The HMAC construction: A decade later

Ran Canetti
IBM Research
What is HMAC?

- HMAC: A Message Authentication Code based on Cryptographic Hash functions [Bellare-C-Krawczyk96].
- Developed for the IPSec standard of the Internet Engineering Task Force (IETF).
- Currently:
  - incorporated in IPSec, SSL/TLS, SSH, Kerberos, SHTTP, HTTPS, SRTP, MSEC, ...
  - ANSI and NIST standards
  - Used daily by all of us.
Why is HMAC interesting?

- “Theoretical” security analysis impacts the security of real systems.

- Demonstrates the importance of modelling and abstraction in practical cryptography.

- The recent attacks on hash functions highlight the properties of the HMAC design and analysis.

- Use the HMAC lesson to propose requirements for the next cryptographic hash function.
Organization

- Authentication, MACs, Hash-based MACs
- HMAC construction and analysis
- Other uses of HMAC:
  - Pseudo-Random Functions
  - Extractors
- What properties do we want from a “cryptographic hash function”??
The goal: Any tampering with messages should be detected.

“If B accepts message m from A then A has sent m to B.”

- One of the most basic cryptographic tasks
- The basis for any security-conscious interaction over an open network
Elements of authentication

The structure of typical cryptographic solutions:

- **Initial entity authentication:**
  The parties perform an initial exchange, bootstrapping from initial trusted information on each other. The result is a secret key that binds the parties to each other.

- **Message authentication:**
  The parties use the key to authenticate exchanged messages via message authentication codes.
• A and B obtain a common secret key \( K \)
• A and B agree on a keyed function \( F \)
• A sends \( t = F_K(m) \) together with \( m \)
• B gets \( (m', t') \) and accepts \( m' \) if \( t' = F_K(m') \).
Message Authentication Codes:
A definition

The MAC game:
• Key $K$ chosen at random
• An attacker can adaptively ask queries $m$ and get $F_K(m)$.
• $F$ is a good MAC function if the attacker is unable to “predict” $F$, i.e. generate $(m', F_K(m'))$ for an unqueried $m'$.

Definition can be quantified, counting:
- Number and length of queries
- Local computation
- Probability of success.
Message Authentication Codes: A definition

The MAC game:

- Key K chosen at random
- An attacker can adaptively ask queries $m$ and get $F_K(m)$.
- $F$ is a good MAC if the attacker is unable to "predict" $F$, i.e. generate $(m',F_K(m'))$ for an un-queried $m'$.

Definition can be quantified, counting:
- Number and length of queries
- Local computation
- Probability of success.

Note: this is a weaker requirement than pseudorandom functions.
The IP Security effort (1993-)

- An initiative of the Internet Engineering Task Force (IETF)
- Goal: provide a ubiquitous mechanism for securing internet traffic:
  - Common to all Internet traffic
  - Sits in the OS kernel, thus always available (but also hard to deploy and modify)
  - Can be easily used by network components (routers, NAT boxes, firewalls, etc.)
A central challenge in 1995: Find a good Message Authentication Code

Requirements:

• Very fast on a variety of platforms
• Ubiquitously available
• Not susceptible to US export controls
• Secure...
MACs for IPSec: Available options

- DES in CBC-MAC mode:
  - Relatively slow in software
  - Only 64-bit MACs
  - Export controls limit to 40-bit keys

- MACs based on “cryptographic hash functions (CHF)” such as MD5, SHA1, RIPEMD.
  - CHFs are anyway incorporated in most libraries
  - Very fast in software
  - Not susceptible to export controls
  - “Nice” security properties

The choice was clear. But, how to do it securely?
Cryptographic Hash Functions
Basics: The common structure of CHFs

- Iterated applications of a basic element, the “compression function” $h$, using the Merkle-Damgard (“cascade”) structure.
- Initialize via a fixed $s$-bit value IV.

$$H_k(x_1...x_n) = \begin{cases} 
H_{H_k(x_1...x_{n-1})}(x_n) & n > 1 \\ 
h_k(x_1) & n = 1 
\end{cases}$$

$H(x) = H_{IV}(x)$

- $b = 512$
  - MD5: $s=128$
  - SHA1, RIPEMD: $s=160$
Security properties of CHFs

Main design goal was collision resistance:
Infeasible to find \( x, y \) with \( H(x) = H(y) \).

*Theorem [Damgard89]:*
If \( h_k \) is collision resistant on \( b \)-bit inputs, then \( H_k \) is collision resistant for any input length.

But:

- Used in many situations that require different, “ad-hoc” security properties.
- Treated like “magic functions”: Output is assumed to be random and completely uncorrelated with the input.
MACs from CHFs

Main question:
How to incorporate a secret key in a public function?
MACs from CHFs

Main question:
How to incorporate a secret key in a public function?

- Proposal 1- Prepend the key: \[ \text{Prep}_k(m) = H(k|m) \]
  - If \( H \) is a “random function” then Prep is a secure MAC.
  - But, Prep is susceptible to “extension attacks”: let \( |m_1| = |m_2| = b \). Then obtain \( t = \text{Prep}_k(m_1) \), and compute \( \text{Prep}_k(m_1|m_2) = h_t(m_2) \).
  - Still, the proposal was quite popular.
    (“Packet headers always include the length, thus the attack is not practical.”)
Proposal 2 - Append the key:

$$\text{App}_k(m) = H(m|k)$$

- Prevents extension attacks.
- If $h$ is a "random function" then $\text{App}$ is secure MAC.
- But, strongly depends on collisions resistance of $H$.
  (k enters the computation only at the very end.)

Can we do better?
MACs from CHFs

- **Proposal 3** - Prepend and append the key:
  \[
  \text{Env}_k(m) = H(k|m|k) \quad \text{[RFC 1828, Aug95]}
  \]
  - To align or not to align? [Preneel-VanOorschot95]
  - What are the assumptions on H/h?

- **Proposal 4** - Start with Env, and add key-related operations to h  [Preneel-VanOorschot95]

None of the above had sound security analysis...
HMAC
Towards HMAC: The NMAC construction

\[ \text{NMAC}_{k_1,k_2}(m) = H_{k_1}(H_{k_2}(m)) \]

- **Idea 1:** Incorporate the key via the IV. Better for modeling and analysis. Follows the design of the underlying CHF.
- **Idea 2:** Use two independent keys. Indeed, each key has a different role in the analysis.
Performance of NMAC

- Internal application of H: Same as plain hashing of the message
- External application of H: Single run of $h$.

The overhead of the external application is negligible for long messages (packets), and tolerable even for small packets.
Security of NMAC (I)

Approach: reduce to weak properties of h.

Assume an attacker A that breaks NMAC. That is:

- A asks sees NMAC_{k_1,k_2}(m_1), NMAC_{k_1,k_2}(m_2),...
  for adaptively chosen m_1,m_2,...
- A generates m',NMAC_{k_1,k_2}(m') for a new m'.

Then:
- If H_{k_2}(m')=H_{k_2}(m_i) for some i, then A has found a collision in H_{k_2}, with an unknown k_2.
- Else, A managed to “predict” h_{k_1}, without either knowing k_1 nor directly seeing the input.

More precisely...
Weak collision resistance

- **H is weak collision resistant (WCR)** if, given oracle access to $H_k$ for a random $k$, it is infeasible to find $x, y$ such that $H_k(x) = H_k(y)$.

  By itself, equivalent to finding collisions with a *known* random key. (First get $k' = H_k(m)$ for a random $m$, and then find a collision in $H_{k'}()$.)

- **H is very WCR** if, given oracle access to $H_{k_1}(H_{k_2}())$ for a random $k_1, k_2$, it is infeasible to find $x, y$ such that $H_{k_2}(x) = H_{k_2}(y)$. 
Security of NMAC (II)

NMAC is a secure MAC as long as:

- $h_k$ is a secure MAC on $b$-bit messages.
- $H_k$ is very weak collision resistant.

Note: Analysis is quantitatively tight.

- No increase in # queries or running time,
- Adversarial success probability is at most the sum of the assumed success probabilities.
Downsides of NMAC:

- Need to change the IV, thus change existing libraries that include CHFs.
- Key is long (256 or 320 bits).

HMAC gets around these, at the price of an additional mild assumption on h.
The HMAC construction

\[ \text{HMAC}_k(m) = H( k \oplus \text{opad} \mid H(k \oplus \text{ipad} \mid m) ) \]

|k|=s (128 or 160)

opad = 0x36 repeated to make b bits
ipad = 0x5c repeated to make b bits
⊕ is bitwise exclusive or

Note:
- key is short
- keying is only via the input, so no change in existing code.
- Performance: 2 additional applications of h.
Security of HMAC

By reduction to the security of NMAC.

Recall: \( \text{HMAC}_k(m) = H(k \oplus \text{opad} \mid H(k \oplus \text{ipad} \mid m)) \)

\( \text{NMAC}_{k_1,k_2}(m) = H_{k_1}(H_{k_2}(m)) \)

Notice: \( \text{HMAC}_k(m) = \text{NMAC}_{k_1,k_2}(m) \),

where \( k_1 = H(k \oplus \text{opad}), \ k_2 = H(k \oplus \text{ipad}) \).

Thus, assuming that:

\[ G(k) = H(k \oplus \text{opad}), H(k \oplus \text{ipad}) \]

is a pseudorandom generator from \( s \) bits to \( 2s \) bits, we have that HMAC is a MAC function if NMAC is.
Looking back: HMAC as a tradeoff

HMAC is a tradeoff between “theoretical elegance” and practical needs:

- The underlying assumptions on the CHF are not the most “elegant” possible.
- Construction is not the most efficient possible.

