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 17/18

Reprise from the last modules

Sie kennen die Unterschiede zwischen Safety und Security (und Datenschutz)

Sie verstehen den Unterschied zwischen Inhalten und Metadaten und den Schutzbedarf von beiden

Sie kennen das primäre Ziel der IT-Sicherheit und operationale Risiken

Sie kennen unterschiedliche Angreifermodelle

Sie wissen, was Identifikation und Authentisierung ist

Sie kennen die unterschiedlichen Aspekte von Authentisierungsschemata

Sie kennen drei Authentisierungsfaktoren und wissen, wie sie implementiert sind

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)

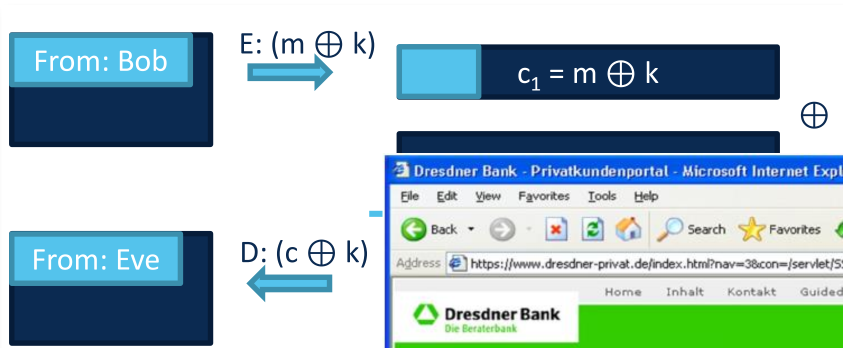
Secure MACs from hash functions and PRFs

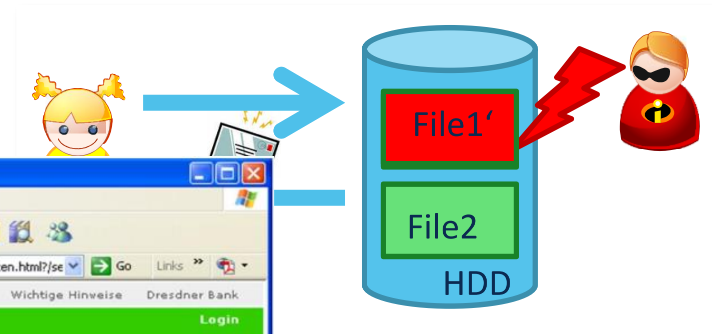
The HMAC

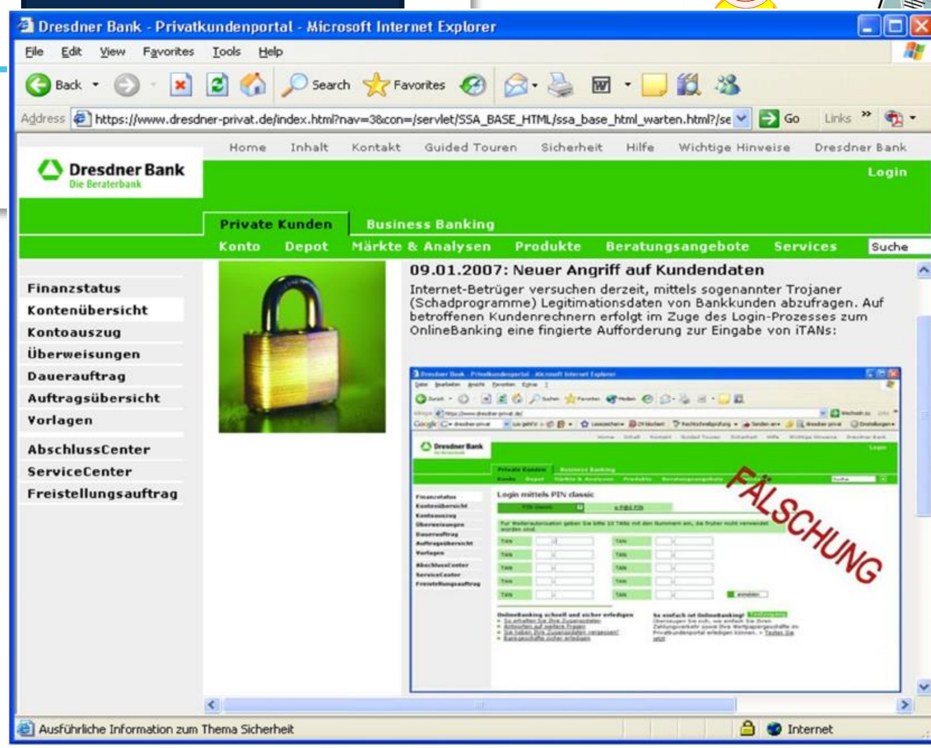
Integrity and Authenticity

Let‘s assume we can keep messages confidential

Integrity of messages not given







Message Integrity

Algorithms:

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

Verify V: M x T → {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) $|_{1,\dots,6}$

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

Is this secure? Why (not)?

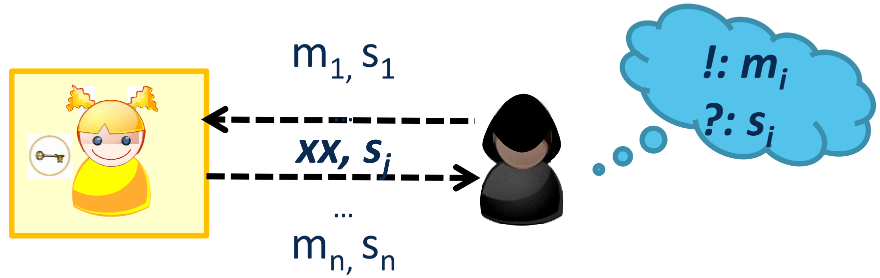
Security depends on |S|

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)

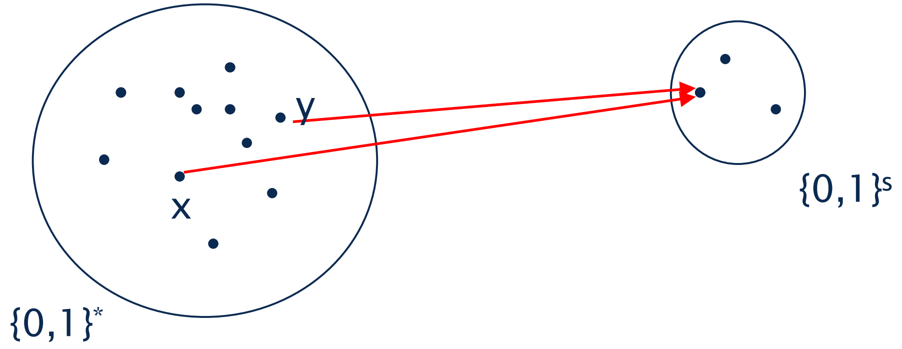
Hash $H: M \rightarrow S$ with $M = \{0,1\}^*$ and $S = \{0,1\}^s$

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*)

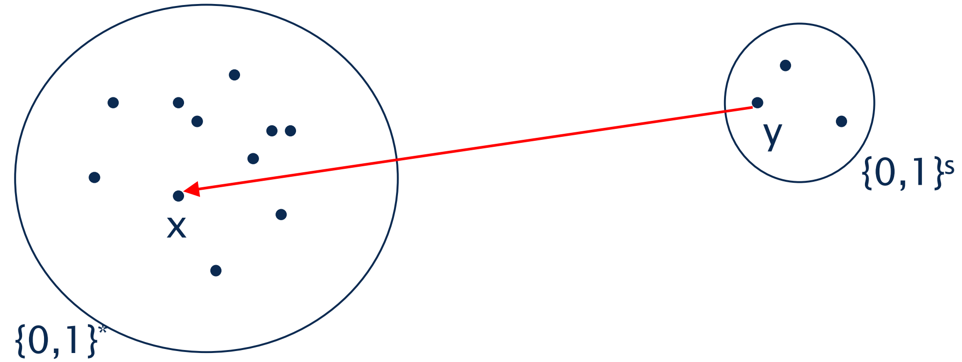
- Has collision resistance



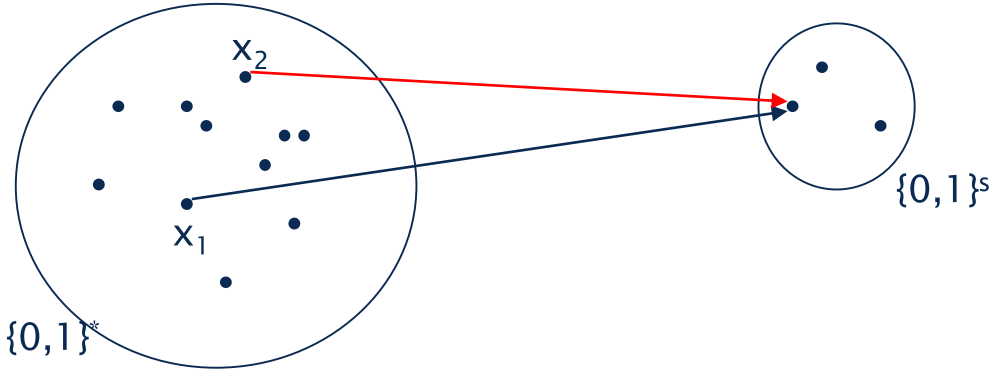
Cryptographic Hash Functions

Further requirements:

- Pre-image resistance



- 2nd pre-image resistance



Revisiting Requirements for Cryptographic h

Cryptographic hash functions:

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

Three core security requirements:

- **Collision** resistance
  - Collision: Given $h(\cdot)$, find m_1, m_2 with $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$
  - $Adv_{CR}(A, h(\cdot))$: $Pr[Collision] \leq min\{|M|,|S|\}^{½}$
    - Common assumption: |M| >> |S| , hence in $O(2^{S/2})$ (due to birthday paradox)

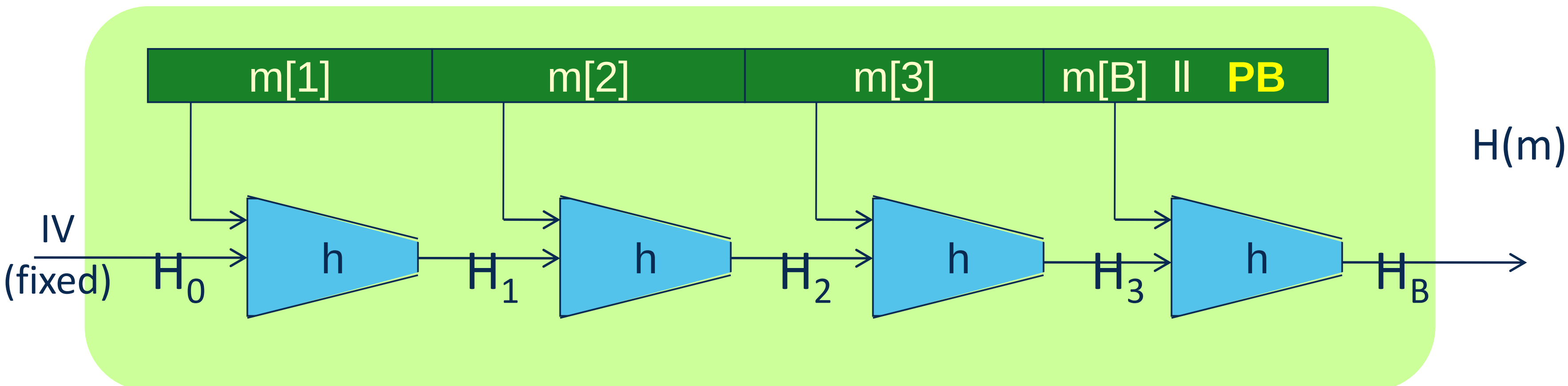
- **Pre-image**, **second pre-image** resistance
  - Pre-image: given $h(m)$ find $h^{-1}(h(m)) = m$
  - 2^{nd} pre-image: given m_1 find m_2 such that $m_1 \neq m_2$ and $h(m_1) = h(m_2)$
  - $Adv_{(S)PR}(A, h(\cdot)) \leq |S| + \varepsilon$, (cf. assumption above, hence $O(2^S)$)

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: = $1000...0 \;||\; \underbrace{\quad L \quad}_{\text{64 bits}}$

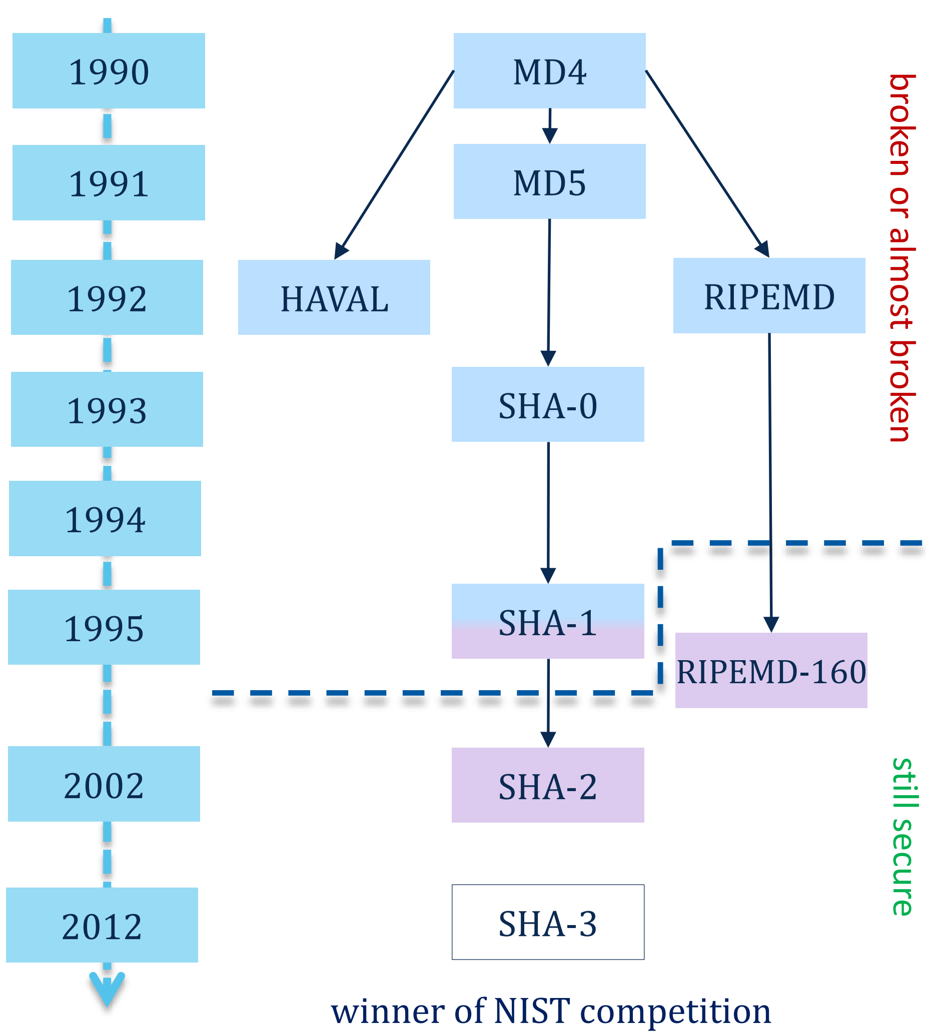
Construct H of B= $\lceil L/s \rceil$ iterations of h:



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

Proof: either $\underbrace{M=M‘,}_{\text{no collision}}$ or $\underbrace{H_{B-i}(m[B-i])=H_{B-i}(m‘[B-i])}_{\text{collision on h}}$

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

End of Interlude, back to MACs!

So can we use these hash functions directly as a MAC?

Quick answer: *no, we need some secret (recall: CRC)!*

Intermediate answer is *„for special cases, yes“:*

Assume a public repository of files

and a ***public read-only space*** with their hashes

package name F_1

package name F_2

…

package name F_n

read-only public space

$H(F_1)$ $H(F_2)$ $H(F_n)$

User can verify validity of the contents of a packet

Adversary cannot forge packets for given $H(F_i)$ (collision!)

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

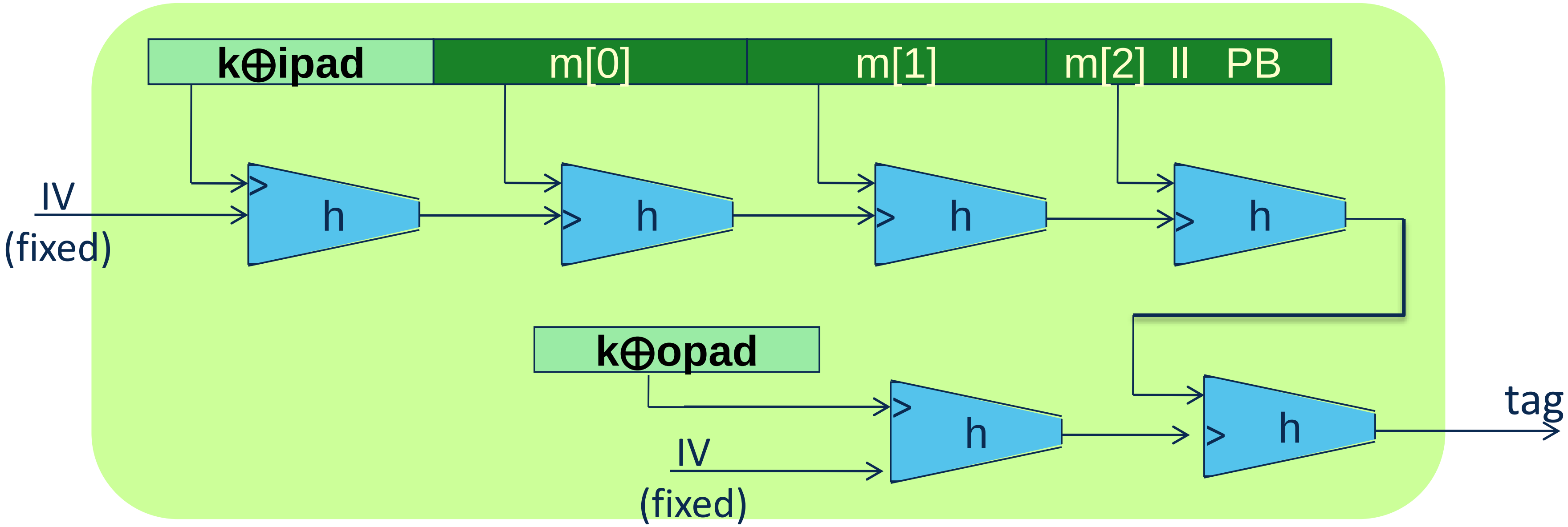
$$s' = h(m||m') = h(s||m'||PB')$$

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

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 know signatures already.....

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

References

Dan Boneh, „Introduction to Cryptography“, course materials, Stanford University, 2016

Stefan Katzenbeisser, „Einführung in Trusted Systems“, teaching materials, TU Darmstadt, 2015

Mark Manulis, „Introduction to Cryptography“, teaching materials, TU Darmstadt, 2011