A New Pseudorandom Generator from Collision-Resistant Hash Functions

Alexandra Boldyreva & Virendra Kumar

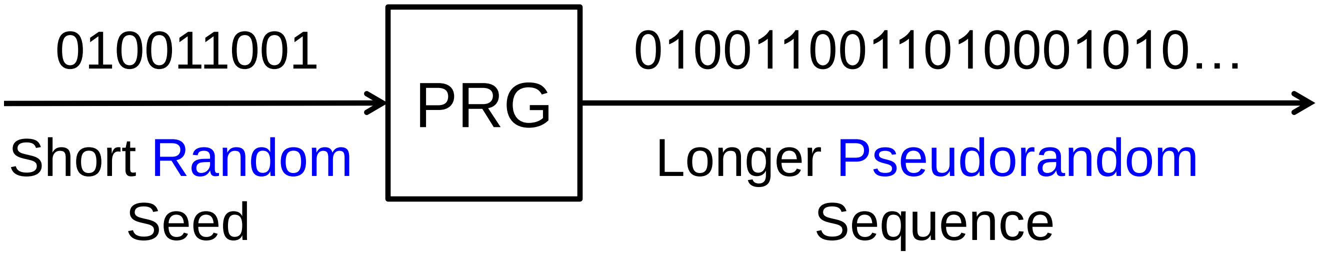
Georgia Institute of Technology

Session ID: CRYP-301

Session Classification: Advanced

RSA CONFERENCE 2012

What's a Pseudorandom Generator (PRG)?



- A bit string is truly random if all the possible values are equally likely.
- A bit string is pseudo random, if no polynomial-time machine can tell it apart from truly random.

Why are PRGs Important?

- If you use cryptographic schemes, then most likely you need to generate randomness
  - Cryptographic keys need to be random
  - Many schemes need random padding
  - Schemes involving passwords need random salt
  - And so on …
- Well, so what?
  - Improperly generated randomness can have devastating effects
  - Insecurity of Netscape's SSL Implementation, RSA Modulus Generation, etc. [GW '96, LHA$^+$ '12]

Uses of PRG

- Apply PRG on a small random seed, obtain large amounts of pseudorandom bits, and use them as keys, pads, salts, etc.
- PRGs can also be used to build more complex cryptographic primitives
  - Pseudorandom Functions [Goldreich et al. '86]
  - Bit Commitment [Naor '91]
  - And so on …

Our Objective

- To construct a PRG from a Hash Function (HF) that is:
  - Efficient
  - Provably secure
  - Relies on reasonable assumptions like Collision-Resistance (CR)
  - Instantiable from standard HFs like SHA-1

(HF is a function whose range is smaller than the domain, also referred as compression function)

Outline of the Talk

- Known Constructions of PRG
- Blum-Micali-Yao PRG
- Our PRG
  - Construction
  - Assumptions and Efficiency Improvement
  - Proof Idea
- Conclusion

Known Constructions of PRG: Why none of them meet our requirements?

Theoretical PRGs

- Introduced by Blum and Micali, later formalized into its current form by Yao (BMY PRG) in 1982
  - Most efficient among theoretical PRGs, but based on One-Way Permutation (OWP)
  - Hash function (HF) is clearly not a permutation, so BMY PRG doesn't meet our requirements
- Later constructions based on different types of One-Way Function (OWF) are all inefficient.

Relaxing the Permutation Requirement

- Regular one-way function [GKL '88, HHR '06]
  - Construction similar to BMY PRG, but additionally uses re-randomizing functions in every iteration
  - Larger seed length and computationally less efficient
- Any one-way function [HILL '99, Holenstein '06, HHR '06]
  - Extremely inefficient both computationally and in terms of seed length

(A function is regular, if the pre-image set of all the elements in the range are of the same size.)

Theoretical PRGs (Seed Length Comparison)

	Seed Length	Assumptions
Hastad et al. '99 (Holenstein '06)	$O(m^8)$	Any OWF
Haitner et al. '06	$O(m^7)$	Any OWF
Holenstein '06	$O(m^5)$	Exponential OWF
Haitner et al. '06	$O(m^2)$	Exponential OWF
Goldreich et al. '88	$O(m^3)$	Regular OWF
Haitner et al. '06	$O(m \log m)$	Regular OWF
Blum-Micali-Yao '82	$O(m)$	OWP

m is the input length of the underlying function.

Security vs. Seed Length

Say, we only trust OWF with input size 128 bits to be secure by current standards.

Then, Hastad et al.'s PRG is secure only for seed of size (ignoring constants) at least 2^{56} bits!

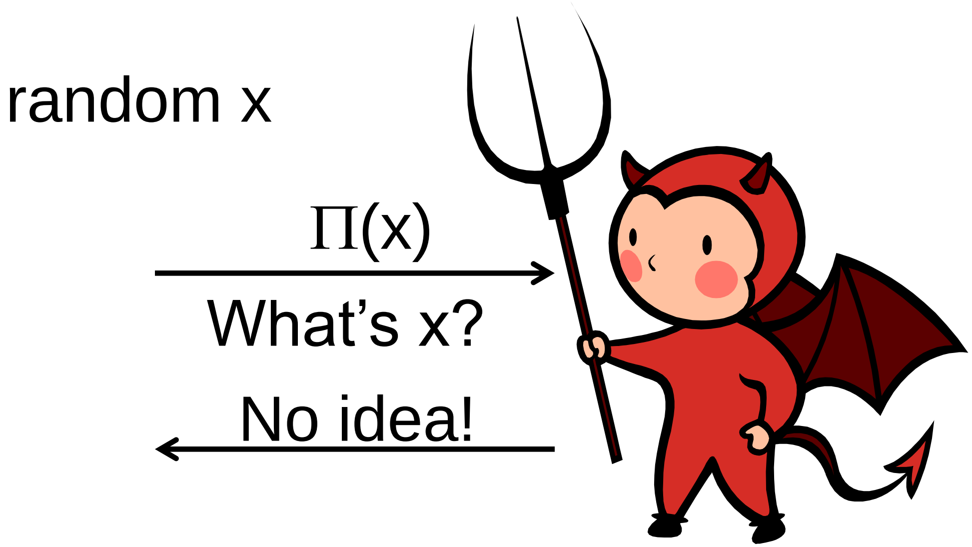
Standardized PRGs Based on HFs [FIPS '94]

