

# MAC Algorithms: State of the Art and Recent Developments

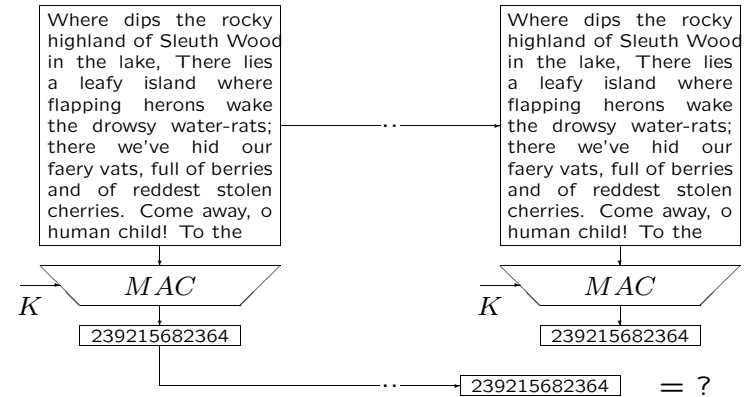
Prof. Bart Preneel

Katholieke Universiteit Leuven bartDOTpreneel(AT)esatDOTkuleuvenDOTbe

<http://homes.esat.kuleuven.be/~preneel>

May 2008

MAC = hash function with secret key



3

## MAC Algorithms: overview

1. definition

2. applications

3. attacks

4. constructions

- based on block ciphers
- based on hash functions
- dedicated MACs
- based on authentication codes/universal hash functions

## MAC: definition (1)

### Message Authentication Code

= hash function with secret key:

1. description of  $h$  public
2.  $X$  arbitrary length  $\Rightarrow$  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
- selective or existential

## MAC: definition (2)

A MAC is secure if, for an adversary who does not know  $K$ , it is computationally infeasible to perform an existential forgery under an adaptive chosen text attack.

A MAC is  $(\epsilon, t, q, q', L)$  secure if, an adversary who does not know  $K$ , and

- can spend time  $t$  (operations),
- can obtain the MAC for  $q$  texts of this choice,
- can observe the MAC for  $q'$  texts (not of his choice),
- and can obtain the result of  $q''$  verification queries on text-MAC pairs of his choice.

(each message of length  $L$ ),  
cannot produce an existential forgery with probability of success  $> \epsilon$ .

5

## Applications

MAC versus digital signature:

- non-repudiation
- key management
- + performance/computational effort
- + size of MAC and of keys

- banking
- Internet security: IP security
- electronic purses + authorization for credit cards

6

## Attack: exhaustive key search

try all values of the key  $K$

- $\#X, h_K(X)$  pairs  $\approx k/m$
- $\#$  attempts  $\approx 2^{k-1}$
- a recovered key is only valuable within its lifetime
- but allows for arbitrary forgery

long term security: 80...100 bits

Attack complexity:  $[2^k, k/m, 0, 0]$   
or  $[2^{2k/3}, 0, k/m, 0]$  with  $2^k$  offline work

notation: [work, known texts, chosen texts, on-line verifications]

7

## 'Attack': guess MAC

success probability  $\max(1/2^m, 1/2^k)$

but:

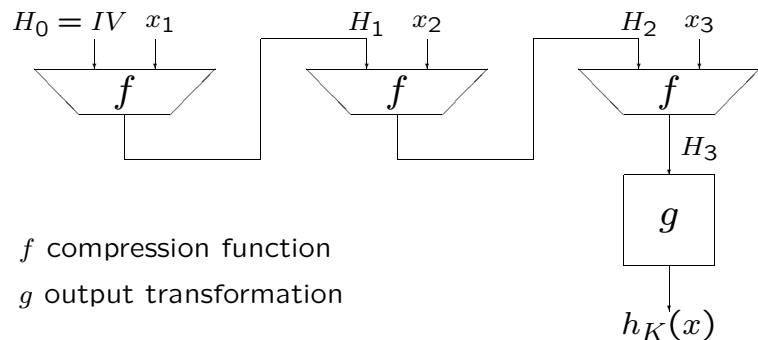
- not verifiable
- requires on-line verification

depending on application:  $m, k \geq 32 \dots 64$

Attack complexity:  $[0, 0, 0, \min(2^m, 2^k)]$

8

## Attack: birthday forgery attack (1)



unambiguous padding of input to multiple of block length  
 divide input into blocks  $x_1, x_2, \dots, x_t$

9

## Birthday forgery attack (2)

find  $(x, x')$ :  $h(x) = g(H_t) = h(x') = g(H'_t)$

**internal collision:**  $H_t = H'_t$

**external collision:**  $H_t \neq H'_t$  but  $g(H_t) = g(H'_t)$

note: if  $g$  is bijective, there are no external collisions

**Lemma 1** An internal collision for an iterated MAC can be used for a forgery which requires only a single chosen text.

