HMAC

Keyed Hash Functions as MACs

- have desire to create a MAC using a hash function rather than a block cipher
  - because hash functions are generally faster
  - not limited by export controls unlike block ciphers
- hash includes a key along with the message
- original proposal:

```
KeyedHash = Hash(Key|Message)
```

  - some weaknesses were found with this
- eventually led to development of HMAC

HMAC

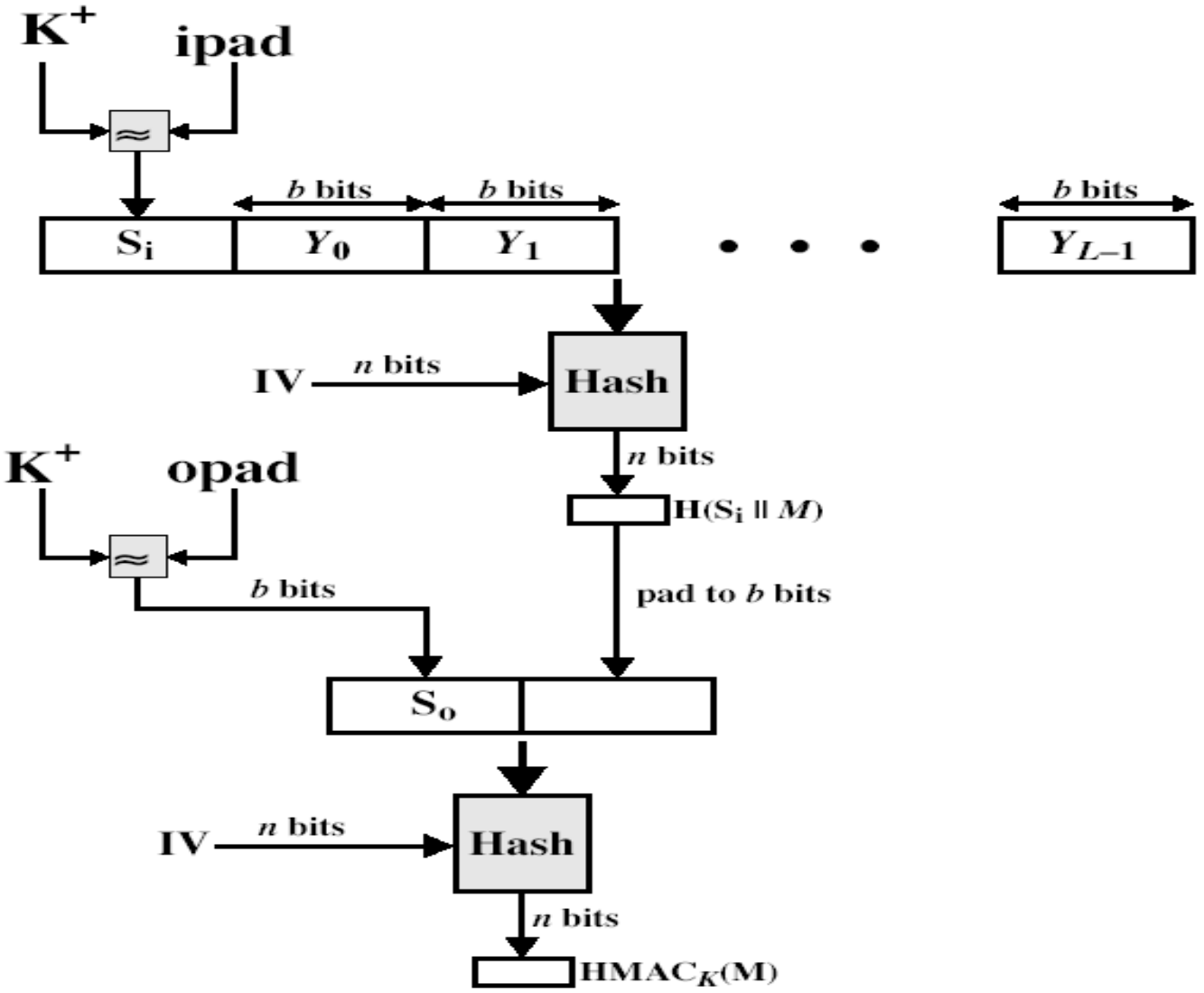
- specified as Internet standard RFC2104
- uses hash function on the message:

$$HMAC_K = Hash[(K^+ \text{ XOR opad}) \,||\, Hash[(K^+ \text{ XOR ipad})||M)]]$$

- where K^+ is the key padded out to size
- and opad, ipad are specified padding constants
- overhead is just 3 more hash calculations than the message needs alone
- any of MD5, SHA-1, RIPEMD-160 can be used

HMAC (cont'd)

- HMAC Design Objectives [RFC2104]
  - To use available hash functions.
  - To allow for easy replaceability of the embedded hash function
  - To preserve the original performance
  - To use and handle keys in simple way
  - To have a well understood cryptographic analysis of the strength of the authentication mechanism


