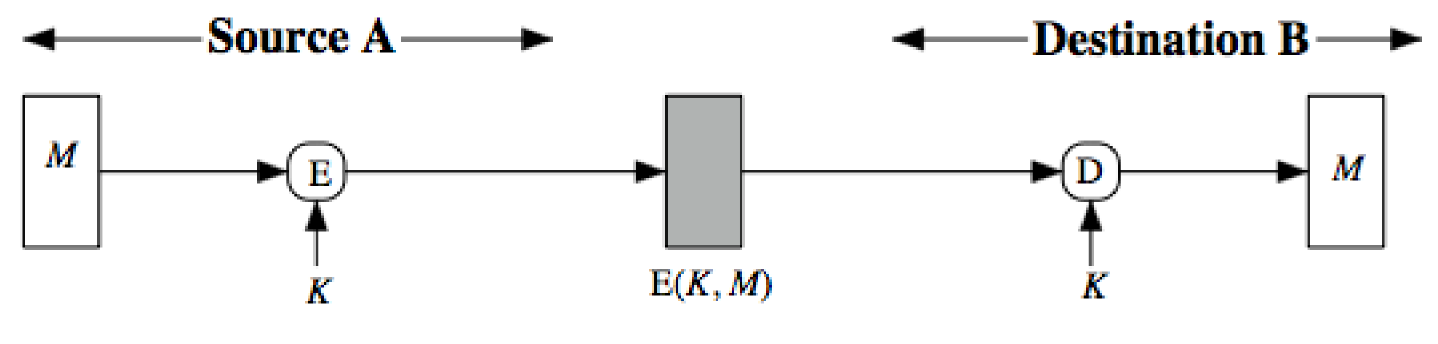
- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

Road Map

- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

Symmetric Message Encryption

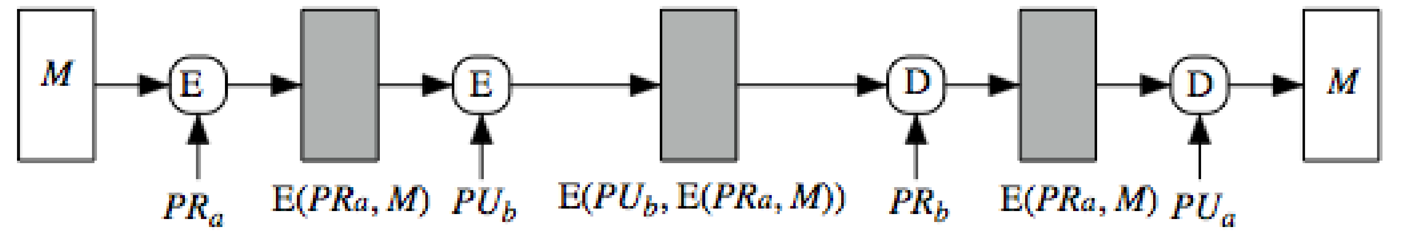
- encryption can also provides authentication
- if symmetric encryption is used then:
  - receiver know sender must have created it
  - since only sender and receiver know key used
  - know content cannot have been altered...
  - ... if message has suitable structure, redundancy or a suitable checksum to detect any changes



(a) Symmetric encryption: confidentiality and authentication

Public-Key Message Encryption

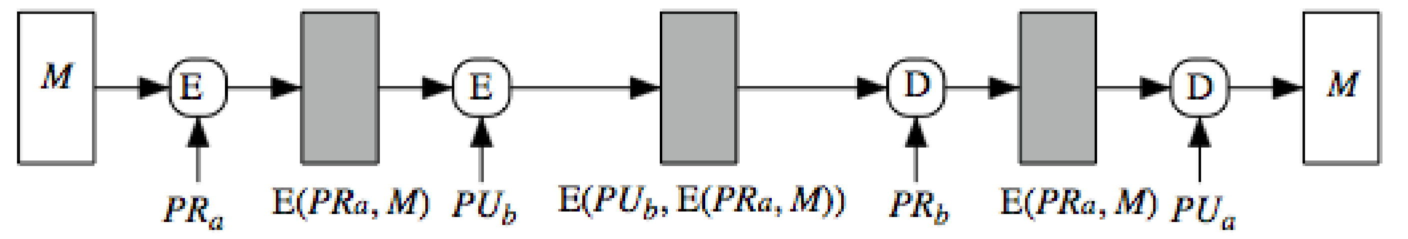
- if public-key encryption is used:
  - encryption provides no confidence of sender
    - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message



(d) Public-key encryption: confidentiality, authentication, and signature

Public-Key Message Encryption

- Dirty little detail on PKCS
  - Every time you encrypt, size expands
  - Due to protections in PKCS#1
- So signing (by encryption) then encrypting, the size is more than doubled!



(d) Public-key encryption: confidentiality, authentication, and signature

Road Map

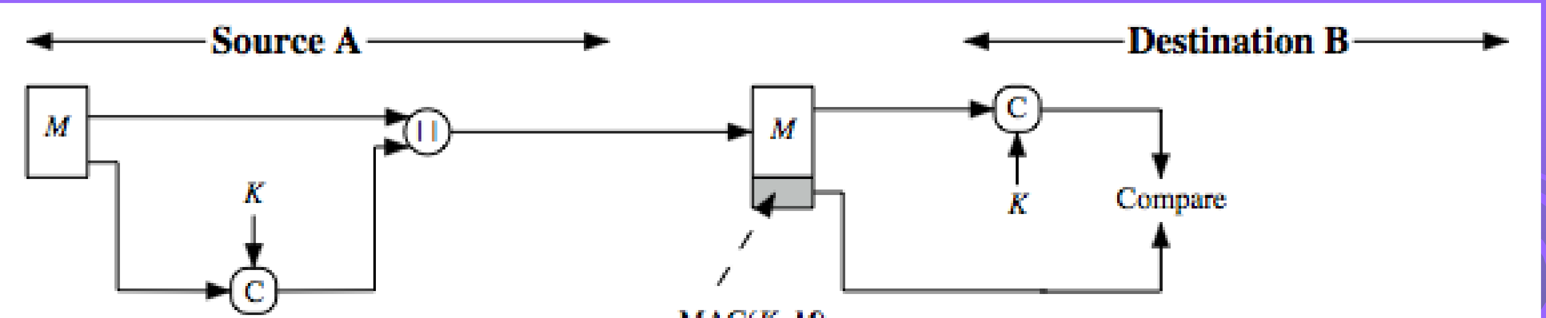
- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

Message Authentication Code (MAC)

- generated by an algorithm that creates a small fixed-sized block
  - depending on both message and secret key
  - like encryption though need not be reversible
- appended to message as a "**signature**"
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender

Message Authentication Code

- a small fixed-sized block of data
  - generated from message + secret key
  - MAC = C(K,M)
  - appended to message when sent



(a) Message authentication

Message Authentication Codes

- as shown the MAC provides authentication
- can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before, but see Generic Composition

