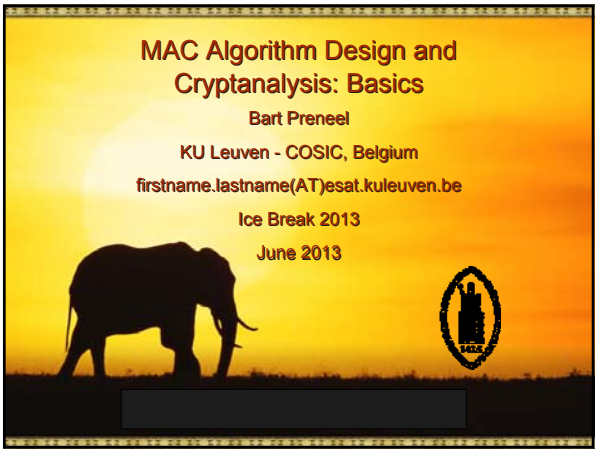
MAC Algorithm Design and Cryptanalysis: Basics

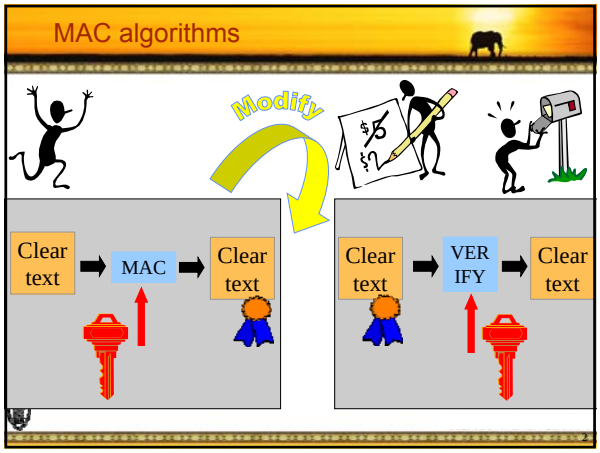
Bart Preneel
KU Leuven - COSIC, Belgium
firstname.lastname(AT)esat.kuleuven.be

Ice Break 2013

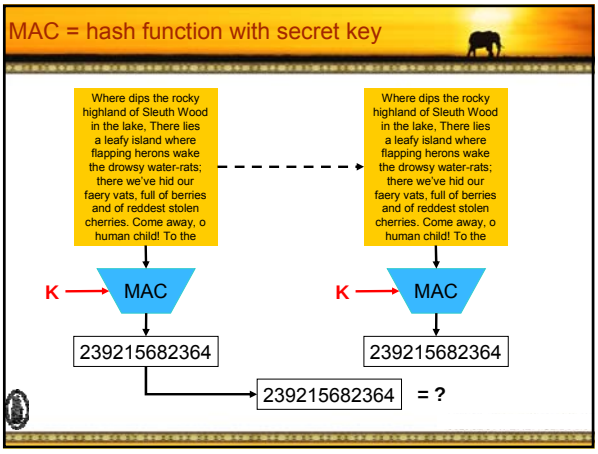
June 2013



MAC algorithms



MAC = hash function with secret key



MAC: Definition

Message Authentication Code

= hash function with secret key:

1. Description of h public
2. X arbitrary length ⇒ fixed length m (32 . . . 160 bits)
3. Computation of $h_K(X)$ "easy" given X and K
4. Computation of $h_K(X)$ "hard" given only X, even if a large number of pairs $\{X_i, h_K(X_i)\}$ is known

- Calculation of $h_K(X)$ without knowledge of secret key: *forgery* (verifiable or not verifiable)

MAC: generic attacks

1. **Guess MAC**: ± same as for hash function
   - On-line verification only
   - Not verifiable
   - Success probability max ($1/2^m$, $1/2^k$)
2. **Exhaustive key search**: ± same as for block cipher
   - # X, $h_K(X)$ pairs ≈ k/m
   - # attempts ≈ 2^{k-1}
3. **Birthday paradox on iterated MAC algorithms**

