SEED SECURITY LABS

Hash Length Extension Attack Lab

General problem

- Some hash function constructions are iterative in the sense that they update the hash calculation result of previous parts of the document ("message") to hash.
  - They keep updating up to the last part (block) of the message.
- This feature would allow the (trivial) calculation of the hash of a composed message *M1* || *M2*[1] starting from the hash of *M1*.
  - And so forth with additional messages, *M1* || *M2* || *M3*...

[1] concatenation of M1 with M2

...General problem

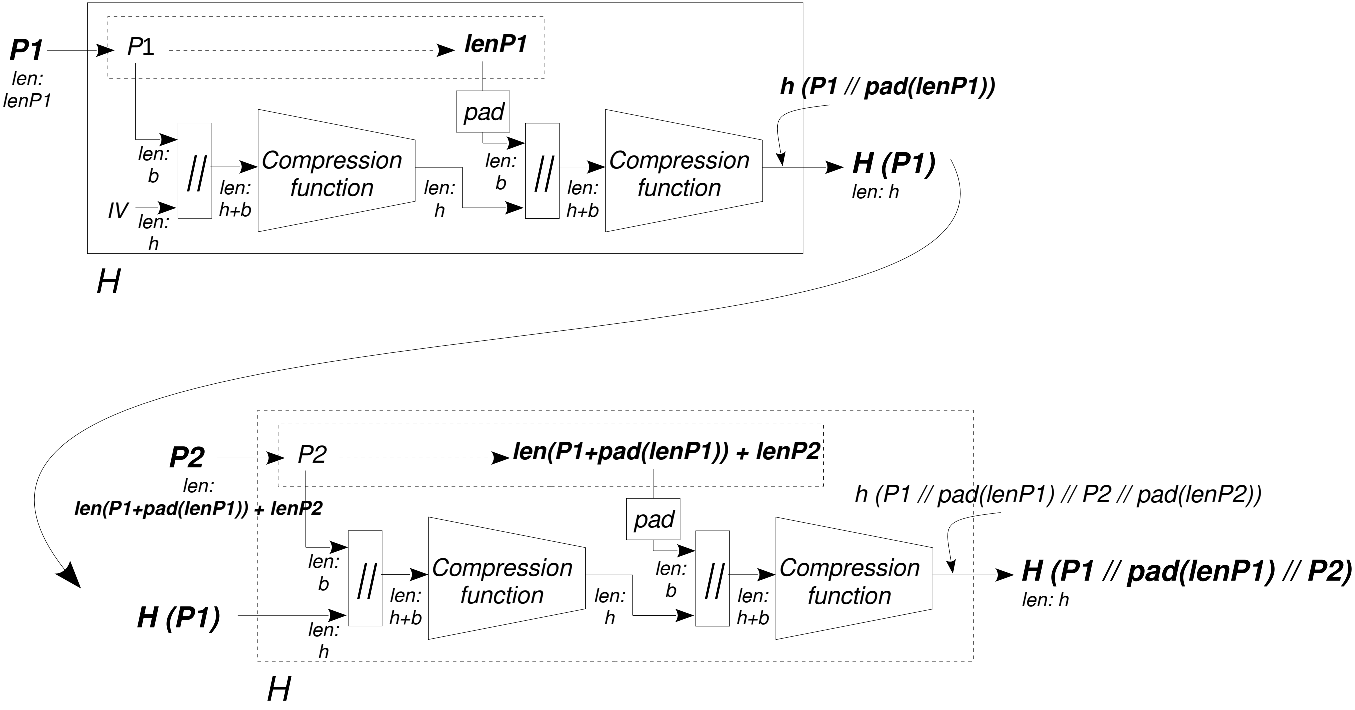
- A Message Integrity Code, MIC[1], based on a hash function of this type[2] and calculated as *hash* (*K* || *M*), with *K* secret, and *M* the document to protect from unauthorized modification, is vulnerable to the so called "*Hash Length Extension Attack*":
    - Forging the MIC of an extended message M || N is very easy.
- To prevent this vulnerability, hash functions with constructs of this type use a strengthening feature: the adding of a last hashing stage with *padding* bytes that include the size of the original message! That is the case of the Merkle–Damgård construction.
    - Nevertheless, even with this protection, a *Hash Length Extension Attack* is still possible!

[1] synonym of Message Authentication Code, MAC
[2] SHA-256 is an example of such hash!

Hash Length Extension Attack procedure

- The attacker needs to have access to the inner parts of the hash mechanism, namely:
    - for inserting the previous hash *H*(*P1*) in the internal state;
    - for updating the internal counter of message length.
    - see [FIG]

...Hash Length Extension Attack procedure


P1
len:
lenP1
P1
lenP1
pad
h (P1 // pad(lenP1))
len:
b
IV
len:
h
//
len:
h+b
Compression
function
len:
h
len:
b
//
len:
h+b
Compression
function
H (P1)
len: h
H
P2
len:
len(P1+pad(lenP1)) + lenP2
P2
len(P1+pad(lenP1)) + lenP2
pad
h (P1 // pad(lenP1) // P2 // pad(lenP2))
len:
b
H (P1)
len:
h
//
len:
h+b
Compression
function
len:
h
len:
b
//
len:
h+b
Compression
function
H (P1 // pad(lenP1) // P2)
len: h
H