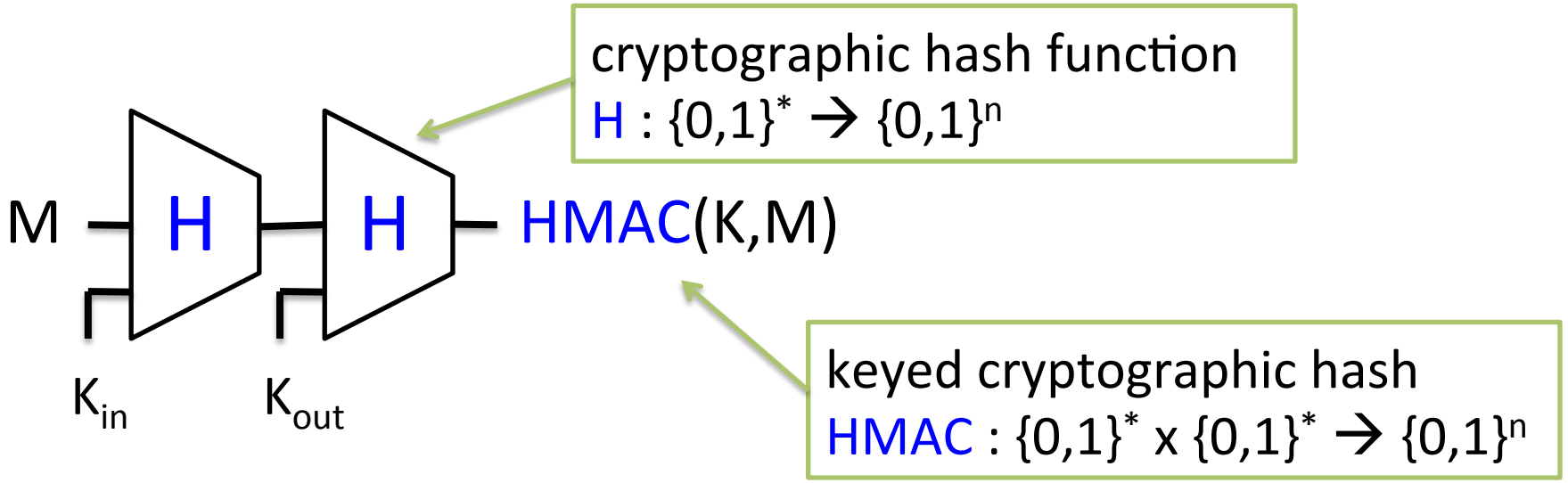
To Hash or Not To Hash Again?

(In)differentiability Results for H^2 and HMAC

Yevgeniy Dodis (New York University)
Thomas Ristenpart (University of Wisconsin)
John Steinberger (Tsinghua University)
Stefano Tessaro (MIT)



HMAC construction

[Bellare, Canetti, Krawczyk 1996]

K_{in} , K_{out} derived from K (details to come)

$HMAC(K,M) = H(K_{out} \,||\, H(K_{in} \,||\, M))$

Designed for message authentication with secret keys …

… but now used in a variety of ways

Usage of HMAC

Disk encryption uses PKCS#5:
HMAC(pw, salt)

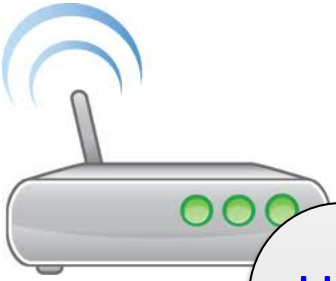
TLS connection

TLS handshake:
HMAC(g^{xy} , …)

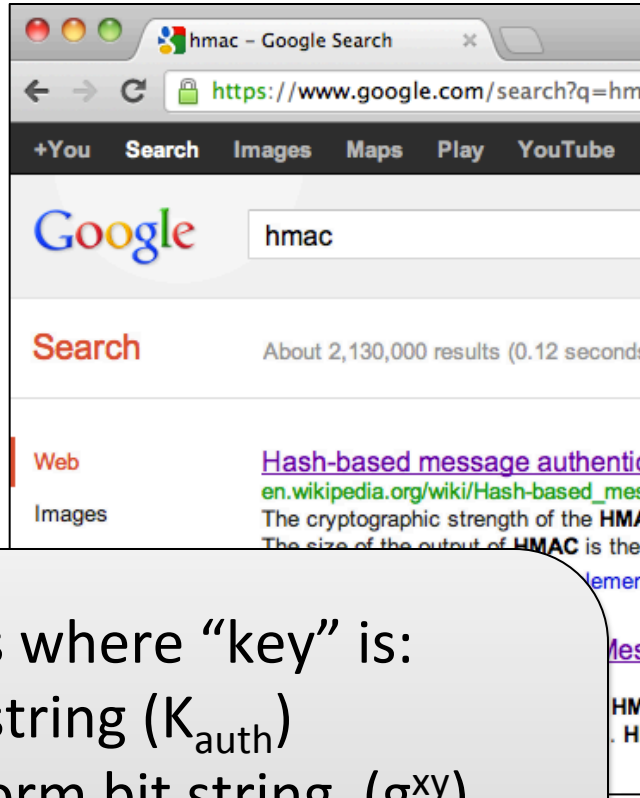
TLS record layer:
HMAC(K_{auth}, Ctxt)

IPSec tunnel

IKEv2 (HKDF):
HMAC(nonces, g^{xy})



WPA WiFi protocol uses PKCS#5:
HMAC(pw, ssid)



HMAC used in settings where "key" is:

- uniform, secret bit string (K_{auth})
- secret, but not uniform bit string (g^{xy})
- low-entropy, variable-length secrets (pw)
- public (nonces)
- …

Security of HMAC?

When K is secret, uniform, fixed-length bit string?

Standard-model proofs of PRF/unforgeability security

[Bellare, Canetti, Krawczyk 1996] [Bellare 2006]

When input includes high-entropy, secret bit string?

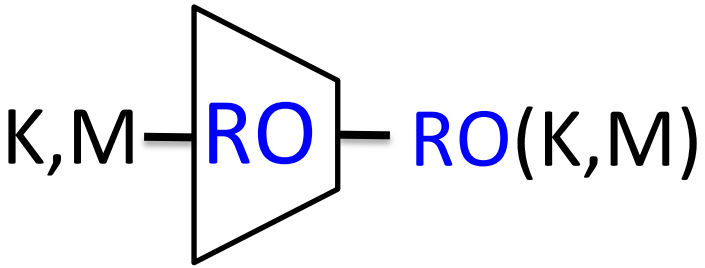
Standard-model proofs that HMAC is a randomness extractor

[DGKHR 2004] [FPZ08] [K10]

Security of HMAC?

Elsewhere?

Proofs model HMAC as a (keyed) random oracle (RO) [BR93]



Each pair K,M mapped to uniformly chosen output RO(K,M)

Examples:

HKDF/IKEv2 [K10]	HMAC(salt,S)
PKCS#5 [BRT12]	HMAC(pw,salt)
Hedged crypto [RY10]	HMAC(R,M)

Public value

Non-uniform, varying length secret. Often low entropy.

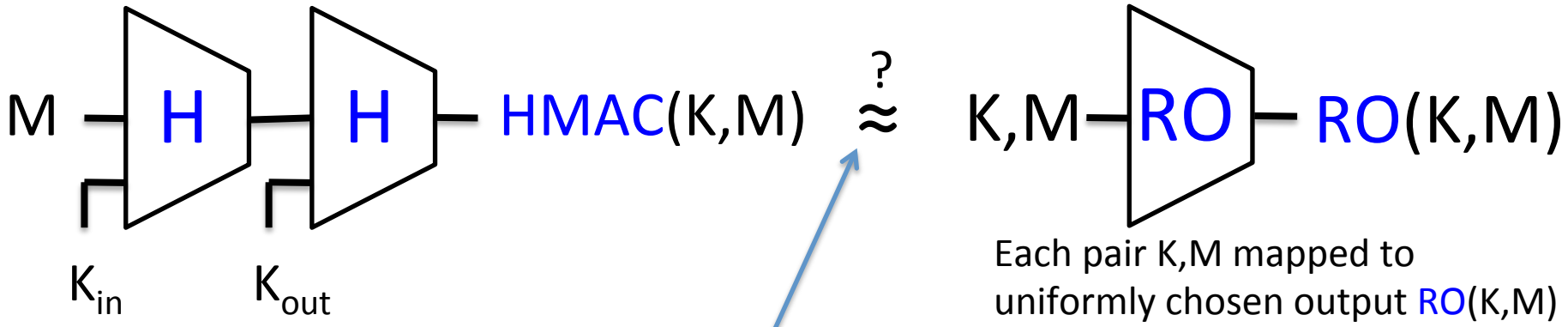
Adversarially controlled

Practitioners expect that HMAC behaves "randomly" for any K,M

HMAC as a RO

Let's (generously) assume H is perfectly secure (itself a random oracle)

Does HMAC "behave like" a RO?