Message Authentication Codes

- why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (e.g. archival use)
- note that a MAC is not a digital signature
  - Does NOT provide non-repudiation

MAC Properties

- a MAC is a cryptographic checksum

  $$\texttt{MAC} = \texttt{C}_{\texttt{K}}(\texttt{M})$$

  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- is a many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message

Security of MACs

- like block ciphers have:
- **brute-force** attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf. key search) or MAC
    - at least 128-bit MAC is needed for security

Security of MACs

- **cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative
- more variety of MACs so harder to generalize about cryptanalysis

Keyed Hash Functions as MACs

- want a MAC based on a hash function
  - because hash functions are generally faster
  - crypto hash function code is widely available
- hash includes a key along with message
- original proposal:

  `KeyedHash = Hash(Key|Message)`

  - some weaknesses were found with this
- eventually led to development of HMAC

Problem with Keyed Hash

- `KeyedHash = Hash(Key|Message)`
- Recall hash function works on blocks
- Let M = Key | Message | Padding and M
  $M=M_1\ M_2\ \ldots\ M_L$, where $|M_i|$ = Blocksize
  $Hash=H(H(\ldots H(H(IV,M_1),M_2),\ldots,M_L)$
- But can add extra block(s) M_{L+1} by
  $Hash'=H(Hash,M_{L+1})$
- Unless formatting prevents it…
  … but still best to use HMAC!

Road Map

- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

HMAC Design Objectives

- use, without modifications, hash functions
- allow for easy replacement of embedded hash function
- preserve original performance of hash function without significant degradation
- use and handle keys in a simple way.
- have well understood cryptographic analysis of authentication mechanism strength

HMAC

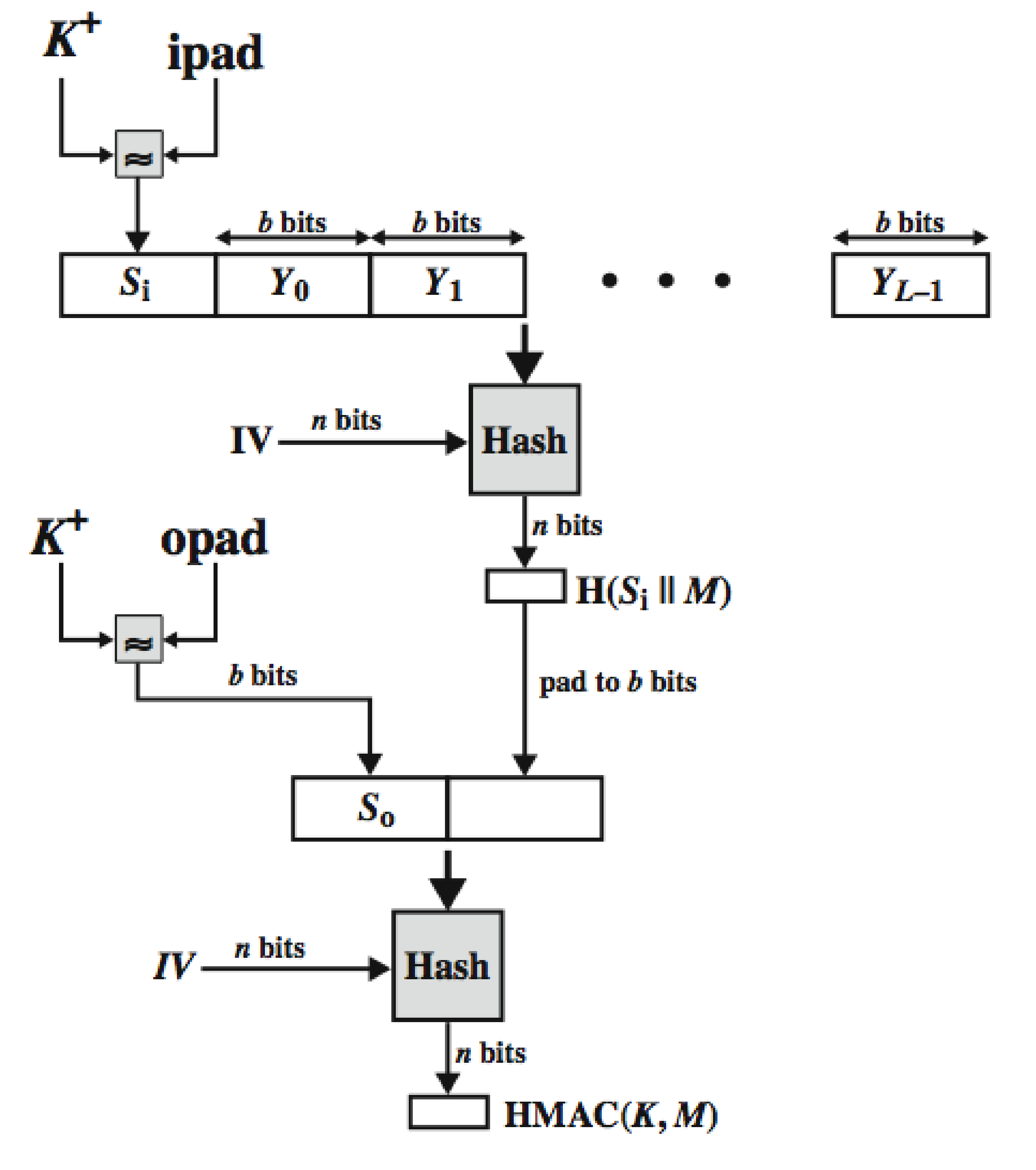
- specified as Internet standard RFC2104
- uses hash function on the message:

  `HMACK(M)= Hash[(K+ XOR opad) ||`
  `Hash[(K+ XOR ipad) || M)] ]`

  $$\mathtt{HMAC}_K(\mathtt{M}) = \mathtt{Hash}[(\mathtt{K}^+ \ \mathtt{XOR}\ \mathtt{opad})\ ||\ \mathtt{Hash}[(\mathtt{K}^+\ \mathtt{XOR}\ \mathtt{ipad})\ ||\ \mathtt{M})]\ ]$$

  - where $\mathtt{K}^+$ is the key padded out to block size
  - **opad**, **ipad** are specified padding constants
- overhead is just 3 more hash block calculations than the message needs alone
- any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