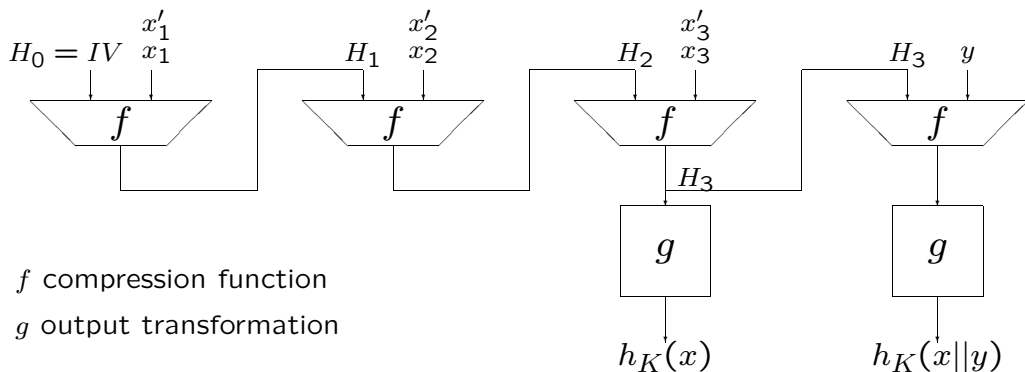
indeed:  $h_K(x \| y) = h_K(x' \| y)$

and thus one can predict the 2nd MAC without knowing  $K$

note: this does NOT work for an external collision.

11

## Attack: birthday forgery attack (2)



unambiguous padding of input to multiple of block length  
 divide input into blocks  $x_1, x_2, \dots, x_t$

10

## Birthday forgery attack (3)

- internal memory ( $H_i$ ):  $n$  bits
- MAC value:  $m$  bits

[Preneel-van Oorschot 95] forgery:  $[0, 2^{n/2}, \leq 2^{n-m}, 0]$ :

- internal collision after  $2^{n/2}$  text-MAC pairs (by the birthday paradox)
- # external collisions is equal to  $(2^{n/2})^2 / 2^{m+1} = 2^{n-m-1}$  (or 0 if  $g$  is a permutation)
- distinguish internal/external by simulating the attack of lemma 1 (fails with high probability for external collisions only; 2 chosen texts per collision)

small improvement:  $[0, 2^{n/2}, \min(2^{n/2}, 2^{n-m}), 0]$

12

## Birthday paradox

Given a set with  $S$  elements.

Choose  $r$  elements at random (with replacements), with  $r \ll S$ .

Find probability  $p$  that there at least two identical elements.

$$q = 1 - p = 1 \cdot \frac{S-1}{S} \cdot \frac{S-2}{S} \dots \frac{S-(r-1)}{S} = \prod_{k=1}^{r-1} \left(1 - \frac{k}{S}\right)$$

$$\ln q = \sum_{k=1}^{r-1} \ln \left(1 - \frac{k}{S}\right) \approx - \sum_{k=1}^{r-1} \frac{k}{S} = -\frac{r(r-1)}{2S}$$

$$p = 1 - \exp\left(-\frac{r(r-1)}{2S}\right) \quad \text{if } r = \sqrt{S} : p = 39\%$$

birthday paradox: for  $S = 365$  and  $r = 23$ ,  $p = 1/2$

intuition: number of pairs of elements is  $r(r-1)/2$ .

13

## Birthday forgery attack (4)

Let  $h$  be an iterated MAC with  $n$ -bit chaining variable and  $m$ -bit result.

An internal collision for  $h$  can be found with

$\sqrt{2} \cdot 2^{n/2}$  known texts

$v$  chosen texts.

$v = 0$  if  $g$  is a permutation;

$v = \min\left(2^{n/2}, 2\left(2^{n-m}\left(1 - \frac{1}{e}\right) + \left\lfloor \frac{n-1}{m-1} \right\rfloor\right)\right)$  otherwise.

14

## Birthday forgery attack (5)

Let  $h$  be an iterated MAC with  $n$ -bit chaining variable,  $m$ -bit result, a compression function  $f$  which behaves like a random function (for fixed  $x_i$ ), and output transformation  $g$ .

An internal collision for  $h$  can be found using  $u$  known text-MAC pairs, where each text has the same substring of  $s \geq 0$  trailing blocks, and  $v$  chosen texts.

$$u = \sqrt{2/(s+1)} \cdot 2^{n/2}$$

$v = 0$  if the output transformation  $g$  is a permutation or if  $s+1 \geq 2^{n-m+6}$

$$v = 2 \left( \frac{2^{n-m}}{s+1} \cdot \left(1 - \frac{1}{e}\right) + \left\lfloor \frac{n-1 - \log_2(s+1)}{m-1} \right\rfloor \right),$$

15

## Birthday forgery attack (6)

practical?

- not all environments allow for chosen texts
- optimization reduces number of known texts
- extension to key recovery is more dangerous

how to preclude? see later

16

# MAC algorithms based on a block cipher

ISO 9797-1 (2001)

- 6 variants of CBC-MAC
- 3 padding schemes
  - padd with zeroes
  - padd with 1 (always) followed by zeroes
  - prepend length in bits and padd with zeroes

Other schemes

- XOR MAC
- PMAC
- 3GPP-MAC
- XCBC

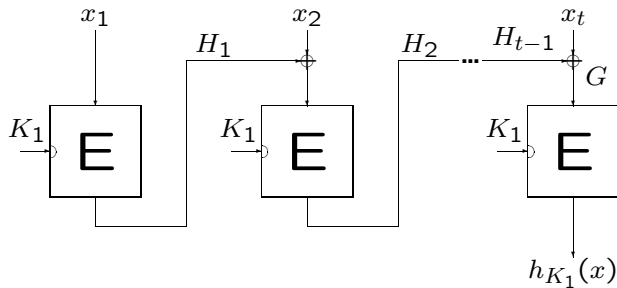
# CBC-MAC: Algorithm 1 (2)

security with DES:

- key search:  $[2^{56}, 2, 0, 0]$
- key recovery using Ic:  $[2^{43}, 2^{43}, 0, 0]$
- guess MAC:  $[0, 0, 0, \min(2^{56}, 2^m)]$
- birthday forgery attack (even if triple-DES):
  - $m = 64$ :  $[0, 2^{32}, 1, 0]$
  - $m = 32$ :  $[0, 2^{32}, 2^{33}, 0]$
- trivial forgery for  $m = 64$  if no special operation for last block
- improved attack for  $m = 32$ :  $[0, 2, 2^{16}, 0]$  [Knudsen97]

much smaller than expected!