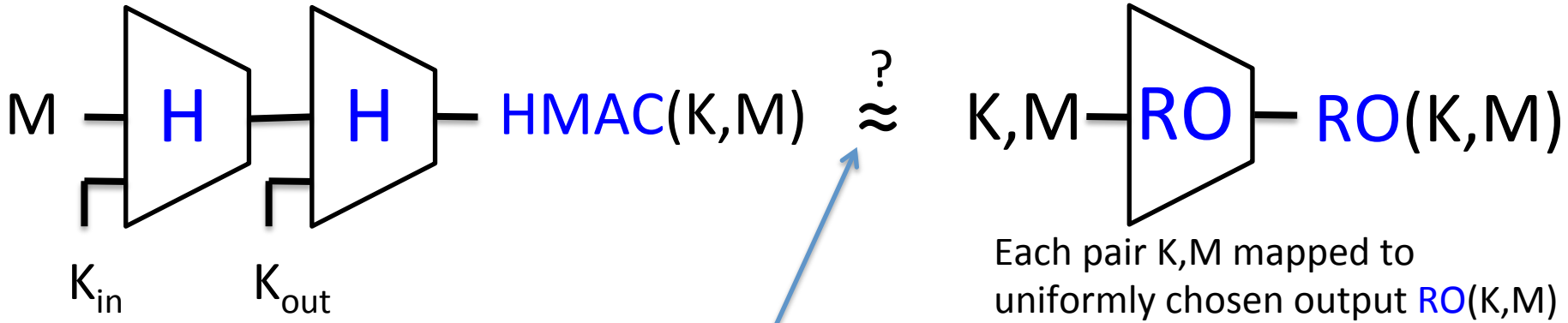


Formalize "behave like" RO via indifferentiability [MRH04]

HMAC and H^2 as ROs

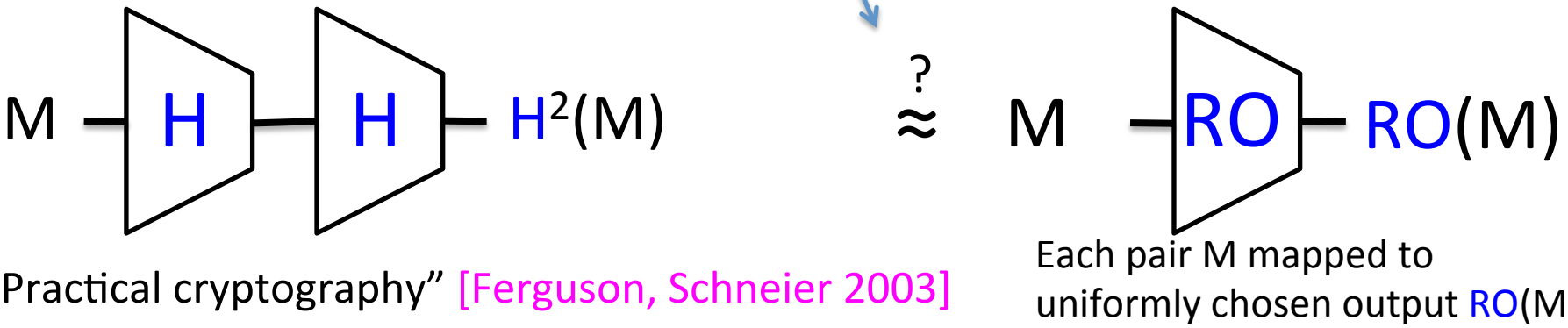
Let's (generously) assume H is perfectly secure (itself a random oracle)

Does HMAC "behave like" a RO?



Formalize "behave like" RO via indifferentiability [MRH04]

Does H^2 "behave like" a RO?



"Practical cryptography" [Ferguson, Schneier 2003]

One expects the answers to be "yes"

Strong positive intuition:

- Both designed to prevent length-extension attacks
- Compose RO with itself, result should behave like a RO

Confusingly, they refer to this as the "HMAC Construction". It's not HMAC

[CDMP05] prove $H^2(0^d \,||\, M)$ indifferentiable from a RO, when H is Merkle-Damgard-style hash function (e.g., SHA-256)

[K10] suggests that the [CDMP05] proof extends to the real HMAC

Summary of our results

Uncover that HMAC has ***weak key pairs***

(1) Colliding key pairs $K \neq K'$ s.t. HMAC(K,M) = HMAC(K',M)