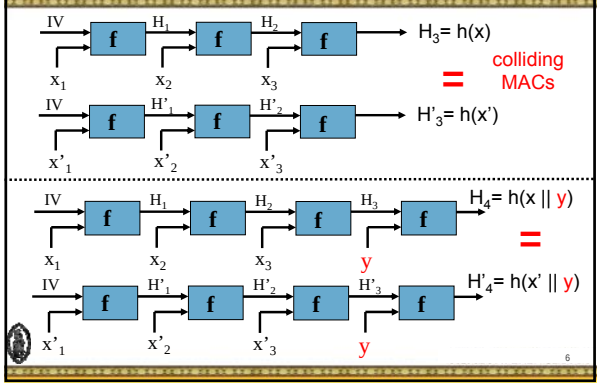
   Internal memory n bits; result m bits

   (output transformation g)

   Forgery after $2^{n/2}$ known and ≤ 2^{n-m} chosen texts

Iterated MAC algorithms

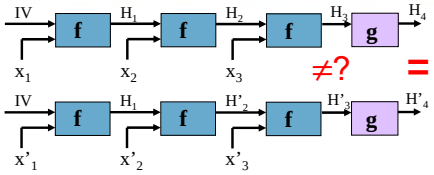
length extension: if one knows h(x), easy to compute h(x ||y) without knowing x or IV



Collision attack on iterated MAC algorithms

- Collision in MAC values leads to trivial forgery after 1 chosen text-MAC pair
  - indeed: h(x) = h(x') $\Rightarrow$ h(x || y) = h(x' ||y)
- If an opponent queries h(x||y), he can forge h(x' || y)
- MAC value of m bits: need $2^{m/2}$ known text-MAC pairs to find a MAC collision

It. MAC algorithms with output transformation

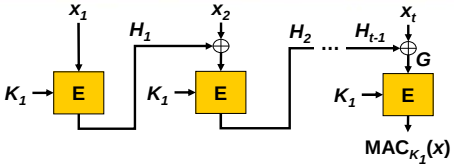


- If g is a injective (fewer input bit than output bits):
  - h(x) = H_4 = H'_4 = h(x') but it may be that $H_3 \neq H'_3$
- If $H_3 \neq H'_3$ the attack will likely fail: h(x || y) ≠ h(x' || y)
- Conclusion: attack requires that $H_3 = H'_3$ (**internal collision**)

Collision attack on iterated MAC algorithms

- Solution: simulate the first attack
- For all MAC collisions (h(x) =h(x')) also ask for h(x || y) and h(x' ||y)
- If h(x || y) = h(x' ||y), we have likely found an internal collision (a collision for H_3 in our example)
- Attack complexity: $2^{n/2}$ known text-MAC pairs and 2^{m-n} chosen text-MAC pairs

MAC based on a block cipher: CBC-MAC



- Standards (ANSI, ISO, IEC)
- Proof of security by [Bellare-Kilian-Rogaway]
- *m* = 32 . . . 64 bits
- Special operation for last block is essential: EMAC, LMAC or CMAC (cf. infra)

Based on a block cipher: CBC-MAC (2)

Security with DES:

- Key search: 2^{56} encryptions
- Key recovery using Ic: 2^{43} known texts
- Guess MAC: max($1/2^{56}$, $1/2^m$)
- Birthday forgery attack (even if triple-DES):
  - *m* = 64: 2^{32} known and 1 chosen text
  - *m* = 32: 2^{33} chosen texts
- Improved attack for *m* = 32: 2^{17} chosen texts and 2 known texts [Knudsen97]

Much smaller than expected!