HMAC Overview



HMAC Security

- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints

Road Map

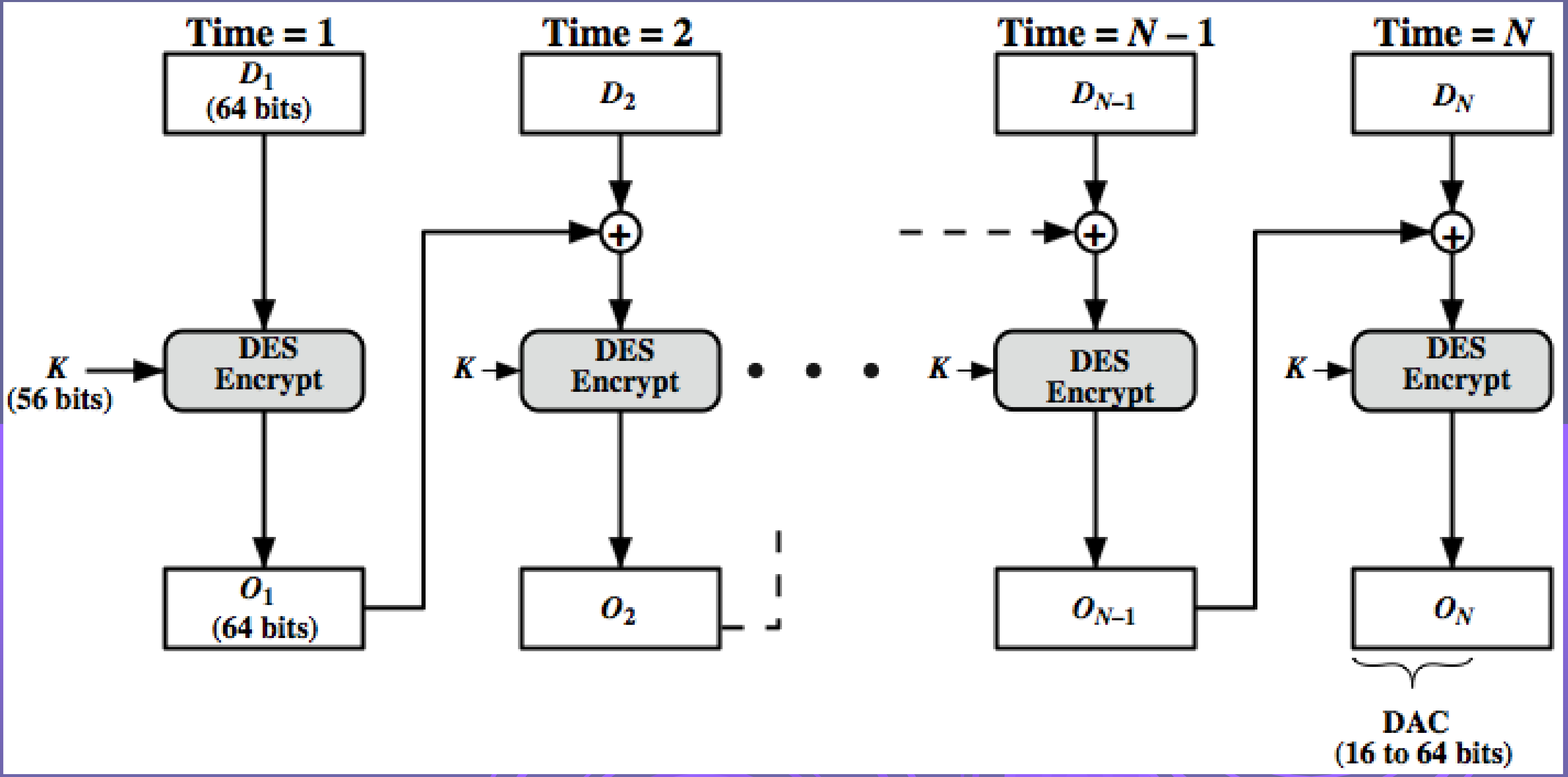
- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits ($16 \leq M \leq 64$) of final block
- but final MAC is now too small for security…

… can use message blocks in reverse order…

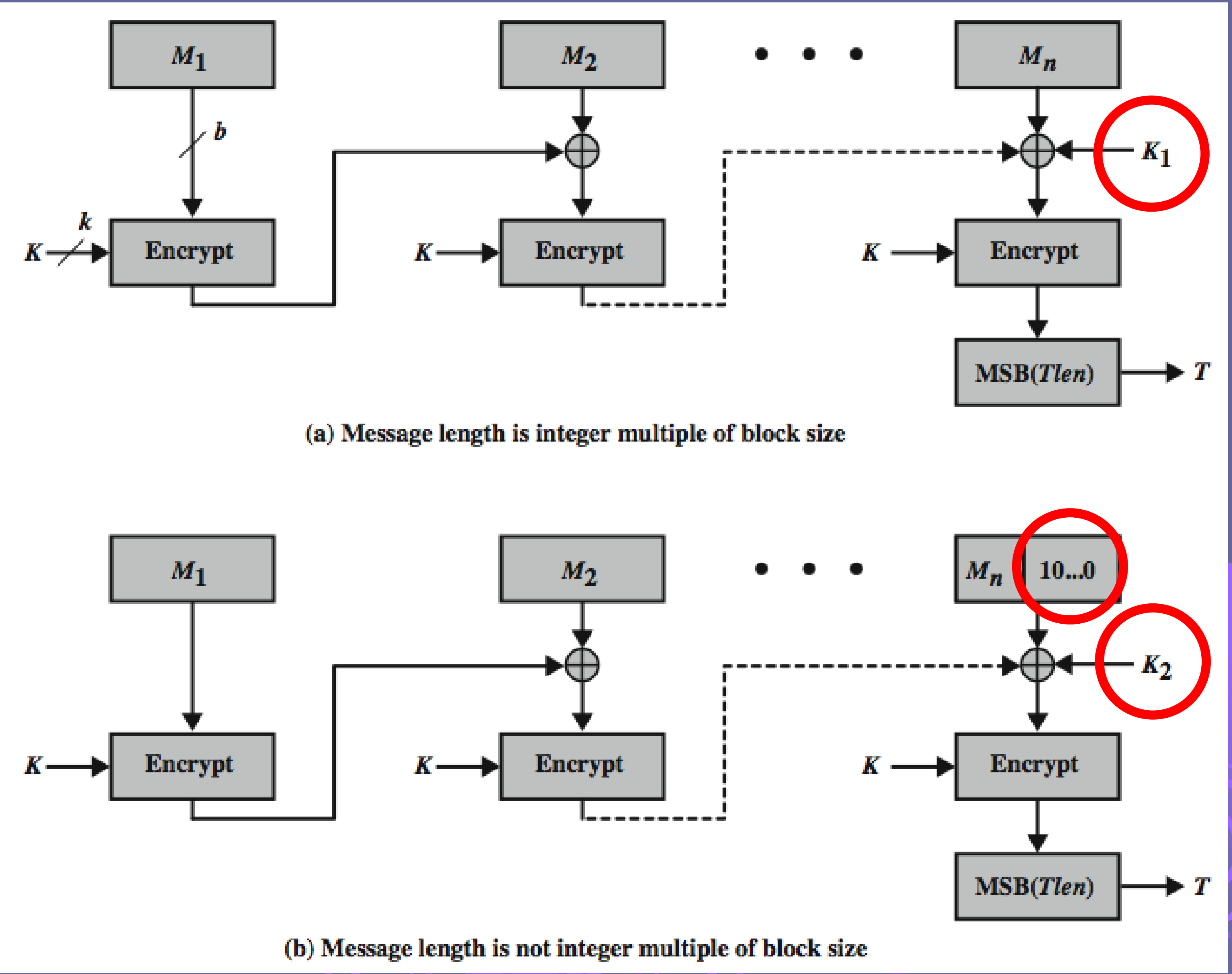
Data Authentication Algorithm



CMAC

- previously saw the DAA (CBC-MAC)
- widely used in govt & industry
- but has message size limitation
- can overcome using 2 keys & padding
- thus forming the Cipher-based Message Authentication Code (CMAC)
- adopted by NIST SP800-38B

