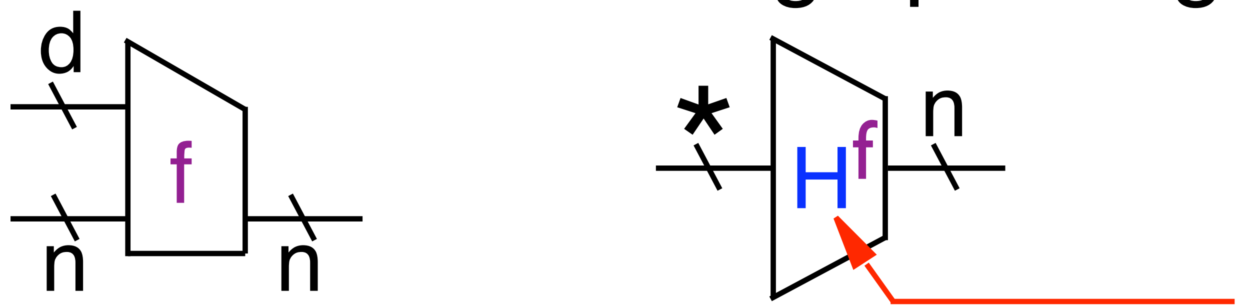
Multi-Property-Preserving Hash Domain Extension and the EMD Transform (Enveloped Merkle-Damgård)

Mihir Bellare and Thomas Ristenpart
University of California at San Diego
Security and Cryptography Lab

Full version appears on ePrint

Current hash function design paradigm



Compression function

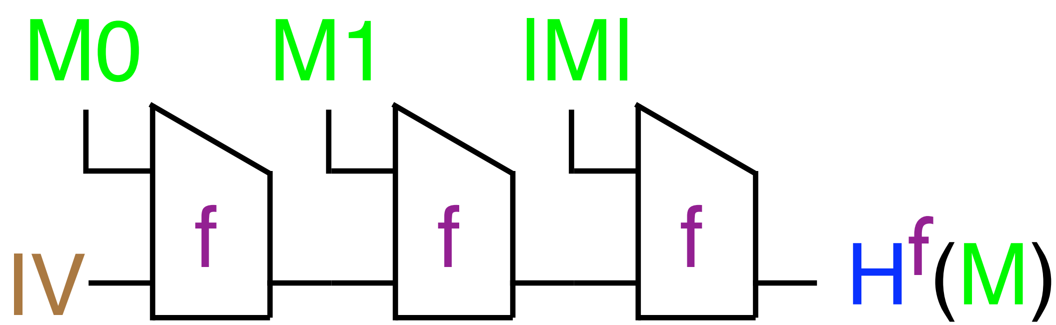
Hash function

Domain extension transform

One wants a transform H that is collision-resistance preserving (**CR-Pr**):

f is **CR** $\Rightarrow$ H^f is **CR**

E.g. H = MD_+ (Merkle-Damgård w/str)



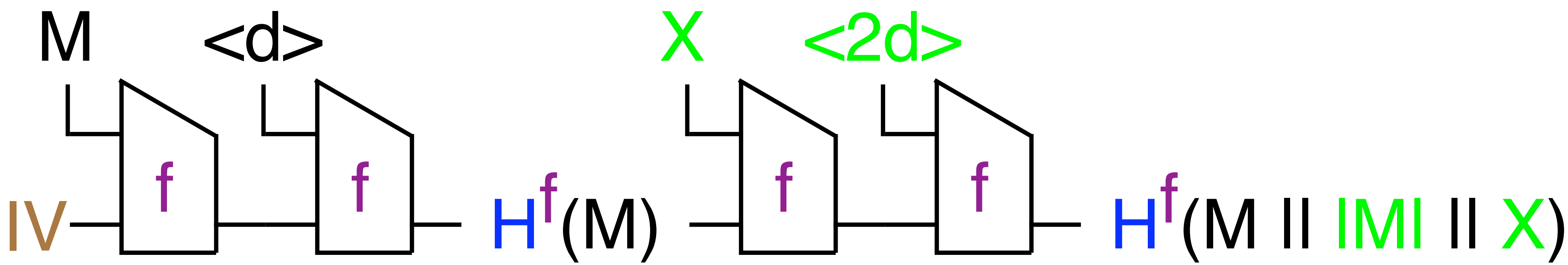
Used in MD4, MD5, SHA-1, SHA-256, etc.