Based on a block cipher: CBC-MAC (3)

Security with AES-128:

- Key search: 2^{128} encryptions
- Guess MAC: $1/2^m$
- Birthday forgery attack:
  - *m* = 128: 2^{64} known and 1 chosen text
  - *m* = 64: 2^{66} chosen texts
- Improved attack for *m* = 64: 2^{33} chosen texts and 2 known texts [Knudsen97]

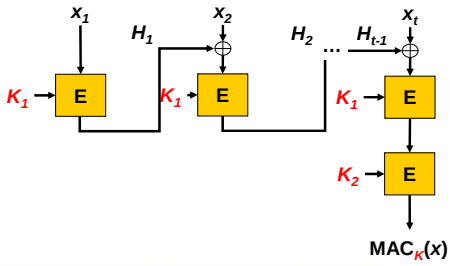
Acceptable for most applications

Exercises: CBC-MAC forgery

- simple CBC-MAC is not secure on message spaces of variable length
  - exercise: consider a 1-block input x consisting of n bits and assume that you know $MAC_K(x)$; show that it is possible to find the MAC for a specific 2-block input
  - exercise: consider two 1-block inputs x and x' consisting of n bits and assume that you know $MAC_K(x)$ and $MAC_K(x')$; show that it is possible to find the MAC for a specific 2-block input

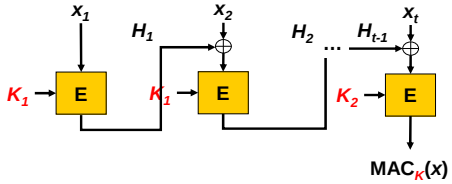
MAC based on a block cipher: EMAC

Better way to process last block: encrypted MAC (EMAC) [RIPE'93][Petrank-Rackoff'98]



MAC based on a block cipher: LMAC

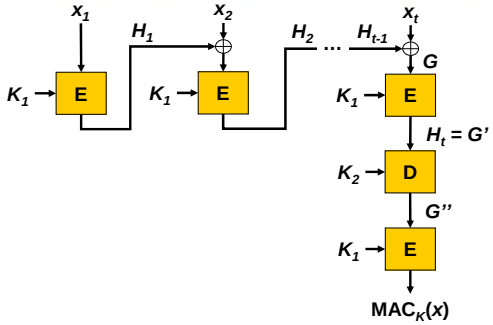
An even better way to process last block: LMAC [Handschuh-Preneel'06]



NIST: CMAC

- Description: use simple CBC-MAC but
  - Derive two keys from K_1: $K_2 = E\ K_1(0)$ and K'_2 is derived from K_2 with a simple finite field operation
  - XOR K_2 or K'_2 to the last plaintext block (first choice if no padding, second choice if there is padding)
- Evaluation
  - This saves 1 key schedule and 1 encryption (if the length of the plaintext is an exact multiple of the block length)
  - Price to pay is robustness: K_2 and K'_2 can be recovered with an internal collision attack
  - Banks send the value $K_2 = E\ K_1(0)$ as (public) key confirmation value!
- Note on name
  - This was called OMAC by its designers [Iwata-Kurosawa]
  - OMAC is an optimized version of TMAC which is an optimized version of XCBC [Black-Rogaway'00]

MAC based on a block cipher: retail MAC



Based on a block cipher: retail MAC (2)

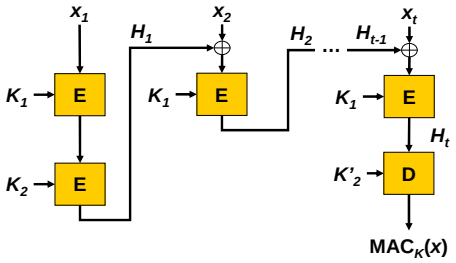
Security with DES and $m = 64$:

- Key search: 2^{112} encryptions
- Guess MAC: $\max(1/2^{56}, 1/2^m)$
- (first attack is based on guessing K_1)
- Birthday forgery attack: 2^{32} known and 1 chosen text
- Improved key recovery [Preneel-van Oorschot-Knudsen]
  - $2^{32.5}$ known texts and $3 \cdot 2^{56}$ off-line encryptions
  - 1 known text + 2^{56} MAC verifications + 2^{57} off-line encryptions

Solution: triple-DES in first and last round?

