Merkle-Damgård construction

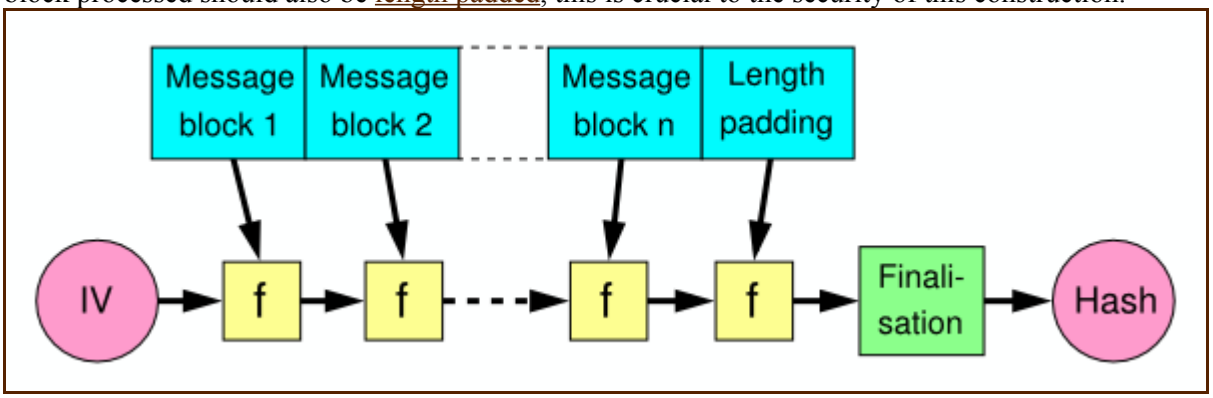
From Wikipedia, the free encyclopedia

Jump to: navigation, search

In cryptography, the **Merkle-Damgård construction** is a method used inside cryptographic hash functions. Most widely used hash functions, including SHA-1 and MD5 uses the Merkle-Damgård construction.

A hash function must be able to process an arbitrary-length message into a fixed-length output. This can be achieved by breaking the input up into a series of equal-sized blocks, and operating on them in sequence using a *compression function*. The compression function can either be specially designed for hashing or be built from a block cipher.

The last block processed should also be length padded, this is crucial to the security of this construction.



In the diagram, the compression function is denoted by *f*, and transforms a fixed length input to an output of the same size. The algorithm starts with an initial value, the initialization vector (IV). The IV is a fixed value (algorithm or implementation specific). For each message block, the compression (or compacting) function *f* takes the result so far, combines it with the message block, and produces an intermediate result. The last block is padded with zeros as needed and bits representing the length of the entire message are appended. (This is called length padding, see below for detailed example.)

To harden the hash further the last result is then often fed through a *finalisation function*. The finalisation function can have several purposes such as compressing a bigger internal state (the last result) into a smaller output hash size or to guarantee a better mixing and avalanche effect on the bits in the hash sum. The finalisation function is often built by using the compression function.

[edit]

Security characteristics

The popularity of this construction is due to the fact, proven by Merkle and Damgård, that if the compression function *f* is collision resistant, then so is the hash function constructed using it. Unfortunately, this construction also has several undesirable properties:

- Length extension - once an attacker has one collision, he can find more very cheaply.
- Second-preimage attacks against long messages are always much more efficient than brute force.
- Multicollisions (many messages with the same hash) can be found with only a little more work than collisions.
- "Herding attacks" (first committing to an output h, then mapping messages with arbitrary starting values to h) are possible for more work than finding a collision, but much less than would be expected to do this for a random oracle.

[edit]

Length padding example

Let's say the message to be hashed is "Wikipedia" and the block size of the compression function is 8 bytes (64 bits). To be able to feed the message to the compression function the last block needs to be zero padded to a full block. So we get two blocks looking like this:

```
Wikipedi  a0000000
```

But this is not enough since it would mean that for instance the message "Wikipedia00" would get the same hash sum. Therefore also the length of the message is added in an extra block, so we get three blocks like this:

```
Wikipedi  a0000000  00000009
```

Now that is a bit wasteful since it means hashing one extra block. So there is a slight speed optimisation that most hash

algorithms use. If there is space enough among the zeros padded to the last block the length value can instead be padded there. Like this:

```
Wikipedi  a0000009
```

Note that to avoid confusion the hash algorithm must use a fixed bit-size for the length value, say 40-bit. So the length value padded in the end really is "00009" not just "9".

[edit]

See also

- Cryptographic hash function
- Hash functions based on block ciphers
- Ralph Merkle - One of the two inventors of the Merkle-Damgård structure.
- Ivan Damgård - The other inventor of the Merkle-Damgård structure.

Cryptographic hash functions and **Message authentication codes (MACs)** - edit

Hash algorithms: Gost-Hash | HAS-160 | HAVAL | MDC-2 | MD2 | MD4 | MD5 | N-hash | RIPEMD | SHA family | Snefru | Tiger | WHIRLPOOL

MAC algorithms: HMAC | CBC-MAC | OMAC/CMAC | PMAC | UMAC | Data Authentication Code | Poly1305-AES

Standardization: CRYPTREC | NESSIE **Misc:** Hash functions based on block ciphers | Avalanche effect

Go Search

Toolbox

- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- Cite this article