Integrity and authentication

Integrity of messages

Goal: Ensure change of message by attacker can be detected
Key tool: *Cryptographic hash function*

Definition

A *cryptographic hash function* is a function from bitstrings of [almost] arbitrary length to bitstring of a small, fixed length (e.g. 160 bits or 256 bits) such that

- For any message or string x, $h(x)$ is easy to compute.
- h is *one-way*, i.e., it is hard to invert, in the sense that for any y it is computationally infeasible to find an x such that $y = h(x)$
- h is *collision-resistant*, i.e., for any x it is computationally infeasible to find an x' such as $h(x) = h(x')$.

Uses of hash functions

Example:
MD5 used for verifying integrity of ubuntu packages
Each package comes with MD5-hash.

Idea is to confirm MD5-hash by a secure channel (phone? another website?) or to sign hashes with private key (see later).

Collision-resistance

- A hash function necessarily has loads of collisions. This can be seen just by considering the size of the input message space and the size of the output message space.
  - For SHA-1, the input is a message up to length $2^{64} - 1$ bits.
  - The output is a message of length 160 bits.

  So, on average, each possible output corresponds to a vast number of inputs!
- Collision-resistance means that it is computationally hard to find a single collision. If a single collision is found, the hash function is considered broken.

Collision-resistance

Collision resistance implies that changing just one bit of the input should "completely change" the output; e.g., that might mean that typically half of the output bits are changed.

Example

```
echo "The quick brown fox jumps over the lazy dog" | sha1sum
```

```
be417768b5c3c5c1d9bcb2e7c119196dd76b5570
```

```
echo "The quick brown fox jumps over the lazy dof" | sha1sum
```

```
71451f4d87fe0e866ec8ebc57f5379b23fa2921f
```

In binary:

```
10111110010000010111011101101000101101011100001111000101110...
01110001010001010001111101001101100001111111111000011101000...
```

Of the 60 bits shown, 27 have changed.

"Birthday paradox"

Suppose there are 23 people in the room.

- The chance that someone has **my** birthday is approximately 0.06.
- The chance that **some two** of them have the same birthday is much higher: it is approximately 0.5.

Suppose there are 50 people in the room.
Then these probabilities become 0.13 and 0.96.

One-way vs collision-resistant

One-way: given y, infeasible to find x such that $h(x) = y$.
Collision-resistant: Infeasible to find distinct x and x' such that $h(x) = h(x')$.

The work needed to break collision-resistance is much less than the work needed to break one-way. If it takes 2^n operations to break one-way, then it takes only approximately $2^{n/2}$ operations to break collision resistance.

The "birthday paradox" is one way to understand this distinction.

Moral: a hash function that outputs n bits has at most $n/2$ bits of security.

Example Hash Functions

MD4 128-bits hash length. Proposed 1990. Collisions found 1995.

MD5 128-bits hash length. Proposed 1992. Collisions found 2004 after years of effort. Nowadays, collisions can be found in seconds on an ordinary PC.