CMAC Overview



Road Map

- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

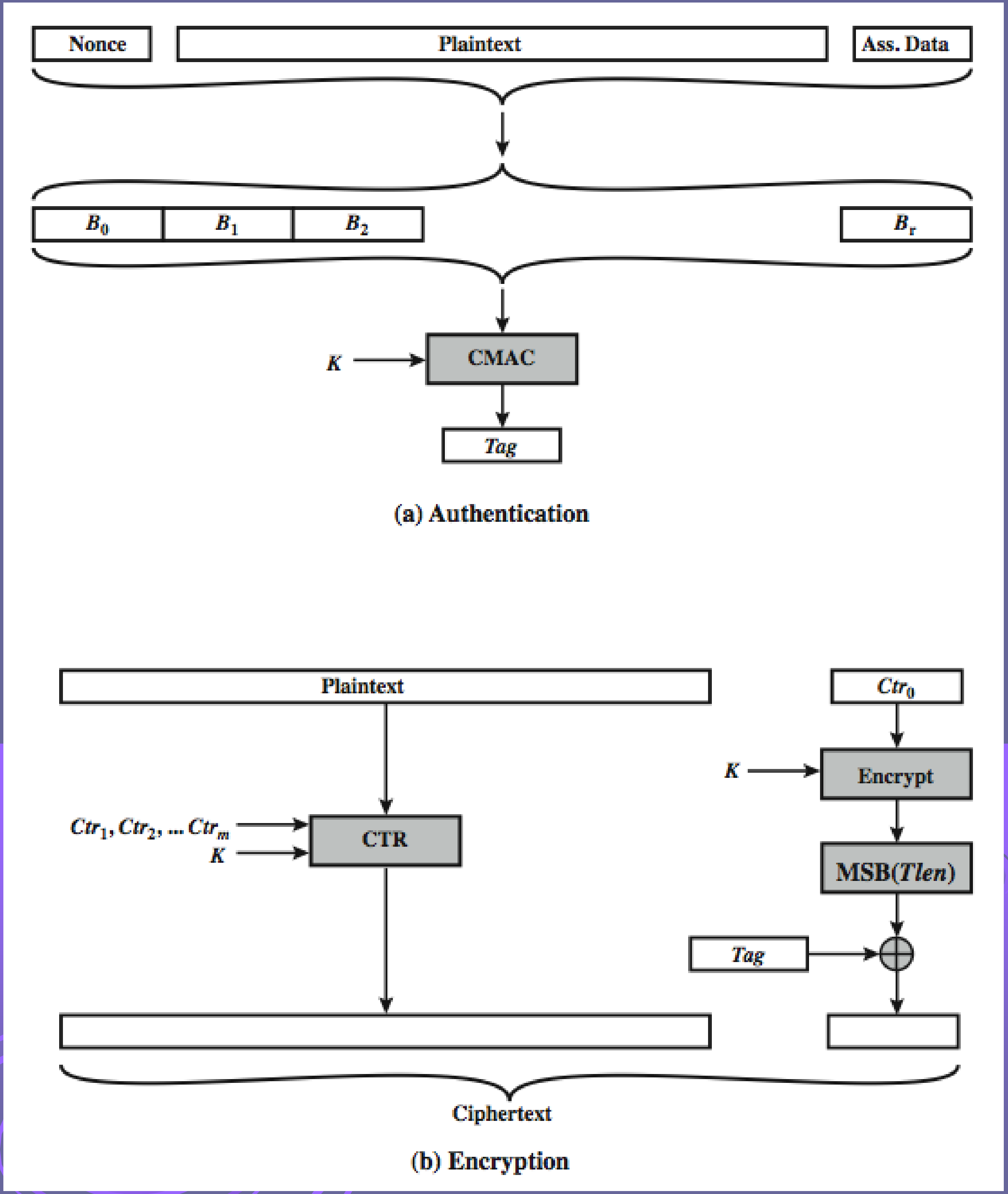
Authenticated Encryption

- simultaneously protect confidentiality and authenticity of communications
  - often required but usually separate
- approaches
  - Hash-then-encrypt: E(K, (M || H(M))
  - MAC-then-encrypt: E(K2, (M || MAC(K1, M))
  - Encrypt-then-MAC: (C=E(K2, M), T=MAC(K1, C)
  - Encrypt-and-MAC: (C=E(K2, M), T=MAC(K1, M)
- decryption /verification straightforward
- but security vulnerabilities with all these

Counter with Cipher Block Chaining-Message Authentication Code (CCM)

- NIST standard SP 800-38C for WiFi
- variation of encrypt-and-MAC approach
- algorithmic ingredients
  - AES encryption algorithm
  - CTR mode of operation
  - CMAC authentication algorithm
- single key used for both encryption & MAC

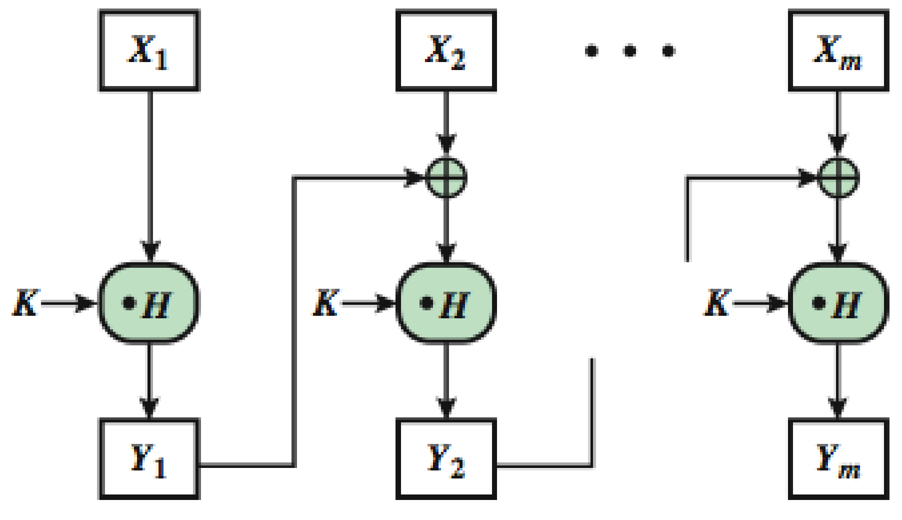
CCM Operation



Galois/Counter Mode (GCM)

