MAC Algorithms

Prof. Bart Preneel
Katholieke Universiteit Leuven
bartDOTpreneel(AT)esatDOTkuleuvenDOTbe
http://www.esat.kuleuven.be/~preneel

August 2006

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 = hash function with secret key

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:
forgey
   - 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,
- and can observe the MAC for $q'$ texts (not of his choice),

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

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

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: 75...90 bits

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

notation
work - known texts - chosen texts - on-line verifications

‘Attack’: guess MAC

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

but:
- not verifiable
- requires on-line verification

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

Attack complexity: $[0, 0, 0, \min(2^m, 2^k)]$
**Attack: birthday forgery attack (1)**

![Diagram of birthday forgery attack (1)]

- $H_0 = IV \ x_1$
- $H_1 = f(x_1) \ f$
- $H_2 = f(x_2) \ f$
- $H_3 = \text{output transformation}$

- $H_0 = IV \ x_1$
- $H_1 = f(x_1) \ f$
- $H_2 = f(x_2) \ f$
- $H_3 = \text{output transformation}$

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

**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.

**Attack: birthday forgery attack (2b)**

![Diagram of birthday forgery attack (2b)]

- $H_0 = IV \ x_1$
- $H_1 = f(x_1) \ f$
- $H_2 = f(x_2) \ f$
- $H_3 = \text{output transformation}$

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

**Birthday forgery attack (3)**

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

[Preneel-van Oorschot] 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: $[0, 2^{n/2}, \min(2^{n/2}, 2^{n-m}), 0]$
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.

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

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

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

birthday paradox: for $S = 365$ and $r = 23$, $p = 1/2$
intuition: number of pairs of elements is $r(r-1)/2$.

Birthday forgery attack (4)

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.

$$u = \sqrt{2/((s + 1) \cdot 2^n/2}$$
$$v = 0 \text{ if the output transformation } g \text{ is a permutation or }$$
$$v = 2 \left(2^{n-m} (1 - \frac{1}{e}) + \left\lfloor \frac{n - 1 - \log_2(s + 1)}{m - 1} \right\rfloor \right),$$

Birthday forgery attack (5)

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

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
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 (1)

\[
\begin{align*}
E_{K_1}(x_1) \quad &\quad H_1 \quad x_2 \\
E_{K_1}(H_1) \quad &\quad H_2 \\
E_{K_1}(H_{t-1}) \quad &\quad G \\
h_{K_1}(x) \quad &\quad x_t
\end{align*}
\]

- proof of security for fixed length inputs by [Bellare-Kilian-Rogaway'94]
- \( m = 32 \ldots 64 \) bits

CBC-MAC: Algorithm 1 (2)

security with DES:

- key search: \([2^{56}, 2, 0, 0]\)
- key recovery using lc: \([2^{43}, 2^{43}, 0, 0]\)
- 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!

Why a special operation for the last block?

\( x \) consists of a single block

- \( \text{MAC}_{K}(x) \) is known
- then \( \text{MAC}_{K}(x || (x \oplus \text{MAC}_{K}(x))) = \text{MAC}_{K}(x) \).

\( x, x' \) consist of a single block

- \( \text{MAC}_{K}(x) \) and \( \text{MAC}_{K}(x') \) are known
- \( \text{MAC}_{K}(x || (x' \oplus \text{MAC}_{K}(x))) = \text{MAC}_{K}(x') \).

\( x, x', \) and \( Y \) fall on block boundaries

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

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:
  $\text{MAC}_K(x||Y) = \text{MAC}_K(x'||Y \oplus (\Delta \alpha_m || \Delta \beta_{n-m}))$
  with $\Delta \alpha = \alpha \oplus \alpha'$ and $\Delta \beta = \beta \oplus \beta'$.