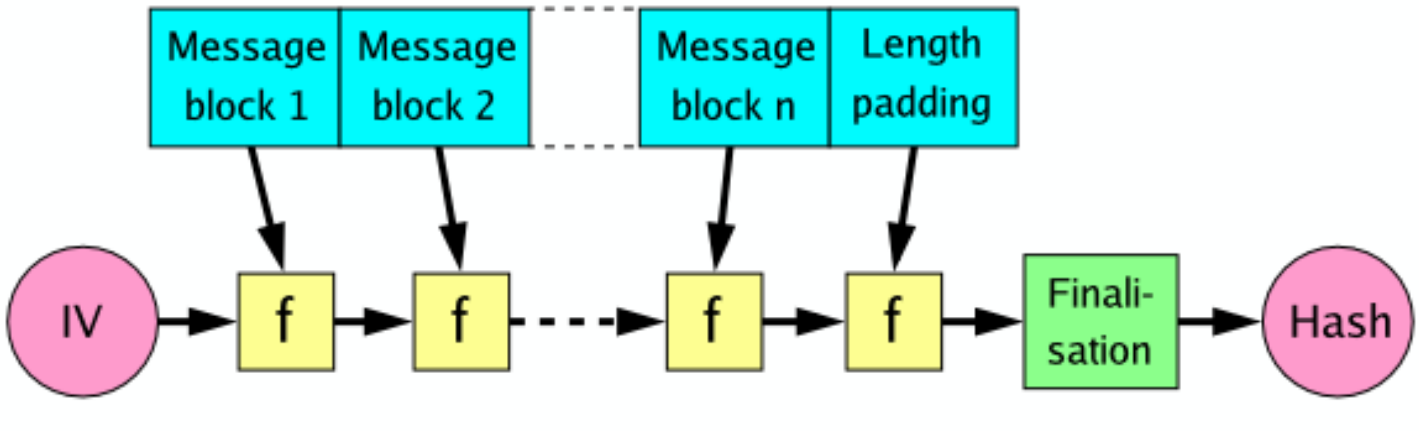
SHA-1 160-bits hash length. Proposed 1995. ~~No collisions have been found. However, it has been shown that they could be found with around 2^{65} operations – vastly less than the theoretical 2^{80}.~~ Collisions found in February 2017. No longer recommended to be used (but still in widespread use).

SHA-2 224, 256, 384, or 512 bits hash length. Proposed 2001. No collisions found for any of the bit lengths. Still considered secure, though there are "erosions".

SHA-3 variable digest size. Proposed 2015.

The Merkle-Damgård Construction

Merkle-Damgård Construction produces a cryptographic hash function from a *compression function* shown as f below.
Idea: Apply compression function repeatedly



Source: Wikipedia

MD4, MD5, SHA-1 and SHA-2 all use the Merkle-Damgard construction. Only SHA-3 does something completely different.

The algorithm for MD4

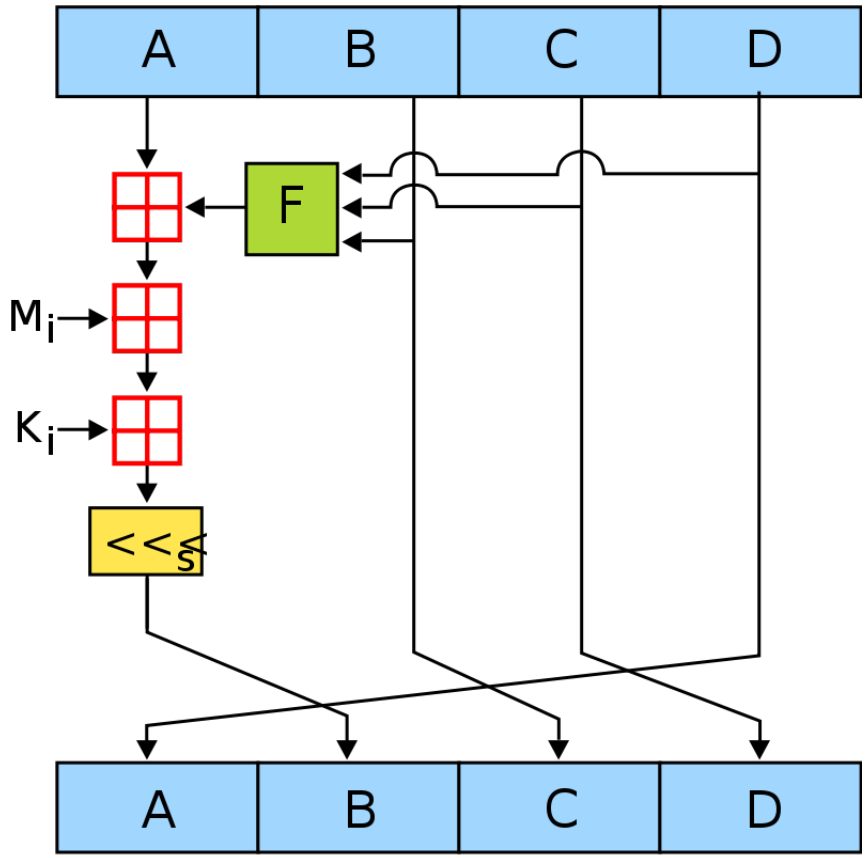
- First, the message is padded to a length that is 64 bits less than a multiple of 512 bits. The padding is 1 followed by a string of 0s.
- A 64 bit representation of the length of the (unpadded) message is added.
- A, B, C, D are initialised to some fixed constants (they're just part of the definition of MD4).
- The message is split into 512 bit blocks. Each block is processed in turn:
  - The compression function is applied to A,B,C,D and the current block, resulting in a new value of A,B,C,D.
- The hash value is the final value of A,B,C,D.

