size_t Does Matter

Hash Length Extension Attacks Explained

Mika Boström

<bostik@iki.fi>, <mika.bostrom@smarkets.com>

dc4420, 2015-10-29

Cryptographic Hash Properties

- Digest Size (n bits)
- Input Block Size (m bits)
- m > n
- Input processed block at a time
- Mutates internal state
- In other words: blocks are chained
- Merkle-Damgård: last block padded, includes number of bytes processed

Hash Length Extension

- ”Append data to a keyed hash, without knowing the key, and calculate a valid hash with your data included”
- Or, programmer friendly:
  - H1 = H(key + data + padding)
  - Transmit H1, data
  - Attacker: append $EVILDATA, calculate H2
  - Transmit: H2, (data+$EVILDATA)
  - Receiver: calculate H = (key + received data)
    - **H = H2**

SHA-1 Properties

- 160-bit output
- 512-bit input block
- Merkle-Damgård construct
  - Yes, that Merkle

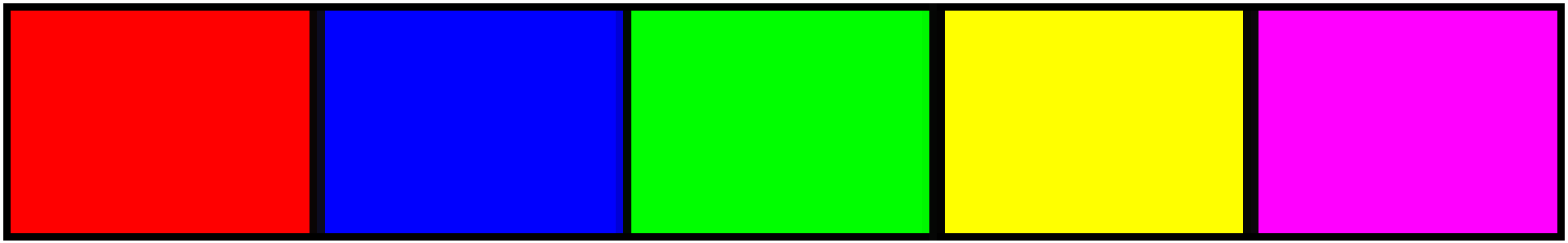
SHA-1 Internal State

```
struct SHA1State {
        uint32 A;
        uint32 B;
        uint32 C;
        uint32 D;
        uint32 E;
}
```

SHA-1 Internal State

```
struct SHA1State {
        uint32 A;
        uint32 B;
        uint32 C;
        uint32 D;
        uint32 E;
}
```

SHA-1 Final Hash … visualised



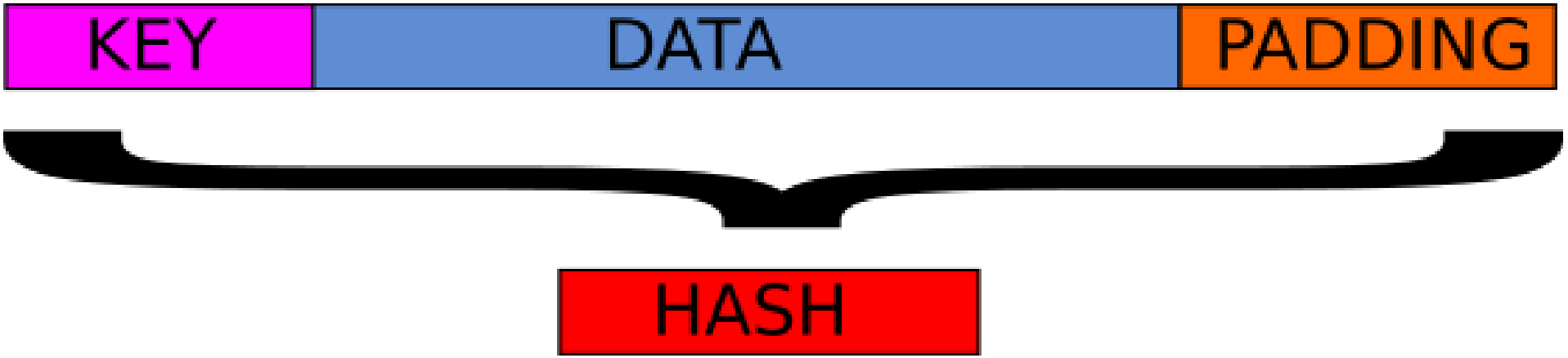
Uint32 A Uint32 B Uint32 C Uint32 D Uint32 E

Keyed Hash

- Secret shared key
- Known payload data
- Hash = H(key + data)

Looks a bit like … salted hash?

Hash Extension Illustrated



Hash Extension Illustrated

- Key + padding missing
- Padding: includes number of bytes hashed
- Guess key length, calculate padding!

Hash Extension Illustrated

Initialize registers to known state...

Append own data...

And calculate new hash

Hash Extension Illustrated

- Hash is valid over the whole of preceding data, with the key prefixed
- Attacker did not need to know the shared key
- Effect of **EVIL DATA** depends on implementation
- *Would you guarantee your implementation handles every possible case of malformed but accepted-as-good input?*

Morale Of The Story

- Keyed hash as authentication method: broken
- Just use HMAC instead
- … even with SHA-3
- … because someone could plug a vulnerable hash into the construct

- *Applied crypto is a world of cargo-culting*

Trivia: Also Vulnerable

- MD5 (*d'oh*)
- SHA-256
- SHA-512
- RIPEMD-160

Trivia: Not Vulnerable

- SHA-384 (truncated)
- SHA-256/512 (truncated)
- SHA-3 (incomplete state export)

Code Gone Wild

- https://github.com/stephenbradshaw/hlextend
- https://github.com/bwall/HashPump
- https://github.com/iagox86/hash_extender
- Just to name a few

Question Time