Generic Related-key Attacks for HMAC

Thomas Peyrin, Yu Sasaki and Lei Wang

Nanyang Technological University - Singapore
NTT - Japan

Asiacrypt 2012

Beijing, China - December 5, 2012
Outline

Introduction
- What is HMAC
- Current state of HMAC

A generic related-key attack on HMAC
- Distinguish-R attack
- Intermediate internal state recovery
- Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
Outline

Introduction

What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
Outline

Introduction

What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
**HMAC and NMAC (Bellare et al. - 1996)**

A MAC outputs an $n$-bit value from a $k$-bit key $K$ and an arbitrary long message $M$.

\[
\text{NMAC}(K_1, K_2, M) = H(K_2, H(K_1, M))
\]
**HMAC and NMAC (Bellare et al. - 1996)**

A MAC outputs an $n$-bit value from a $k$-bit key $K$ and an arbitrary long message $M$.

\[
\text{HMAC}(K, M) = H(K \oplus \text{opad} \ || \ H(K \oplus \text{ipad} \ || \ M))
\]
HMAC and NMAC (Bellare et al. - 1996)

A MAC outputs an $n$-bit value from a $k$-bit key $K$ and an arbitrary long message $M$.

$$\text{HMAC}(K, M) = H(K \oplus \text{opad} \parallel H(K \oplus \text{ipad} \parallel M))$$
Outline

Introduction
What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
## Known dedicated attacks on HMAC

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Target</th>
<th>Size</th>
<th>#Rounds</th>
<th>Comp.</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>MD4</td>
<td>128</td>
<td>Full</td>
<td>$2^{121.5}$</td>
<td>[KBPH06]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>MD5</td>
<td>128</td>
<td>33/64</td>
<td>$2^{126.1}$</td>
<td>[KBPH06]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>MD5</td>
<td>128</td>
<td>Full</td>
<td>$2^{97}$</td>
<td>[WYWZZ09]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>3p HAVAL</td>
<td>256</td>
<td>Full</td>
<td>$2^{228.6}$</td>
<td>[KBPH06]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>4p HAVAL</td>
<td>256</td>
<td>102/128</td>
<td>$2^{253.9}$</td>
<td>[KBPH06]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>SHA0</td>
<td>160</td>
<td>Full</td>
<td>$2^{109}$</td>
<td>[KBPH06]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>SHA1</td>
<td>160</td>
<td>43/80</td>
<td>$2^{154.9}$</td>
<td>[KBPH06]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single key</td>
<td>SHA1</td>
<td>160</td>
<td>50/80</td>
<td>$2^{153.5}$</td>
<td>[RR08]</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Related Key</td>
<td>SHA1</td>
<td>160</td>
<td>58/80</td>
<td>$2^{158.74}$</td>
<td>[RR08]</td>
</tr>
<tr>
<td>Inner key rec.</td>
<td>Single Key</td>
<td>MD4</td>
<td>128</td>
<td>Full</td>
<td>$2^{63}$</td>
<td>[CY06]</td>
</tr>
<tr>
<td>Inner key rec.</td>
<td>Single Key</td>
<td>SHA0</td>
<td>160</td>
<td>Full</td>
<td>$2^{84}$</td>
<td>[CY06]</td>
</tr>
<tr>
<td>Inner key rec.</td>
<td>Single Key</td>
<td>SHA1</td>
<td>64</td>
<td>34/80</td>
<td>$2^{32}$</td>
<td>[RR08]</td>
</tr>
<tr>
<td>Inner key rec.</td>
<td>Single Key</td>
<td>3p HAVAL</td>
<td>256</td>
<td>Full</td>
<td>$2^{122}$</td>
<td>[LCKSH08]</td>
</tr>
<tr>
<td>Full key rec.</td>
<td>Single Key</td>
<td>MD4</td>
<td>128</td>
<td>Full</td>
<td>$2^{95}$</td>
<td>[FLN07]</td>
</tr>
<tr>
<td>Full key rec.</td>
<td>Single Key</td>
<td>MD4</td>
<td>128</td>
<td>Full</td>
<td>$2^{77}$</td>
<td>[WOK08]</td>
</tr>
</tbody>
</table>
Known generic attacks on HMAC

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Cost Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal forgery</td>
<td>$2^n$ computations (ideal)</td>
</tr>
<tr>
<td>Existential forgery</td>
<td>$2^{l/2}$ computations (not ideal)</td>
</tr>
<tr>
<td>Distinguishing-R</td>
<td>$2^{l/2}$ computations (not ideal)</td>
</tr>
<tr>
<td>Distinguishing-H</td>
<td>$2^l$ computations (ideal)</td>
</tr>
</tbody>
</table>
Known generic attacks on HMAC

Existential forgery attack costs $2^{l/2}$ computations (not ideal)

The procedure

- **step 1:** query $2^{l/2}$ messages and gather all pairs $(M, M')$ that collides on the output
- **step 2:** for all colliding pairs, append an extra random message block $M_1$ and check if this new message pair $(M||M_1, M'||M_1)$ collides as well. Pick one such pair.
- **step 3:** append another extra random message block $M_2$ and query the MAC for message $M||M_2$. Then it is equal to the MAC for message $(M'||M_2)$
Known generic attacks on HMAC

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Generic Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal forgery</td>
<td>Single Key</td>
<td>$2^n$</td>
</tr>
<tr>
<td>Existential forgery</td>
<td>Single Key</td>
<td>$2^{l/2}$</td>
</tr>
<tr>
<td>Dist.-R</td>
<td>Single Key</td>
<td>$2^{l/2}$</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Single Key</td>
<td>$2^l$</td>
</tr>
</tbody>
</table>
### Known generic attacks on HMAC

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Generic Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal forgery</td>
<td>Related Key</td>
<td>$2^n$ ?</td>
</tr>
<tr>
<td>Existential forgery</td>
<td>Related Key</td>
<td>$2^{l/2}$ ?</td>
</tr>
<tr>
<td>Dist.-R</td>
<td>Related Key</td>
<td>$2^{l/2}$ ?</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Related Key</td>
<td>$2^l$ ?</td>
</tr>
</tbody>
</table>
Outline

Introduction

What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
Outline

Introduction
What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
What weakness to attack?

NMAC

\[ M \rightarrow \text{Hash} \rightarrow \text{Hash} \rightarrow \text{NMAC} \]

A generic related-key attack on HMAC
What weakness to attack?

HMAC

A generic related-key attack on HMAC
What weakness to attack?

HMAC
(with key $K$)
What weakness to attack?

HMAC
(with key $K' = K \oplus \text{ipad} \oplus \text{opad}$)
What weakness to attack?

HMAC
(with key $K' = K \oplus \text{ipad} \oplus \text{opad}$)
What to detect?

**HMAC**

*(with key K and arbitrary message)*
What to detect?

HMAC
(with key $K$ and $n$-bit message)
What to detect?

HMAC
(with key $K$ and $n$-bit message)
What to detect?

HMAC

(with $K$ and $K' = K \oplus \text{ipad} \oplus \text{opad}$ and $n$-bit message)
What to detect?

HMAC

(with $K$ and $K' = K \oplus \text{ipad} \oplus \text{opad}$ and $n$-bit message)
What to detect?

Functions $f(g(x))$ and $g(f(x))$ have a particular cycle structure:

there is a 1-to-1 correspondence between cycles of $f(g(x))$ and $g(f(x))$
How to detect the cycle structure?

⇒ by measuring cycles length

The game played (distinguishing-R in the related-key model):
The attacker can query two oracles, $F_K$ and $F_{K'}$, that are instantiated either with $\text{HMAC}_K$ and $\text{HMAC}_{K'}$, or with two independent random functions $R_K$ and $R_{K'}$. He must obtain non-negligible advantage in distinguishing the two cases:

$$\text{Adv}(A) = |\Pr[A(\text{HMAC}_K, \text{HMAC}_{K'}) = 1] - \Pr[A(R_K, R_{K'}) = 1]|$$
The attack

First step (walk A)

Start from an $n$-bit random input message, query $F_K$, and keep querying as new message the MAC just received. Continue so for about $2^{n/2} + 2^{n/2-1}$ queries until getting a collision among the MACs received.

If no collision is found, or if the collision occurred in the $2^{n/2}$ first queries, the attacker outputs 0.
The attack

Second step (walk B)
Do the same for oracle $F_{K'}$. 

\[
\begin{array}{c}
\overset{f}{\longrightarrow} g \quad \overset{\text{HMAC}_K}{\longrightarrow} \quad \overset{\text{HMAC}_{K'}}{\longrightarrow}
\end{array}
\]

At least $2^{n/2}$ elements in the structure

Walk B

Zb
The attack

Third step (colliding walk A and walk B)
If the cycle of walk A has the same length as the one from walk B, then output 1. Otherwise output 0.
The advantage of the attacker is non-negligible and the complexity of the distinguisher is about $2^{n/2} + 2^{n/2-1}$ computations for each of the first and second phase, thus about $2^{n/2+1}$ computations in total.

We implemented and verified the distinguisher. With SHA-2 truncated to 32 bits, we found two walks A and B that have the same cycle length of 79146 elements with $2^{17}$ computations. The best previously known attack for HMAC instantiated with SHA-2 truncated to 32 bits required $2^{128}$ computations.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Target</th>
<th>Old Generic Complexity</th>
<th>New Generic Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist.-R</td>
<td>Related Key</td>
<td>Wide-pipe</td>
<td>$2^{l/2}$</td>
<td>$2^{n/2+1}$</td>
</tr>
</tbody>
</table>
Outline

Introduction

What is HMAC
Current state of HMAC

A generic related-key attack on HMAC

Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
How to recover the intermediate internal state?

We would like to know some of the intermediate internal state of \( \text{HMAC}_K \) and \( \text{HMAC}_{K'} \).

Inside a colliding cycle for \( \text{HMAC}_K \) and \( \text{HMAC}_{K'} \), the input or output queries to \( \text{HMAC}_K \) are intermediate internal state of \( \text{HMAC}_{K'} \) (and vice-versa) ... but we don’t know which one it is, so we need to synchronize the cycles.
Synchronized and Unsynchronized cycles

There are two cases for a collision between walk A and walk B:

- collision in the tail
- collision in the cycle

If the collision happens in the tail, then the cycles are directly synchronized.
Synchronized and Unsynchronized cycles

We just build walk A and walk B with a tail long enough, such that the collision is likely to happen in the tail.

The procedure

- **step 1 (build walk A):** same as before, but just ensure that tail in walk A has size at least \(2^n/2-2\)
- **step 2 (build walk B):** same as step 1, but with queries to \(K' = K \oplus \text{ipad} \oplus \text{opad}\)
- **step 3:** check if the cycle have the same length, and if so, there is a good chance that it happened in the tail. Then you can recover the intermediate internal states.
Results - internal state recovery for HMAC

The complexity of the internal state recovery is about $2^{n/2+2}$ queries and $2^{l-n+1}$ computations in total.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Target</th>
<th>Old Generic Complexity</th>
<th>New Generic Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist.-R</td>
<td>Related Key</td>
<td>Wide-pipe</td>
<td>$2^{l/2}$</td>
<td>$2^{n/2+1}$</td>
</tr>
<tr>
<td>Inner state rec.</td>
<td>Related Key</td>
<td>Narrow or Wide</td>
<td>$2^n$</td>
<td>$2^{n/2+2} + 2^{l-n+1}$</td>
</tr>
</tbody>
</table>
Outline

Introduction
What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
Existential forgery and distinguish-H attack

- once we have recovered an internal state, forging a valid MAC is easy
- if we can recover an internal state, then distinguish-H is easy

The complexity to forge a valid MAC or distinguish-H is the complexity of the internal state recovery

\((2^{n/2} + 2^{l-n} + 1)\) computations

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Target</th>
<th>Old Generic Complexity</th>
<th>New Generic Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist.-R</td>
<td>Related Key</td>
<td>Wide-pipe</td>
<td>2^{l/2}</td>
<td>2^{n/2+1}</td>
</tr>
<tr>
<td>Inner state rec.</td>
<td>Related Key</td>
<td>Narrow or Wide</td>
<td>2^n</td>
<td>2^{n/2+2} + 2^{l-n+1}</td>
</tr>
<tr>
<td>Ex. forgery</td>
<td>Related Key</td>
<td>Wide-pipe</td>
<td>2^{l/2}</td>
<td>2^{n/2+2} + 2^{l-n+1}</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Related Key</td>
<td>Narrow or Wide</td>
<td>2^l</td>
<td>2^{n/2+2} + 2^{l-n+1}</td>
</tr>
</tbody>
</table>
Outline

Introduction

What is HMAC
Current state of HMAC

A generic related-key attack on HMAC
Distinguish-R attack
Intermediate internal state recovery
Existential forgery and distinguish-H attack

Patching HMAC and Conclusion
Our results

Our attacks on HMAC work when the key has length $m$, or $m - 1$ because $ipad = 0x3636 \cdots 36$ and $opad = 0x5C5C \cdots 5C$

$\implies$ The choice of $ipad$ and $opad$ was in fact important

<table>
<thead>
<tr>
<th>Attack</th>
<th>Key Setting</th>
<th>Target</th>
<th>Old Generic Complexity</th>
<th>New Generic Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist.-R</td>
<td>Related Key</td>
<td>Wide-pipe</td>
<td>$2^{l/2}$</td>
<td>$2^{n/2} + 1$</td>
</tr>
<tr>
<td>Inner state rec.</td>
<td>Related Key</td>
<td>Narrow or Wide</td>
<td>$2^n$</td>
<td>$2^{n/2+2} + 2^{l-n+1}$</td>
</tr>
<tr>
<td>Ex. forgery</td>
<td>Related Key</td>
<td>Wide-pipe</td>
<td>$2^{l/2}$</td>
<td>$2^{n/2+2} + 2^{l-n+1}$</td>
</tr>
<tr>
<td>Dist.-H</td>
<td>Related Key</td>
<td>Narrow or Wide</td>
<td>$2^l$</td>
<td>$2^{n/2+2} + 2^{l-n+1}$</td>
</tr>
</tbody>
</table>
Patching HMAC

1st try:
We use a different IV for the hash function in the inner and outer call ...
... but that would require to change the $H$ definition and implementations

2nd try:
We truncate the HMAC output ...
... but having a smaller output reduces the expected security

Our solution:
Just prepend a "0" bit to the message $M$:

- no more possible for the attacker to synchronize the computation chains: the inner and outer function are made distinct
- no need to change the specification of $H$, even better: can be done on top of HMAC implementations
- almost zero performance drop
Thank you for your attention!