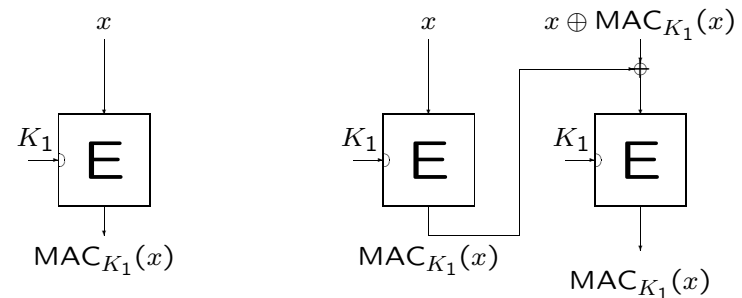
# CBC-MAC: Algorithm 1 (1)



- ANSI 1982, FIPS 185, ISO 1987, ISO/IEC 1989
- proof of security for fixed length inputs by [Bellare-Kilian-Rogaway 94]
- $m = 32 \dots 64$  bits

# Why special operation on last block? (1)

Assume for simplicity  $|x| = m = n$



## Why a special operation for the last block? (2)

$x$  consists of a single block

- $\text{MAC}_K(x)$  is known
- then  $\text{MAC}_K(x \parallel (x \oplus \text{MAC}_K(x))) = \text{MAC}_K(x)$ .

$x, x'$  consist of a single block

- $\text{MAC}_K(x)$  and  $\text{MAC}_K(x')$  are known
- $\text{MAC}_K(x \parallel (x' \oplus \text{MAC}_K(x))) = \text{MAC}_K(x')$ .

$x, x'$ , and  $Y$  fall on block boundaries

- $\text{MAC}_K(x)$ ,  $\text{MAC}_K(x \parallel Y)$ , and  $\text{MAC}_K(x')$  are known
- $\text{MAC}_K(x' \parallel Y') = \text{MAC}_K(x \parallel Y)$   
if  $Y' = Y \oplus \text{MAC}_K(x) \oplus \text{MAC}_K(x')$ .

21

## Knudsen's attack for $m < n$ (1)

probabilistic variant of the attack

- 2 known texts:  $\text{MAC}_K(x) = \alpha_m$  and  $\text{MAC}_K(x') = \alpha'_m$
- output of last encryption =  $\alpha_m \parallel \beta_{n-m}$  and  $\alpha'_m \parallel \beta'_{n-m}$ ,  
with  $\alpha, \alpha'$  known and  $\beta, \beta'$  unknown.
- chosen texts:  $r$  pairs of the following form:  
 $x \parallel \alpha_m \parallel \gamma_{n-m}$  and  $x' \parallel \alpha'_m \parallel \gamma'_{n-m}$ .
- input block for the last encryption ( $G$ ) is of the form  $0_m \parallel (\beta \oplus \gamma)_{n-m}$   
and  $0_m \parallel (\beta' \oplus \gamma')_{n-m}$

22

## Knudsen's attack for $m < n$ (2)

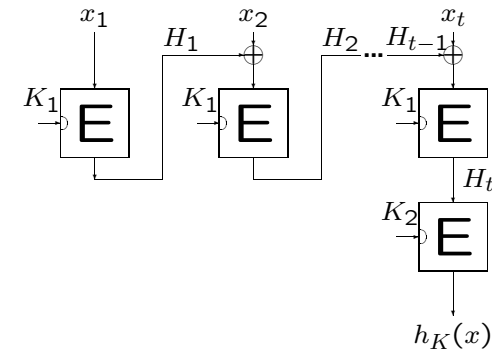
- if for a pair  $\beta \oplus \gamma = \beta' \oplus \gamma'$ , the two MACs for that pair will be equal; this is easy to detect.
- this happens with probability 0.63 if  $r = \sqrt{2^{n-m}}$
- based on the collision, we can compute  $\beta \oplus \beta' := \gamma \oplus \gamma'$
- this leads to a forgery as follows:  
 $\text{MAC}_K(x \parallel Y) = \text{MAC}_K(x' \parallel (Y \oplus (\Delta\alpha_m \parallel \Delta\beta_{n-m})))$   
with  $\Delta\alpha = \alpha \oplus \alpha'$  and  $\Delta\beta = \beta \oplus \beta'$ .

2 known texts,  $2 \cdot 2^{(n-m)/2} + 1$  chosen texts

23

## CBC-MAC: Algorithm 2 (1)

RIPEMD-MAC [RIPE 93] + EMAC (DMAC) [Petrank-Rackoff 98]



24

## CBC-MAC: Algorithm 2 (2)

security with DES:

- key search:  $[2^{56}, 2, 0, 0]$
- guess MAC:  $[0, 0, 0, \min(2^{56}, 2^m)]$
- birthday forgery attack (even if triple-DES):
  - $m = 64$ :  $[0, 2^{32}, 1, 0]$
  - $m = 32$ :  $[0, 2^{32}, 2^{33}, 0]$

much smaller than expected!

