Message authentication codes

Martin Stanek

Department of Computer Science
Comenius University
stanek@dcs.fmph.uniba.sk

Cryptology 1 (2021/22)

Content

Introduction

- message authentication code (MAC)
  - data integrity and authenticity
  - faster than digital signatures, suitable for per-packet use
  - shared-key (symmetric) construction

$$A \xrightarrow{m,\ \mathrm{Mac}_k(m)} B$$

- MAC ~ “hash function with a key”
  - key is necessary to compute the value of MAC
  - verification by recomputation and comparison
  - no non-repudiation property (!)
- requirements: efficiency & security
- remark: using MAC alone does not prevent replay attacks (!)
  - sequential message number, timestamp, etc.

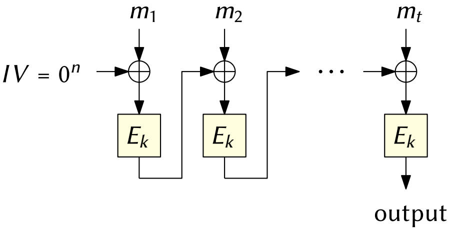
Security of MAC (informally)

- formal definition of MAC uses three algorithms: Gen, Mac, Vrf
- PPT attacker A, with oracle access to Mac_k (for random k)
- existentially unforgeable under an adaptive chosen message attack:
  - the probability that any attacker A produces a pair (m, h) such that $\mathrm{Mac}_k(m) = h$ (and A did not query the oracle with m) is negligible

- MAC uses a key, therefore the birthday attack is not applicable
  - output of MAC can be shorter than output of a hash function
  - for example IPSec: HMAC-SHA1-96 (truncated HMAC)

CBC-MAC

- MAC constructed from a block cipher
- initial attempt:



- secure for fixed length inputs (assuming PRP property of E)

CBC-MAC (2)

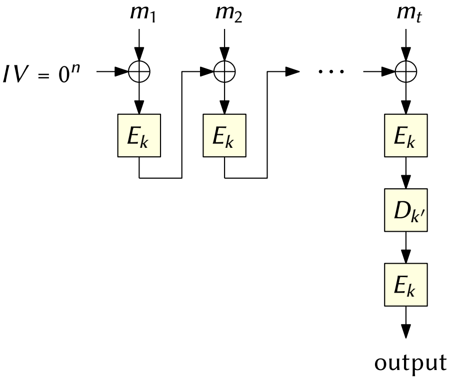
- insecure for variable length inputs:
  1. A queries Mac_k oracle with 1-block messages m and m'
  2. A obtains $h = \text{Mac}_k(m) = E_k(m)$ and $h' = \text{Mac}_k(m') = E_k(m')$
  3. A queries the oracle with two-block message $m \,||\, x$ and obtains $h^* = E_k(E_k(m) \oplus x)$
  4. Let us compute MAC for two-block message $m' \,||\, h \oplus h' \oplus x$:

$$E_k(E_k(m') \oplus E_k(m) \oplus E_k(m') \oplus x) = E_k(E_k(m) \oplus x) = h^*$$

  i.e. A knows the valid MAC for this message without asking the oracle

CBC-MAC (3)

- how to fix CBC-MAC:



- two different keys k, k'
  - derive k' from k, e.g. $k' = \overline{k}$, $k' = E_k(k)$ …
  - or derive two keys from a single key: $k_1 = E_k(1)$, $k_2 = E_k(2)$

CMAC

- authentication mode of block ciphers, approved by NIST (SP 800-38B)
- simplified presentation
  - assuming that the input length is divisible by block length (padding and slightly different subkey used otherwise)
  - $m = m_1, \dots, m_t$
  - $l = E_k(0)$; $k_1 = \mathrm{MSB}(l)\ ?\ (l \ll 1) \oplus R : l \ll 1$
  - R is a constant depending on block length, e.g. $R_{128} = 0^{120}10000111$
  - the last block is transformed: $m'_t = m_t \oplus k_1$
- CBC processing (starting with $C_0 = 0$):
  1. $C_i = E_k(C_{i-1} \oplus m_i)$, for $i = 1, \dots, t-1$
  2. $C_t = E_k(C_{t-1} \oplus m'_t)$ final round
  3. output: C_t (can be truncated)

MAC construction based on hash functions

- natural but (often) insecure approaches (let H be a hash function):
  1. $\mathrm{Mac}_k(m) = H(k \,||\, m)$
     using some iterated H (e.g. MD-based) allows the attacker to compute MAC for an extended message
  2. $\mathrm{Mac}_k(m) = H(m \,||\, k)$
     using some iterated H (e.g. MD-based) means that finding collision implies colliding MAC (security of MAC reduces/weakens to collision resistance)

  easy to propose other ideas, e.g. $H(k \,||\, m \,||\, k)$ …security proof?
  (btw. some weaknesses were identified even in this construction)

HMAC

- MAC construction based on hash functions
- the most popular / used MAC today (SSL/TLS, SSH, IPSec, ...)
- provable security (if underlying compression function is PRF)
- construction:

$$\mathrm{HMAC}_k(m) = H(k \oplus \mathrm{opad} \,||\, H(k \oplus \mathrm{ipad} \,||\, m))$$

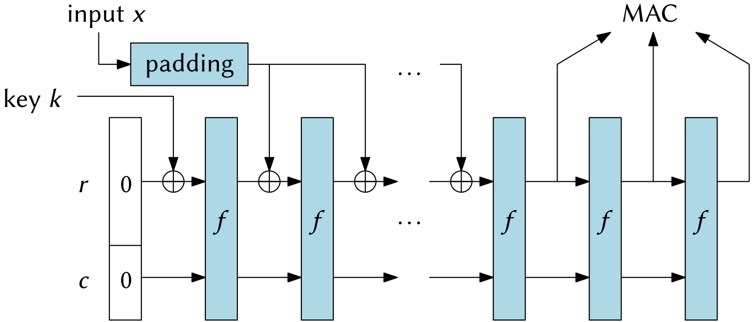
  - opad/ipad – block-length outer/inner padding (0x5c5c.../ 0x3636...), i.e. 64 bytes for MD5, SHA-1 or SHA-256
- almost as fast as underlying hash function (just 3 additional blocks)
- truncation of output possible (e.g. used in IPSec)

Combined construction

- another approach: $\mathrm{Mac}_k(m) = E_k(H(m))$
- provable security, if E is PRP and H is collision resistant
- problems:
  - stronger assumptions than HMAC (thus no reason to use it)
  - block ciphers usually with short block length n and because of collisions, the bit-security is just $n/2$
  - for example AES with 128-bit block (and truncated hash) leads to 64-bit security

MAC from sponge construction

- KMAC – Keccak MAC (NIST SP 800-185, 2016)
- basic idea of MAC from sponge hash function:



Secure channel (1)

- confidentiality & integrity/authenticity
- usually both needed for a secure communication
- authenticated encryption – specific modes of a block cipher
- encryption (standard confidentiality modes) & MAC
  - How to combine them properly?
- options (we use two independent keys k_1, k_2):
  1. EtM (Encrypt then MAC, e.g. IPSec): $c = E_{k_1}(m), \langle c, \mathrm{Mac}_{k_2}(c) \rangle$
  2. MtE (MAC then Encrypt, e.g. SSL/TLS): $E_{k_1}(m \,||\, \mathrm{Mac}_{k_2}(m))$
  3. E&M (Encrypt and MAC, e.g. SSH): $\langle E_{k_1}(m), \mathrm{Mac}_{k_2}(m) \rangle$
- recent situation:
  - SSL: authenticated encryption (GCM); only AEAD ciphers for TLS 1.3
  - SSH: authenticated encryption (GCM): e.g. aes128-gcm@openssh.com

Secure channel (2)

- theory: the correct approach is EtM
- the problem with MtE:
  - use any secure MAC
  - cipher: encoding n bit plaintext to $2n$ bits (each bit is replaced by two bits: $0 \mapsto 00$; $1 \mapsto 01$ or 10 randomly) and then a secure synchronous stream cipher is applied (even one-time pad can be used)
  - a pair 11 observed while decoding means "invalid ciphertext"
  - the cipher is IND-CPA if underlying cipher is (for symmetric ciphers)
  - flipping first two bits in the ciphertext is a correct ciphertext $\Leftrightarrow$ the first bit of the plaintext is 1
  - we assume that an attacker can distinguish correct/incorrect MAC situation
  - *remark: more practical scenario with SSL/TLS with MtE and oracle padding attack (padding vs. MAC error – learning something about plaintext)*
- E&M: MAC algorithm can leak information about the plaintext