But:

- Provides convincing and sound arguments that breaking HMAC would mean a complete break of the CHF.
- Design is simple and does not require change of existing code.
Other uses of HMAC

Once HMAC became readily available, people started to use it in different ways... e.g.:

- **Pseudorandom function (PRF):** for “key expansion”: generate multiple PR keys from a single short key. In IPSec, TLS, SSH, KERBEROS...

- “Collision-resistant PRF”: In TESLA (stream authentication for the MSEC secure multicast standard).

- “Computational randomness extractor”: For deriving pseudo-random keys from somewhat random keying material.

Will talk on the uses as a PRF and an Extractor.
Pseudo-random functions

PRFs are keyed functions that behave like random functions as long as the key is random and secret.

More formally, PRFs are defined via a game:

- Oracle $O$ is fixed to either $F_K$ for a random key $K$, or a random function $R$ with the same domain and range.
- An attacker can adaptively ask queries $m$ and get $O(m)$.
- $F$ is a good PRF if the attacker is unable to tell whether it interacts with $R$ or with $F_K$.

![Diagram of the game](image)
HMAC as a PRF

Fact 1: If the compression function $h_K$ is a PRF on $b$-bit inputs then the cascade $H_K$ is a PRF on variable size inputs, as long as no query is a prefix of another [Bellare-C-Krawczyk97].

Fact 2: If $h_K$ is a PRF on $b$-bit inputs and $H_K$ is Almost Universal (AU) on $v$-size inputs, then $\text{NMAC}_K$ is a PRF on $v$-size inputs [Bellare05]. ($H_K$ is AU if for any $x,y$ $\text{Prob}_K(H_K(x)=H_K(y))$ is negl.)

Fact 3: If $h_K$ is a PRF on $b$-bit inputs then $\text{NMAC}_K$ is AU [Bellare05].

→ If $h_K$ is a PRF on $b$-bit inputs then $\text{NMAC}_K$ is a PRF on $v$-size inputs.
→ If in addition $G(k)=H(k\oplus\text{opad}),H(k\oplus\text{ipad})$ is a PRG then $\text{HMAC}_K$ is a PRF on $v$-size inputs.
The extraction problem

Some key exchange protocols generate “defective keys”:
  • Have much “computational entropy”, but
  • Are not pseudorandom.

Goal: Extract a pseudorandom key.
Main example: Diffie-Hellman exchanges

Public: Algebraic group $G$, generator $g$

Choose $x$ in $[1..|G|]$

Choose $y$ in $[1..|G|]$

Output $(g^x)^y = g^{xy}$
Properties of the generated key ($g^{xy}$)

The Decisional Diffie-Hellman (DDH) assumption implies:

$$(g, g^x, g^y, g^{xy}) \sim (g, g^x, g^y, g^r)$$

But:

- DDH is a strong assumption.
- Even under DDH, $g^{xy}$ is pseudorandom only in the group $G$, which is often embedded in a much larger group (e.g., $Z_p$).
- Even in best case, when $|G|=q, \ p=2q+1$, we only have that $g^{xy}$ is pseudorandom in a small subset of $\{0,1\}^k$.
- When the exchange is not authenticated by external mechanisms (e.g., in the MQV or HMQV protocols) the guarantees are even weaker.
Common practice

Hash using a CHF and hope for the best...

If the CHF is modeled as a random oracle then everything is ok.

But, can we do better?
Randomness extractors

Input:
- A “defective random source”, namely a value drawn from a distribution with substantial entropy,
- A short truly random value.

Output:
- A value that is statistically close to random.

A computational variant [Dodis-Gennaro-Hastad-Krawczyk-Rabin05]:

Input:
- A (secret) value drawn from a distribution with substantial “computational entropy”,
- A (public) truly random value.

Output:
- A (secret) pseudorandom value
HMAC as an extractor

Assume the compression function $h_k$ is a c-extractor from b-bit inputs to s-bit outputs, with an s-bit public random input.

Then:

- The cascade $H_k$ is a c-extractor from $v$-length input to s-bit outputs, as long as each input block has sufficient c-entropy given all subsequent blocks [DGHKR05, CG88].
- NMAC and HMAC behave similarly, when assuming in addition that $h$ is a PRF from s-bits to s-bits with b-bit key.
Using HMAC as an extractor

Applicable when the parties have some trusted public randomness (e.g., the protocol involves exchanging public authenticated random nonces).

Here do: \[ k = \text{HMAC}_r(g^{xy}) \]
where \( r \) is the public randomness (e.g., concatenation of nonces).

\( K \) is guaranteed to be pseudorandom as long as \( g^{xy} \) has enough c-entropy.

- Indeed, HMAC is used this way in IPSec's IKE.
Open question:

What to do when there is no trusted public randomness?

Here the best we know today is to model the CHF as a random oracle.

Can we do better?
HMAC as a Random Oracle

HMAC was designed to get away from unnecessary random oracle modeling.

Still, it turns out that the HMAC/NMAC constructions can be used to extend Random Oracles [Coron-Dodis-Malinaud-Punya05]:

- If $h$ is a random oracle on $b$-bit inputs, then:
  - The cascade $H$ of $h$ is a random oracle on variable-length inputs, as long as queries are prefix-free.
  - The HMAC/NMAC constructions are Random Oracles on variable-length inputs.
Recent attacks on CHFs

The [Wang-Yu-Yin05] collision attacks against MD5 and SHA1 imply:

- Can find collisions in current functions in time $2^{O(60)}$.

- Same approach seems to work for a random, public IV (but needs a “human in the loop” for each new IV).
Implications on HMAC:

- Another reminder that $H$ is not a Random Oracle (and not even $h$).
- Weak collision resistance (with secret IV) is somewhat affected, due to the extension attack.
- Very weak collision resistance does not seem to be affected.
- Neither the PRF nor the MAC assumptions on $h$ seem to be affected.
- The c-extraction assumption on $h$ seems unaffected.

In contrast, other suggestions of hash-based MACs are seriously affected.
Lessons for a new CHF:
Lessons for a new CHF:

- Make the IV part of the interface.
  (OK to fix a single IV for interoperability, but explicitly allow applications to choose their own IV.)
Lessons for a new CHF:

- Make the IV part of the interface.
  (OK to fix a single IV for interoperability, but explicitly allow applications to choose their own IV.)
- The compression function should be designed to be:
  - A PRF when keyed via the chaining variable
  - A PRF when keyed via the input
Lessons for a new CHF:

- Make the IV part of the interface.
  (OK to fix a single IV for interoperability, but explicitly allow applications to choose their own IV.)
- The compression function should be designed to be:
  - A PRF when keyed via the chaining variable
  - A PRF when keyed via the input
- The compression function should be a good extractor
Lessons for a new CHF:

- Make the IV part of the interface.
  (OK to fix a single IV for interoperability, but explicitly allow applications to choose their own IV.)
- The compression function should be designed to be:
  - A PRF when keyed via the chaining variable
  - A PRF when keyed via the input
- The compression function should be a good extractor
- The cascade design is a good one: preserves important properties
Lessons for a new CHF:

• Make the IV part of the interface.
  (OK to fix a single IV for interoperability, but explicitly allow applications to choose their own IV.)
• The compression function should be designed to be:
  • A PRF when keyed via the chaining variable
  • A PRF when keyed via the input
• The compression function should be a good extractor
• The cascade design is a good one: preserves important properties
• Make the output length parameterizable:
  • For collision resistance larger output is easier
  • For PRF, extractor smaller output is easier
Lessons for a new CHF:

- Make the IV part of the interface.
  (OK to fix a single IV for interoperability, but explicitly allow applications to choose their own IV.)
- The compression function should be designed to be:
  - A PRF when keyed via the chaining variable
  - A PRF when keyed via the input
- The compression function should be a good extractor
- The cascade design is a good one: preserves important properties
- Make the output length parameterizable:
  - For collision resistance larger output is easier
  - For PRF, extractor smaller output is easier

Perhaps we want different functions for different applications?
Summary: Why is HMAC interesting?

- An example where “theoretical” security analysis has impact on acceptability and practical security.
- Demonstrates the importance of modeling and abstraction in practical cryptography: Different models of the same construction bring different results, all useful.
- The recent attacks on hash functions highlight the properties of the HMAC design and analysis.
- Can use the HMAC lesson to propose requirements for the next cryptographic hash function.
Basic structure of the IPSec protocol:

- **Key exchange**: Two peers obtain a common secret key in an authenticated way. (Application layer protocol)
- **Data protection**: Encryption and authentication. (IP layer protocol: Each packet encoded and decoded individually.)
- **Per-packet transforms**:
  - Authentication header (AH): Authentication only
  - ESP: Authentication and/or encryption

Seems simple enough. But turns out to be far from that...
IP: the common denominator of the Internet

... Telnet HTTP DNS NTP audio/video ...

TCP UDP

IP

... Ethernet Token Ring ...

...
HMAC as a standard

After much discussion and debate, HMAC was accepted as the mandatory-to-implement MAC function for IPSec (RFC 2104).

- Rare example of a security standard where “theoretical” modeling and analysis has helped acceptance as standard.

Other IETF standards that incorporate HMAC: TLS, SHTTP, SSH, HTTPS, KERBEROS, SRTP,…

NIST standard: FIPS 198
ANSI standard: X9.71

Incorporated in practically any browser and OS today.