Extension attack

Let H = MD_+ and message M unknown to adversary

$$X, |M|, H^f(M) \xrightarrow{\text{easy}} H^f(M \,||\, |M| \,||\, X)$$

e.g. if $|X| = |M| = d$, then:



So what?

- Does not affect **CR**
- But means that H^f does not "behave like" a RO

Extension attack

Let $H = MD_+$ and message M unknown to adversary

$$X, |M|, H^f(M) \xrightarrow{\text{easy}} H^f(M \,\|\, |M| \,\|\, X)$$

$$X, |M|, RO(M) \xrightarrow{\text{hard}} RO(M \,\|\, |M| \,\|\, X)$$

So what?

- Does not affect **CR**
- But means that H^f does not “behave like” a RO
  This is true even if f is a RO.

[CDMP05]:

- Hash functions widely used as ROs
  e.g. RSA-OAEP [BR94], RSA-PSS [BR96]
  used in PKCS#1 v2.1

- Should (minimally) validate this use
  assuming compression function f is a RO

To that end they ask for domain extension transforms H which are (what we call) pseudo-random-oracle preserving (**PRO-Pr**):

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

indifferentiable
[MRH04]

[CDMP05]:

- Hash functions widely used as ROs
  e.g. RSA-OAEP [BR94], RSA-PSS [BR96]
  used in PKCS#1 v2.1

- Should (minimally) validate this use
  assuming compression function f is a RO

To that end they ask for domain extension transforms H which are (what we call) pseudo-random-oracle preserving (**PRO-Pr**):

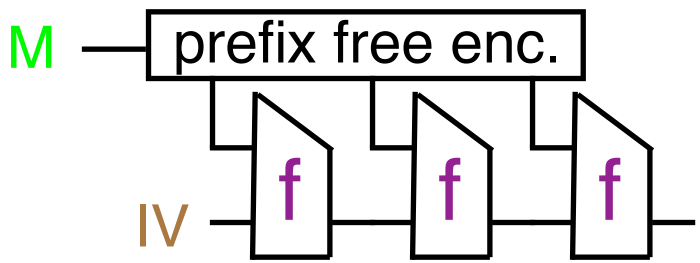
$$f \approx RO \Rightarrow H^{f} \approx RO$$

indifferentiable
[MRH04]

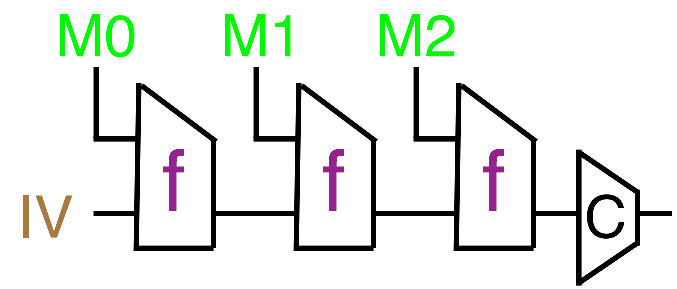
PRO's only exist in the random oracle model

$H = MD_+$ is <u>not</u> **PRO-Pr** (due to extension attack)

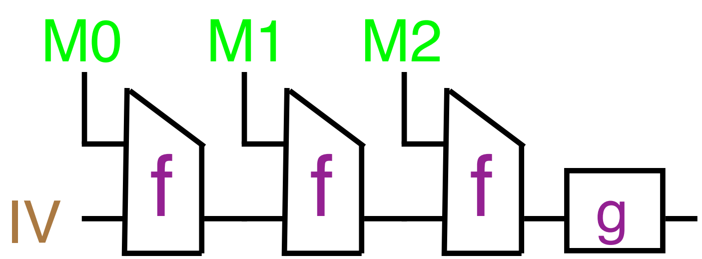
[CDMP05] present several new **PRO-Pr** transforms:



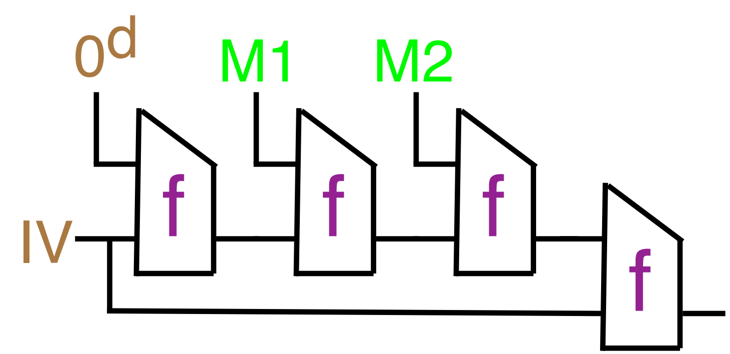
Prefix-free MD



Chop transform



NMAC construction



HMAC construction

PRO-Pr is a desirable property: Important for usage of hash functions as ROs.