## CBC-MAC: Algorithm 3 (retail MAC) (2)

security with DES and  $m = 64$ :

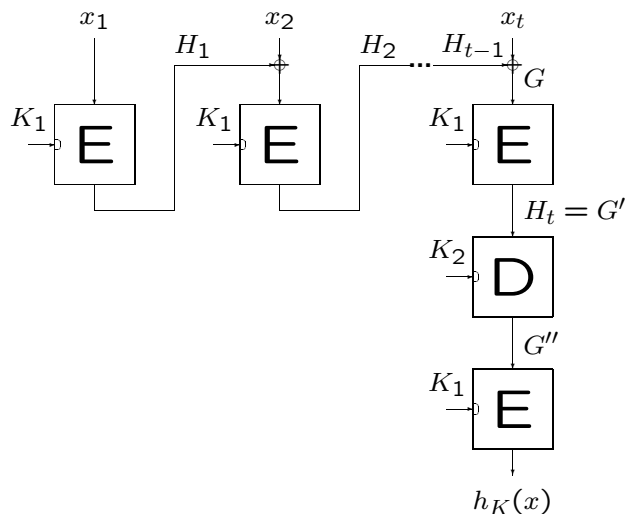
- key search:  $[2^{112}, 2, 0, 0]$
- guess MAC:  $[0, 0, 0, \min(2^{112}, 2^m)]$
- birthday forgery attack:  $[0, 2^{32}, 1, 0]$  (or  $[0, 2^{32}, 2^{33}, 0]$ )
- improved key recovery [Preneel-van Oorschot-Knudsen]
  - $[3 \cdot 2^{56}, 2^{32}, 0, 0]$
  - $[3 \cdot 2^{56}, 1, 0, 2^{56}]$

solution: triple-DES in first and last round?

25

27

## CBC-MAC: Algorithm 3 (retail MAC) (1)



26

## Key recovery attack on retail MAC (1)

- collect  $2^{32}$  known text-MAC pairs (e.g., of 2 blocks)
- with probability 0.39 there is a collision  $(x, x')$
- find the keys  $K_1$  for which the input to the last triple encryption ( $G$ ) is the same for  $x$  and  $x'$ 
  - with high probability, there will be only 1 solution
  - work factor:  $2 \cdot 2^{56}$  encryptions
- use  $K_1$  to compute  $G'$  and  $G''$
- find the key  $K_2$ :  $2^{56}$  encryptions

$2^{32.5}$  known texts and  $3 \cdot 2^{56}$  encryptions

28

## Key recovery attack on retail MAC (2)

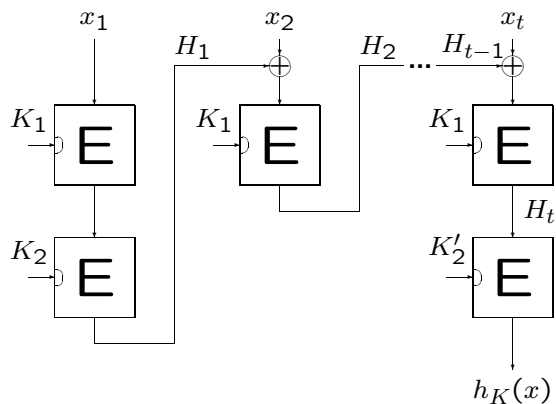
- one known text  $\alpha = \text{MAC}_K(x)$ , say of 2 blocks  $x_1, x_2$
- guess  $K_1$ 
  - compute the value of  $G$  (input of the last triple encryption)
  - choose  $x'_1 \neq x_1$  and  $x'_2$  such that the same value for  $G$  is obtained, or  $x'_2 = E_{K_1}(x'_1) \oplus G$
  - ask a MAC verification device:  $\text{MAC}_K(x'_1 \| x'_2) = \alpha$ ?
  - if yes, the guess for  $K_1$  was right (with high probability)
- use  $K_1$  to compute  $G'$  and  $G''$
- find the key  $K_2$ :  $2^{56}$  encryptions

1 known text,  $3 \cdot 2^{56}$  encryptions and  $2^{56}$  MAC verifications

29

## CBC-MAC: Algorithm 4 (Mac-DES) (1)

[Knudsen-Preneel 98]



30

## CBC-MAC: Algorithm 4 (Mac-DES) (2)

security with DES and  $m = 64$ :

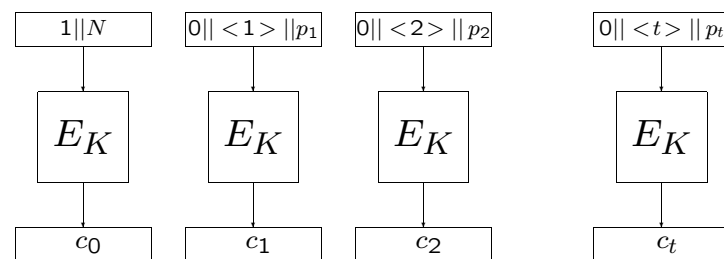
- key search:  $[2^{112}, 2, 0, 0]$
- guess MAC:  $[0, 0, 0, \min(2^{112}, 2^m)]$
- birthday forgery attack:  $[0, 2^{32}, 1, 0]$ , for  $m = 32$ :  $[0, 2^{32}, 2^{33}, 0]$
- improved key recovery [Coppersmith-Mitchell-Knudsen 00]:  $[2^{59}, 2^{33}, 3 \cdot 2^{49}, 0]$ , for  $m = 32$ :  $[2^{64}, 0, 2^{63}, 2^{57}]$

stronger against key recovery than MAC Algorithm 3

31

## XOR MAC

[Bellare-Guérin-Rogaway 95]



$\Sigma, N$  with  $\Sigma = c_0 \oplus c_1 \oplus c_2 \oplus \dots \oplus c_t$

- stronger security reduction than CBC-MAC (linear decrease of security with number of blocks)
- incremental and parallelizable
- twice as slow as CBC-MAC (for 32-bit length field with DES)
- MAC twice as long

32

## OCB mode and PMAC (1)

[Rogaway-Bellare-Black-Krovetz 01]

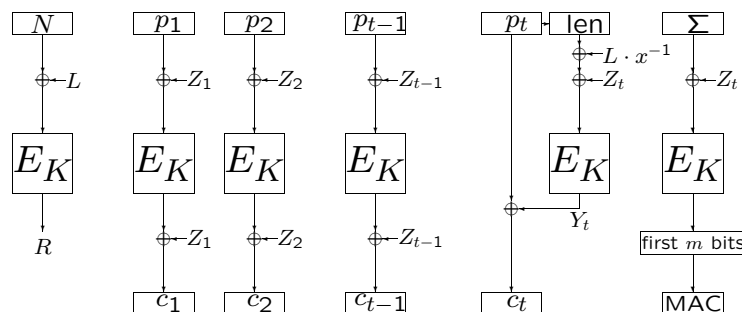
- authenticated encryption: indistinguishability under chosen plaintext attack and authenticity of ciphertexts
- randomized encryption using a nonce  $N$ : nonce can be used only once but does not need to be unpredictable
- one block cipher key (but 3 keys)
- any input length: no need for multiple of block length
- fully parallel, preprocessing possible
- minimal ciphertext expansion: MAC and nonce
- only two extra block cipher calls

PMAC: variant with only MAC

optimization: [Rogaway 04] (Asiacrypt 2004)

33

## OCB mode and PMAC (2)



$$\Sigma := p_1 \oplus \dots \oplus p_{t-1} \oplus c_t || 0^* \oplus Y_t$$

$$\text{MAC} := \text{first bits of } E_K(\Sigma \oplus Z_t)$$

security bound:  $O\left(\frac{q'^2}{2^n} + \frac{1}{2^m}\right)$  with  $q'$  total number of blocks

34

## OCB mode and PMAC (3)

input:  $p_1 \dots p_{t-1} p_t$  ( $p_t$  can be incomplete)

1.  $L := E_K(0^n)$  and  $R := E_K(N \oplus L)$
2. for  $i := 1$  to  $t$  do  $Z_i := L_i \oplus R$
3. for  $i := 1$  to  $t - 1$  do  $c_i := E_K(p_i \oplus Z_i) \oplus Z_i$
4.  $Y_t := E_K(\text{len}(p_t) \oplus L_{-1} \oplus Z_t)$
5.  $c_t := Y_t \oplus p_t$
6.  $\Sigma := p_1 \oplus \dots \oplus p_{t-1} \oplus c_t || 0^* \oplus Y_t$
7.  $\text{MAC} := \text{first bits of } E_K(\Sigma \oplus Z_t)$

$$L_1 := L, L_i := L_{i-1} \cdot x^{\text{ntz}(i)}, L_{-1} := L \cdot x^{-1}$$

multiplication in  $\text{GF}(2^{128})$  with  $f(x) = x^{128} + x^7 + x^2 + x + 1$

$\text{ntz}(i)$ : number of trailing zeroes in binary representation of  $i$

35

## 3GPP-MAC and XCBC

### 3GPP-MAC:

CBC-MAC where  $\text{MAC} = \text{leftmost } m \text{ bits of } E_{K'}(\Sigma)$

with  $\Sigma = c_0 \oplus c_1 \oplus c_2 \oplus \dots \oplus c_t$

Birthday attack still applies  $[0, 2^{32}, 2^{33}, 0]$  if  $m = 64$ .

### XCBC [Black-Rogaway 00]: 3-key construction

CBC with XOR of  $K_2$  to final input block ( $H_{t-1}$ ) if complete block and  $K_3$  if padding was needed

### XCBCX [Black-Rogaway 00]:

extra XOR of result with  $K_2/K_3$  may increase strength against exhaustive key search à la DES-X

### OMAC [Iwata-Kurosawa 03] renamed as CMAC by NIST:

derive  $K_2/K_3$  from  $K$  (à la PMAC)

Birthday attack still applies

!!  $L := E_K(0^n)$  is used by banks as key confirmation value

36

## Impact of AES

- Exhaustive key search no longer relevant, hence MAC Algorithms 3 and 4 are not needed
- Main 'concern' for CBC-MAC variants (only exception is XOR MAC) is birthday forgery attack with  $2^{64}$  known texts
- One but last encryption in EMAC can be dropped [Handschuh-Preneel 06]: LMAC

37

## Countermeasures against birthday attack

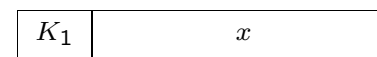
best attacks known for AES are indicated (= conjected security level)

- increase the size of the internal memory  $n$  (Algorithm 5)  $[0, 2^{84}, 2, 0]$  (double computation)
- randomize the first block of the data (R-MAC) [Semanko01]  $[0, 2^{84}, 2, 0]$  (double MAC length)
- truncation: improves security against key recovery but adds very limited protection against birthday forgery
- prepending the length: does not work
- prepending a serial number prior to padding (with length before or after) does not work if serial numbers are in clear [Brincat-Mitchell 01]:  $[0, 1, 2^{65}, 0]$
- derive key in output transformation from a serial number

