TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Betriebssysteme und ***Sicherheit***

Stefan Köpsell, Thorsten Strufe

Modul 5: Mechanismen – Integrität

Disclaimer: large parts from Mark Manulis, Dan Boneh, Stefan Katzenbeisser

Dresden, WS 15/16

Reprise from the last modules

You have an overview of cryptography and cryptology

You know different adversary models and their corresponding games

You know what symmetric cryptography is

You recall the difference of stream and block ciphers

You can explain the OTP and constructions for stream ciphers

You can prove that the OTP has perfect secrecy

You can tell PRFs and PRP apart and you know constructions for block ciphers

You can explain different modes of operation and their properties

Module Outline

Verification of message integrity as a goal

Adversary and security models

Hashes and cryptographic hash functions

Collisions and how to create them (also: the birthday paradox)

The Merkle-Damgard construction , real hash functions (MD5, SHA-1)

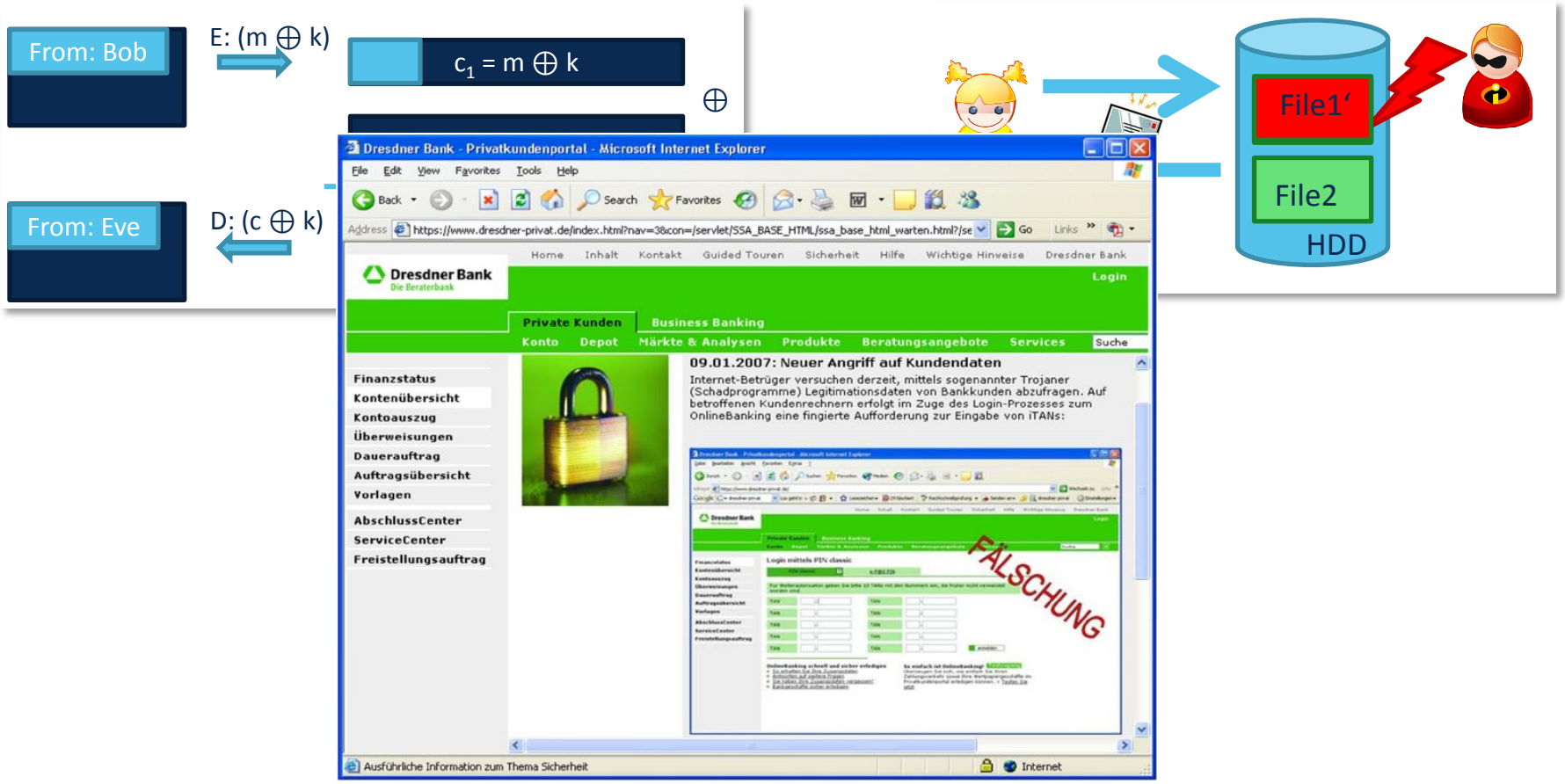
Secure MACs from hash functions and PRFs