MAC based on a block cipher: Mac-DES (1)

[Knudsen-Preneel'98]



Based on a block cipher: Mac-DES (2)

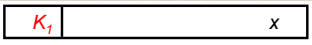
Security with DES and m = 64:

- Key search: 2^{112} encryptions
- Guess MAC: max($1/2^{112}$, $1/2^m$)
- Birthday forgery attack: 2^{32} known and 1 chosen text
- Improved key recovery [Coppersmith-Mitchell-Knudsen2000]:
  - 2^{48} chosen texts and 2^{59} off-line encryptions

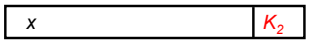
Included in ISO/IEC 9797-1 (revision, 1999)

MAC: based on an MDC?

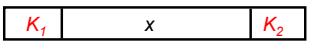
- **Secret prefix**: $h(K_1||x)$

  Prepend length to avoid that one can compute $h(K_1||x||y)$ from $h(K_1||x)$ without knowing K_1
- **Secret suffix**: $h(x||K_2)$

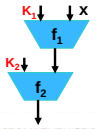
  

  Off-line attacks on h
- **Envelope**: $h(K_1||x||K_2)$

  Risky: less secure than h
- Better variants:
  - **MDx-MAC** and **H-MAC**:
  - $h_K(X) = h(h(K_1||x)||K_2)$

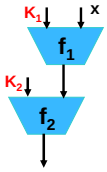
HMAC



HMAC

- HMAC keys through the IV (plaintext) [Kim+'06]
  - collisions for MD5 invalidate current security proof of HMAC-MD5
  - new attacks on reduced version of HMAC-MD5 and HMAC-SHA-1

	Rounds in f2	Rounds in f1	Data complexity
Haval-4	128	102 of 128	2^{254} CP
MD4	48	48	2^{72} CP + 2^{77} time
MD5	64	33 of 64	$2^{126.1}$ CP
MD5	64	64	2^{51} CP & 2^{100} time (RK)
SHA-0	80	80	2^{109} CP
SHA-1	80	53 of 80	$2^{98.5}$ CP



no problem yet for most widely used schemes

Information-theoretic authentication

Authentication codes (AC): unconditionally secure

= independent of computational power of opponent

- Research area since mid 1970s
- Widely believed to be impractical:
  - Use key only once
  - Sometimes very large keys
  - Security level against forgery is at most half the key size

In 1990s series of new schemes:

Polynomial evaluation, Toeplitz, bucket hashing, MMH, UMAC,. . .

GMAC: polynomial authentication code

(NIST SP 800-38D 2007 + 3GSM)

- keys K_1, $K_2 \in GF(2^{128})$
- input x: $x_1, x_2, \ldots, x_t$, with $x_i \in GF(2^{128})$

$$g(x) = K_1 + \Sigma_{i=1}^{t} x_i \cdot (K_2)^i$$

- in practice: compute $K_1 = AES_K(n)$ (CTR mode)
- properties:
  - lightweight and/or fast in software and hardware (support from Intel/AMD)
  - not very robust w.r.t. nonce reuse, truncation, MAC verifications, due to reuse of K_2 *(not in 3GSM!)*
  - efficient through reuse of fast arithmetic
    - (Intel/AMD) PCLMULQD: 10.68 cycles/byte [Käsper-Schwabe09]
  - weak keys [Saarinen11][Cid-Procter'13]
  - versions over GF(p) (e.g. Poly1305-AES) seem more robust

UMAC RFC 4418 (2006)

- key $K, k_1, k_2 .., k_{256} \in GF(2^{32})$ *(1024 bytes)*
- input x: $x_1, x_2, \ldots, x_{256}$, with $x_i \in GF(2^{32})$

$g(x) = prf_K(h(x))$

$h(x) = \left(\sum_{i=1}^{512} (x_{2i-1} + k_{2i-1}) \bmod 2^{32} \cdot (x_{2i} + k_{2i}) \bmod 2^{32} \right) \bmod 2^{64}$