K+
ipad
b bits
b bits
b bits
Si
Y0
Y1
YL–1
IV
n bits
Hash
n bits
K+
opad
H(Si || M)
b bits
pad to b bits
So
IV
n bits
Hash
n bits
HMACK(M)

HMAC Algorithm

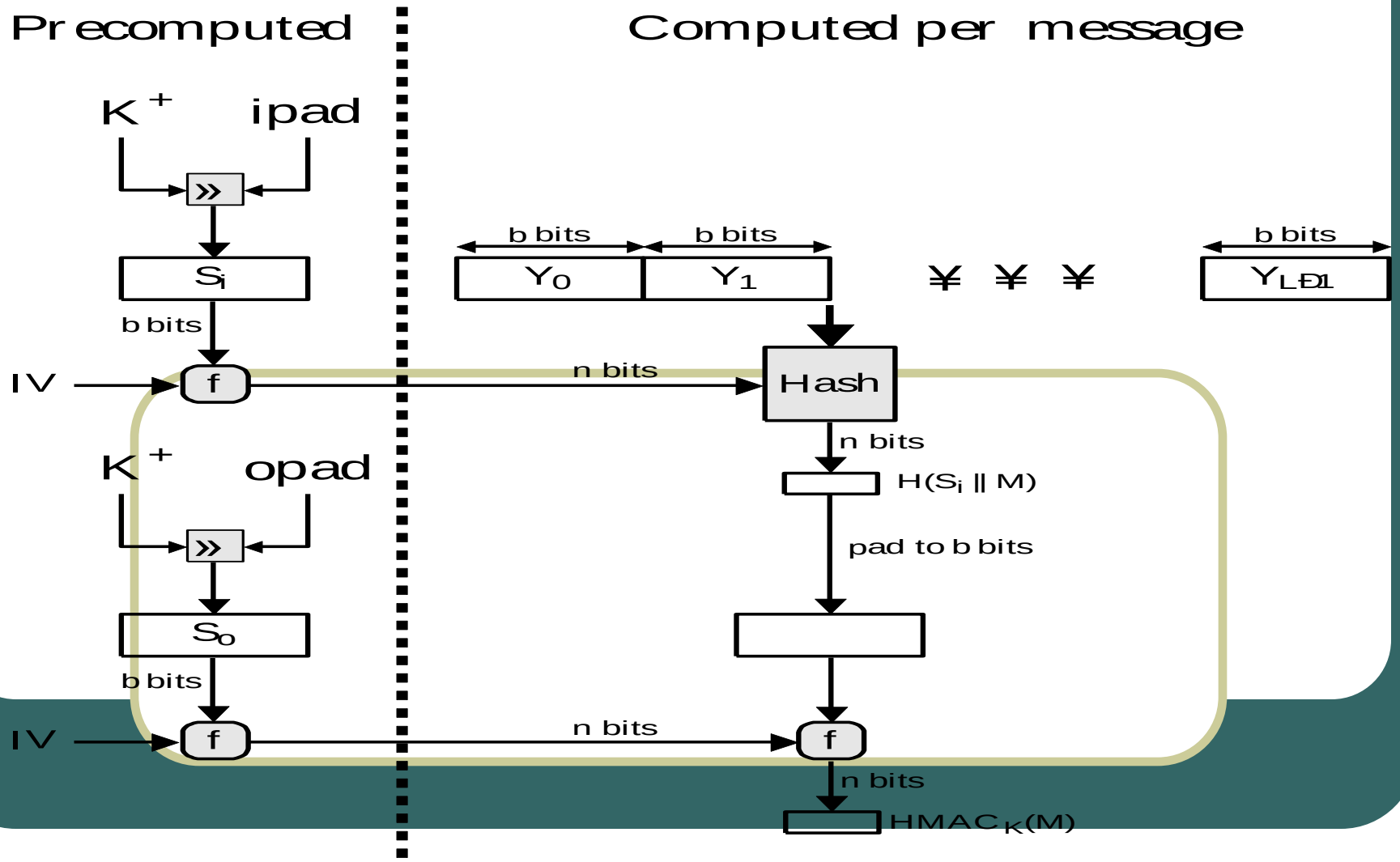
$$HMAC_K = H[(K^+ \oplus opad) \| H[K^+ \oplus ipad \| M]]$$

1. Append zeros to the left end of K to create a *b*-bit string K^+
2. XOR K^+ with ipad to produce the *b*-bit block S_i
3. Append M to S_i
4. Apply H to the stream generated in step 3

Algorithm (cont′d)

5. XOR K^+ with opad to produce the *b*-bit block S_o
6. Append the hash result from step 4 to S_o
7. Apply H to the stream generated in step 6 and output the result

EFFICIENT IMPLEMENTATION of HMAC



HMAC Security

- know that the security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints