Design Paradigms for Building Multi-Property Hash Functions

Thomas Ristenpart

UCSD Security and Cryptography Lab

Multi-property hash functions

One hash function with many security properties

Design that reflects usage:

-digital signatures (collision resistance)
-message authentication (unforgeability)
-key derivation (PRF)
-instantiate random oracles (pseudorandom oracle)

How to build multi-property hash functions?

Three parts:

1) Multi-property-preserving (MPP) domain extension transforms

[Bellare, Ristenpart 06]

2) Dedicated-key setting & MPP transforms in it

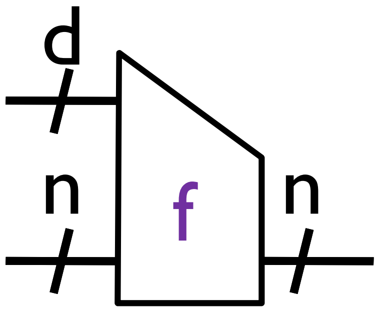
[Bellare, Ristenpart 07]

3) Building provably-CR hash functions

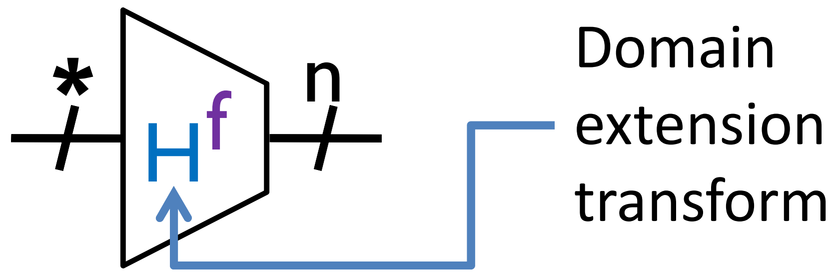
[Ristenpart , Shrimpton 07]

Current hash function design paradigm

Compression Function



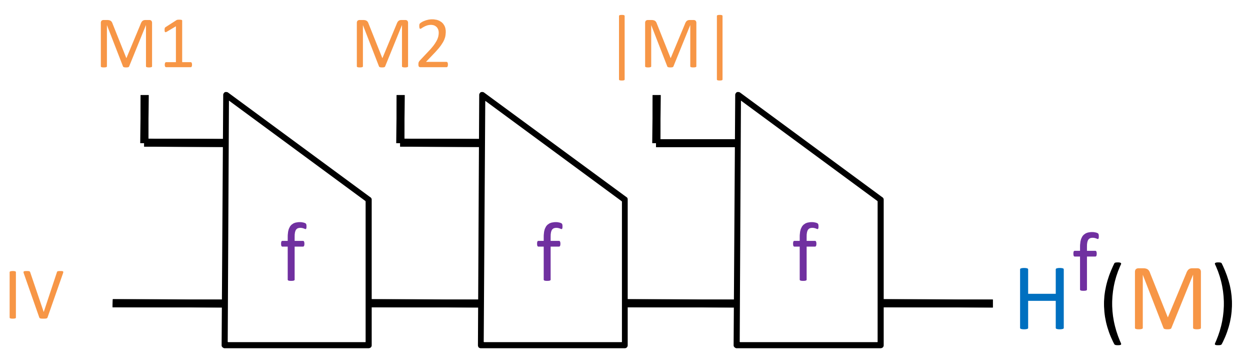
Hash Function



Want H to preserve collision-resistance

$$\textbf{CR-Pr:}\quad f \ \textbf{CR} \Rightarrow H^f \ \textbf{CR}$$

E.g., H = Str. MD (SMD) preserves **CR:**



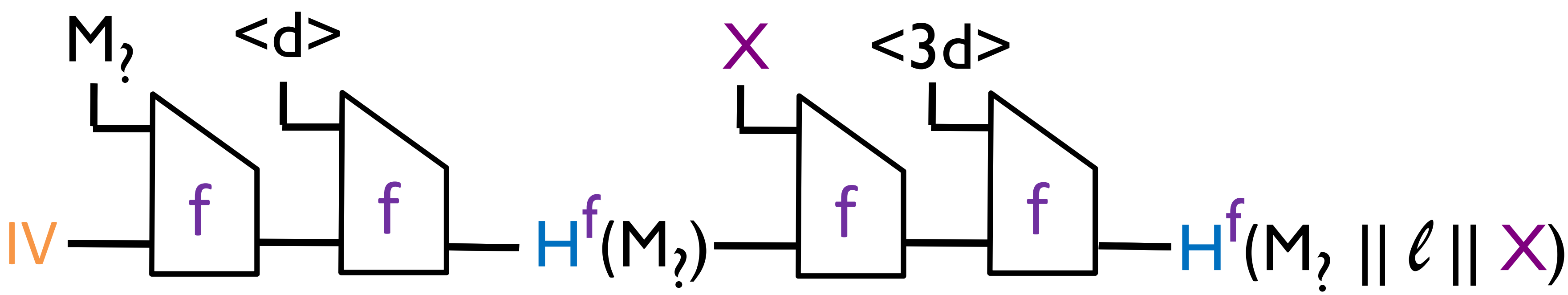
Used in MD-x,SHA-1, SHA-256, …

Extension attack

Let H = MD$^+$ and message $M_?$ unknown to adversary

$$X, \ell = |M_?|, H^f(M_?) \xrightarrow{\text{easy}} H^f(M_? \,||\, \ell \,||\, X)$$

e.g. $|X| = |M_?| = d$



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

$$X, \ell = |M_?|, H^f(M_?) \xrightarrow{\text{easy}} H^f(M_? \,||\, \ell \,||\, X)$$

$$X, \ell = |M_?|, RO(M_?) \xrightarrow{\text{hard}} RO(M_? \,||\, \ell \,||\, X)$$

So what?

Does not affect **CR**

But means that H^f does not “behave like” a RO

True even if f is ideal

[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**):

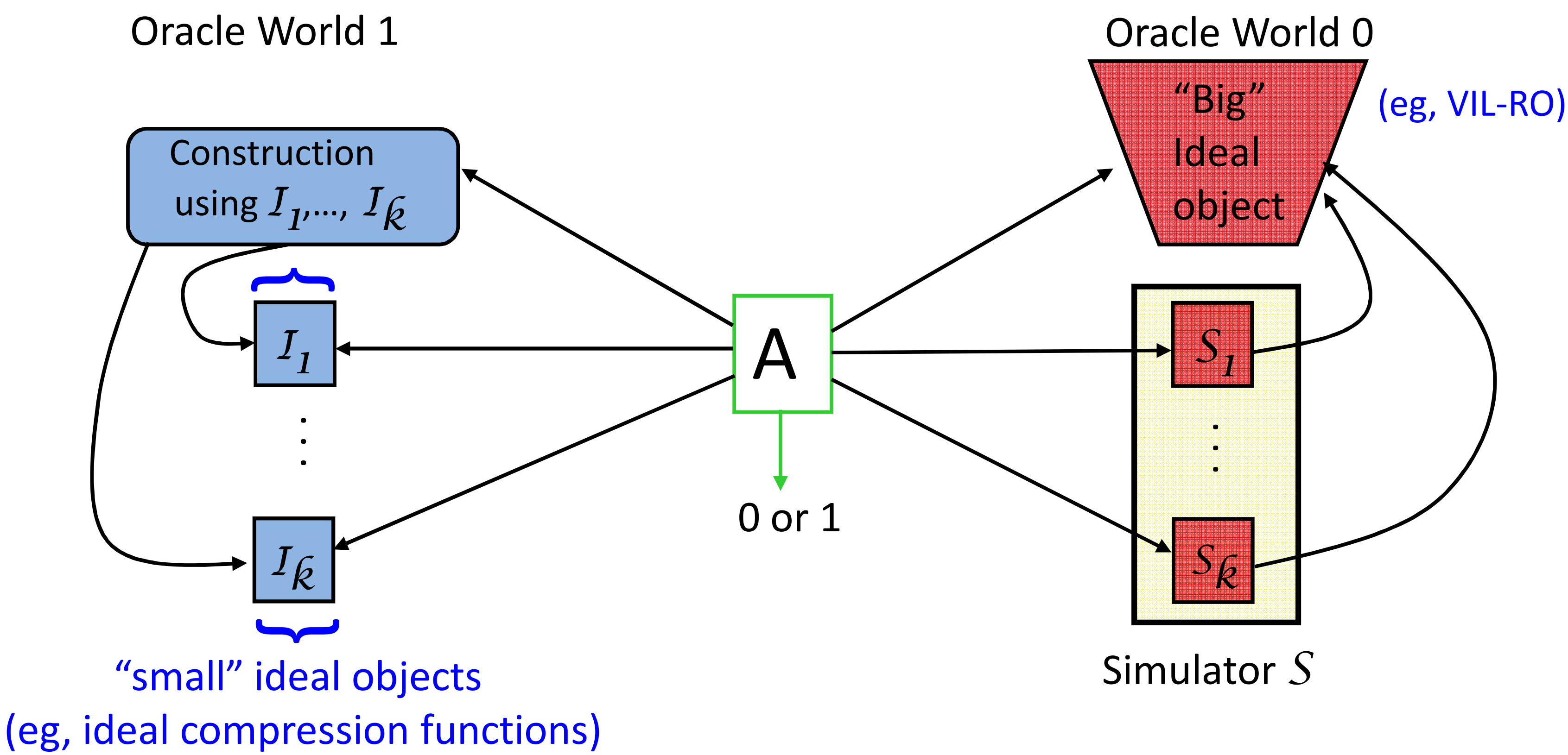
$$f = \mathrm{RO} \Rightarrow H^f \approx \mathrm{RO}$$

PRO

Intuitively: $\approx$ means "behaves like"

Indifferentiability

[MRH04]

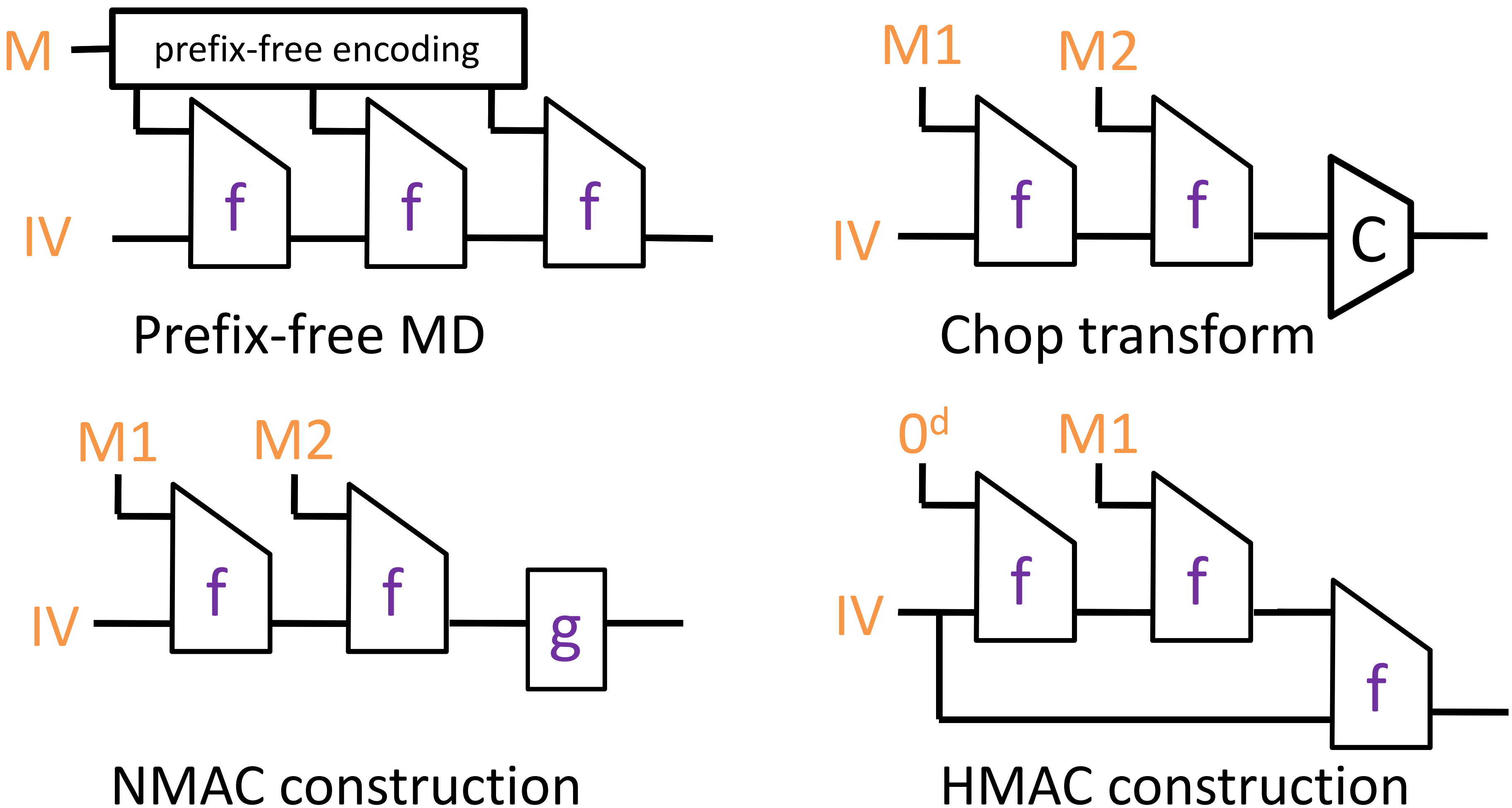


Construction is "indifferentiable" from the Big Ideal Object if there exists a (reasonable) simulator that makes this game hard for every (reasonable) adversary.

"Big" ideal object is RO and indifferentiable
$\Rightarrow$ construction is PRO

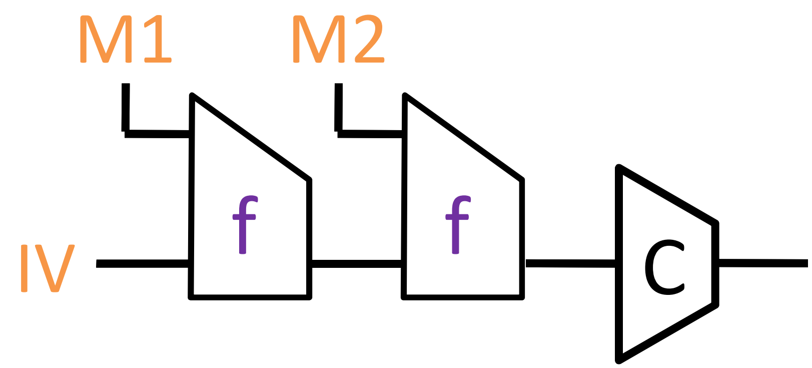
$H = MD_{+}$ is not **PRO-Pr** (due to extension attack)

Several new **PRO-Pr** transforms proposed: [CDMP05]



Intuition for Chop transform being **PRO-Pr**

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

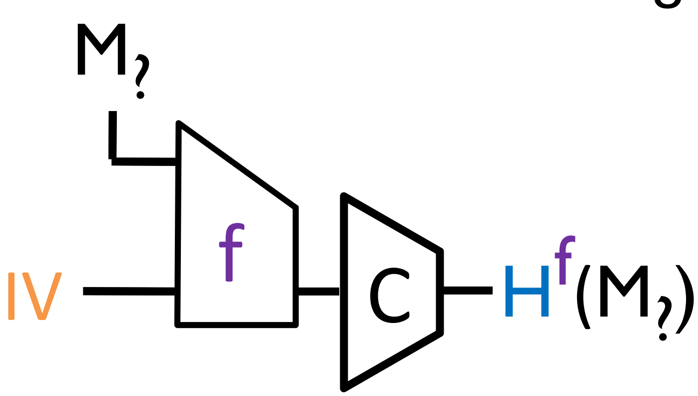


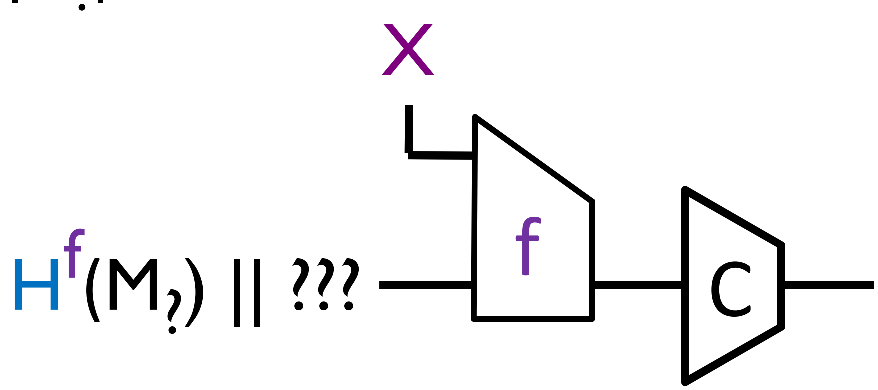
Giving only a fraction of output bits hides structure.

Extension attack fails:

$$X, H^f(M_?) \xrightarrow{\text{hard}} H^f(M_? \,||\, X)$$

e.g. $|X| = |M_?| = d$





M. Bellare, T. Ristenpart. *Multi-Property-Preserving Hash Domain Extension and the EMD Transform.* ASIACRYPT 2006.

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

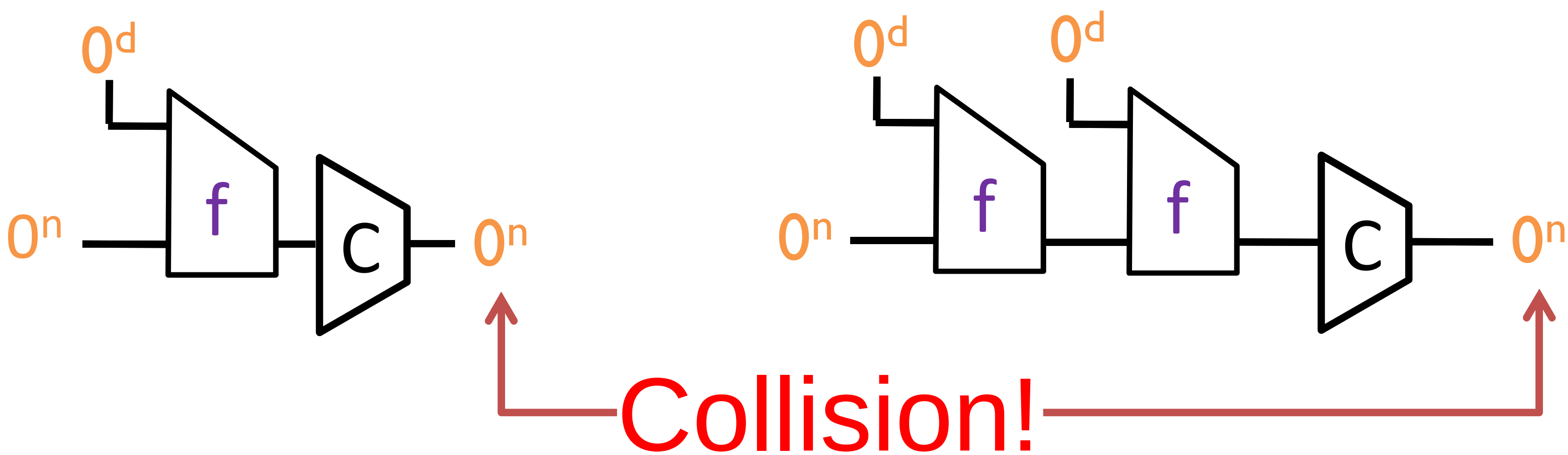
$$\textbf{PRO-Pr} \not\Rightarrow \textbf{CR-Pr}$$

Chop transform is not **CR-Pr**

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

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

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



Similar counter-examples for 3 other transforms

What this means

For **CR**, guarantee of transforms that are (only) **PRO-Pr** is worse than that of SMD

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 hash functions built with proposed transforms

PRO-Pr in review...

+ Important for building hash functions used as ROs

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

Weaker **CR** guarantee: bad for any uses where **CR** is needed for security!

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

Solves the previous problems with (only) **PRO-Pr** transforms!

	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 <u>just</u> **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 <u>both</u> **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 (best) security guarantees for as many applications as possible

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

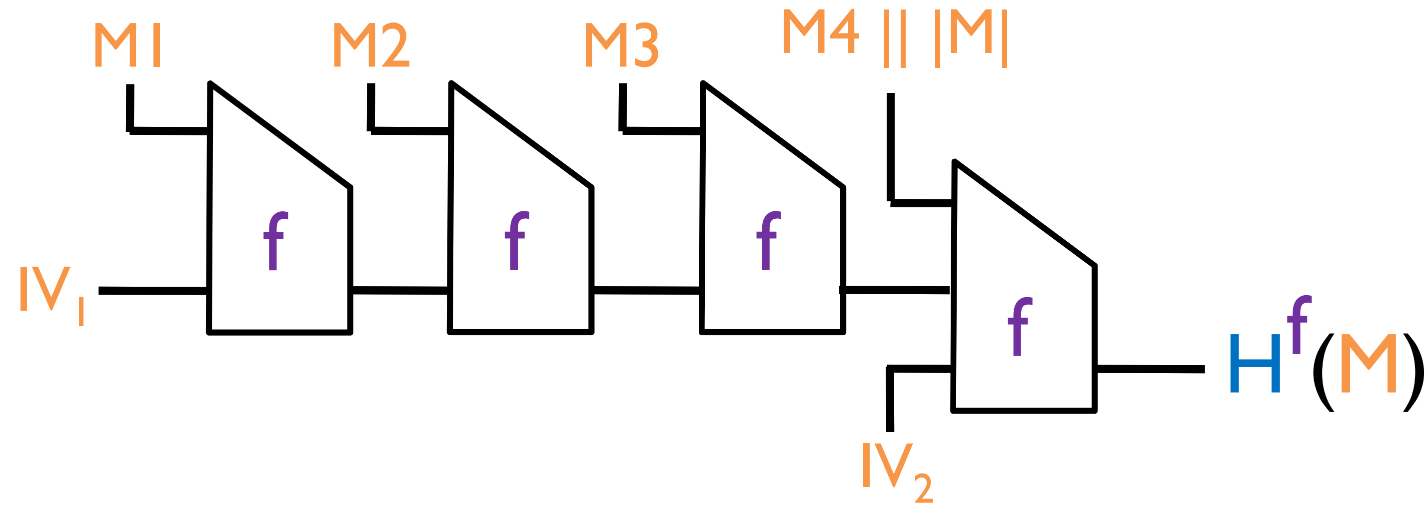
Minimally, we suggest building a single transform H that is simultaneously

1) **CR-Pr** f is **CR** $\Rightarrow$ H^f is **CR**

2) **PRO-Pr** f = **RO** $\Rightarrow$ $H^f \approx$ **RO**

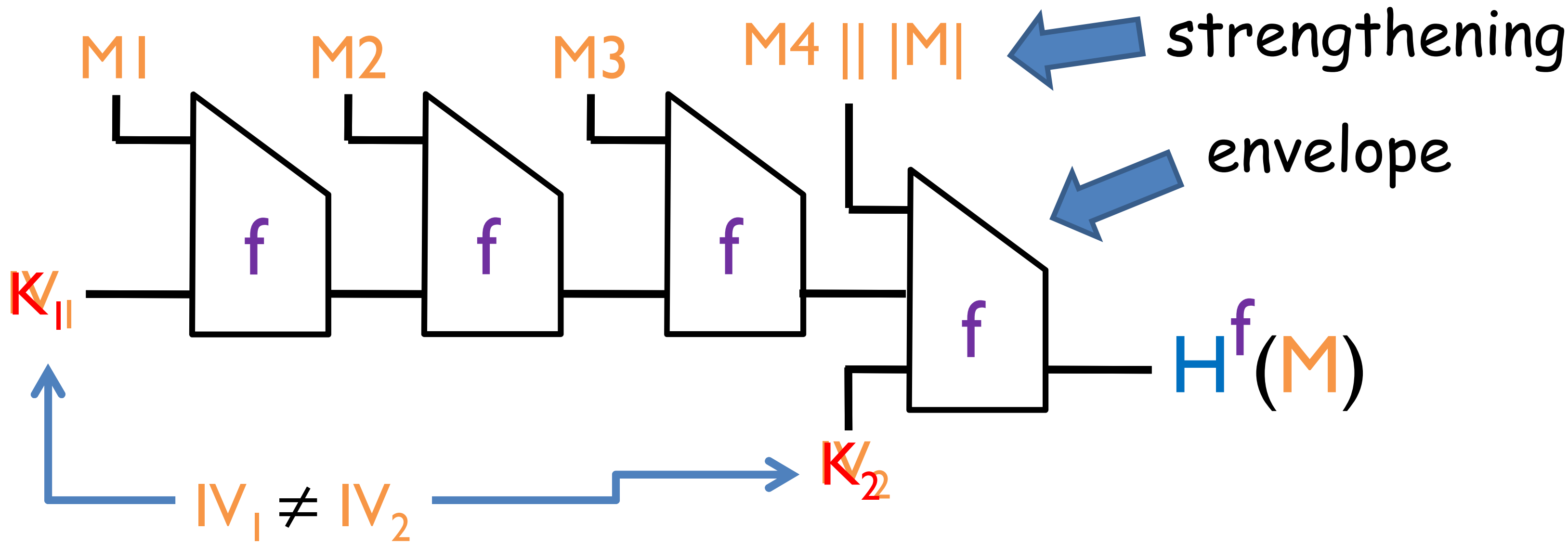
3) **PRF-Pr** f is **PRF** $\Rightarrow$ H^f is **PRF**

The Enveloped MD (EMD) transform



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

The Enveloped MD (EMD) transform



EMD is **CR-Pr**

EMD is **PRO-Pr**

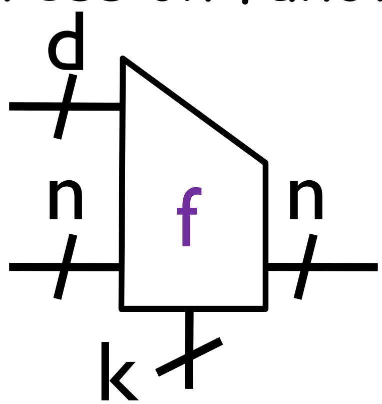
EMD is **PRF-Pr**

Transform	Citation	CR-Pr	PRO-Pr	PRF-Pr
Merkle-Damgard	[M89,D89]	No	No	No
Str. Merkle Damgard	[M89,D89]	Yes	No	No
Prefix-Free MD	[CDMP05]	No	Yes	Yes
Chop transform	[CDMP05]	No	Yes	?
NMAC Construction	[CDMP05]	No	Yes	?
HMAC Construction	[CDMP05]	No	Yes	?

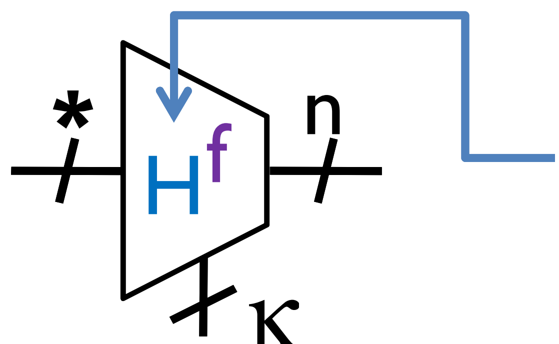
Enveloped MD	[BR06]	Yes	Yes	Yes

Another design setting: dedicated keys

Dedicated-key compression function



Dedicated key hash function

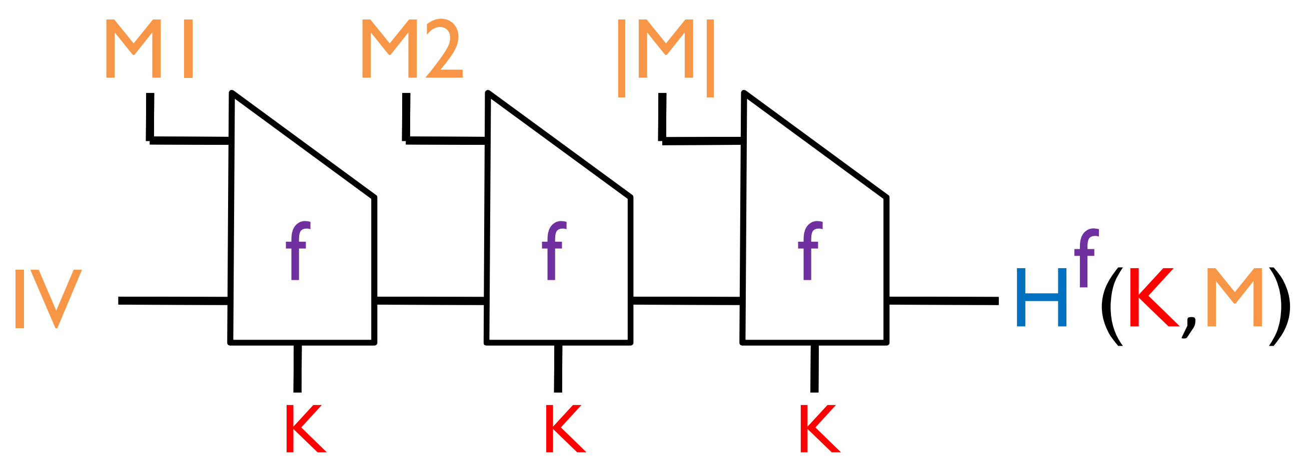


Domain extension transform

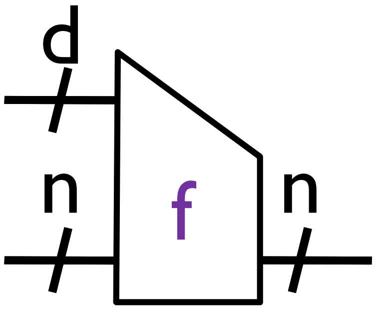
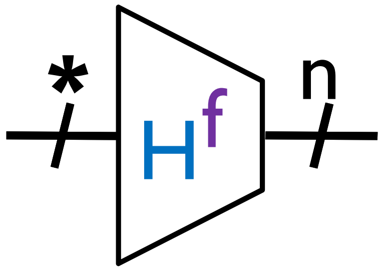
$\kappa \geq k$

E.g., H = Str. MD (SMD) becomes:

IV is not a key



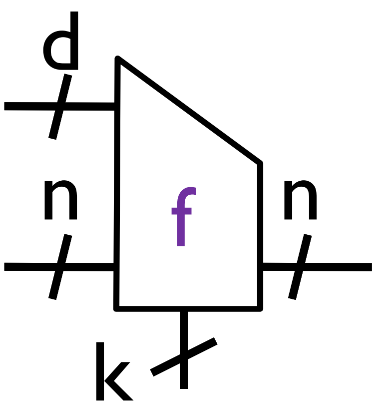
Two settings: comparison

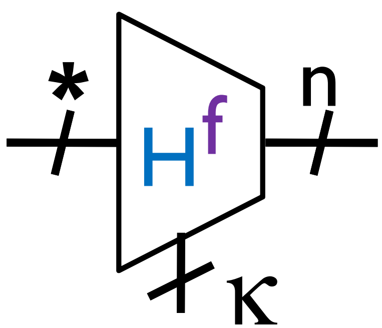
		Public usage (signatures/ROs)	Secret usage (MACs, PRF)
		$H^f(\bullet)$	$H^f(K \parallel \bullet)$

"Traditional"

keying

Dedicated key





public

Public usage (signatures/ROs)	Secret usage (MACs, PRF)
$H^f(K, \bullet)$	$H^f(K, \bullet)$

Traditional vs. dedicated key: just theoretical?

"just a theoretical distinction"

Keyless **CR** functions don't exist

Pigeon-hole principle means

$\exists$A s.t. outputs M $\neq$ M' with H(M) = H(M')
(hardcoded in it)

Solution:

1) Build hash functions **in keyless setting**
2) Analysis in **dedicated key setting**

"foundations-of-hashing dilemma" [R06]

Dedicated keys provides easy solution!

[R06] Doesn't matter in practice (for **CR**)

M. Bellare, T. Ristenpart. *Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms.* ICALP 2007.

Highlight advantages/disadvantages of hash functions in dedicated key setting

(novel advantages of practical impact)

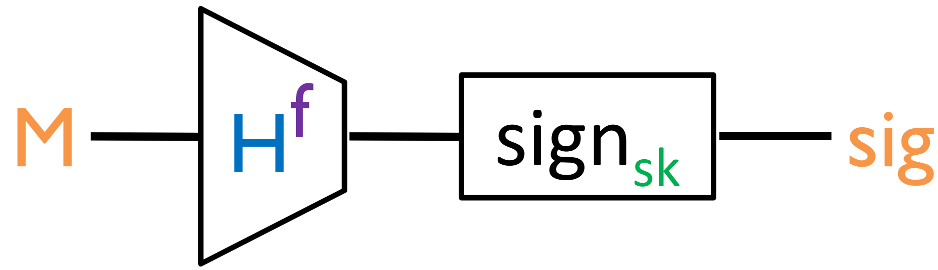
Analyze transforms in dedicated-key setting

(from an MPP perspective)

Hash function heterogeneity:

Users can select independent hash instances

E.g., digital signatures…



User i publishes
public key = vk

verification key

Adversary:

Find collisions for H^f $\Rightarrow$ **all users** compromised

Work:

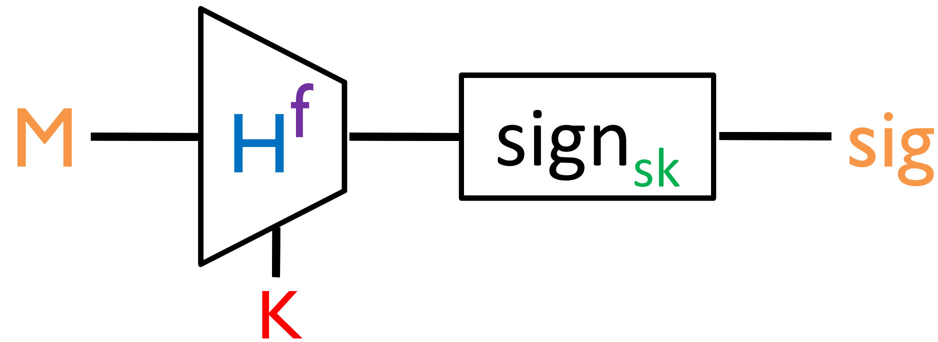
$\sim 2^{61}$

Fix: deploy new hash function

Hash function heterogeneity:

Users can select independent hash instances

E.g., digital signatures…



User i publishes
public key = (vk, K)

verification key
(for signing part)

hash key

Adversary: attacking Bob and Sue

Find collisions for $K_{Bob} \Rightarrow$ breaks Bob's DS

Find collisions for $K_{Sue} \Rightarrow$ breaks Sue's DS

Work:

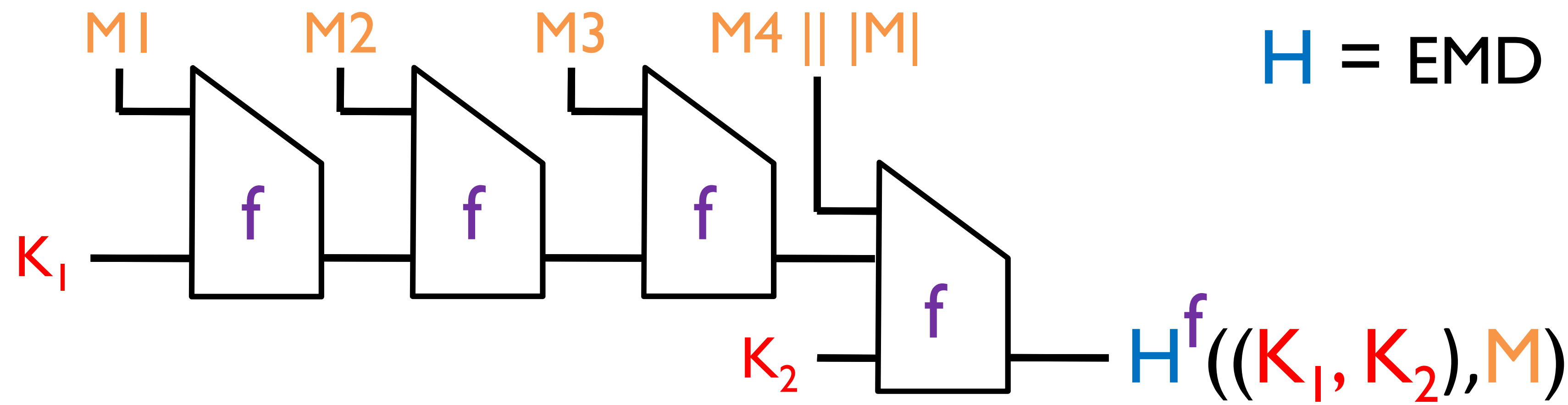
$\sim 2^{61}$

$+ \sim 2^{61}$

Total: $\sim 2^{62}$

(temporary) Fix: choose new keys

Message authentication: "traditional" setting



H = EMD

Security lost

Security rests on f being a good PRF:

$\cancel{f \textbf{ PRF} \Rightarrow H^f \textbf{ PRF} \Rightarrow H^f \textbf{ MAC}}$

Security guarantee worse than

f ***MAC*** $\Rightarrow H^f$ ***MAC***

Still okay

Breaking f as **PRF** $\not\Rightarrow$ Breaking f as ***MAC***

"Traditional" setting:

no (known) transforms preserve ***MAC***

Dedicated key setting:

several efficient transforms preserve ***MAC***

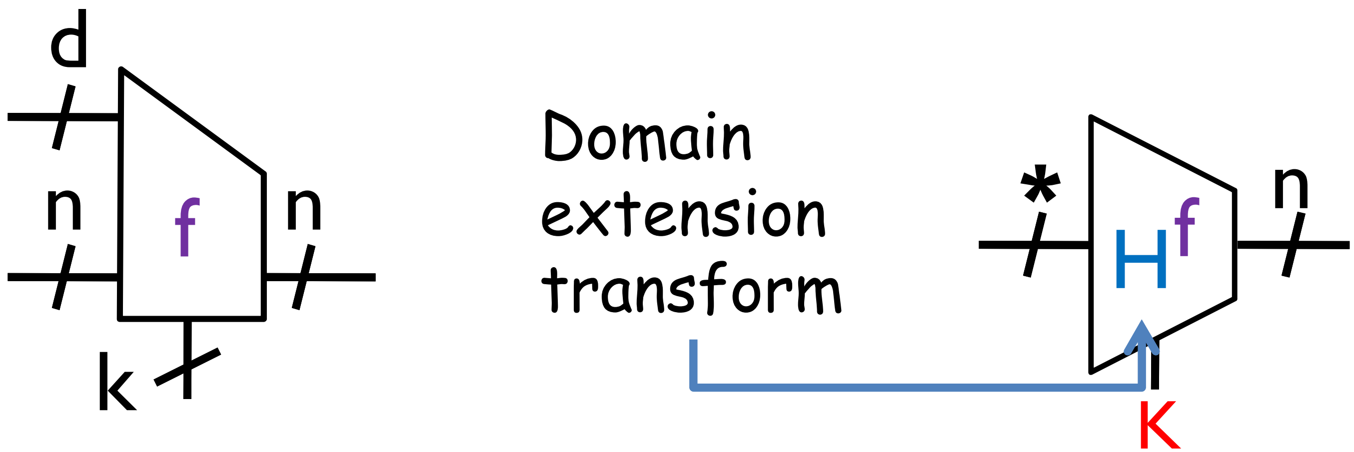
[AB99] [MS05a] [MS05b]

For message authentication this means stronger security guarantees!

What about downsides of dedicated-key?

Efficiency loss: (n+d) per block vs. (n+d+k) per block

MPP dedicated-key transforms



Want multi-property-preserving transform:

CR-Pr: f CR $\Rightarrow$ H^f CR

PRO-Pr: f PRO $\Rightarrow$ H^f PRO

PRF-Pr: f PRF $\Rightarrow$ H^f PRF

MAC-Pr: f MAC $\Rightarrow$ H^f MAC

TCR-Pr: f TCR $\Rightarrow$ H^f TCR

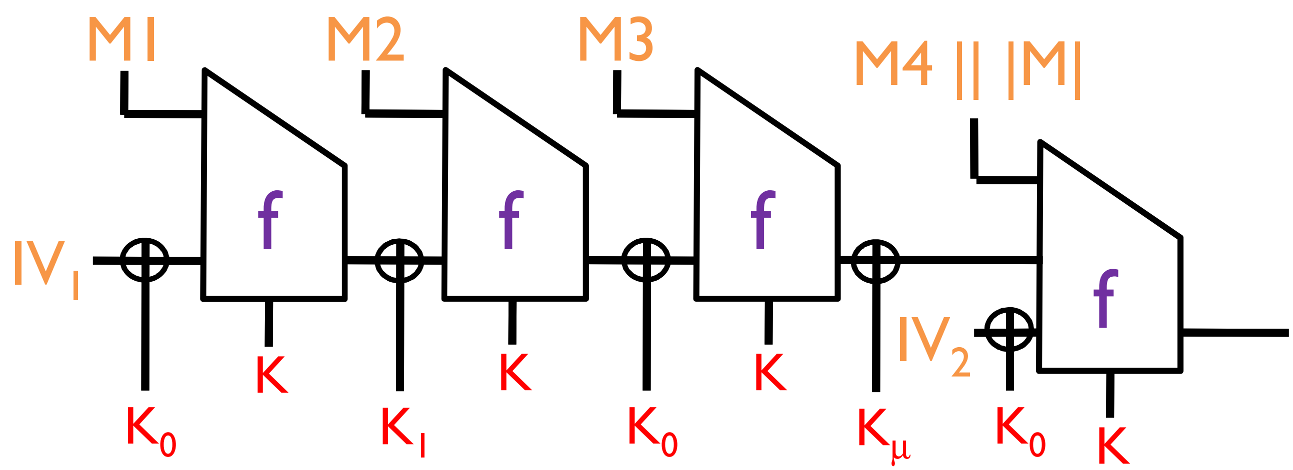
Transform	Citation	CR-Pr	PRO-Pr	PRF-Pr	MAC-Pr	TCR-Pr	Key bits
Merkle-Damgard	[M89,D89]	No	No	Yes	No	No	k
Str. Merkle Damgard	[M89,D89]	Yes	No	Yes	No	No	k
Prefix-Free MD	[MS05]	No	Yes	Yes	Yes	No	k
Shoup	[S00]	Yes	No	Yes	No	Yes	k log σ
Str. Nested Iteration	[AB99]	Yes	Yes	Yes	Yes	No	2k
Nested Iteration	[MS05]	No	Yes	Yes	Yes	No	2k
Chain Shift	[MS05]	No	Yes	Yes	Yes	No	k

Str. Chain Shift	[BR07]	Yes	Yes	Yes	Yes	No	k
Enveloped Shoup	[BR07]	Yes	Yes	Yes	Yes	Yes	k log σ + k

MPP Transforms

Transform	Citation	CR-Pr	PRO-Pr	PRF-Pr			Key bits
Enveloped MD	[BR06]	Yes	Yes	Yes			2n (PRF)
					MAC-Pr	TCR-Pr	
Str. Chain Shift	[BR07]	Yes	Yes	Yes	Yes	No	k
Enveloped Shoup	[BR07]	Yes	Yes	Yes	Yes	Yes	k log σ + k

Transforms all share similar structure:



What about f?

NIST competition --- many new options

Another approach:
use **provably-CR** compression function

$$f(m) = x^m \bmod N$$

large composite

fixed base

Guarantee: collisions against f $\Rightarrow$ factoring N

Ex.: VSH [CLS05], FFT hash [LMPR06],
expander graph hashing [CGL06]

Provably-CR functions not like ROs!

Let $m_?$ be message unknown to adversary

$$f(m_?) \xrightarrow{\text{easy}} f(2m_?)$$

Because: $f(2m_?) = (x^{m_?})^2 \bmod N = f(m_?)^2 \bmod N$

$$RO(m_?) \xrightarrow{\text{hard}} RO(2m_?)$$

So applying MPP transform to f ...

Gives: provably CR function

Does **NOT** give: pseudorandom oracle

T. Ristenpart, T. Shrimpton. *How to Build a Hash Function from any Collision-Resistant Function.* Asiacrypt 2007.

Is there a generic method for turning CR functions into good RO's?

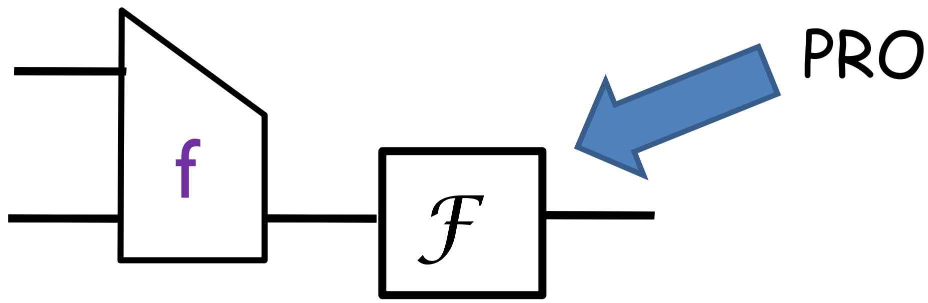
Important:
don't lose standard model CR guarantee!!!

Build a function that is *simultaneously*:

1. a PRO in an idealized model
2. provably CR in the standard model

A simple approach that doesn't work

Compose CR function with a suitable RO instantiation



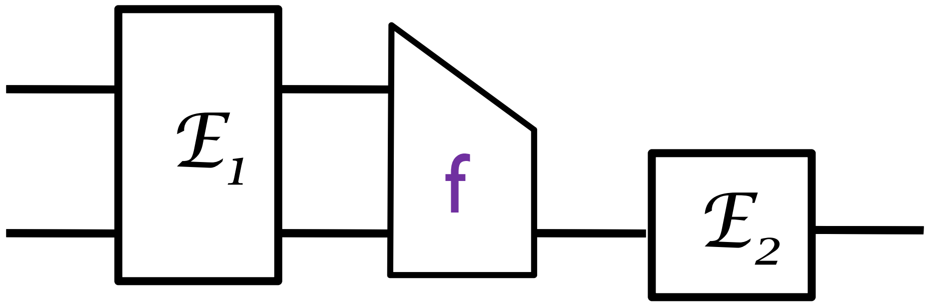
This might seem sufficient... but...

$\mathcal{F}(f(\bullet))$ being CR $\Rightarrow$ $\mathcal{F}$ must be CR (in standard model)

$\mathcal{F}$ = RO $\not\Rightarrow$ $\mathcal{F}(f(\bullet))$ is a PRO [CDMP05]

Mix-Compress-Mix construction

Sandwhich f between two **injective** “mixing steps”



This works:

$$f \text{ CR} + \mathcal{E}_1, \mathcal{E}_2 \text{ injective} \Rightarrow \mathcal{E}_2(f(\mathcal{E}_1(\bullet))) \text{ is CR}$$

$$f \text{ CR,balanced} + \mathcal{E}_1, \mathcal{E}_2 \text{ ideal} \Rightarrow \mathcal{E}_2(f(\mathcal{E}_1(\bullet))) \text{ is PRO}$$

Intuition is clear:

mixing steps hide any structure of f

How to build good mixing steps?

Mixing steps: Injective (by construction)

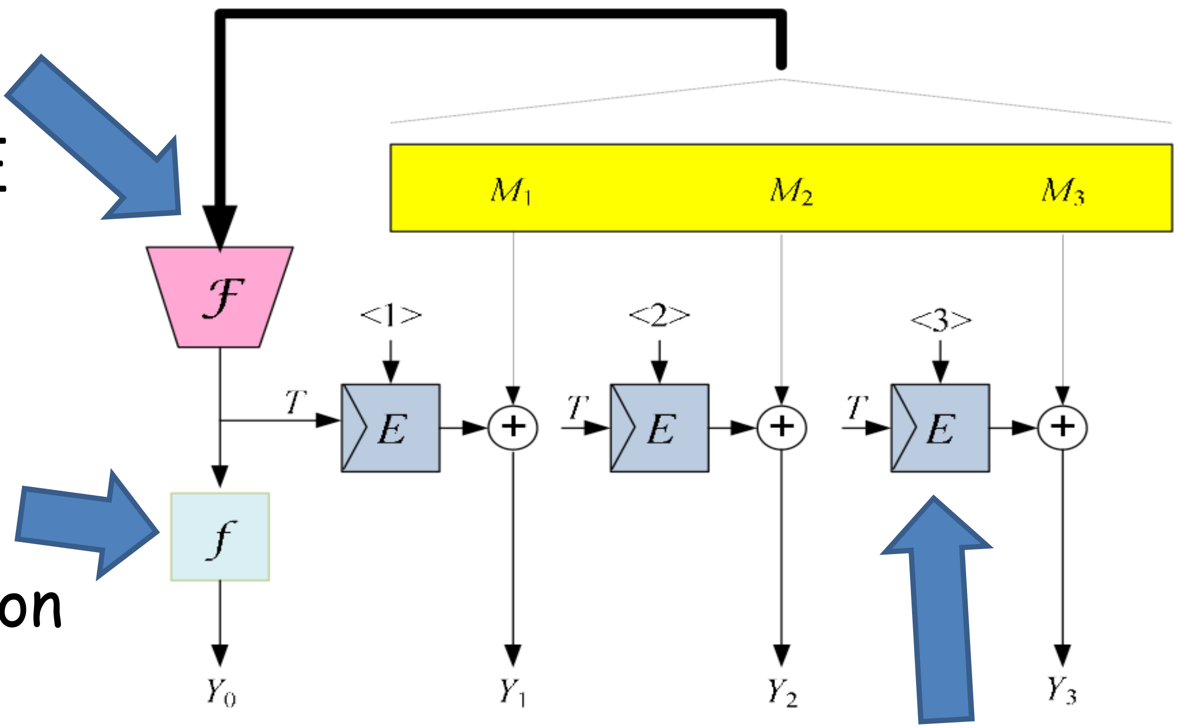
PRO in ideal model

TE (Tag-Encipher) construction:

VIL-PRO

built w/ E

[CDMP05, CLNY06]

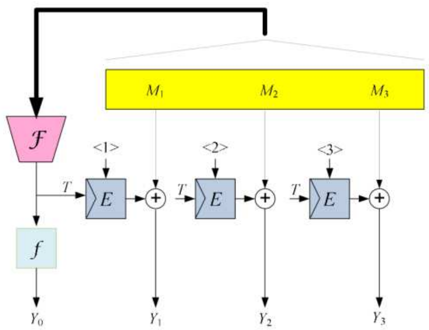


trapdoor

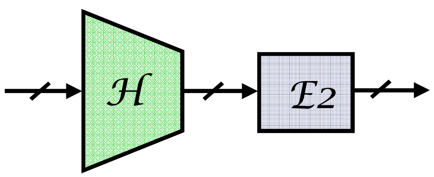
permutation

ideal cipher

Open questions…



Are there more efficient constructions of mixing steps?



In practice, this seems like it should be fine. Are the definitions “off”?

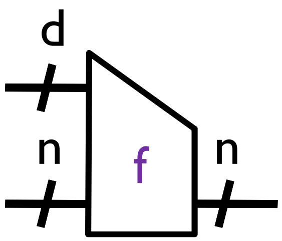
✓ provable CR
✓ PRO

What about PRF?
Other properties?

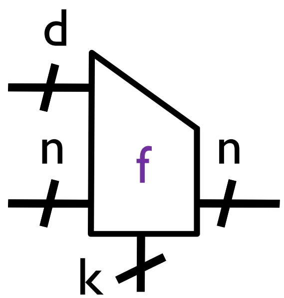
Summary

Explored new approaches to building cryptographic hash functions with broad security

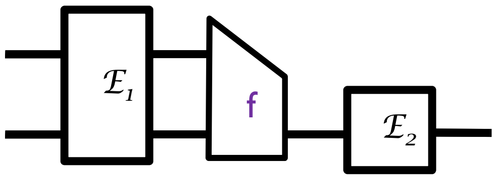
Hash function should meet each security property of interest under weakest (possible) assumptions!



"Traditional" MPP transforms:
Enveloped MD



Dedicated-key MPP transforms:
str. Chain-Shift,
Enveloped Shoup



provably-CR hash functions:
MCM, **TE**

New tools for building multi-purpose hash functions!

M. Bellare, T. Ristenpart. *Multi-Property-Preserving Hash Domain Extension and the EMD Transform.* ASIACRYPT 2006.

M. Bellare, T. Ristenpart. *Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms.* ICALP 2007.

T. Ristenpart, T. Shrimpton. *How to Build a Hash Function from any Collision-Resistant Function.* Asiacrypt 2007.

Available: http://www.cse.ucsd.edu/~tristenp/