- properties
  - software performance: 1-2 cycles/byte
  - forgery probability: $1/2^{30}$ (provable lower bound)
  - [Handschuh-Preneel08] full key recovery with 2^{40} verification queries (no nonce reuse needed!)
  - Similar attack applies to WMAC polynomial variant

Information-theoretic authentication

- simple
- very high speeds
  - UMAC/VMAC: up to 0.5-2 cycles/byte for long messages;
  - poly1305-AES up to 4-5 cycles/byte
- parallelizable
- hardware support by Intel (GCM)
- use key only once!
- consecutive keys can be generated with an additive stream cipher
  - but then the unconditional security is lost
- speed comes at cost of large keys (e.g. UMAC): key reuse??
- if part of keys is reused: key recovery attacks
- not robust: nonce reuse can also lead to key recovery

Conclusion: use polynomial hash functions but avoid key reuse as specified in several standards

Authenticated encryption

- Default modes: ECB/CBC/CFB/OFB and CTR
- Needed for network security, but only fully understood by crypto community around 2000 (too late)
- Standards have been selected recently:
  - CCM: CTR + CBC-MAC [NIST SP 800-38C]
  - GCM: CTR + GMAC [NIST SP 800-38D]
- Both are suboptimal

Issues:
- associated data
- parallelizable
- on-line
- provable security

- IAPM
- XECB
- OCB
- GCM
- CCM
- (EAX)

patented

MDC ↔ MAC

Authentication without secrecy

- **MAC**: obvious solution
- **MDC**: protect authenticity of hash result

Authentication with secrecy

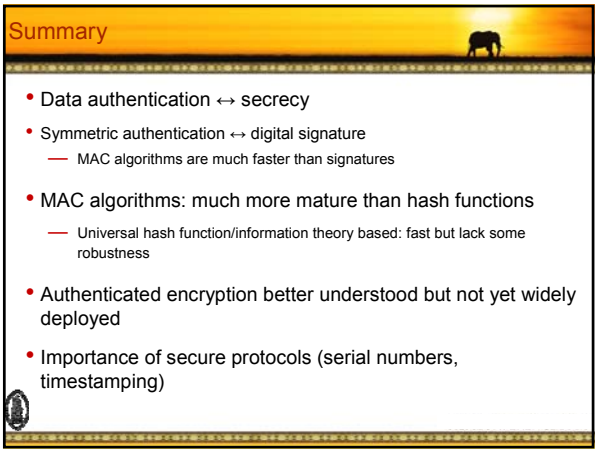
- **MAC**: needs 2 INDEPENDENT keys (MAC then encrypt, encrypt then MAC or MAC then encrypt then MAC)
- **MDC**: only 1 key, but important security risks: avoid this approaches
- Clear choice: authenticated encryption mode
  - Many solutions: OCB, IAPM, XECB, GCM, CCM,. . .

Performance

- Modern processor
  - HMAC, MDx-MAC: 13.1 cycles/byte for SHA-1 and 15.8 cycles/byte for RIPEMD-160
  - CBC-MAC: 43 cycles/byte for DES and 7-14 cycles/byte for AES
  - Universal hash functions: 2-4 cycles/byte
- Better performance than encryption if one is willing to pay the price in robustness

Practical recommendations

- CBC-MAC variant of AES (LMAC or CMAC)
- HMAC-RIPEMD-160 or HMAC-SHA-1
- Universal hash function based
  - GMAC but replace K_2 for every message as in 3G
  - Poly1305-AES


Summary
• Data authentication ↔ secrecy
• Symmetric authentication ↔ digital signature
— MAC algorithms are much faster than signatures
• MAC algorithms: much more mature than hash functions
— Universal hash function/information theory based: fast but lack some robustness
• Authenticated encryption better understood but not yet widely deployed
• Importance of secure protocols (serial numbers, timestamping)