(2) Ambiguous key pairs no inner/outer H domain separation

$K \neq K'$ s.t. $K_{in} = K'_{out}$ $K'_{in} = K_{out}$

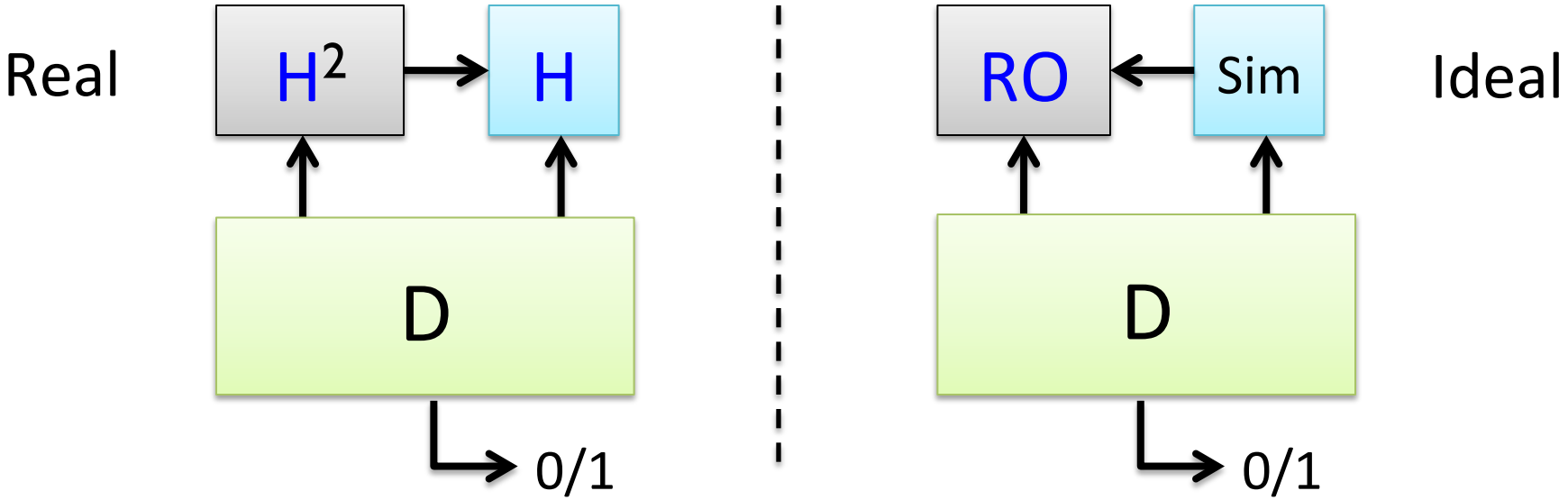
H^2 and HMAC w/ ambiguous key pairs have similar issues

- At best weak concrete security from indifferentiability
- Upper bound proof that H^2 is "weakly" indifferentiable
- Example (vulnerable) setting: mutual proofs of work

Avoid weak key pairs in HMAC, get strong indifferentiability

[CDMP05]: formalize "behave like a RO" via indifferentiability framework of [MRH04]