MACs using block ciphers (CBC, NMAC, HMAC)

Integrity and Authenticity

So far messages can be kept confidential

Integrity of messages not given



Message Integrity

Algorithms:

Tag S: $M \rightarrow T$ $M = \{0,1\}^n$; $S = \{0,1\}^t$ with $n >> t$

Verify V: $M \times T \rightarrow \{yes,no\}$

Bad Examples

Cross sum:

f(x) : calculate the cross sum of all bytes in the message

Is this secure? Why (not)?

f(5 23) = f(23 5)

CRC:

tag ← CRC(m) ; Verify tag: return CRC(m) == tag

Is this secure? Why (not)?

Integrity requires a secret

Adversary can create new message and recompute CRC

Simple Encryption:

tag ← Enc(k,m) | $_{1,\ldots,6}$ Verify tag: return tag == Enc(k,m) | $_{1,\ldots,6}$

Is this secure? Why (not)?

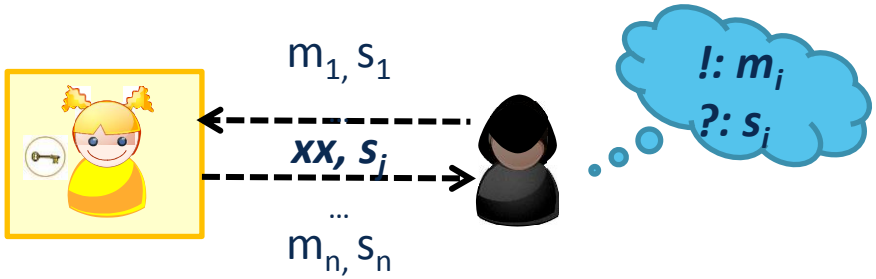
Security depends on |S|

Adversary can guess tag for a message in 2^6

Existential Forgery

Chosen Message Attack:

- given s_1, s_2, … s_n for chosen m_i



Existential Forgery:

Produce ***some*** new valid tuple (m,s) (any message, even gibberish)

⇒ adversary cannot produce a valid tag for a new message

⇒ adversary cannot even produce (m,t') for (m,t) and t' ≠ t

Attack results in general:

Exist. forgery < selective forgery < universal forgery < total break

Interlude: Collision Resistant Hash Functions

Goal:

Map a message of arbitrary length to a characteristic digest (fingerprint)

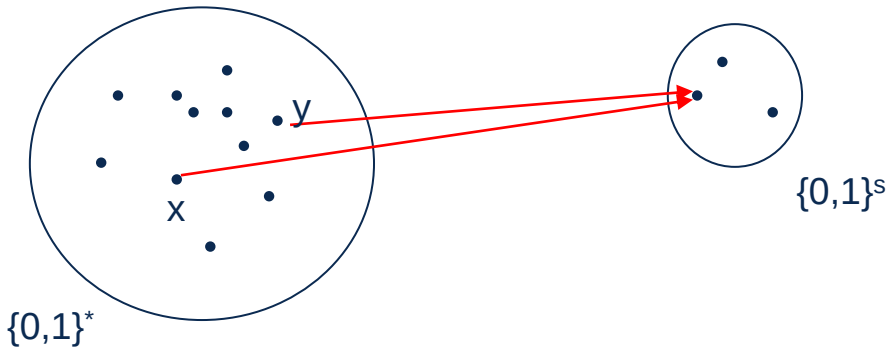
Hash $H: M \rightarrow S$ with $M = \{0,1\}^*$ and $S = \{0,1\}^s$