38

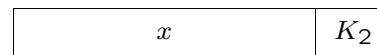
## MAC: based on a hash function (MDC) (1)

**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:**  $h(x||K_2)$

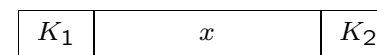


off-line attacks on  $h$

39

## MAC: based on a hash function (MDC) (2)

**envelope:**  $h(K_1||x||K_2)$



- provable security based on pseudo-randomness of compression function  $f$
- forgery (for MD5):  $2^{64}$  known texts
- key recovery:  $2^{66}$  known texts and  $2^{20}$  chosen texts

**MDx-MAC** (SHA-1-MAC, RIPEMD-160-MAC)

- stronger pseudo-random properties: key in each iteration
- $K_1$  and  $K_2$  in separate blocks: precludes key recovery

40

## MAC: based on a hash function (MDC) (3)

**NMAC:**  $h_{K_1 \| K_2}(X) = h_{K_1}(h_{K_2}(x))$

**HMAC:**  $h_K(X) = h(K \oplus \text{opad} \| h(K \oplus \text{ipad} \| x))$

security proof:

- $h$  collision resistant for secret  $IV$  (\*)
- $f$  secure MAC if  $H_{i-1}$  is secret key
- need also  $f$  pseudo-random function if  $H_{i-1}$  is secret key for efficient implementation

problem: security proof no longer valid after attack by Wang et al. (\*)  
fixed by [Bellare 06]: it is sufficient to assume that  $f$  pseudo-random function if  $H_{i-1}$  is secret key  
(then HMAC is PRF and thus a secure MAC)

recent improvements: ENMAC [Patel 03] and BNMAC [Yasuda 07]

41

## Attacks on NMAC and HMAC

[Wang-Ohta-Kunihiro 08] outer key recovery + forgery

NMAC/HMAC MD4:  $2^{72}$  chosen texts and  $2^{77}$  steps

NMAC MD5:  $2^{75}$  chosen texts and  $2^{76}$  steps (rel. key)

[Rechberger-Rijmen 07]

NMAC MD5: key recovery  $2^{45}$  chosen texts and  $2^{100}$  steps (rel. key)

NMAC SHA-1: key recovery 34/80 rounds with  $2^{155}$  chosen texts and  $2^{157}$  steps (rel. key)

HMAC SHA-1: forgery for 37/80 rounds with  $2^{65}$  data

HMAC SHA-1: inner key recovery for 53/80 rounds with  $2^{100}$  data and time

Further improvements to be expected.

42

## Property preservation?

Do properties of a compression function with a secret key imply properties of an iterated MAC algorithm? [An-Bellare 99]

CBC-MAC (EMAC): secure MAC if block cipher is PRF or PRP but not secure if block cipher is a MAC

Merkle-Damgard: preserves MAC property and PRF property

43

## Dedicated MACs

Message Authenticator Algorithm (ISO 8731-2: 1987):

- [Davies and D. Clayden 83]
- key size  $k = 64$ , internal memory  $m = 64$ , result  $n = 32$
- typical speed: 2 Mbyte/s (50% of MD4)
- special mode for messages  $\geq 1024$  bytes

best known attacks:

- forgery:  $2^{24}$  messages of 1 Kbyte
- key recovery:  $2^{32}$  chosen texts,  $2^{44} \dots 2^{51}$  multiplications
- several large classes of weak keys

44

## Other dedicated MAC algorithms

- KHF (cryptanalyzed by Wagner)

fixed length input

- COMP-128 (GSM example)
- CAVE (CDMA)
- Secure-ID
- Two-Track-MAC

45

## Information-theoretic authentication

authentication codes (AC)/universal hash functions  
[1970s (Zobrist/Simmons/Carter-Wegman)]

- advantages
  - provably secure: only combinatorial
  - extremely fast (10-15 times faster than AES/HMAC)
  - parallelizable and incremental
- disadvantages
  - use key only once
  - sometimes very large keys: low key agility
  - security level in bits against forgery is at most half the key size

46

## Information-theoretic authentication

[Black-Halevi-Krawczyk-Krovetz-Rogaway 99]

[...] “since the combinatorial property of the universal hash-function family is mathematically proven (making no cryptographic hardness assumptions), it needs no “over-design” or “safety margin” the way a cryptographic primitive would. Quite the opposite: the [UMAC] hash-function family might as well be the fastest, simplest thing that one can prove universal.”

47

## Example: polynomial authentication code

(Change of notation:  $k$  is key rather than key size in bits)

- key  $k'$ ,  $k \in GF(2^n)$
- split  $x$  into  $x_1, x_2, \dots, x_t$ , with  $x_i \in GF(2^n)$
- note  $\ell = t \cdot n$

$$g(x) = k' + \sum_{i=1}^t x_i \cdot k^i$$

Pr(success of forgery after seeing 1 text/MAC pair) =  $(\ell/n)/2^n = t/2^n$

In practice: value  $k$  can be reused

48

## Step 1: Compress

family of functions  $g_k : A \rightarrow B$  with  $a = |A|$  and  $b = |B|$ .

$B, \star$  an Abelian group

Let  $\epsilon$  be any positive real number.

$g_k$  is an  $\epsilon$ -almost universal class (or  $\epsilon$ -AU class)  $\mathcal{G}$  of hash functions if  $\forall x, x' \neq x \in A$

$$\Pr_k \{g_k(x) = g_k(x')\} \leq \epsilon.$$

$g_k$  is an  $\epsilon$ -almost  $\star$  universal class (or  $\epsilon$ -A $\star$ U class)  $\mathcal{G}$  of hash functions if  $\forall x, x' \neq x \in A$  and  $\forall \Delta \in B$

$$\Pr_k \{g_k(x) = g_k(x') \star \Delta\} \leq \epsilon.$$

49

## Step 1: Compress (2)

functions that are  $\epsilon$ -AU

- $g_k(x) = \sum_{i=0}^t x_i \cdot k^i$  with  $k, x_i \in \text{GF}(2^r)$  or  $\text{GF}(p)$

functions that are  $\epsilon$ -A $\star$ U

- $g_k(x) = \sum_{i=1}^t x_i \cdot k^i$  with  $k, x_i \in \text{GF}(2^r)$  or  $\text{GF}(p)$
- MMH:  $g_k(x) = (\sum_{i=1}^t x_i \cdot k_i) \bmod p$   
 $x_i, k_i \in \mathbf{Z}_{2^{32}}$  and  $p = 2^{32} + 15$  (inner sum mod  $2^{64}$ ) [Halevi-Krawczyk97]
- NMH:  $g_k(x) = (\sum_{i=1}^{t/2} (x_{2i-1} + k_{2i-1}) \cdot (x_{2i} + k_{2i})) \bmod p$   
 $x_i, k_i \in \mathbf{Z}_{2^{32}}$  and  $p = 2^{32} + 15$  [Wegman-Carter 81 and Halevi-Krawczyk 97]
- NH:  $g_k(x) = (\sum_{i=1}^{t/2} ((x_{2i-1} + k_{2i-1}) \bmod 2^w) \cdot ((x_{2i} + k_{2i}) \bmod 2^w)) \bmod 2^{2w}$   
 $x_i, k_i \in \mathbf{Z}_{2^w}$  [BHKKR 99]

50

## Step 2: Replace addition $k' +$

pseudorandom function family  $f_{k'}$   $\rightsquigarrow$  computational security

Option 1:  $\text{MAC}_{k||k'}(x) = f_{k'}(g_k(x))$  with  $g \in \text{AU}$

Option 2:  $\text{MAC}_{k||k'}(x) = f_{k'}(n) \star g_k(x)$  with  $g \in \text{A}\star\text{U}$   
need nonce but better security

Option 3:  $\text{MAC}_{k||k'}(x) = f_{k'}(n||g_k(x))$  with  $g \in \text{AU}$   
need nonce and larger input of  $f$

51

## Observations on universal hash functions for MAC

- reuse of  $k$  is common in practice, in particular if  $k$  is large
- nonce  $n$  is supposed to be unique; what if it isn't?
  - nonce is always used twice (generation/verification)
  - nonce reuse for verification (e.g., if random numbers)
  - nonce reuse for MAC generation more problematic
- weak keys:  $\forall x, x', x' \neq x \Pr_k \{g_k(x) = g_k(x') \star \Delta\} \leq \epsilon$  does not imply that  $\forall k \Pr_{x, x'} \{g_k(x) = g_k(x') \star \Delta\} \approx 1/|B|$ . For some keys  $\bar{k}$

$$\Pr_{x, x'} \{g_{\bar{k}}(x) = g_{\bar{k}}(x') \star \Delta\} \gg 1/|B|$$

- simple combinatorial scheme: partial knowledge of  $k$  may be devastating (e.g., verification by leaking first 2 key bytes)
- messages of special form reduce hash value to subspace

52

## Related Work

[Coppersmith 96] Finding a small root of a bivariate integer equation; factoring with high bits known.

[Bellare-Goldwasser 97] Verifiable partial key escrow

[McGrew-Fluhrer 05] Multiple forgery attacks

[Preneel-vanOorschot-Knudsen 96] Key recovery attacks on ANSI retail MAC

[Blackburn-Paterson 04] Cryptanalysis of a MAC due to Cary and Venkatesan

[Black-Cochran 06] Focus on reforgeability

UMAC RFC 4418 contains warnings:

- be careful if too many wrong MAC values
- “once an attempted forgery is successful, it is possible, in principle, that subsequent messages under this key may be easily forged. This is important to understand in gauging the severity of a successful forgery, **even though no such attack on UMAC is known to date.**”

Even two appendices full of warnings for GCM NIST SP 800-38D

53

## Polynomial hash

$$g_k(x) = \sum_{i=1}^t x_i \cdot k^i \text{ with } k, x_i \in \text{GF}(2^n)$$

GCM NIST SP-800 38-D:

- Option 2:  $\text{MAC}_{k||k'}(x) = \text{trunc}_\tau(\text{AES}_{k'}(n) \oplus g_k(x))$
- $k = \text{AES}_{k'}(000\dots 00)$

trivial weak key:  $k = 0$  (extremely unlikely)

trivial to verify a guess for  $k$ , even if we do not know  $k'$  (ok for GCM)

if order of  $k$  divides  $l < t$ : swap of two blocks leaves  $g_k(x)$  unchanged

[Joux 06] attack on GCM (nonce reuse by sender):

$$\text{compute } \text{MAC}_{k||k'}(x) \oplus \text{MAC}_{k||k'}(x') = g_k(x) \oplus g_k(x') = g_k^*(x, x')$$

$k$  is one of the  $t$  roots of the polynomial  $g_k^*$

54

## Polynomial hash (2)

Joux: “replacing the counter encryption for MACs by the classical encryption with the block cipher usually used with Wegman-Carter MACs seems a safe option.”