Indifferentiability from a RO



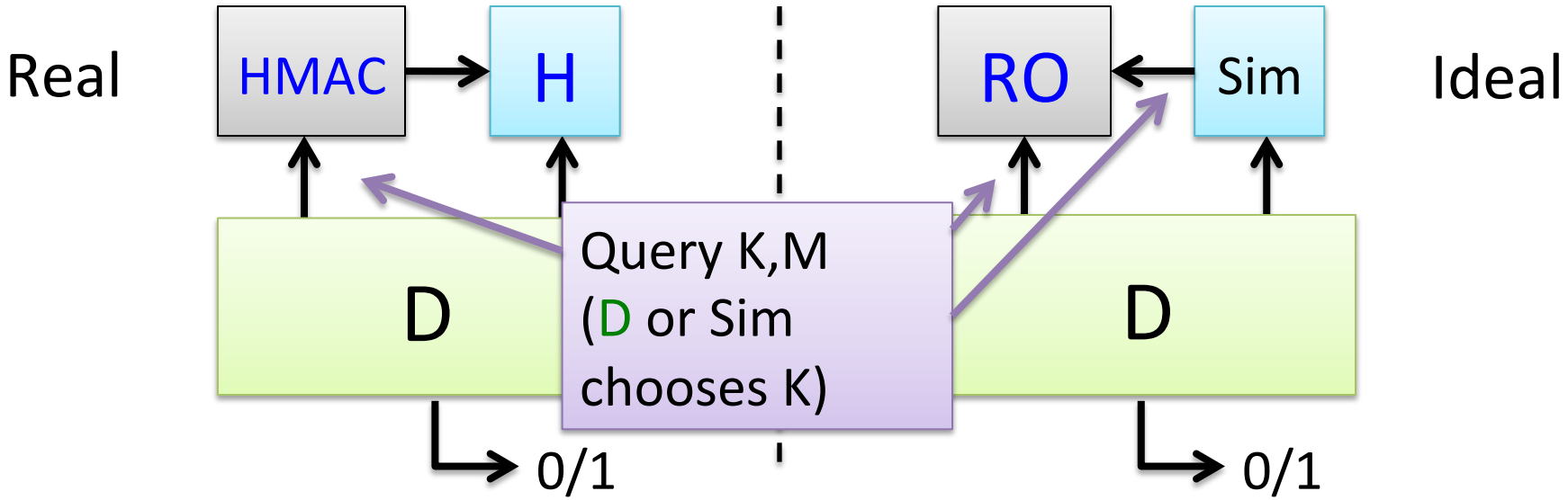
H , RO both random functions

Indifferentiable if exists efficient Sim s.t. for all efficient D

$Adv^{indiff}(H^2,D,Sim) = Pr[Real(D) => 1] – Pr[Ideal(D) => 1] < \varepsilon$

[CDMP05]: formalize "behave like a RO" via indifferentiability framework of [MRH04]

Indifferentiability from a RO



H , RO both random functions

Indifferentiable if exists efficient Sim s.t. for all efficient D

$Adv^{indiff}(H^2,D,Sim) = Pr[Real(D) => 1] - Pr[Ideal(D) => 1] < \varepsilon$

$Adv^{indiff}(HMAC,D,Sim) = Pr[Real(D) => 1] - Pr[Ideal(D) => 1] < \varepsilon$

[CDMP05]: formalize "behave like a RO" via indifferentiability framework of [MRH04]

(Conjectured) HMAC indifferentiability

+

[BRT12],[K10],[RY10] proofs of security for applications using RO

[MRH04] composition theorem

Applications proven secure using HMAC (H still ideal)

Rules out attacks that abuse structure of HMAC

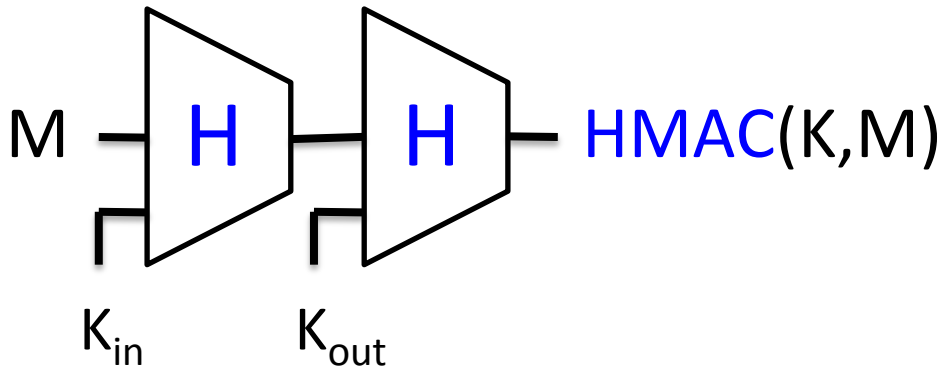
Limitations:

- [RSS11]: composition only applies to "single-stage" games
- Concrete security of indifferentiability important!

Normal indifferentiability: Sim makes O(q) queries
Weak indifferentiability: Sim makes >O(q) queries

A closer look at HMAC

Let H : {0,1}* -> {0,1}^{n} be parameterized by a block length d ≤ n



HMAC(K,M):
If |K| > d then K' <- H(K)
else K' <- K
K'' <- K' || $0^{d-|K'|}$
K_{in} <- K'' ⊕ ipad
K_{out} <- K'' ⊕ opad
Return H(K_{out} || H(K_{in} || M))

ipad ≠ opad fixed strings

A closer look at HMAC

Let H : {0,1}* -> $\{0,1\}^n$ be parameterized by a block length d ≤ n

HMAC(K,M):
If |K| > d then K' <- H(K)
else K' <- K
K'' <- K' || $0^{d-|K'|}$
K_{in} <- K'' ⊕ ipad
K_{out} <- K'' ⊕ opad
Return H(K_{out} || H(K_{in} || M))

ipad ≠ opad fixed strings

A closer look at HMAC

Let H : {0,1}* -> {0,1}^{n} be parameterized by a block length d ≤ n

1) Colliding key pairs

Any K1 ≠ K2 such that

HMAC(K1,M) = HMAC(K2,M)