Cryptographic Hash Functions

Requirements for secure hash functions:

- Efficient algorithm to evaluate *H(x)*
- creates chaos (slight changes in *m* yield large differences in *s*)

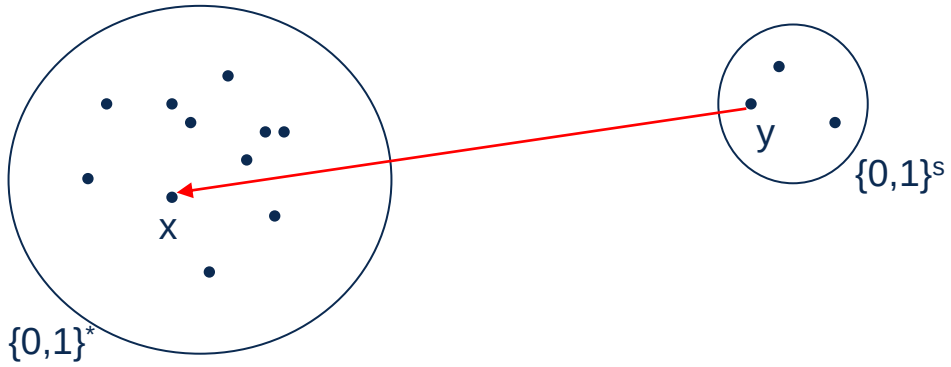
- Has collision resistance



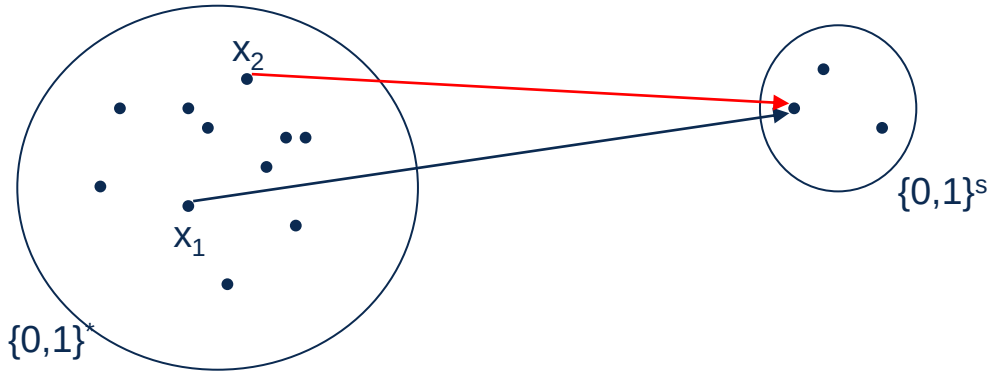
Cryptographic Hash Functions

Further requirements:

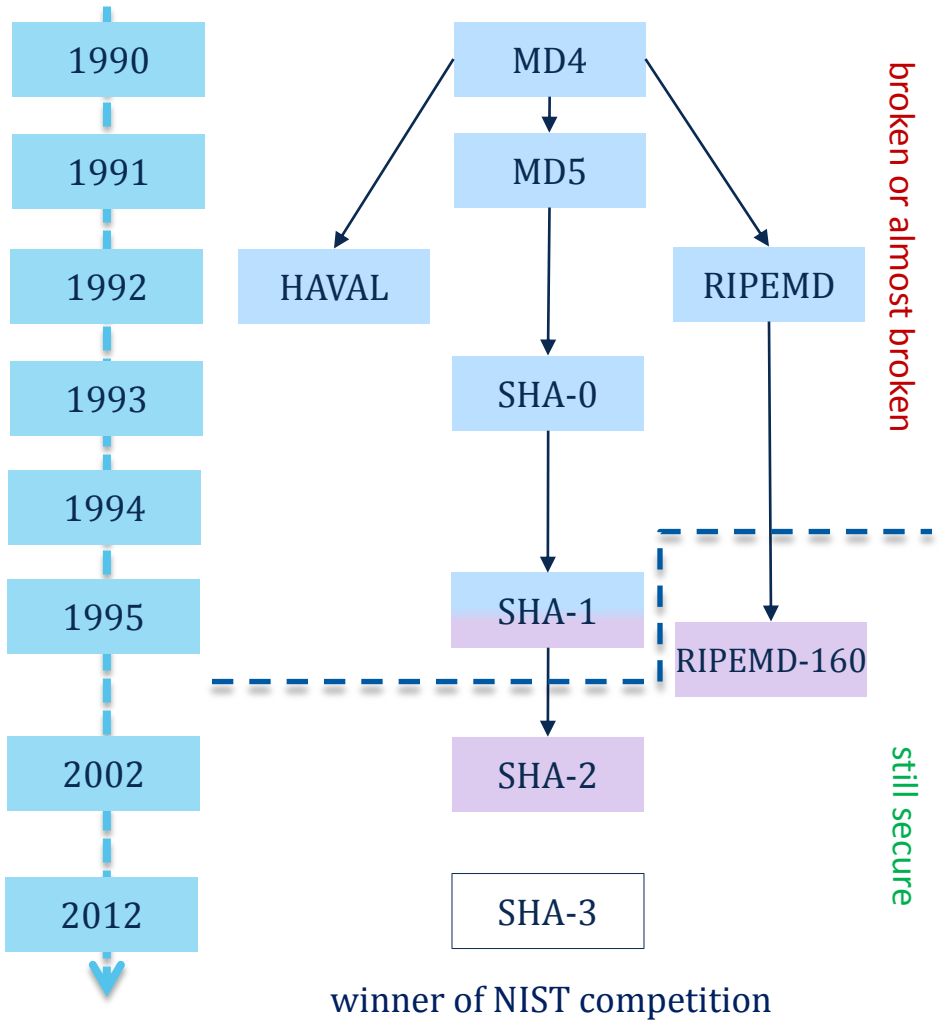
- Pre-image resistance



- 2nd pre-image resistance



A brief history of Hash Functions



MD4 s = 128 bits
collisions in $O(2^8)$, preimages in $O(2^{102})$

MD5 s = 128 bits
collisions in $O(2^{32})$
known colliding documents, certificates

HAVAL s = 128, 160, 192, 224, 256 bits
collisions on HAVAL-128 in $O(2^6)$

RIPEMD s = 128 bits
collisions are known

SHA-0 s = 160 bits
collisions in $O(2^{39})$, replaced by SHA-1 in '95
meanwhile collisions in 1 hour

SHA-1 s = 160 bits
collisions in $O(2^{63})$ – $O(2^{69})$
still secure in practice

SHA-2 supports s = 224, 256, 384, 512 bits

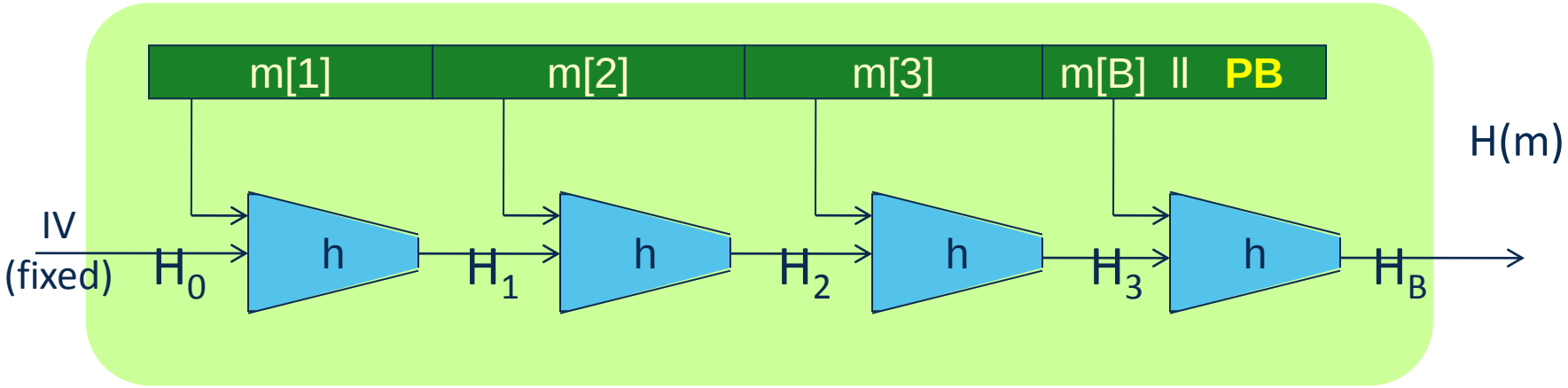
The Merkle-Damgard construction

Given a compression function $h : \{0,1\}^{2s} \longrightarrow \{0,1\}^{s}$ and

Input $m \in \{0,1\}^*$ of length L and PB: = 1000…0 || L

64 bits

Construct H of B= $\lceil L/s \rceil$ iterations of h:



If h is a fixed length CRHF, then H is an arbitrary length CRHF

Proof: either M=M‘, or $H_{B-i}(m[B-i])=H_{B-i}(m'[B-i])$

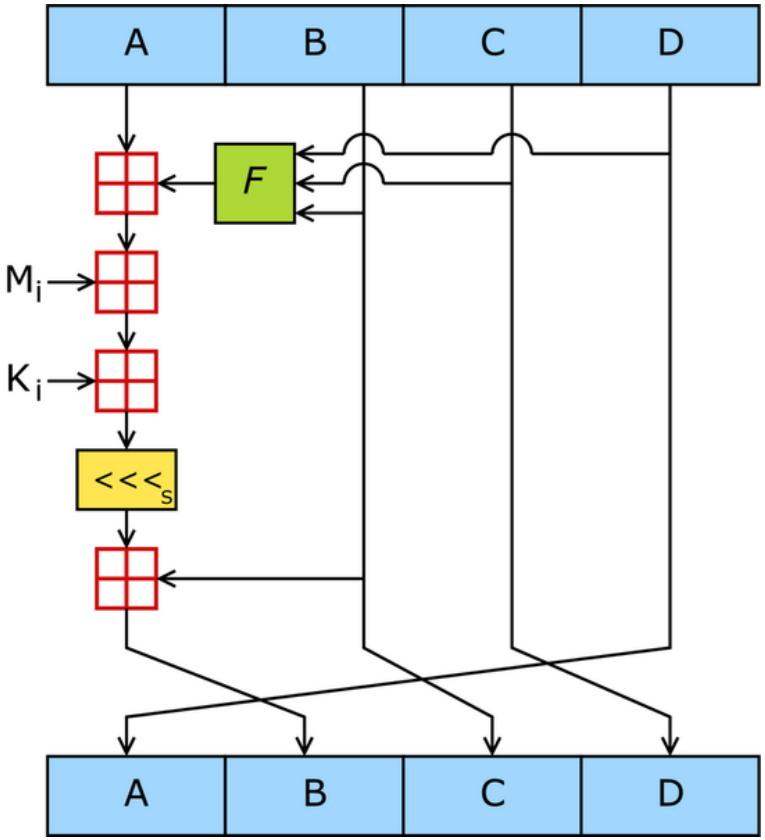
no collision

collision on h

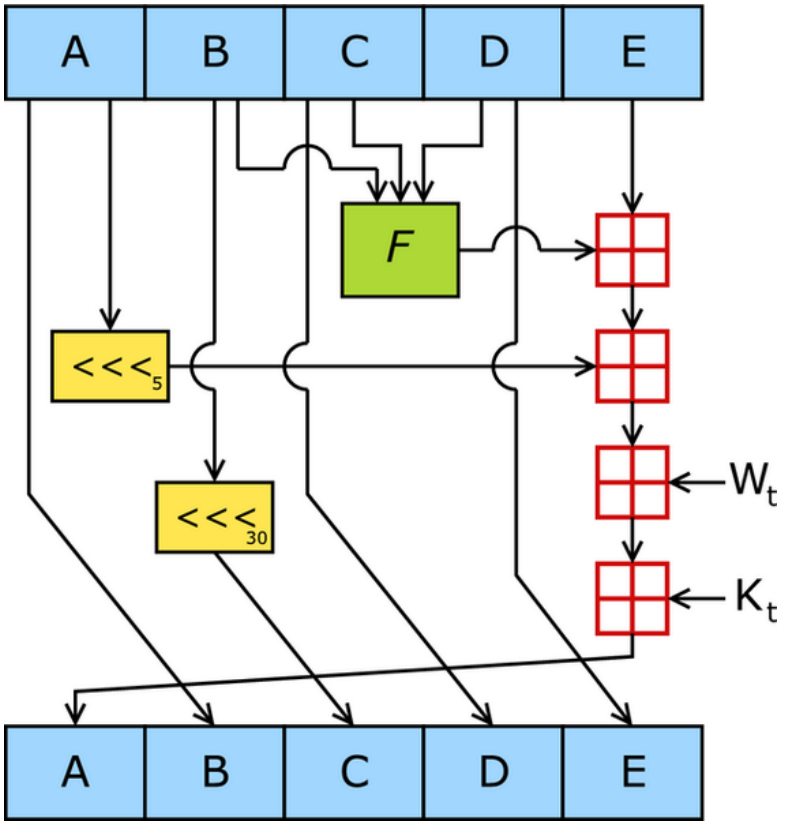
Popular Merkle Damgard constructions

While SHA-2 is also a MD construction, SHA-3 isn't...

Fixed length h (for arbitrary length m, as MD):



MD5 (128 bit)



SHA-1 (160 bit)

Creating secure MACs, a first attempt

MAC: signing alg. **S(k,m)⟶t** and verification alg. **V(k,m,t) ⟶0,1**

First Idea: keyed hash functions, MAC: H(k||m)

Recall: Secure hash is collision resistant

But a secure MAC needs to be ***unforgeable***

Consider Merkle-Damgard construction: s = H(m|| PB)

Feasible chosen message attack:

A – [m] –> C $s = h(k||m||PB)$

C <– [s] – A

$s' = h(m||m') = h(s||m'||PB')$

A – (m||m', s') –> C *and wins the game!*

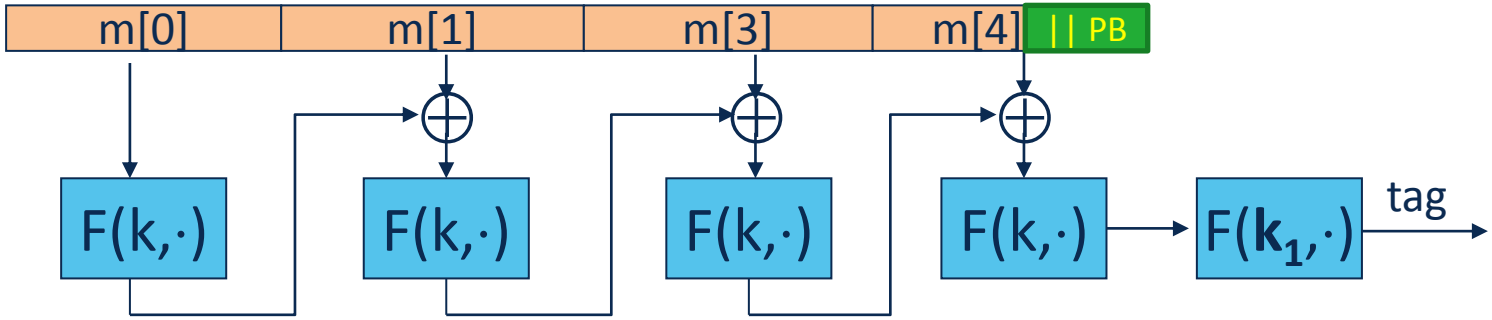
Creating MACs using known PRF/PRPs

Recall chaining of compression functions, block ciphers

Let‘s create a ***MAC*** from AES (a PRP for small messages)!

Let $F: K \times X \longrightarrow X$ be a PRF, define new PRF $F_{ECBC}: K^2 \times X^{\leq L} \longrightarrow X$

CBC-MAC (rawCBC)



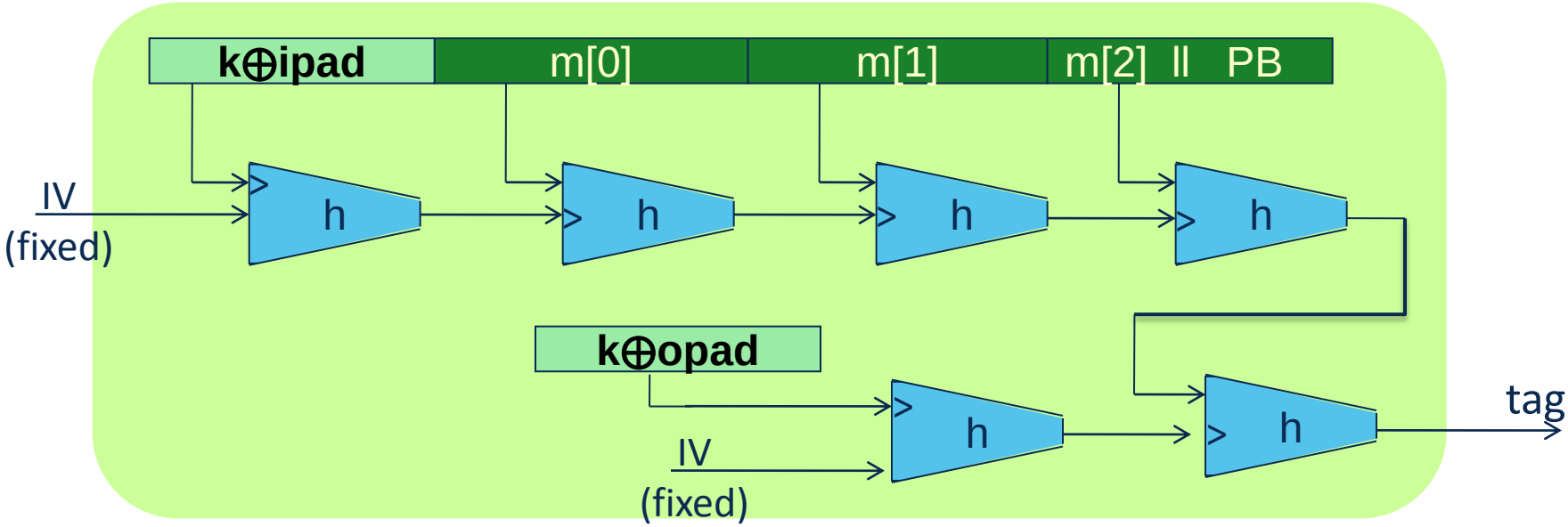
CBC-MAC insecure if |m| ≠ multiple of block size

Solve by padding

The HMAC (RFC 2104)

Hashing is fast, but H(k||m) insecure

Solution: encase message with keys!



HMAC: $S(k, m) = H(\ k \oplus opad \ ||\ \mathbf{H(k \oplus ipad \ ||\ m)}\ )$

(… used in TLS, IPsec,…)

Concluding: Security through MACs

MACs verify integrity of messages

S(k,m) ; V(k,m,t) → secret key must be used, known to verifier

MAC hard to forge without secret key, but *integrity purely mutual*:

- Once key is disclosed, receiver can create arbitrary new tags!
- ⇒ Proof of origin not towards third parties (no non-repudiation!)

But to achieve this, we already know RSA signatures…

Summary

You can explain the goals and ideas of message integrity

You know different adversary and security models for MACs

You have seen different constructions of hash functions

You specifically can explain the Merkle-Damgard construction

You know how to create collisions (and why that‘s bad)

You can construct and explain the details of CBC-MAC and HMAC