Hash functions and data integrity

Andreas Bomke

June 12, 2001

Definition 1. A **message authentication code (MAC)** is a family of functions h_k, where k is a secret key, with

- **ease of computation**
  ($k, x \mapsto h_k(x)$ easy to compute)

- **compression**
  ($h_k : \{0, 1\}^* \to \{0, 1\}^n$ for a fixed n)

- **computation resistance**
  for any k (unknown to the adversary): given some pairs $(x_i, h_k(x_i))$ it is computationally infeasible to compute any new pair $(x, h_k(x))$ with $x \neq x_i$
  ($\Rightarrow$ key non-recovery)

Types of forgery:

- **selective** forgery: the adversary is able to produce a new pair $(x, h_k(x))$ for an x of his choice

- **existential** forgery: the adversary has no control over x
  potential use: replace a monetary amount known to be small by a random number

Exhaustive search:
For a t-bit key and a n-bit MAC the probability to correctly guess a MAC is at least

$$max(2^{-t}, 2^{-n})$$

(because we may simply try all keys or all MACs, whatever is shorter)

Example (Cipher Block Chaining MAC).

given: data x, block cipher E, secret MAC key k

pad and divide x in n-bit blocks $x_1, \ldots, x_t$

compute $H_1 := E_k(x_1)$

$H_i := E_k(H_{i-1} \oplus x_i), 2 \leq i \leq t$

output H_t

Trivial Example of existential forgery:

let $x \in \{0,1\}^n, E_k : \{0,1\}^n \to \{0,1\}^n$;

given $y := MAC(x) = E_k(x) \in \{0,1\}^n$ and $MAC(y)$,

we also know $MAC(x||0^n) = E_k(E_k(x) \oplus 0^n) = E_k(E_k(x)) = MAC(y)$

Example (Construction of a MAC using an MDC).

given $x = x_1 \dots x_t$ and an iterated MDC h defined as $H_0 := const, H_i := f(H_{i-1}, x_i), h(x) := H_t$,

define $MAC_k(x) := h(k||x)$
$\rightarrow$ for any single block y, we can deduce $MAC_k(k||x||y)$ as $f(MAC_k(x), y)$ without knowing k.

define $MAC'_k(x) := h(x||k) = f(h(x), k)$
$\rightarrow$ consider an offline birthday attack, where an adversary may find $x' \neq x''$ with $h(x') = h(x'')$ in $O(2^{n/2})$ evaluations of h (without knowing k). Then, given $MAC'_k(x')$, he also knows $MAC'_k(x'')$.

define $MAC''_k(x) := h(pad(k)||h(pad(k)||x))$
$\rightarrow$ no attacks are known
(padding is used to ensure that the internal computation involves at least two iterations)

Data integrity and message authentication

data integrity $:=$ property whereby data has not been altered since it was created by an authorized source

message authentication $:=$ authentication whereby a party is corroborated as the original source of specified data

message authentication $\Rightarrow$ data integrity

Using MACs alone, message re-use or replay is undetectable.

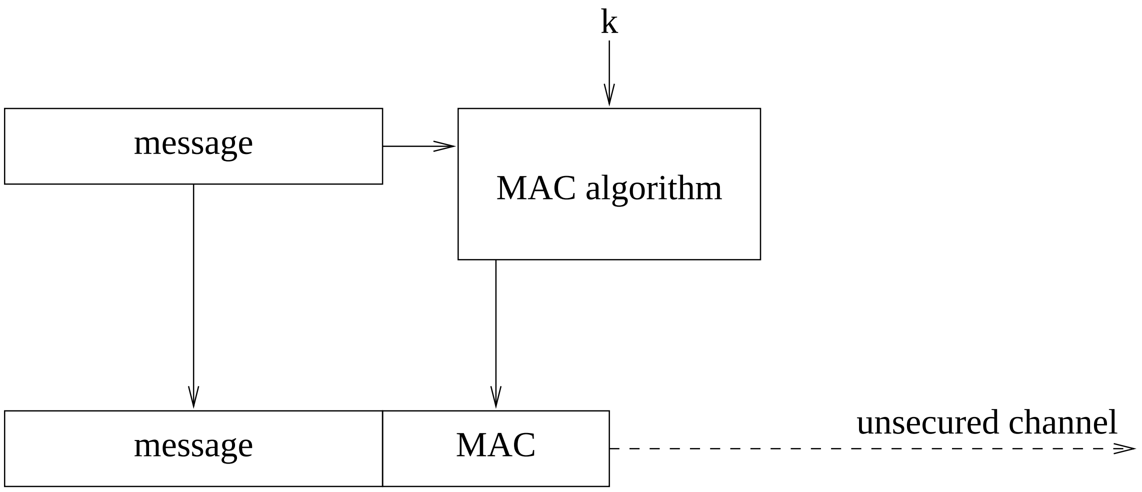
transaction authentication $:=$ message authentication with uniqueness and timeliness guarantee; can be achieved using e.g. sequence numbers, random nonces, or timestamps.

- Means to achieve data integrity against non-malicious threats (e.g. noisy channels with transmission errors), such as checksums, error-correcting codes, and cyclic redundancy codes, don't work against malicious threats (i.e. adversarial manipulation).

- Encryption alone does not guarantee data integrity!
  Simple example: ciphertext blocks of any block cipher (in ECB mode) can be re-ordered giving valid ciphertext.