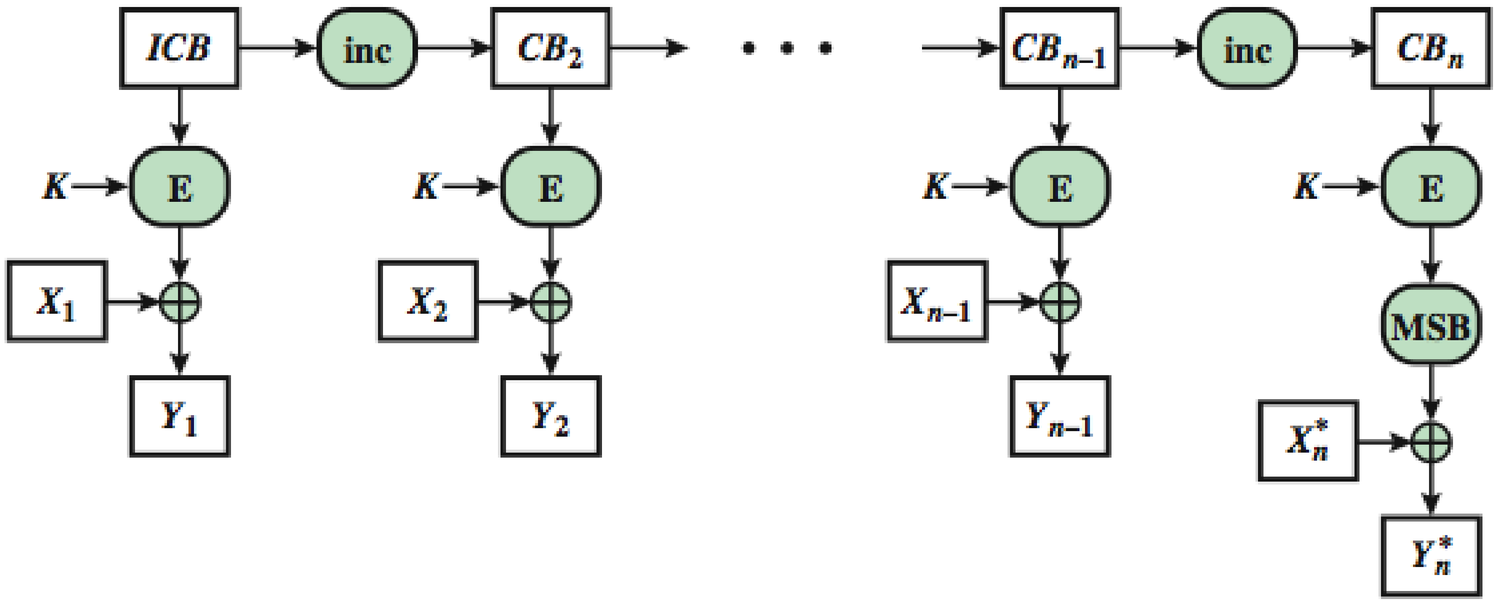
- NIST standard SP 800-38D, parallelizable
- message is encrypted in variant of CTR
- ciphertext multiplied with key & length over $GF(2^{128})$ to generate authenticator tag
- have GMAC MAC-only mode also
- uses two functions:
  - GHASH - a keyed hash function
  - GCTR - CTR mode with incremented counter

GCM Functions



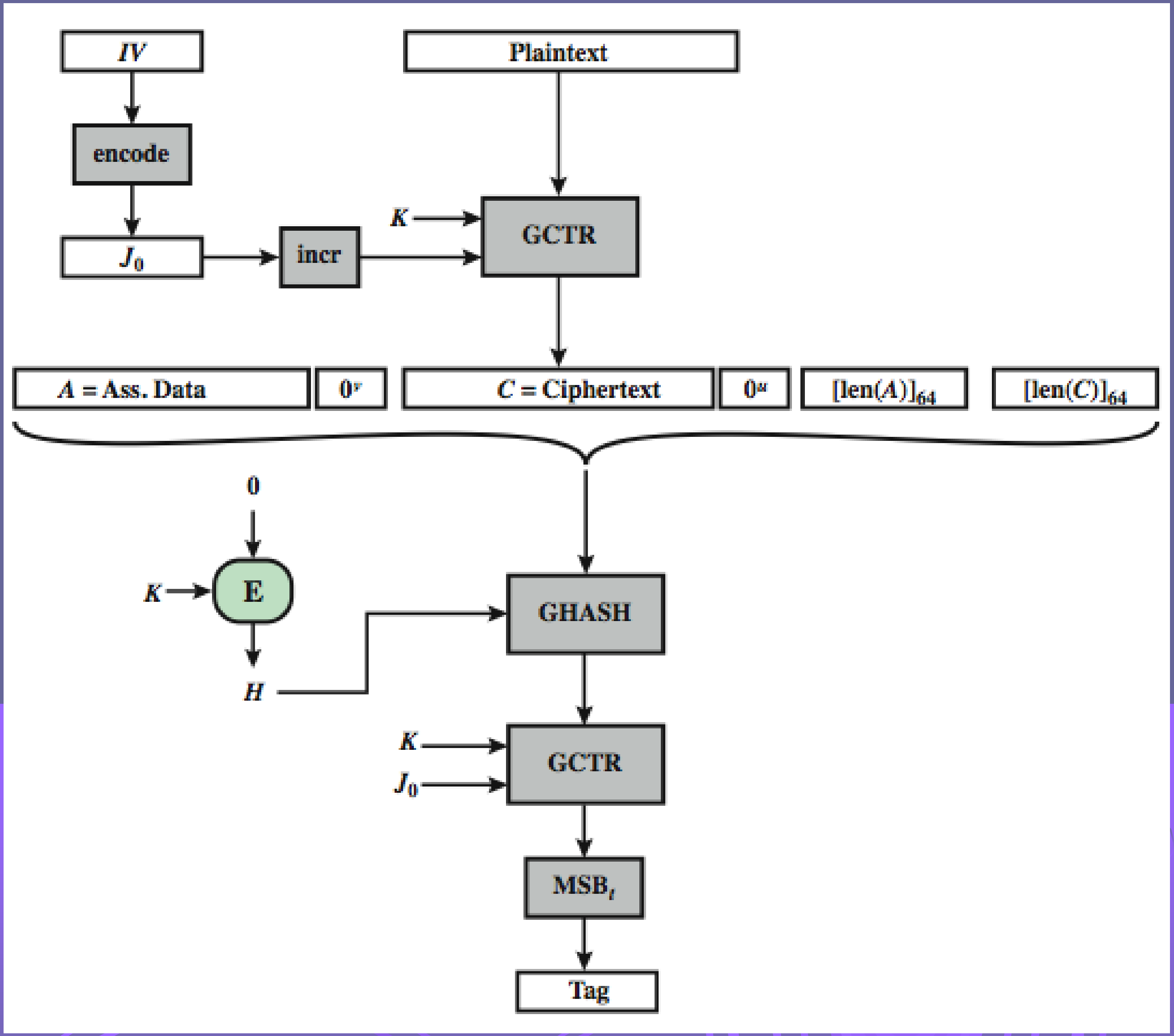
(a) $\mathrm{GHASH}_H(X_1 \| X_2 \| \ldots \| X_m) = Y_m$

GCM Functions



(b) $\mathrm{GCTR}_K(ICB, X_1 \| X_2 \| \ldots \| X_n^*) = Y_n^*$

GCM Mode Overview



Authenticated Encryption

- Want confidentiality and integrity/authenticity
- Use combination of encryption
  - but how?
- Generic Composition:
  - "Foolproof" ways to combine (compose) encryption and MAC to achieve AE
  - Trouble is, fools are so clever!

