Security and Cryptography 1

Stefan Köpsell, Thorsten Strufe

Module 6: Integrity

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

Dresden, WS 14/15
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 and some real hash functions (MD5, SHA-1)

Block ciphers as compression functions

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

\[
E: (m \oplus k) \rightarrow c_1 = m \oplus k
\]

\[
D: (c \oplus k) \rightarrow m
\]

File1' HDD

File2
Message Integrity

Algorithms:

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

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

\[ \text{tag} \leftarrow \text{CRC}(m) \]

*Is this secure? Why (not)?*

Adversary can create new message and recompute CRC

**Simple Encryption:**

\[ \text{tag} \leftarrow \text{Enc}(k,m) \mid _{1,...,6} \]

*Is this secure? Why (not)?*

Adversary can guess tag for a message in \(2^6\)
**Existential Forgery**

**Chosen Message Attack:**
- given $s_1, s_2, \ldots, s_n$ for chosen $m_i$

(versions are: known key / known signature attacks)

**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 $t \neq t$
Defining the MAC game

For a MAC $I=(S,V)$ and adversary $A$, where $(S,V)$ additionally take $k$:

- **Chal.** $k \leftarrow K$
- **Adv.**

\[
\begin{align*}
  m_1 & \in M & m_2, \ldots, m_q \\
  t_1 & \leftarrow S(k,m_1) & t_2, \ldots, t_q \\
  (m,t) & \leftarrow A(m_1, t_1, \ldots, m_q, t_q)
\end{align*}
\]

\[
b = \begin{cases} 
  1 & \text{if } V(k,m,t) = \text{`yes'} \text{ and } (m,t) \not\in \{(m_1,t_1), \ldots, (m_q,t_q)\} \\
  0 & \text{otherwise}
\end{cases}
\]

**Def:** $I=(S,V)$ is a **secure MAC** if for all “efficient” $A$:

\[
\text{Adv}_{MAC}[A,I] = \Pr[\text{Chal. outputs } 1] \leq \varepsilon
\]

**Variations:**
- Selective Forgery
- Universal Forgery
- Total Break

So how can we build a secure MAC?
Interlude: Collision Resistant Hash Functions

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

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

• has an efficient algorithm to evaluate H(x)
• is an „onto“ function (surjective, Im(H) = S)
• avoids collision ( maps uniformly to S)
• creates chaos (slight changes in m yield large differences in s)
Cryptographic Hash Functions

Further properties / requirements of hash functions for security:

• Compression is irreversible

• Collision resistance
Cryptographic Hash Functions

Further requirements to hash functions for security:

• Pre-image resistance

\[ \{0,1\} \rightarrow \{0,1\}^s \]

• 2nd pre-image resistance

\[ \{0,1\}^s \rightarrow \{0,1\}^s \]
Let $H: M \rightarrow S$ be a hash function \((|M| >> |S|)\)

A **collision** for $H$ is a pair $m_0, m_1 \in M$ such that:

$$H(m_0) = H(m_1) \text{ and } m_0 \neq m_1$$

A function $H$ is **collision resistant** if for all (explicit) “eff” algs. $A$:

$$\text{Adv}_{\text{CR}}[A,H] = \Pr[A \text{ outputs collision for } H] \leq \epsilon$$
Brute Forcing Collisions

Do collisions exist?

Yes.

but it should be hard to find collisions (in polynomial time)

Trivial Collision-Finder (Brute Force)
compute \( H \triangleq \{H(m) \mid \text{for all } m \in \{0,1\}^s\} \)
if no collision found compute \( H(m^*) \) for any \( m^* \not\in \{0,1\}^s \)
there must be at least one \( m \) with \( H(m) \in H \) such that \( H(m) = H(m^*) \)

\( s \) must be sufficiently large

\( \text{time needed} \quad O(2^s) \)
Generic Attack on C.R. Hash Function

Let \( H : M \rightarrow \{0,1\}^s \) be a hash function \( (|M| >> 2^s) \)

Generic alg. to find a collision in time \( O(2^{s/2}) \) hashes

Algorithm:

1. Choose \( 2^{s/2} \) random messages in \( M: \ m_1, \ldots, m_{2^{s/2}} \) (distinct w.h.p)
2. For \( i = 1, \ldots, 2^{s/2} \) compute \( t_i = H(m_i) \in \{0,1\}^s \)
3. Look for a collision \( (t_i = t_j) \). If not found, got back to step 1.

How well will this work?
The Birthday Paradox

Let \( r_1, \ldots, r_n \in \{1, \ldots, B\} \) be indep. identically distributed integers.

BP states: when \( n = 1.2 \times B^{1/2} \) then \( \Pr[ \exists i \neq j: r_i = r_j ] \geq \frac{1}{2} \)

\( B=10^6 \)
Generic attack due to the B. P.

H: M → \{0,1\}^s . Collision finding algorithm:
1. Choose $2^{s/2}$ random elements in M: $m_1, ..., m_{2^{s/2}}$
2. For $i = 1, ..., 2^{s/2}$ compute $t_i = H(m_i) \in \{0,1\}^s$
3. Look for a collision ($t_i = t_j$). If not found, got back to step 1.

Expected number of iteration $\approx \ 2^{n/2}$

Running time: $O(2^{n/2})$ (space $O(2^{n/2})$)
A brief history of Hash Functions

- **MD4** \( s = 128 \text{ bits} \)
  - collisions in \( O(2^8) \), preimages in \( O(2^{102}) \)

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

- **HAVAL** \( s = 128, 160, 192, 224, 256 \text{ bits} \)
  - collisions on HAVAL-128 in \( O(2^6) \)
  - broken or almost broken

- **RIPEMD** \( s = 128 \text{ bits} \)
  - collisions are known

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

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

- **SHA-2** supports \( s = 224, 256, 384, 512 \text{ bits} \)
  - still secure

- **SHA-3**

winner of NIST competition
The Merkle-Damgard construction

Given a compression function \( h : \{0,1\}^{2s} \rightarrow \{0,1\}^s \) and
Input \( m \in \{0,1\}^* \) of length \( L \) and \( \text{PB: } = 1000...0 \| L \) with 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 \( H_{B-i}(m[B-i])=H_{B-i}(m'[B-i]) \)
Popular Merkle Damgard constructions

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

- **MD5 (128 bit)**
- **SHA-1 (160 bit)**

While SHA-2 is also a MD construction, SHA-3 isn’t...
Can we build hash functions from block ciphers?
- block ciphers $F : \{0,1\}^k \times \{0,1\}^b \rightarrow \{0,1\}^b$ behave like PRPs
- collision-resistance: the outputs of $F$ are close to uniform
- provable security: reduction to the security of a block cipher
- block ciphers used in practice are efficient
- implementation of a block cipher immediate hash function

*but what about the keys?*

However, block ciphers are reversible!
E: $K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a block cipher.

The **Davies-Meyer** compression function: $h(H, m) = E(m, H) \oplus H$

**Thm:** Suppose $E$ is ideal cipher (collection of $|K|$ random perms.). To find collision $h(H, m) = h(H', m')$ takes $O(2^{n/2})$ evaluations of $(E, D)$.

Equivalent to birthday attack $\rightarrow$ as good as possible.
Further variations

Matyas-Meyer-Oseas

$\begin{align*}
m_i \\
\uparrow \\
\downarrow \\
H_{i-1} \\
\rightarrow g \\
\rightarrow E \\
\oplus \\
\rightarrow H_i
\end{align*}$

Miyaguchi-Preneel

$\begin{align*}
m_i \\
\uparrow \\
\downarrow \\
H_{i-1} \\
\rightarrow g \\
\rightarrow E \\
\oplus \\
\rightarrow H_i
\end{align*}$

... and many insecure one’s..
The efficiency of Davies-Meyer compression function in the Merkle-Damgård transformation depends on the efficiency of the chosen block cipher.

*Example*  
- AES-128 has key length $\kappa = 128$ bits, block length $b = 128$ bits  
- AES-192 has key length $\kappa = 192$ bits, block length $b = 128$ bits  
- AES-256 has key length $\kappa = 256$ bits, block length $b = 128$ bits

**Davies-Meyer with AES**

- $|m_i| = 128$ bits  
- $|m_i| = 192$ bits  
- $|m_i| = 256$ bits  
- $|h_{i-1}| = 128$ bits  
- $|h_i| = 128$ bits

*Efficiency*  
hashing of a 5MB file would require at least $156250 \approx 2^{17}$ AES executions

*Security*  
“birthday attack” $O(2^{64})$

*block ciphers with much larger key lengths and block sizes needed...*
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

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. \( S(k, m) \rightarrow t \) and verification alg. \( V(k, m, t) \rightarrow 0, 1 \)

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:

\[
A \rightarrow [m] \rightarrow C \quad s = h(k || m || PB) \\
C \leftarrow [s] - A \\
s' = h(m || m') = h(s || m' || PB') \\
A \rightarrow (m || m', \ s') \rightarrow C \ and \ wins \ the \ game!
\]
Creating secure MACs using a secure PRF

For a PRF \( F: K \times X \rightarrow Y \) define a MAC \( I_F = (S,V) \) as:

- \( S(k,m) := F(k,m) \)
- \( V(k,m,s): \) return \( (s == F(k,m)) \)

Suppose \( F \) is PRF, \( m \) given, what is the chance of the adv. to guess \( s \)?

\[
\text{Adv}_{\text{MAC}}[A,I_F] = \frac{1}{|Y|}
\]
Security of the simple PRF MAC

**Thm:** If $F: K \times X \rightarrow Y$ is a secure PRF and $1/|Y|$ is negligible (i.e. $|Y|$ is large) then $I_F$ is a secure MAC.

For every efficient MAC adversary $A$ attacking $I_F$, there must be an efficient PRF adversary $B$ attacking $F$, s.t.:

$$\text{Adv}_{\text{MAC}}[A, I_F] \leq \text{Adv}_{\text{PRF}}[B, F] + 1/|Y|$$

$\Rightarrow I_F$ is secure as long as:

- $F$ is a secure PRF (given), and
- $|Y|$ is large ($|Y| \geq 2^{80}$)
Creating MACs using known PRF/PRPs

Recall Merkle-Damgård, PRFs, block ciphers & mode of operation
Let’s create a **MAC** from AES (a PRP for small messages)!

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

CBC-MAC (rawCBC)

CBC-MAC insecure if \(|m| \neq \) multiple of block size
Solve by padding (cmp. hash): \( PB = 10...0 \) (Why not “0...0”)?
Nested MAC

Recall Merkle-Damgad, $k$ instead of fixed IV:

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

Cascade:

Why the last encryption with second key?

Otherwise: $\text{cascade}(k, m | | m') = \text{cascade}(\text{cascade}(k,m) | | m')$

(same holds for the CBC-MAC $\rightarrow$ final encryption with $k_1$)
And finally: The HMAC (RFC 2104)

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

Solution: encase message with keys! (Essentially double NMAC)

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

(used in TLS, IPsec,...)
Concluding: Security through MACs

MACs verify integrity of messages
\[ S(k,m) ; V(k,m,t) \rightarrow \text{secret key must be used, known to verifier} \]

MAC hard to forge without secret key, but *integrity purely mutual*:
- Once key is disclosed, receiver can create arbitrary new tags!
- \( \Rightarrow \) Proof of origin not towards third parties (no non-repudiation!)

But to achieve this, we first have to understand keys better...
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, MD5, SHA-1

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

You can explain how to create secure MACs from hash functions and PRFs

You can construct and explain the details of CBC, NMAC, and HMAC