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

CBC-MAC: Algorithm 2 (1)

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

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) (1)

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

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

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

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

[Knudsen-Preneel’98]

\[ E^{K_1}(x_1) \rightarrow H_1 \rightarrow E^{K_1}(x_2) \rightarrow H_2 \rightarrow \cdots \rightarrow E^{K_1}(x_t) \rightarrow h_K(x) \]

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-Knudsen00]:
  \([2^{59}, 2^{33}, 3 \cdot 2^{49}, 0]\), for \( m = 32 \): \([2^{64}, 0, 2^{63}, 2^{57}]\)

stronger against key recovery than MAC Algorithm 3

XOR MAC

[Bellare-Guérin-Rogaway’95]

\[ \Sigma, N \text{ with } \Sigma = c_0 \oplus c_1 \oplus c_2 \oplus \cdots \oplus c_t \]

- 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)
- MAC twice as long

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: [Rogaway04] (Asiacrypt 2004)
OCB mode and PMAC (2)

\[
\begin{align*}
\Sigma := p_1 \oplus \cdots \oplus p_{t-1} \oplus c_t \| 0^* \oplus Y_t \\
\text{MAC} := \text{first bits of } E_K(\Sigma \oplus Z_t)
\end{align*}
\]

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

OCB mode and PMAC (3)

input: \( p_1 \ldots p_{t-1} p_t \) (\( p_t \) 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. \( \Sigma := p_1 \oplus \cdots \oplus p_{t-1} \oplus c_t \| 0^* \oplus Y_t \)
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 \( GF(2^{128}) \) with \( f(x) = x^{128} + x^7 + x^2 + x + 1 \)

ntz(\( i \)): number of trailing zeroes in binary representation of \( i \)

3GPP-MAC and XCBC

3GPP-MAC:

CBC-MAC where MAC = leftmost \( m \) bits of \( E_{K'}(\Sigma) \) with \( \Sigma := c_0 \oplus c_1 \oplus c_2 \oplus \cdots \oplus c_t \)

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

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
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-Mitchell01]: $[0, 1, 2^{65}, 0]$
- Derive key in output transformation from a serial number

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

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

Prepend length to avoid that one can compute $h(K_1||x||y)$ from $h(K_1||x)$ without knowing $K_1$

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

Off-line attacks on $h$

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

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

- 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

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

**HMAC:** $h_K(X) = h(K_2||h(K_1||x))$

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 [Bellare06]: 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)
Dedicated MACs


- D. Davies and D. Clayden [1983]
- 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, $2^{44} \ldots 2^{51}$ multiplications
- several large classes of weak keys

Other dedicated MAC algorithms

- KHF (cryptanalyzed by Wagner)

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

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, . . .

Information-theoretic authentication (2)

attacks:
- impersonation: $P_i$
- substitution: $P_s$
- deception: $P_d = \max(P_i, P_s)$

simplified model:
- authentication without secrecy
- authenticator ($\simeq$ MAC) added to text
- use key only once
Bounds [Simmons 84]

**Theorem [Authentication Channel Capacity]**

\[ P_i \geq 2^{-I(M;K)} \]
\[ P_s \geq 2^{-H(K|M)} \] (\( \geq 2 \) distinct inputs)

Note: \( H(K) = H(K|M) + I(K;M) \)

**Theorem [square root bound]**

\[ P_d \geq \frac{1}{\sqrt{2^{H(K)}}} \]

---

Example: polynomial authentication code

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

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

for a random AC: \( \Pr(\text{success}) = P_i = 1/2^r \)

forgery after 1 text/AC pair: \( \Pr(\text{success}) = P_s = (\ell/r)/2^r \)

\( P_i = 1/2^{80} \) and \( P_s = 1/2^{64} \): 160-bit key
message shorter than 640 Kbyte

---

Bounds

**bad news**

- The square root bound can only be tight if the inputs are very small (compared to the key)

**good news**

- Wegman & Carter 1977
  use authentication codes with \( P_s > P_i \)
  this leads to much shorter keys
- Johansson et al. 1993 \( P_s > P_i \)
  \( \Rightarrow \) the number of plaintexts grows exponentially with the number of keys.

---

How to use Authentication Codes in practice?

1. Distill essential property of compression phase
2. Replace key addition by pseudo-random function

Credits: the subsequent slides (50-52 and 55) are heavily inspired by Dan Bernstein’s presentation on poly1305-AES: http://cr.yp.to/talks.html#2005.02.15
Step 1: Distill essential property

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

Let $\epsilon$ be any positive real number.

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

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

Step 1: Distill essential property (2)

functions that are $\epsilon$-AU

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

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

- $g_k(x) = \Sigma_{i=1}^t x_i \cdot k_i$ with $k, x_i \in \text{GF}(2^r)$ ($\times = \oplus$)
- $g_k(x) = \Sigma_{i=1}^t x_i \cdot k_i$ with $k, x_i \in \text{GF}(p)$ ($\times = +$)
- same but $x_i \in \text{GF}(2^{32})$ and $p = 2^{127} - 1$ [Bernstein99]
- same but $x_i \in \text{GF}(2^{64})$ and $p = 2^{64} - 59$ but need to re-encode $x_i$ [Krovetz-Rogaway00] - PolyR
- same but $x_i \in \text{GF}(2^{128})$ and $p = 2^{130} - 5$ [Bernstein02] - Poly1305
- same but $x_i \in \text{GF}(2^{96})$ and $p = 2^{127} - 1$ [Kohno-Viega-Whiting03] - CWC
- $g_k(x) = \Sigma_{i=1}^t x_i \cdot k_i \mod p$ with $k_i, x_i \in \text{GF}(p)$ ($\times = +$)

Step 2: Replace addition $k' +$

Choose pseudorandom function family $f_{k'} = f_{k'}(g_k(x))$ hard to distinguish from random function

- example: AES$_{k'}(x)$, MD5($k||x$) (for fixed input length)

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

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

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

Optimization:

use $f$ to derive $k$ and $k'$ from a common master key

Example: Poly1305-AES [Bernstein02-05]

Option 2: MAC$_{k||k'}(x) = f_{k'}(n) + g_k(x)$

- $f(n) = \text{AES}_{k'}(n)$
- $g_k(x) = \Sigma_{i=1}^t x_i \cdot k_i \mod p$
- $k, x_i \in \text{GF}(2^{128})$ and $p = 2^{130} - 5$

If an attacker adaptively chooses $C \leq 2^{64}$ messages (max $\ell$ bytes each) and obtains their MAC values, attempts $F$ forgeries:

$$1 - \Pr\{\text{all forgeries rejected}\} \leq 14F[\ell/16]/2^{106}.$$  

(here we assume AES is perfect)

Example: $\ell = 1536$, $C = F = 2^{64}$: $1 - \Pr \leq 3 \cdot 10^{-10}$

Note: with option 1 we have $1 - \Pr \leq 8C(C + F)[\ell/16]/2^{106}$

With $C = F = 2^{32}$: $3.4 \cdot 10^{-10}$
Performance

very high speeds
- ALRED and Alpha-MAC: 2.5 faster than AES
- poly1305-AES: 4-5 cycles/byte
- UMAC: up to 1-2 cycles/byte for long messages
- VMAC: up to 0.5 cycles/byte on Athlon 64 for long messages

compare to MAC Algorithms (on Pentium):
- 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 15 cycles/byte for AES

Performance (2)

- parallelizable
- if long keys are needed (inner product): low key agility, i.e., inefficient for short messages (poly1305-AES is an exception)

conclusion: attractive, but be aware that unconditional security is lost

HMAC and CBC-MAC: based on universal hash functions?
can also be studied in this model

EMAC: Option 1 with \( g_k(x) = \text{AES}_k(AES_k(x_1) \oplus x_2) \oplus x_3 \) and \( f(x) = \text{AES}_{k'}(x) \)

NMAC-MD5: Option 1 with \( g_k(x) = \text{MD5}(k||x) \) and \( f_k'(x) = \text{MD5}(k'\|x) \)

HMAC-MD5: same as above with key optimization

HMAC-MD4-DJB: Option 3 with \( g_k(x) = \text{MD5}(k||x) \) and \( f_k'(n||x) = \text{MD5}(k'\|n\||x) \)

Conclusions

- Still an active research area — need for security bounds and extreme performance
- With AES/Rijndael, MACs based on block ciphers may increase importance compared to HMAC
- Few dedicated proposals
- Some applications may start using authentication codes (= universal hash functions)