Multi-Property-Preserving Hash Domain Extension: The EMD Transform

Mihir Bellare and Thomas Ristenpart
UCSD Security and Cryptography Lab

NIST Second Cryptographic Hash Workshop
August, 2006

To appear in Asiacrypt 2006
Expanding utility of hash functions

In the beginning, hash functions were designed for use in digital signature schemes...

then used heuristically to instantiate random oracles...

and hash functions were keyed to build message authentication codes...

and now-a-days get used for numerous disparate applications.
Hash functions are *used* like “Swiss Army Knives”

Whether hash function designers like it or not, hash functions are (and will continue to be) used in numerous different ways.

So what should hash function designers do?

Design hash functions to be like “Swiss Army Knives”
The goal:

Build hash functions to be secure for as many applications as possible
Current design paradigm insufficient

1) Compression function

2) Compression function is iterated using MD w/ strengthening

All in-use hash functions use MD w/ str.

because:

\[ f \text{ is CR} \Rightarrow H^f \text{ is CR} \]

But CR does not support usage for many settings!
Building stronger hash functions

• Point out limitations of a natural approach for designing strong hash functions, due to [CDMP05]

• Introduce a new design approach which utilizes multi-property-preserving (MPP) transforms

• Describe a provably-secure MPP transform, EMD, which can be used to build “Swiss Army Knives”
A newer approach

[CDMP05] introduced new design paradigm for hash functions:

1) Assume compression function is a random oracle (RO)

2) Build domain extension transform $H$ such that:

$$ f \approx \text{RO} \Rightarrow H^f \approx \text{RO} $$

Great benefit: directly supports usage of hash functions for instantiating random oracles by fixing a previously-existing gap

4 transforms: [CDMP05] give transforms to enable this approach
A newer approach

[CDMP05] introduced new design paradigm for hash functions:

1) Assume compression function is a random oracle (RO)

2) Build domain extension transform $H$ such that:

$$f \approx \text{RO} \implies H^f \approx \text{RO}$$

“PRO”

Behaving like a RO seems very strong...

is this all we need to build “Swiss Army Knives”?

No, security guarantees worse for most applications!
Limitations of PRO-Pr approach

PRO-Pr approach: $f \approx RO \Rightarrow H^f \approx RO$

Resulting hash function is trivially CR, easily keyed to become PRF, etc....

PRO-Pr approach great for building hash functions to use for instantiating RO’s. What about other settings?
Limitations of PRO-Pr approach

PRO-Pr approach

\[
\begin{align*}
\text{But: only under assumption that } f \text{ is a PRO, which it is provably not! [CGH04]} \\
\text{PRO-Pr, by itself, gives worse guarantee for standard model properties!}
\end{align*}
\]

Resulting hash function is trivially CR, easily keyed to become PRF, etc....
Limitations of PRO-Pr approach

Hash functions built using $H$ that is only PRO-Pr give **worse** security guarantee than MD w/str

**PRO-Pr**

approach

But: only under assumption that $f$ is a PRO, which it is *provably* not! [CGH04]

$f \approx RO \implies H^f \approx RO$

Resulting hash function is trivially CR, easily keyed to become PRF, etc....

compared to...

**MD w/str**

approach

$f$ is CR $\implies H^f$ is CR
Limitations of PRO-Pr approach (cont.)

(Free) Translation: the [CDMP05] design approach results in hash functions which have **worse** security guarantees for applications beyond **instantiating a RO**

In fact: the 4 proposed transforms in [CDMP05] do **not** give guarantees for **CR** and (3 of the them) do **not** give guarantees for being a **PRF** (under standard assumptions)

The problem is focusing only on **PRO-Pr**, and not explicitly including more standard preservation goals
Our approach: use MPP transforms

1) Construct compression function that is \textbf{CR}, “behaves like a \textbf{RO}”, and is a good \textbf{PRF} (when keyed)

2) Build domain extension transform \( H \) such that:

\[
\begin{align*}
\text{f is CR} & \Rightarrow H^f \text{ is CR} & \text{(CR-Pr)} \\
\text{f \approx RO} & \Rightarrow H^f \approx \text{RO} & \text{(PRO-Pr)} \\
\text{f is a PRF} & \Rightarrow H^f \text{ is a PRF} & \text{(PRF-Pr)}
\end{align*}
\]

We call \( H \) a \textit{multi-property-preserving} (MPP)

Note that we include \textbf{PRO-Pr}, because it’s important for instantiating ROs.
MPP approach results in “Swiss Army Knife”

Build a single hash function $H^f$ via the MPP approach and...

<table>
<thead>
<tr>
<th>Usage</th>
<th>Assumption on $f$</th>
<th>Hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>digital signatures</td>
<td>collision-resistance</td>
<td>$H^f$</td>
</tr>
<tr>
<td>instantiating RO’s</td>
<td>“behaves like a RO”</td>
<td>$H^f$</td>
</tr>
<tr>
<td>message</td>
<td></td>
<td>$H^f$</td>
</tr>
<tr>
<td>authentication, key derivation</td>
<td></td>
<td>$H^f$</td>
</tr>
</tbody>
</table>

Minimal set of properties ... perhaps more?
Building an MPP transform

Unfortunately, the [CDMP05] transforms, as specified, are not MPP:

Prefix-free MD: specific prefix-free encodings give **CR-Pr**, and all prefix-free encodings give **PRF-Pr** [BCK96], but has other drawbacks (as described in [CDMP05])

Other 3 transforms: omit strengthening, not **CR-Pr**, and unclear whether **PRF-Pr**

Instead of these...build a new transform that combines techniques for preserving **CR, PRO, and PRF**
The EMD transform

Similar to NMAC in design

Provably...

\[ CR-Pr \quad PRO-Pr \quad PRF-Pr \]

Slightly more efficient than [CDMP05] transforms
<table>
<thead>
<tr>
<th>Transform</th>
<th>CR-Pr</th>
<th>PRO-Pr</th>
<th>PRF-Pr</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain MD</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>[M89,D89]</td>
</tr>
<tr>
<td>Strengthened MD</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>[M89,D89]</td>
</tr>
<tr>
<td>Prefix-free</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>[CDMP05]</td>
</tr>
<tr>
<td>Chop solution</td>
<td>✗</td>
<td>✓</td>
<td>?</td>
<td>[CDMP05]</td>
</tr>
<tr>
<td>HMAC construction</td>
<td>✗</td>
<td>✓</td>
<td>?</td>
<td>[CDMP05]</td>
</tr>
<tr>
<td>NMAC construction</td>
<td>✗</td>
<td>✓</td>
<td>?</td>
<td>[CDMP05]</td>
</tr>
<tr>
<td>EMD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[BeRi06]</td>
</tr>
</tbody>
</table>
Summary

• Motivated developing stronger hash functions, with **broader** security goals

• Pointed out insufficiency of [CDMP05] approach for building stronger hash functions

• Proposed the **multi-property-preserving** approach

• Introduced a proven MPP transform, EMD
Thank you!

Before

After

tristenp@cs.ucsd.edu