The compression function for MD4

- Input: 512 bit block of the message, and current value of A,B,C,D.
- Output: new values of A,B,C,D.
- The 512 bit message is split into 16 chunks $M_0, \ldots, M_{15}$ of 32 bits each
- Three rounds, each of 16 steps, are used to transform A,B,C,D into new values of A,B,C,D. Each of the 16 steps consumes one of the chunks of the message. The rounds make use of three special functions, one for each round:

$$\begin{aligned} F(X,Y,Z) &= (X \wedge Y) \vee (\neg X \wedge Z) \\ G(X,Y,Z) &= (X \wedge Z) \vee (Y \wedge Z) \vee (X \wedge Z) \\ H(X,Y,Z) &= X \oplus Y \oplus Z \end{aligned}$$

One of the 16 steps from one of the rounds of MD4. In the picture, M_i is the 32-bit chunk of the message the step consumes. K_i is a 32-bit constant



Source: Wikipedia

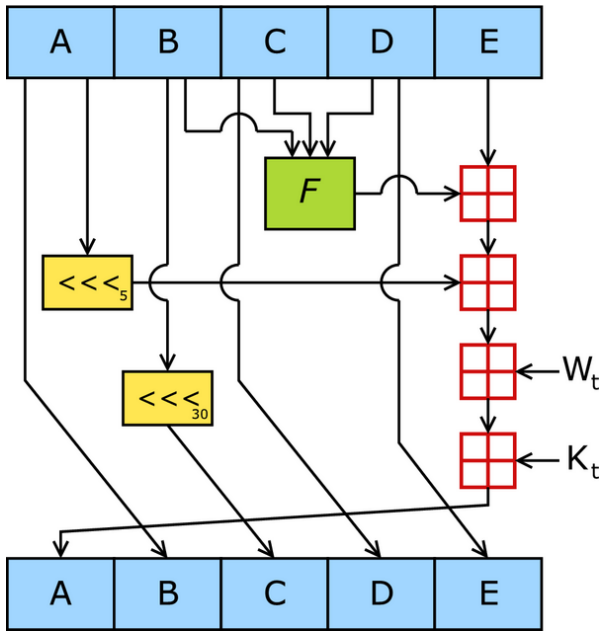
MD5

Same parameters, same initialisation vector
Adds fourth round with a different non-linear function

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$

SHA-1

SHA-1 extends hash size to 160 bits
Extension of MD4 (same non-linear functions used)
Message blocks used differently
Recent usage: TLS, SSL, SSH, BitTorrent