Example: K2 = K1 || 0 |K1| < d

HMAC(K,M):
If |K| > d then K' <- H(K)
else K' <- K
K'' <- K' || $0^{d-|K'|}$
K_{in} <- K'' ⊕ ipad
K_{out} <- K'' ⊕ opad
Return H(K_{out} || H(K_{in} || M))

ipad ≠ opad fixed strings

Simple distinguisher breaking indifferentiability with 2 queries

Possible security issue anywhere variable-length keys used

No colliding keys if use fixed-length keys (and H is CR)

D
Query left oracle on (K,M) to get Y
Query left oracle on (K||0,M) to get Y'
If Y = Y' then ret 1
ret 0

A closer look at HMAC

Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be parameterized by a block length $d \leq n$

1) Colliding key pairs

Any $K1 \neq K2$ such that

$HMAC(K1,M) = HMAC(K2,M)$

Example: $K2 = K1 \,||\, 0 \quad |K1| < d$

HMAC(K,M):
If $|K| > d$ then $K' \leftarrow H(K)$
else $K' \leftarrow K$
$K'' \leftarrow K' \,||\, 0^{d-|K'|}$
$K_{in} \leftarrow K'' \oplus ipad$
$K_{out} \leftarrow K'' \oplus opad$
Return $H(K_{out} \,||\, H(K_{in} \,||\, M))$

ipad ≠ opad fixed strings

2) Ambiguous key pairs

Any $K1 \neq K2$ such that

$K1_{in} = K2_{out} \qquad K2_{in} = K1_{out}$

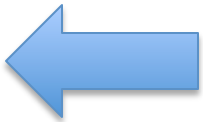
Example: $K2 = K1 \oplus ipad \oplus opad \quad |K1| = |K2| = d$

For all X:
$H(K1_{in} \,||\, X) = H(K2_{out} \,||\, X)$
$H(K2_{in} \,||\, X) = H(K1_{out} \,||\, X)$

No inner/outer H domain separation.

Summary of HMAC indifferentiability

Key space includes	Indifferentiable?
Colliding key pairs	No
Ambiguous key pairs (no colliding key pairs)	At most weak
Keys K of fixed length $\|K\| < d-1$	Yes



Avoids both kinds of weak key pairs:

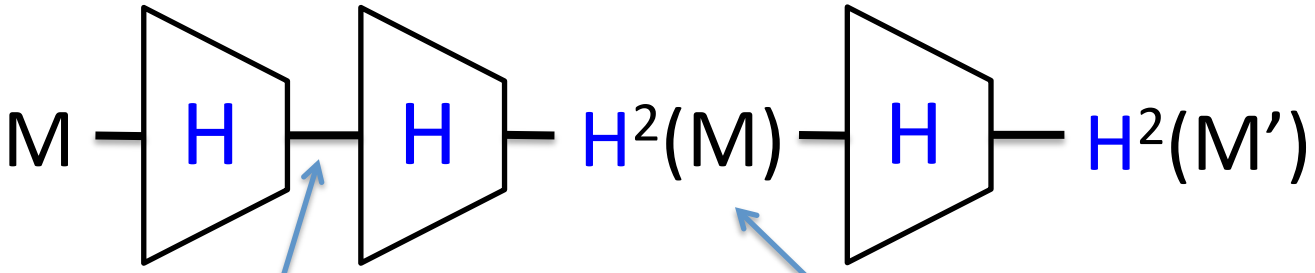
Indifferentiability upper bound (simplified)

For HMAC with restricted keys, exists Sim such that for any D

$$\mathrm{Adv}^{\mathrm{indiff}}(\mathrm{HMAC}, D, \mathrm{Sim}) < O(q^2 / 2^n)$$