Generic Composition

- Classic result by Bellare & Namprempre
- Basic compositions (BN 2000)
  - MAC then Encrypt
  - Encrypt then MAC
  - Encrypt and MAC
- Major result:
  - Only Encrypt then MAC is always safe
  - But caveats – depends on assumptions of encrypt...

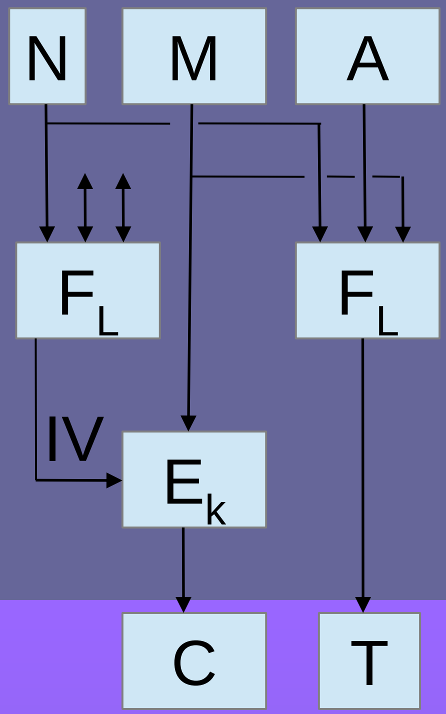
Generic Composition

- Recent reconsideration by Namprempre, Rogaway & Shrimpton (2014)
- 160 possible compositions - A-schemes
    - 8 “favored” A-schemes - always good
    - 1 “transitional” A-scheme - inferior
    - 3 “elusive” A-schemes - not sure
    - 148 are nonsense or wrong
- Convert to B-schemes

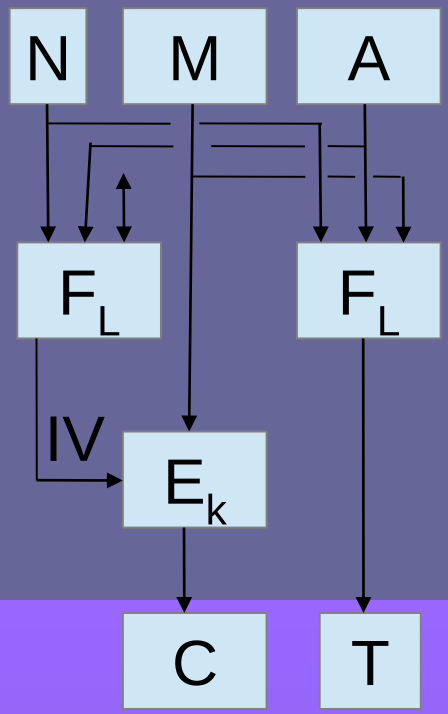
Generic Composition

- A-schemes use
  - IV-based encryption (ivE)
  - Vector MAC (vecMAC)
- B-schemes use
  - IV-based encryption (ivE)
  - String MAC (strMAC)
- Both produce nAE
  - Nonce-based Authenticated Encryption

A-Schemes



Scheme A-1



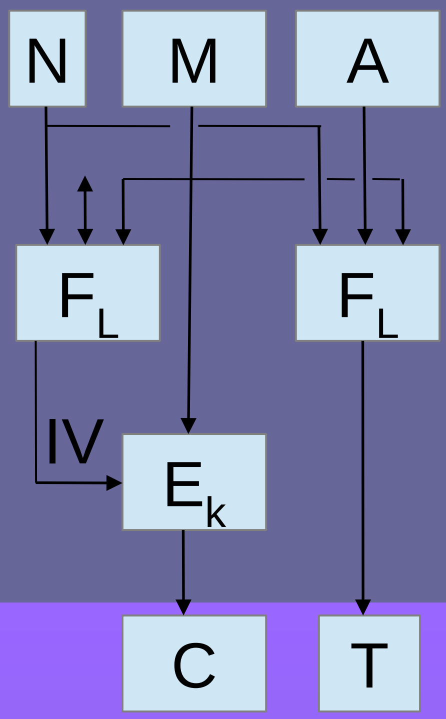
Scheme A-2

N=nonce, M=msg, A=associated data

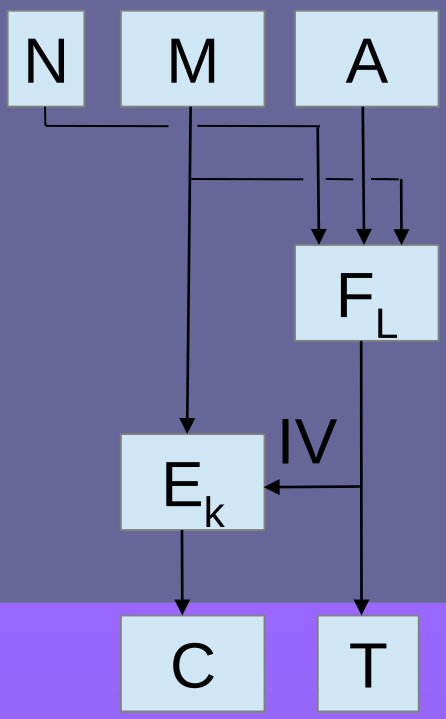
F_L=keyed MAC with key L, E_K = encryption with key K

C = ciphertext, T = tag (MAC value)