Methods for providing data integrity using hash functions

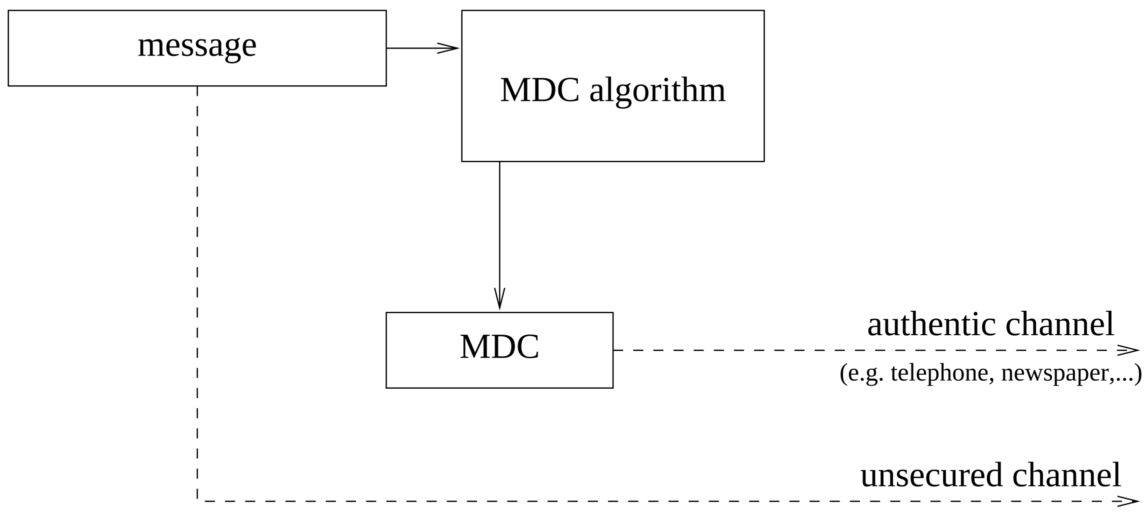
a)



k: secret key shared by the message originator and the intended recipient

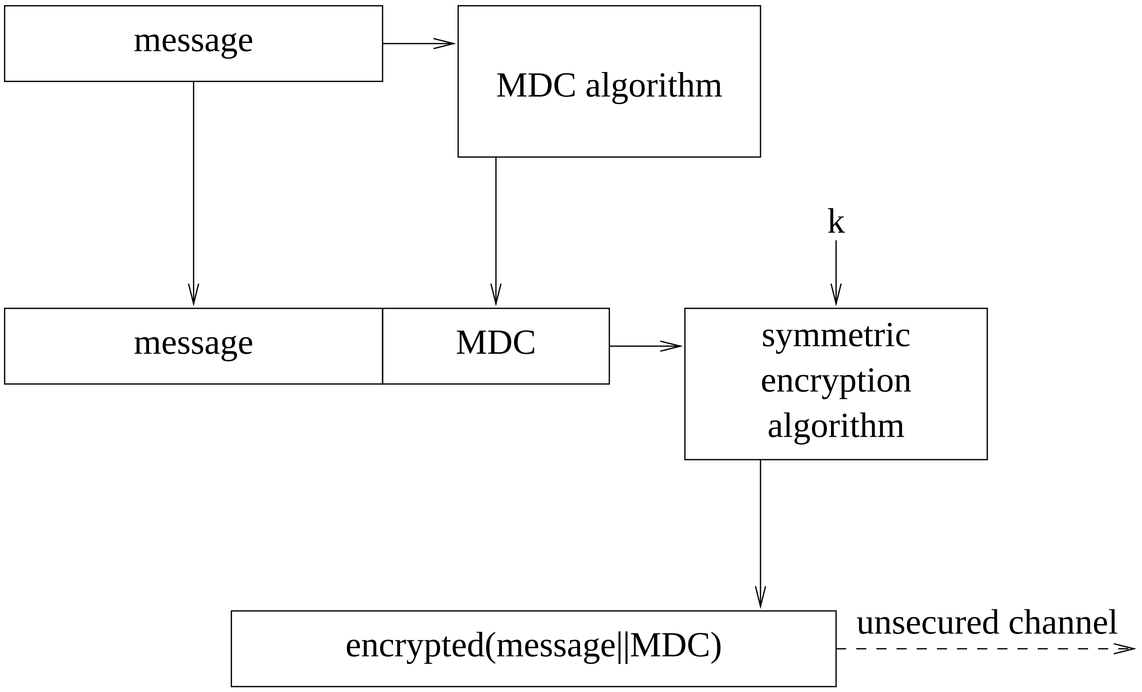
$\rightarrow$ message authentication without privacy

b) given an authentic (but not necessarily private) channel, we can reduce the amount of data to be transmitted authentically using MDCs:



example application: distribution of software or public keys via untrusted networks

c) message authentication combined with confidentiality:



variant: use a second key k' and a MAC $h_{k'}$ to send

$$E_k(x||h_{k'}(x))$$

advantage: ”double security”
disadvantage: unexpected dependencies could lead to security weaknesses

Birthday paradox

When drawing elements randomly, with replacement, from a set of N elements, with high probability a repeated element will be encountered after $O(\sqrt{N})$ selections.

$\rightarrow$ in general it is easier to find collisions than to find pre-images

Usage: a dishonest signer signs x_1 and later claims to have signed x_2 instead.
Or: a dishonest verifier makes somebody sign x_1 and later claims its signature on x_2.

example: collisions for MD5 can be found in $\sim 2^{64}$ operations

Literature

This introductory talk is based on Chapter 9 of

- Menezes, van Oorschot, Vanstone, 1996
  Handbook of Applied Cryptography, CRC Press
  www.cacr.math.uwaterloo.ca/hac