Sim makes O(q) queries

Lack of inner/outer domain separation



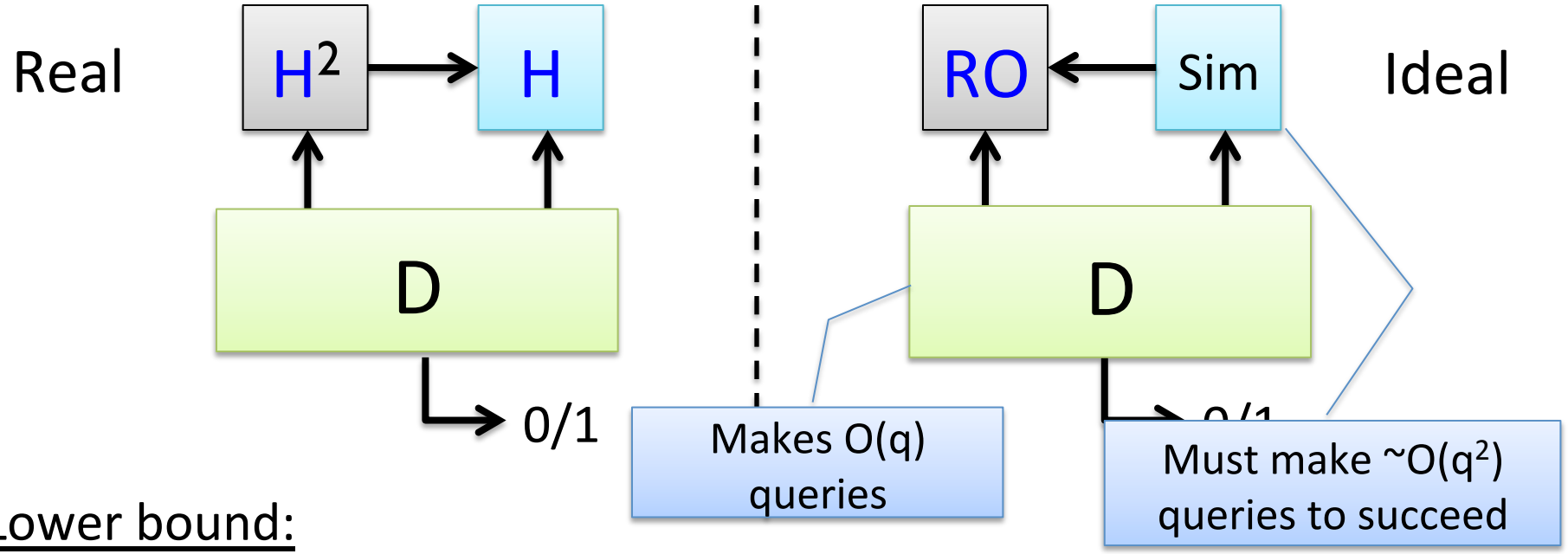
M' = H(M)

Output of H^2 "leaks" an internal value for another hash computation

Worrisome:

- Prior indifferentiable constructions avoid leaking intermediate values
- Extension attacks abused such leaks

Indifferentiability lower and upper bounds (simplified)



Lower bound:

We give D making q_L left queries and q_R right queries such that for any Sim making q_s queries:

$$\mathrm{Adv}^{\mathrm{indiff}}(H^2, D, \mathrm{Sim}) > 1 - q_s / (q_L\, q_R)$$

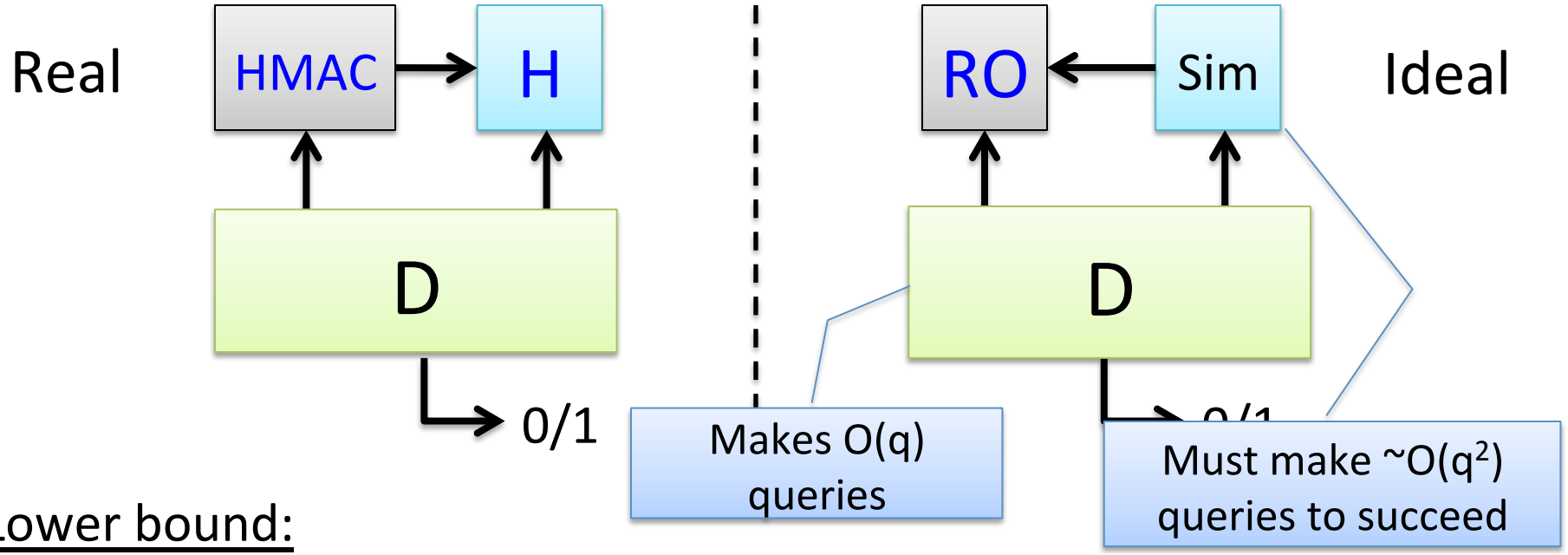
Upper bound:

We give Sim such that for any D making q queries

$$\mathrm{Adv}^{\mathrm{indiff}}(H^2, D, \mathrm{Sim}) < q^2 / 2^n$$

Sim makes $O(q^2)$ queries

Indifferentiability lower and upper bounds (simplified)



Lower bound:

We give D, making q_L left queries and q_R right queries including ambiguous key pairs, such that for any Sim making q_s queries:

$$Adv^{indiff}(HMAC,D,Sim) > 1 - q_s / (q_L\, q_R)$$

Upper bound: ???

Indifferentiability for ambiguous key pairs (but no colliding keys) probably holds with Sim making $O(q^2)$ queries. We don't have a proof.

Does q^2 versus q really matter?

In paper we provide an example:
mutual proofs of work protocol

vulnerable using H^2
or HMAC with
ambiguous key pairs

secure using RO

Vulnerabilities in practice?

Disk encryption uses PKCS#5:
HMAC(pw, salt)

???

TLS connection

OK! TLS handshake:
HMAC(g^{xy} , …)

OK! TLS record layer:
HMAC(K_{auth}, Ctxt)

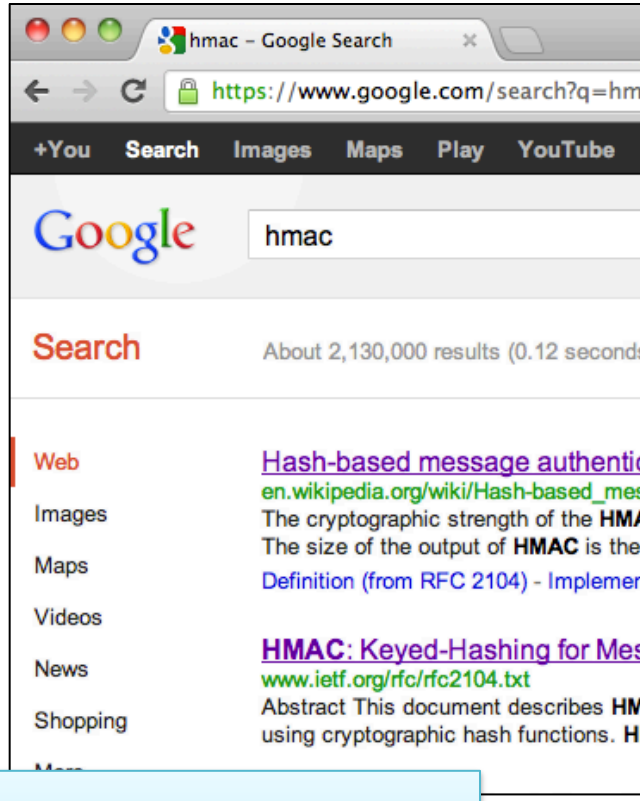
IPSec tunnel

IKEv2 (HKDF):
HMAC(nonces, g^{xy})

OK!

WPA WiFi protocol uses PKCS#5:
HMAC(pw, ssid)

???



Most applications already avoid weak key pairs

Discussion

Results extend to when H is an iterative hash function (e.g., SHA-1, SHA-256, …)

In theory, we can fix H^2 and HMAC, but deployment would be a … hurdle

Summary of our results

Uncover that HMAC has ***weak key pairs***

(1) Colliding key pairs $K \neq K'$ s.t. HMAC(K,M) = HMAC(K',M)

(2) Ambiguous key pairs no inner/outer H domain separation

$K \neq K'$ s.t. $K_{in} = K'_{out}$ $K'_{in} = K_{out}$

H^2 and HMAC w/ ambiguous key pairs have similar issues

- At best weak concrete security from indifferentiability
- Upper bound proof that H^2 is "weakly" indifferentiable
- Example (vulnerable) setting: mutual proofs of work

Avoid weak key pairs in HMAC, get strong indifferentiability