Source: Wikipedia

SHA-2

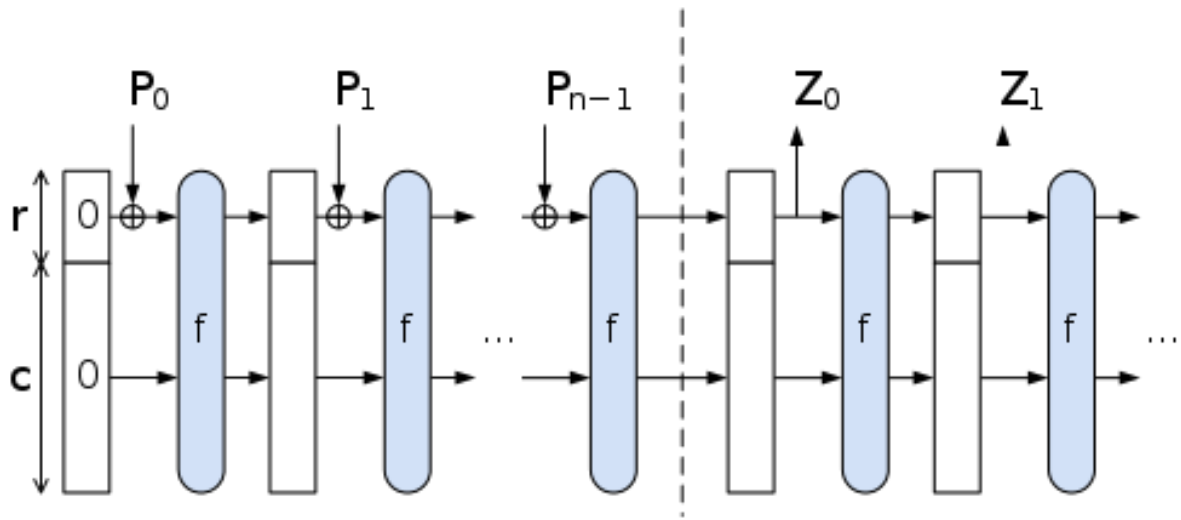
Successor of SHA-1
Introducing more bitwise operations
increasing block sizes
increasing hash length...

but essentially the same ideas.

SHA-3

NIST held a competition for new hash functions, which ended in 2015. Keccak was the winner, and one of its family is now known as SHA-3.

In contrast with previous hash functions, it uses the *sponge construction* rather than the *Merkle-Damgard construction*.



Source: Wikipedia

With this construction, the security parameter (c) and the output length (the length of the Z's added together) can be varied.

SHA-3 uses a 1600-bit *block transformation function f*.

Message Authentication Codes

A hash function can be used to guarantee the *integrity* of messages (e.g., the integrity of downloaded software).

However, a hash function alone is insufficient to guarantee the *authenticity* of messages (i.e., the fact that a message came from a particular source). If you merely use a hash function, the attacker can modify message and recompute hash.

To guarantee authenticity, we use a "message authentication code" – a keyed hash function. Assumption: Alice and Bob share key k

Alice sends to Bob: m, $\mathrm{MAC}_k(m)$.

When Bob receives this message, say m, x, he computes $\mathrm{MAC}_k(m)$ and then checks if $x = \mathrm{MAC}_k(m)$.

How to define a MAC function from a hash function?

How to define MAC from a hash function?

- $\text{MAC}_k(m)$ could be defined as $h(k||m)$. However, this is vulnerable to a "length extension attack". Given m and $h(k||m)$, one can construct m' and $h(k||m')$ (for example, let m' be $m||padding||length(m)||m''$).

  Thus, if Alice sent the message m with $\text{MAC}_k(m)$ using this definition, the attacker could modify the message to m' with $\text{MAC}_k(m')$.

- The constructions $\text{MAC}_k(m) = h(m||k)$. and $\text{MAC}_k(m) = h(k||m||k)$. have also been found to have weaknesses.

HMAC

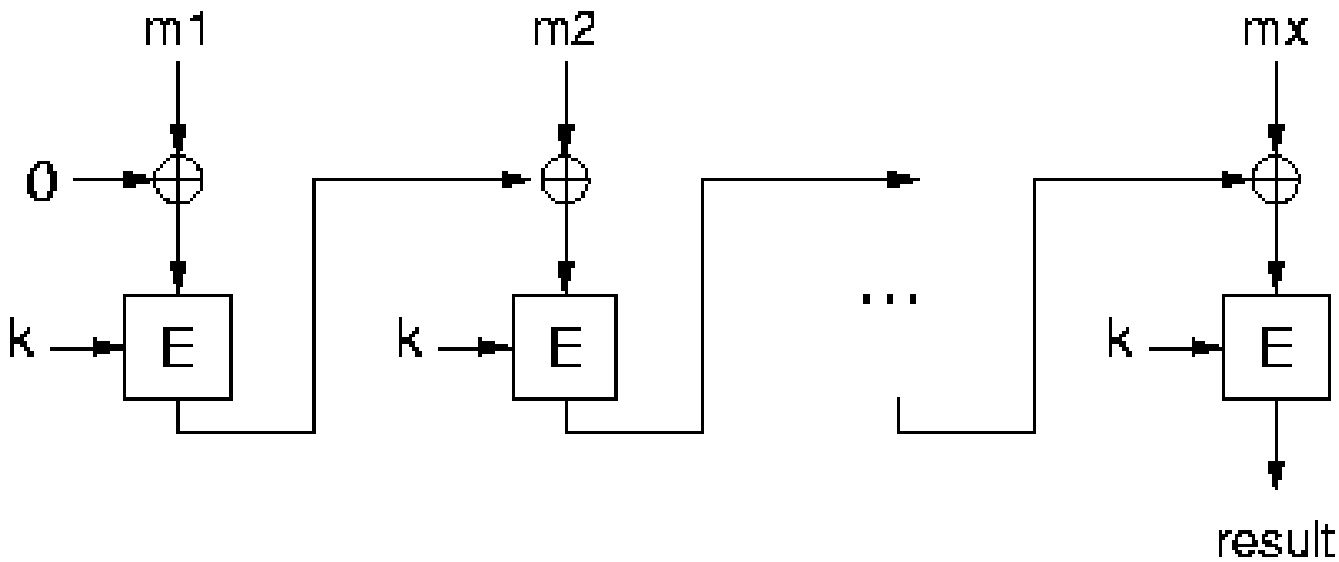
$\mathrm{HMAC}_k(m)$ defined as:

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

Here, the key k is padded with zeros to the blocksize of the hash function, and *ipad* and *opad* are constants of that blocksize.
The values of ipad and opad are not critical to the security of the algorithm, but were defined in such a way to have a large Hamming distance from each other and so the inner and outer keys will have fewer bits in common.
This definition can be shown to have some good security properties: if you can break HMAC, then you can break the underlying hash function.