A-Schemes



Scheme A-3



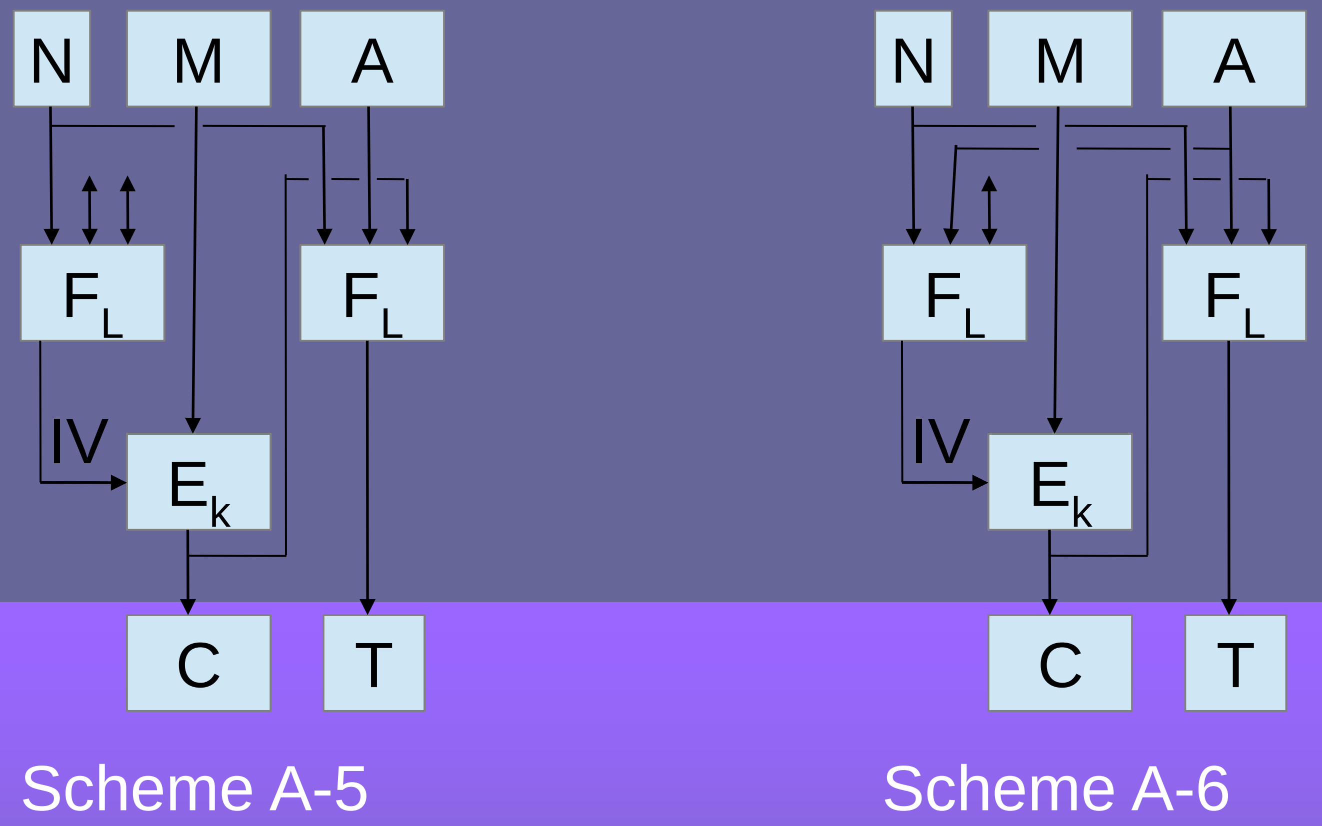
Scheme A-4

N=nonce, M=msg, A=associated data

F_L=keyed MAC with key L, E_K = encryption with key K

C = ciphertext, T = tag (MAC value)

A-Schemes

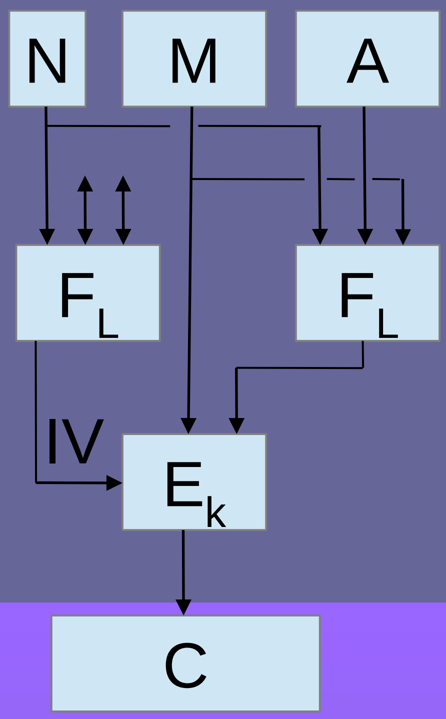


N=nonce, M=msg, A=associated data

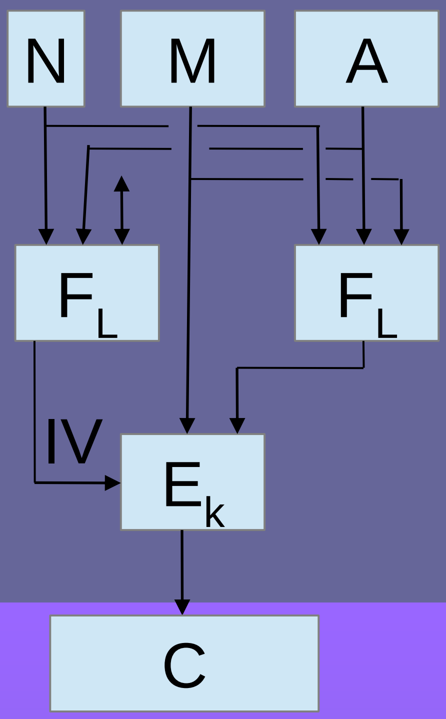
F_L=keyed MAC with key L, E_K = encryption with key K

C = ciphertext, T = tag (MAC value)

A-Schemes



Scheme A-7



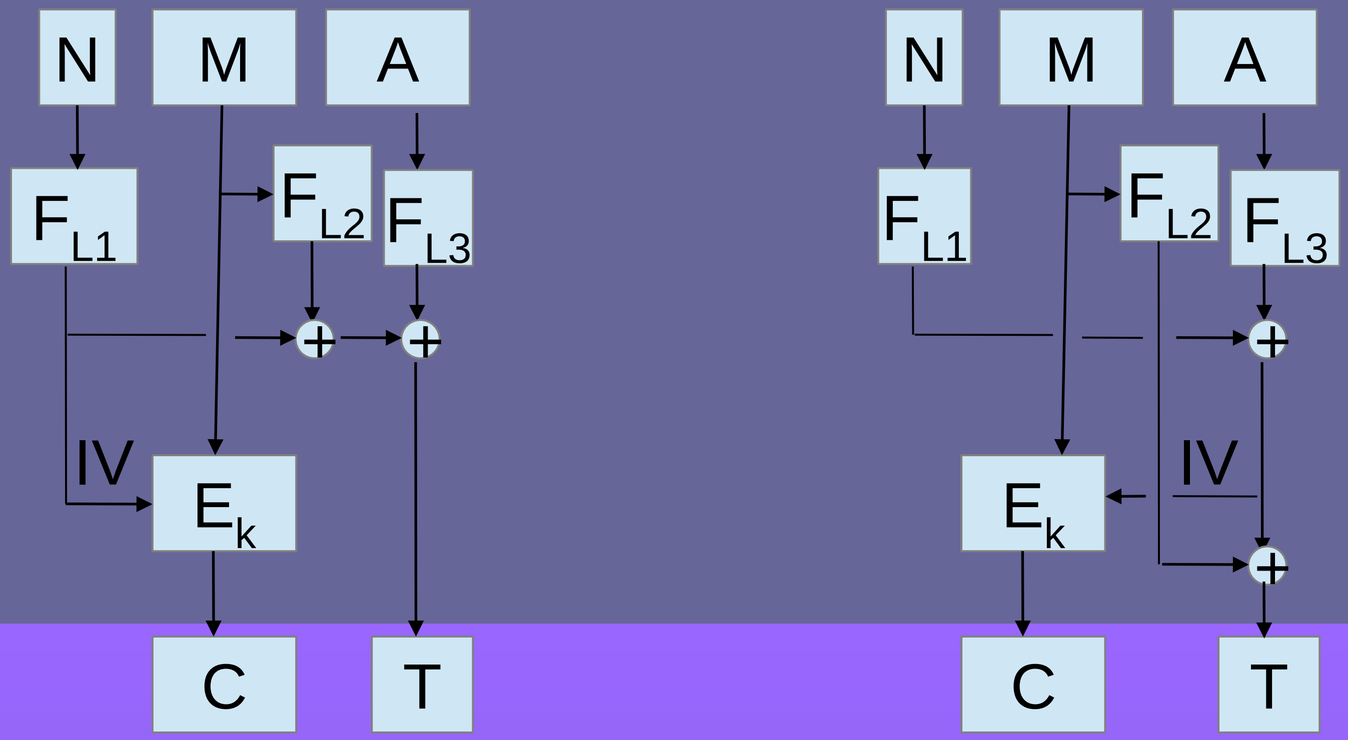
Scheme A-8

N=nonce, M=msg, A=associated data

F_L=keyed MAC with key L, E_K = encryption with key K

C = ciphertext, T = tag (MAC value)

B-Schemes



Scheme B-1

Scheme B-2

N=nonce, M=msg, A=associated data

F_L=keyed MAC with key L, E_K = encryption with key K

C = ciphertext, T = tag (MAC value)

Generic Composition

- 6 more B-schemes
- Built similarly (use XOR and strMAC)
- Bottom line:

Must understand nature of encryption, nonces vs. random values, etc.

Road Map

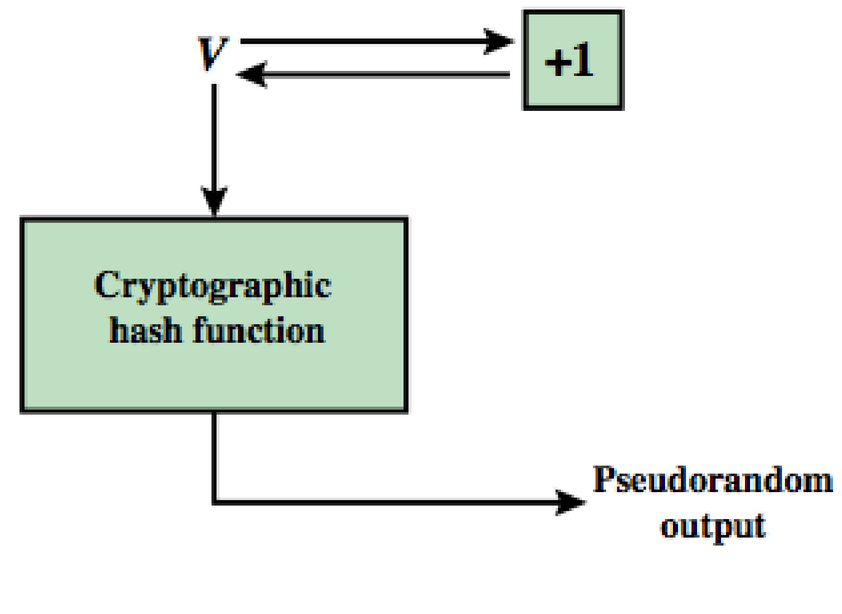
- Topics
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs

Pseudorandom Number Generation (PRNG) Using Hash Functions and MACs

- essential elements of PRNG are
  - seed value
  - deterministic algorithm
- seed must be known only as needed
- can base PRNG on
  - encryption algorithm (Chs 7 & 10)
  - hash function (ISO18031 & NIST SP 800-90)
  - MAC (NIST SP 800-90)

PRNG using a Hash Function

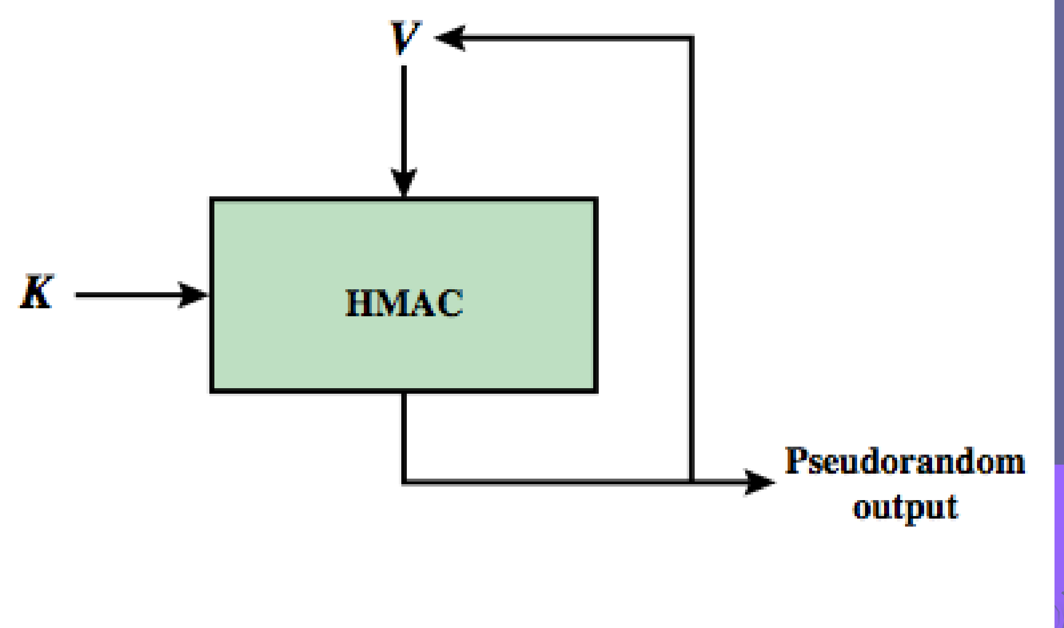
- hash PRNG from SP800-90 and ISO18031
  - take seed V
  - repeatedly add 1
  - hash V
  - use n-bits of hash as random value
- secure if good hash used



(a) PRNG using cryptographic hash function

PRNG using a MAC

- MAC PRNGs in SP800-90, IEEE 802.11i, TLS
  - use key
  - input based on last hash in various ways



(b) PRNG using HMAC

Summary

- have considered:
  - message authentication requirements
  - message authentication using encryption
  - MACs
  - HMAC authentication using a hash function
  - CMAC authentication using a block cipher
  - Generic Composition for Authenticated Encryption
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs