CRYPTOGRAPHY AND ENCRYPTION (CY 371)

Dr Eric Affum

Message Authentication Codes

Message Authentication

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)

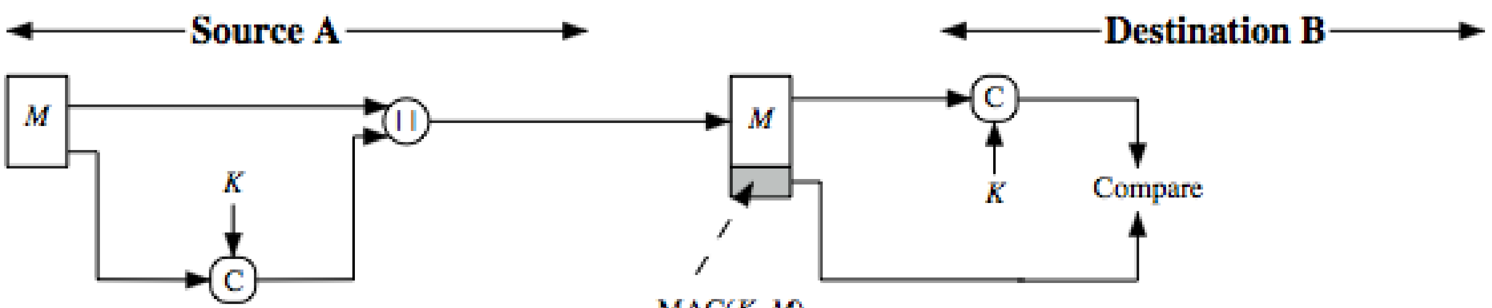
- The three alternative functions used:
  - hash function
  - message encryption
  - message authentication code (MAC)

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

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

  MAC = CK(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 2m/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=M1 M2 … ML, where |Mi| = Blocksize

  Hash=H(H(…H(H(IV,M1),M2),…,ML)
- But can add extra block(s) ML+1 by

  Hash'=H(Hash,ML+1)
- Unless formatting prevents it…

… but still best to use HMAC!

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

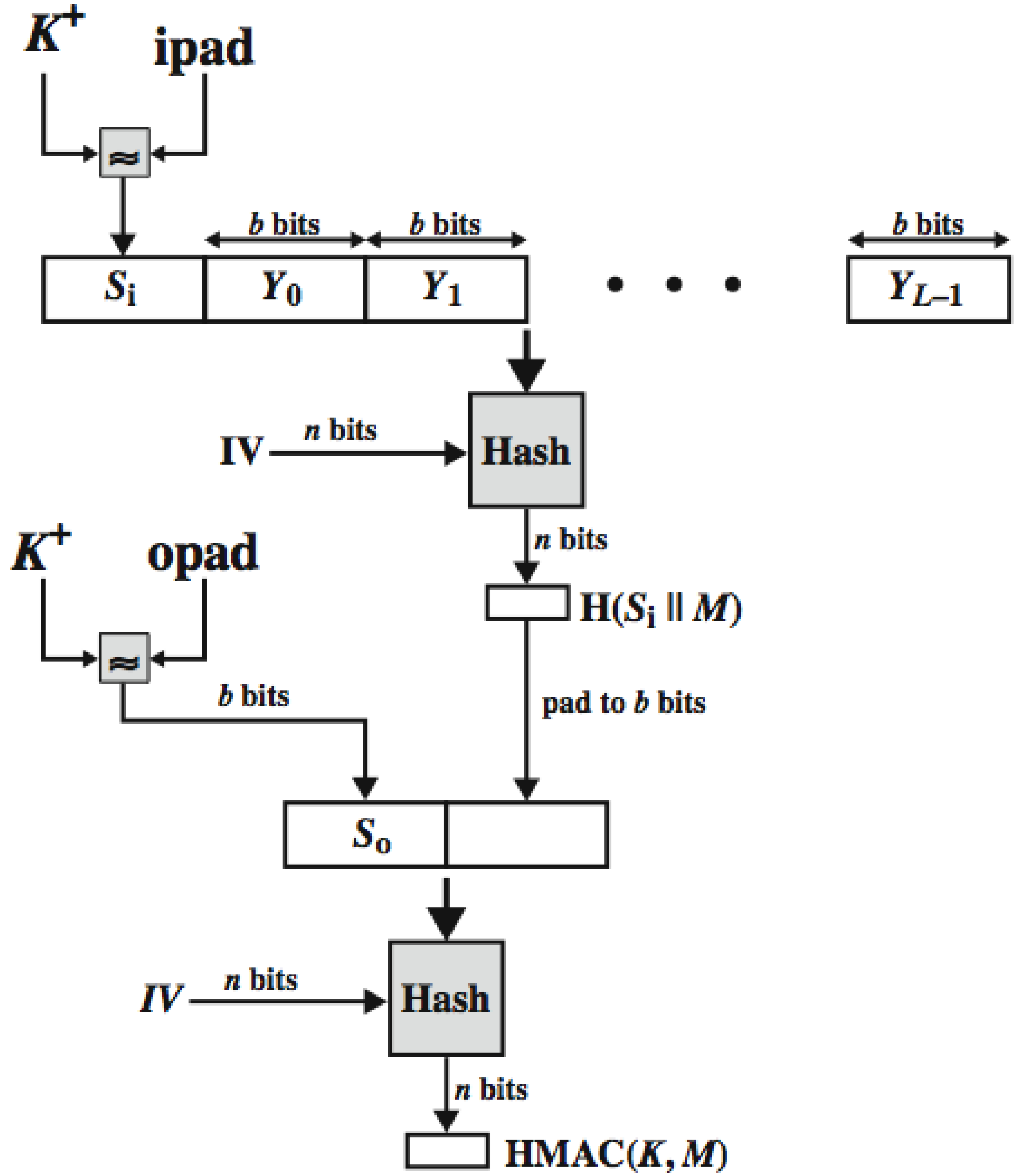
- Specified as Internet standard RFC2104
- uses hash function on the message:

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

Hash[(K+ XOR ipad) || M)]]

  - where 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 algorithmhhhhhh
- 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