But, there is also danger in using **PRO-Pr** transforms...

The same hash functions will be used both as ROs and (just) as **CR** functions.

Will **PRO-Pr** transforms yield CR hash functions?

It might *seem* so:

$$f = \mathrm{RO} \;\Rightarrow\; H^f \approx \mathrm{RO} \;\Rightarrow\; H^f \text{ is } \mathbf{CR}$$

Problem!

PRO-Pr

When f is a real compression function, then

- $f \neq \mathrm{RO}$
- so above does not justify that H^f is **CR**

The problem is real

For each of 4 **PRO-Pr** transforms H proposed in [CDMP05] we show that:

$\exists$f such that

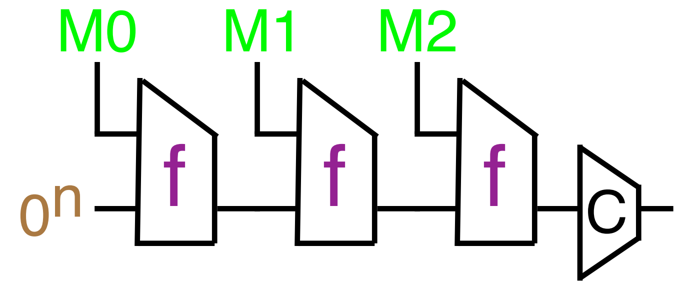
f is **CR** but H^f is not **CR**

In other words

$$\textbf{PRO-Pr} \not\Rightarrow \textbf{CR-Pr}$$

Example: H = chop transform

C outputs first n-s bits of its n bit input

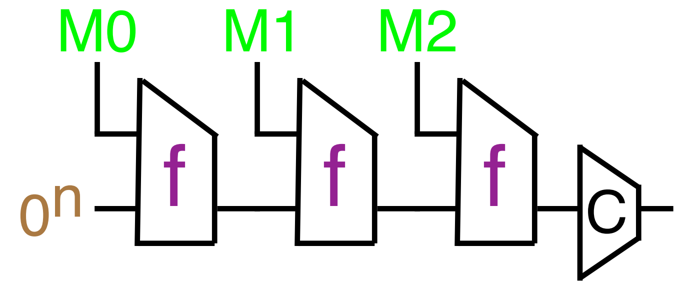


We build a **CR** compression function f for which H^f is not **CR**.

$$\text{Let } f(x) = \begin{cases} 0^n & \text{if } x = 0^{n+d} \\ h(x) \,||\, 1 & \text{otherwise} \end{cases}$$

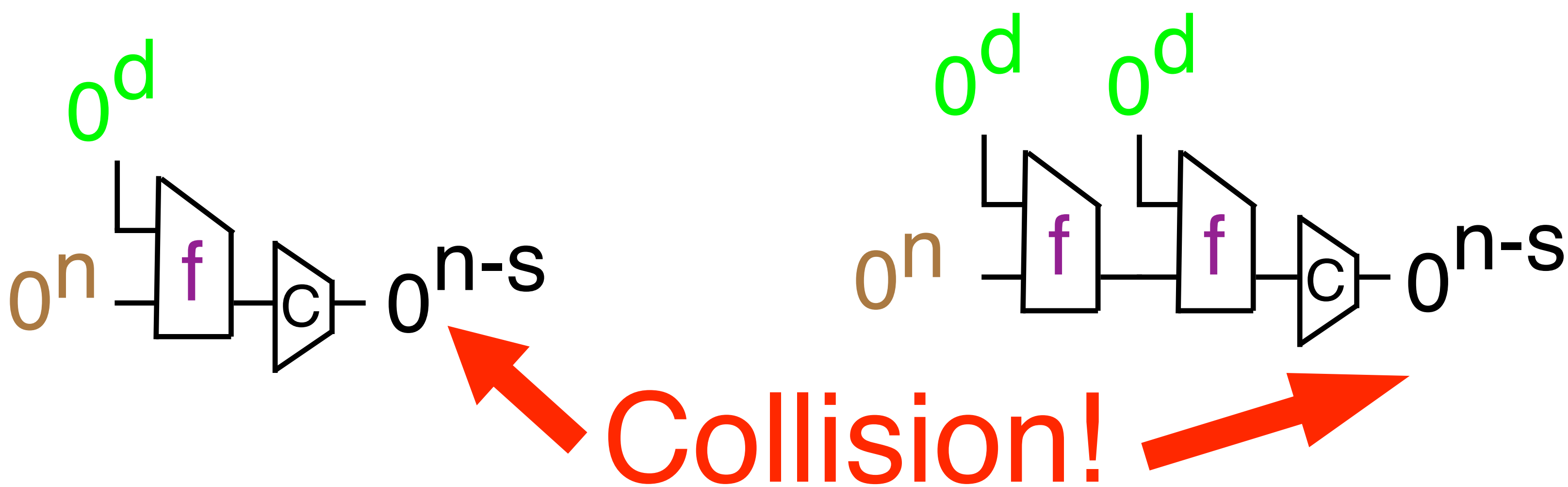
Claim 1: f is **CR** (assuming h is **CR**)

Example: H = chop transform



C outputs first n-s bits of its n bit input

We build a **CR** compression function f for which H^f is not **CR**.



Claim 2: H^f is not **CR**

What this means

For **CR**, guarantee of transforms from [CDMP05] is worse than that of MD_+

Root of problem:

PRO-Pr provides guarantee of security *only in the model* where f = RO.
No guarantee in the standard model!

This speaks against standardizing any of the [CDMP05] transforms

PRO-Pr in review...

+ Important for building hash functions used as ROs

– Does not guarantee H^f is **CR** when f is **CR**

So what types of transforms should we use?

Preserve both **CR** and **PRO**

Natural solution is to require H to be both

1. **CR-Pr** f is **CR** $\Rightarrow H^f$ is **CR**
2. **PRO-Pr** $f = \mathrm{RO} \Rightarrow H^f \approx \mathrm{RO}$

Solves the previous problems with (only) **PRO-Pr** transforms: single hash function good for both uses.

	Random oracles	*Digital signatures*
H is **PRO-Pr**, **CR-Pr**	Alice $\rightleftarrows$ $H^f(.)$ H^f secure if f = RO	Sign($H^f(M)$) H^f secure if f is **CR**
H is just **PRO-Pr**	Alice $\rightleftarrows$ $H^f(.)$ H^f secure if f = RO	Sign($H^f(M)$) H^f secure if f = RO

One can “patch” the [CDMP05] transforms to get them to be both **CR-Pr** and **PRO-Pr**: add strengthening!

but...

Hash functions have all kinds of applications:

CR functions

random oracles

message authentication

key derivation

near-collision resistant functions

one-way functions

others...

Want security guarantees for as many settings as possible

Two very important uses:

message authentication codes (MACs)

key derivation

These require that hash functions be keyed and are good **PRF**s. Does a **CR-Pr**, **PRO-Pr** H suffice?

PRO-Pr transforms again *seem* sufficient:

secret key K

$$f = RO \Rightarrow H^f \approx RO \Rightarrow H^f(K \,||\, .) \text{ is } \mathbf{PRF}$$

Problem!

PRO-Pr

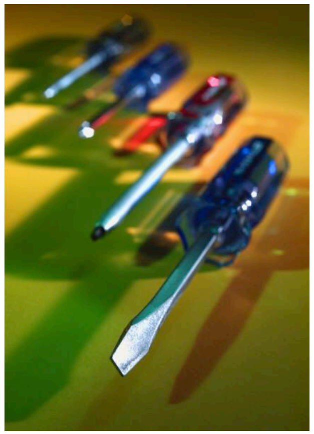
But as before, no guarantee for a real f.

Solution: use multi-property-preserving (MPP) transforms, which simultaneously preserve all properties of interest.

Minimally, we suggest building a single transform H that is simultaneously

1) **CR-Pr** f is **CR** $\Rightarrow$ H^f is **CR**
2) **PRO-Pr** $f = RO \Rightarrow H^f \approx RO$
3) **PRF-Pr** f is **PRF** $\Rightarrow$ H^f is **PRF**

Current situation



Transform	Security	Example Applications
MD w/str	**CR-Pr**	digital signatures
[CDMP05]	**PRO-Pr**	ROs
HMAC/NMAC	**PRF-Pr**	PRF/MAC

Even if one f, must build many hash functions:

- Standardize many hash functions
- Complicates implementations

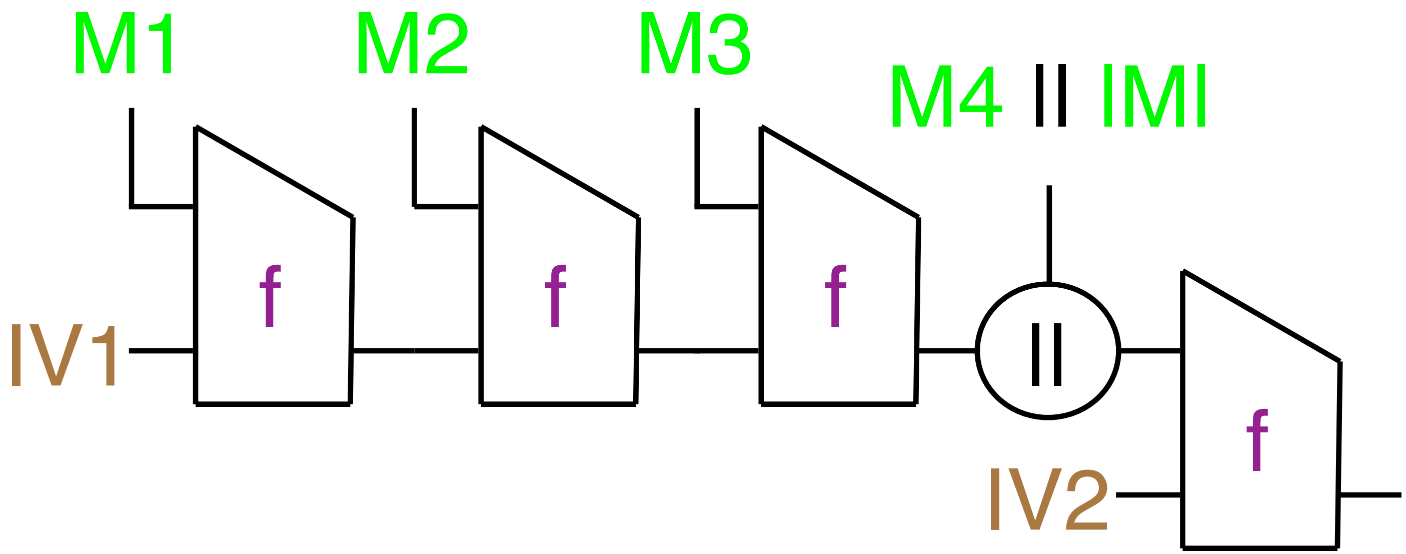
Using MPP approach

Transform	Security	Example Applications
H	**CR-Pr**	digital signatures
	PRO-Pr	ROs
	PRF-Pr	PRF/MAC

Apply H to a single f to build one hash function good for many tasks.

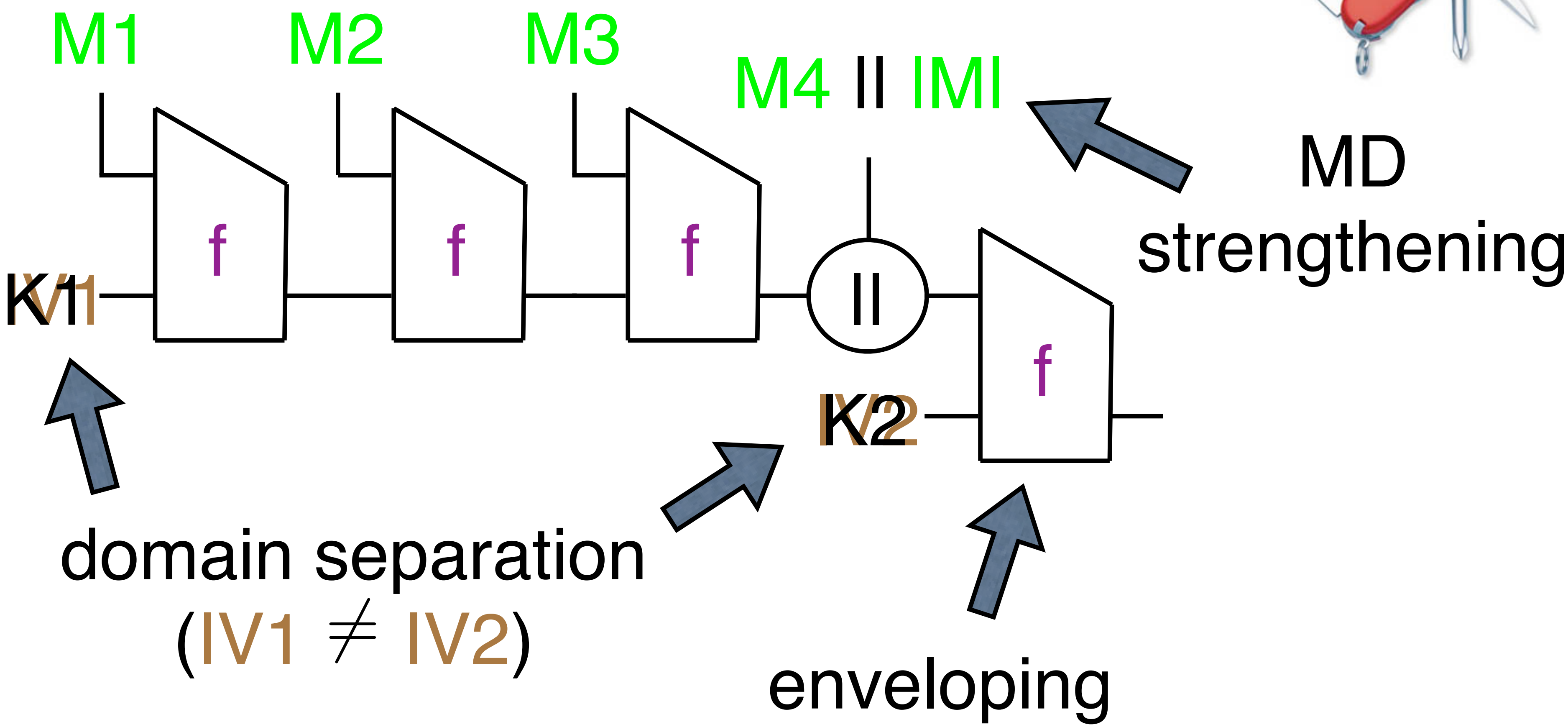
- \+ Standardize just one hash function
- \+ Simplifies implementation choices, one hardware implementation needed

The EMD transform



- Similar in design to NMAC [BCK96], Chain shift construction [MS05].
- Combines several techniques for preserving individual properties.

The EMD transform



EMD is **CR-Pr**

EMD is **PRO-Pr**

EMD is **PRF-Pr**

Theorem **[EMD is CR-Pr]** *Fix n, d, and let $IV1, IV2 \in \{0,1\}^n$ with $IV1 \neq IV2$. Let $f\colon \{0,1\}^{n+d} \to \{0,1\}^n$. Let A be a CR adversary that runs in time t_A. Then there exists an adversary B such that*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathrm{EMD}}(A) \leq \mathbf{Adv}^{\mathrm{cr}}_{f}(B)$$

where B runs in time $t \leq t_A + \mathcal{O}(l)$ where l is the number of blocks in the longer message output by A.

Theorem 5.2 **[EMD is PRO-Pr]** *Fix n, d, and let $IV1, IV2 \in \{0,1\}^n$ with $IV1 \neq IV2$. Let $f = \mathsf{RF}_{d+n,n}$ be a random oracle. Let A be an adversary that asks at most q_L left queries (each of length no larger than ld bits), q_1 right queries with lowest n bits not equal to $IV2$, q_2 right queries with lowest n bits equal to $IV2$, and runs in time t. Then*

$$\mathbf{Adv}^{\mathrm{pro}}_{\mathrm{EMD},\,SA}(A) \leq \frac{(q_L + q_2)^2 + q_1^2 + q_2 q_1}{2^n} + \frac{l q_L^2}{2^n}\,.$$

where the simulator SA, defined in Fig. 4, makes $q_{SA} \leq q_2$ queries and runs in time $\mathcal{O}(q_1^2 + q_2 q_1)$.

Theorem 5.3 **[EMD is PRF-Pr]** *Fix n, d and let $e\colon \{0,1\}^{d+n} \to \{0,1\}^n$ be a function family keyed via the low n bits of its input. Let A be a prf-adversary against keyed* EMD *using q queries of length at most m blocks and running in time t. Then there exists prf-adversaries A_1 and A_2 against e such that*

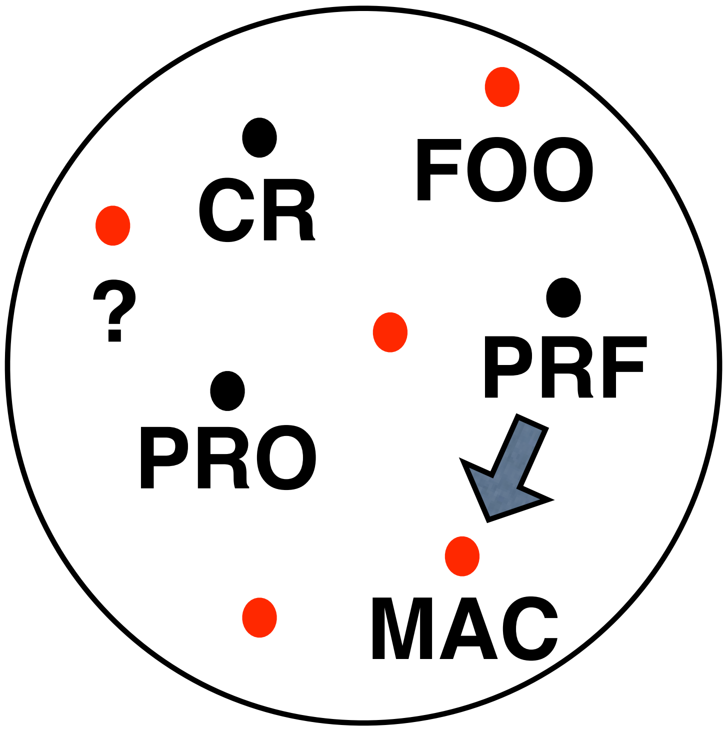
$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{EMD}^e_{K_1,K_2}}(A) \leq \mathbf{Adv}^{\mathrm{prf}}_{e}(A_1) + \binom{q}{2}\left[2m \cdot \mathbf{Adv}^{\mathrm{prf}}_{e}(A_2) + \frac{1}{2^n}\right]$$

where A_1 utilizes q queries and runs in time at most t and A_2 utilizes at most two oracle queries and runs in time $\mathcal{O}(mT_e)$ where T_e is the time for one computation of e.

Transform	**CR-Pr**	**PRO-Pr**	**PRF-Pr**	Efficiency $\lvert M \rvert$ = b >= d
EMD	✓ [BR06]	✓ [BR06]	✓ [BR06]	[(b+1+64+n) / d]
Plain MD	✗	✗	✗	[(b+1) / d]
MD w/str	✓ [D89,M89]	✗	✗	[(b+1+64) / d]
Prefix-free MD	✗	✓ [CDMP05]	✓ [BCK96]	[(b+1) / (d-1)]
Chop solution	✗	✓ [CDMP05]	?	[(b+1) / d]
NMAC construction	✗	✓ [CDMP05]	?	1 + [(b+1) / d]
HMAC construction	✗	✓ [CDMP05]	?	2 + [(b+1) / d]

What about other properties?

Choices to make...



Some properties implied by others (e.g., PRF => MAC)

Should only worry about *useful* properties

Design trade-offs: security versus efficiency

Summary

We propose multi-property-preserving transforms for building the next generation of hash functions

- Minimally a transform H should be **CR-Pr**, **PRO-Pr**, and **PRF-Pr**
- Enables building a single hash function that is good for a variety of applications

We point out that previous **PRO-Pr** transforms are not **CR-Pr** and thus give worse guarantees than MD_+

We describe an efficient MPP transform EMD (Enveloped Merkle-Damgård)

Multi-Property-Preserving Hash Domain Extension
and
the EMD Transform
(Enveloped Merkle-Damgård)