Variant on Joux attack that works even for Option 1 or 3 (more expensive, same cost as forgery, but no nonce reuse):

- obtain 1 MAC value for a text  $x$  of your choice
- choose  $x'$  such that the polynomial with coefficients from  $x - x'$  has  $t$  distinct roots (each time)
- perform a MAC verification query for  $x'$
- after  $2^n/t$  trials you know that  $k$  is one of  $t$  values
- perform another  $t$  MAC verification queries to find out which one of the  $t$

easy to take into account any information you may have on  $k$ .

55

## Polynomial hash (3)

special message attack [Ferguson 05]

- $x_i = 0$  except if  $i = 2^j$  for some  $j$ :  $g_k(x) = \sum_{j=0}^l x_{2^j} \cdot k^{2^j}$
- squaring is a linear operation in  $\text{GF}(2^n)$   
hence we can write the bits of the hash as follows:

$$g_k(x)[.] = \sum_{i^*, j^*, u^*} x_{i^*}[j^*] \cdot k[u^*]$$

- choose  $x, x'$  such that  $\text{trunc}_s(g_k(x)) = \text{trunc}_s(g_k(x'))$  ( $1 \leq s < \tau$ ), independent of the value of  $k$
- submit  $x'$  for verification (same nonce); success prob.  $1/2^{\tau-s}$
- collect  $r \cdot 2^{\tau-s}$  messages (same nonce!) resulting in  $r$  forgeries; each forgery yields  $\tau - s$  linear equations in the key bits.

response by NIST: be careful with wrong MAC values; no error messages

56

## MMH [Halevi-Krawczyk 97]

$$g_k(x) = \left( \sum_{i=1}^t x_i \cdot k_i \right) \bmod p$$

$x_i, k_i \in \mathbf{Z}_{2^{32}}$  and  $p = 2^{32} + 15$  (inner sum mod  $2^{64}$ )

attacks (consider  $t = 2$ ):

- $k_1 = k_2 = 0$ : all messages map to 0.
- $k_i = 0$ : result does not depend on  $x_i$
- $k_1 = k_2$ : can swap  $x_1$  and  $x_2$ : no effect
- $k_1 = \alpha \cdot k_2 \bmod p$ ,  $g_k(x)$  unchanged if one replaces  $(x_1, x_2)$  by  $(x_2/\alpha, x_1 \cdot \alpha)$   
hence a guess for  $\alpha$  can be verified with one verification query
- easy to exploit partial key information  $x_1 k_1 + x_2 k_2 = x'_1 k_1 + x'_2 k_2$

57

## NH-UMAC [BHKKR 99]-VMAC-WMAC (2)

UMAC:  $w = 32$  marginal; ok if output length of 64, 96 or 128 bits.  
(earlier version had  $w = 16$ )

VMAC:  $w = 64$  (probably ok)

WMAC: variant with  $w = 16, 32$  and  $64$

59

## NH-UMAC [BHKKR 99]-VMAC-WMAC

$$g_k(x) = \left( \sum_{i=1}^{t/2} ((x_{2i-1} + k_{2i-1}) \bmod 2^w) \cdot ((x_{2i} + k_{2i}) \bmod 2^w) \right) \bmod 2^{2w},$$

$$x_i, k_i \in \mathbf{Z}_{2^w}$$

- if  $k_{2i-1} = k_{2i}$ : can swap  $x_{2i-1}$  and  $x_{2i}$  without changing result
- if  $((x_{2i-1} + k_{2i-1}) \bmod 2^w) = 0$ ,  $g_k(x)$  is independent of  $x_{2i}$ .
- assume that  $k_{2i} = k_{2i-1} + \Delta \bmod 2^w$  for  $\Delta \in \mathbf{Z}_{2^w}$ ,  
then  $g_k(x)$  unchanged if one replaces  $(x_{2i-1}, x_{2i})$  by  $(x_{2i} + \Delta \bmod 2^w, x_{2i-1} - \Delta \bmod 2^w)$
- recover  $t$  key words with  $t \cdot 2^w$  verification queries
- with an oracle for  $s$  bits of one key word, expected complexity to find one word reduces to  $2^{w-s}$  verification queries.

58

## Summary of Attacks on MACs based on Universal Hash Functions

	weak keys	divide and conquer attacks	partial key information
Polynomial hash $\text{GF}(2^n)$	1	$k$ only	yes
Polynomial hash $\text{GF}(p)$	1	$k$ only	?
MMH	many	$w$ -bit subkey $k_i$	yes
NMH/NH/WH	many	$w$ -bit subkey $k_i$	yes

60

## Conclusions

While universal hash functions have very attractive performance and provable security, they can be very brittle in practice

- avoid reusing keys (Snow3G is good example)
- sender/verifier needs to guarantee/check uniqueness of nonces
- vulnerable to oracle that reveals part of the key (and thus to side-channel attacks)

Some schemes more secure than others. . . (e.g., polynomial over  $GF(p)$ )

EMAC based on AES is slower but more “robust”

- internal collisions ( $2^{64}$  texts) lead to forgeries, but not to key recovery
- no known way to use an oracle that gives access to 32 or 64 key bits
- faster key setup

HMAC: are the compression functions of hash functions suitable pseudo-random functions?