CBC-MAC

CBC-MAC uses CBC mode of operation for block cipher



Source: Wikipedia

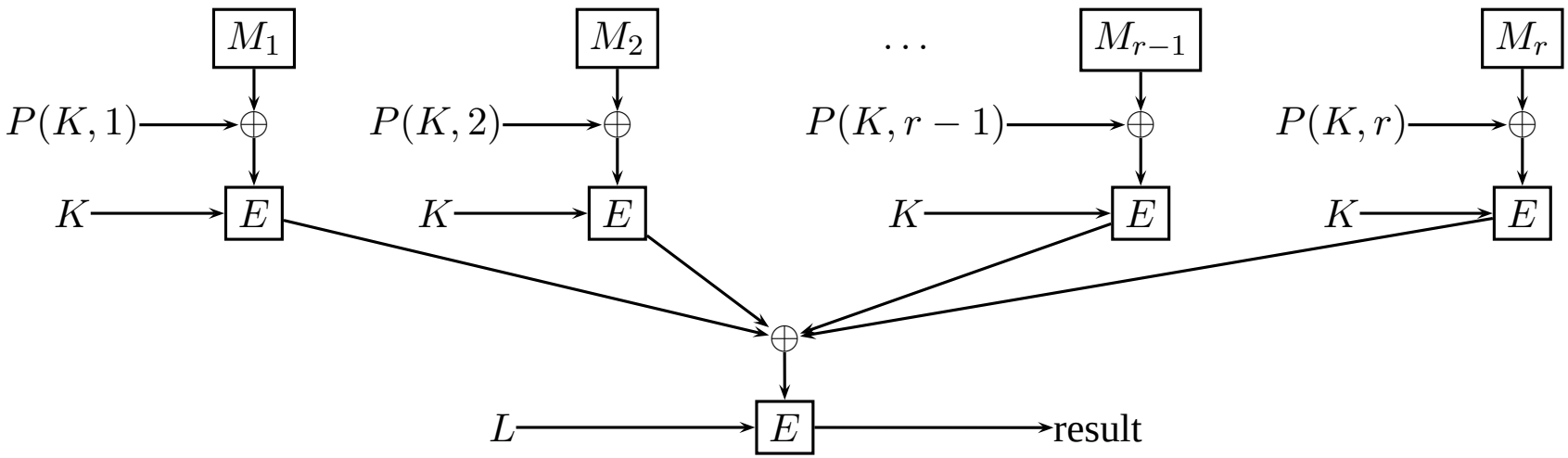
PMAC

Hash functions, HMAC and CBC-MAC are not parallelisable

PMAC addresses this issue

Have two keys K and L

Have function $P(K, i) = K * x^i$ in $\mathbb{F}_{2^n}$



Security of hash function

A hash function is collision-resistant if the attacker can't output a collision.

Reminder: padding scheme for Merkle-Damgard construction with block size n.
Given a message M, add $10\cdots0||msglen$, where $msglen$ is the length of the message represented as a 64 bit number. The number of 0s is the smallest number that brings the padded message to a multiple of the block size.

Theorem

If h is a collision-resistant compression function, and messages are padded as above, then the Merkle-Damgåard construction without a finalisation function produces a collision-resistant hash function.

Security of MAC

Let m be a message. Then $\mathrm{MAC}_k(n)$ is sometimes called the *tag* for m.

We call a MAC function *secure* if an attacker cannot produce a valid (message, tag)-pair which he hasn't seen before. (We assume the attacker doesn't have the key.)

This is called *secure against existential forgery*,

Definition

The MAC-game between challenger and attacker is defined as follows:

- The attacker does some computations and may in the process supply messages $m_1, \ldots, m_n$ to the challenger
- The challenger returns $t_1, \ldots, t_n$ to the attacker , which are the result of creating the MAC for the messages $m_1, \ldots, m_n$.
- The attacker does some more computations and then supplies to the challenger a pair (m, t), which is not equal to any of the pairs $(m_1, t_1), \ldots, (m_n, t_n)$.
- The challenger outputs 1 if t is obtained by creating the MAC for m, otherwise he returns 0.

The attacker wins the MAC-game if the challenger outputs 1.

Definition

We call a MAC *secure* if no attacker can win the MAC-game with non-negligible probability.

Here, as before, the probability is a function of the key length.

Example

CBC-MAC is not secure (unless you add restrictions).

Suppose the attacker possesses (m, t) and (m', t'). Then he can forge a third pair, (m'', t''):

We assume that m' is more than one block long; say
$m' = m'_1||m'_2||\ldots||m'_p$.
Set $m'' = m||(m'_1 \oplus t)||m'_2||\ldots||m'_p$, and $t'' = t'$.
Check that (m'', t'') is a valid message-tag pair.

CBC-MAC result

Theorem

Assume CBC-MAC is used only on messages of a fixed length. If the block cipher used is a secure block cipher, then CBC-MAC is a secure MAC.

Another way to achieve this is to prepend the length in the message.

HMAC and PMAC results

Theorem

If the hash function used is secure, then HMAC is a secure MAC.

Theorem

If the block cipher used is secure, then PMAC is a secure MAC.