- Very efficient
- Small seed length
- Security proof relies on strong assumptions that are not very well studied
  - Underlying function is assumed to be a Pseudo Random Function (PRF)
  - Unreasonable for HF-based PRGs, as HFs not only don't have secret keys, but are usually keyless

Blum-Micali-Yao (BMY) PRG: Construction and Proof Idea

One-Way Permutation (OWP)

- A Permutation Π is One-Way if it is hard to invert the value of Π on a randomly selected input.



Hardcore Bit (HCB)

- A bit of a randomly selected input, say x[i], is HCB w.r.t. a function Π, if it is hard to compute w.p. better than a random guess, given the function's value on that input.

random x

Π(x)

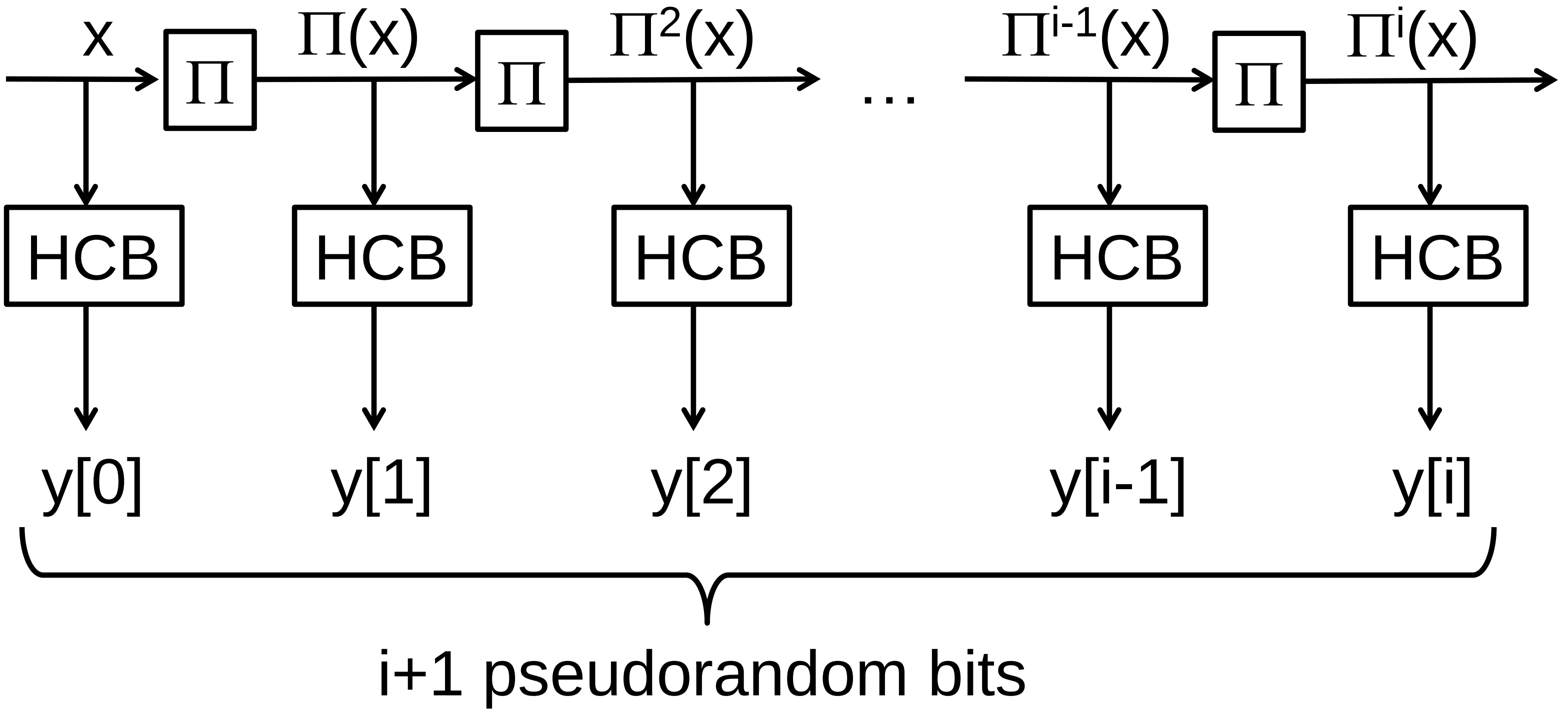
What's x[i]?

Is it 0?

Wait, is it 1?

Blum-Micali-Yao (BMY) PRG

One-Way Permutation Π, random seed x



Goldreich-Levin Hardcore Bit [GL '89]

- For any OWF Φ and random (x, r), the inner product of x and r, $\langle x, r \rangle$ is the HCB of (Φ(x), r).
- Using GL hardcore bit, BMY PRG is:

$$\langle x,r\rangle, \langle \Pi(x),r\rangle, \langle \Pi^2(x),r\rangle, \ldots, \langle \Pi^{i-1}(x),r\rangle,$$

where Π needs to be one-way on iterates, and (x,r) need to be random.

Proof Idea of BMY PRG

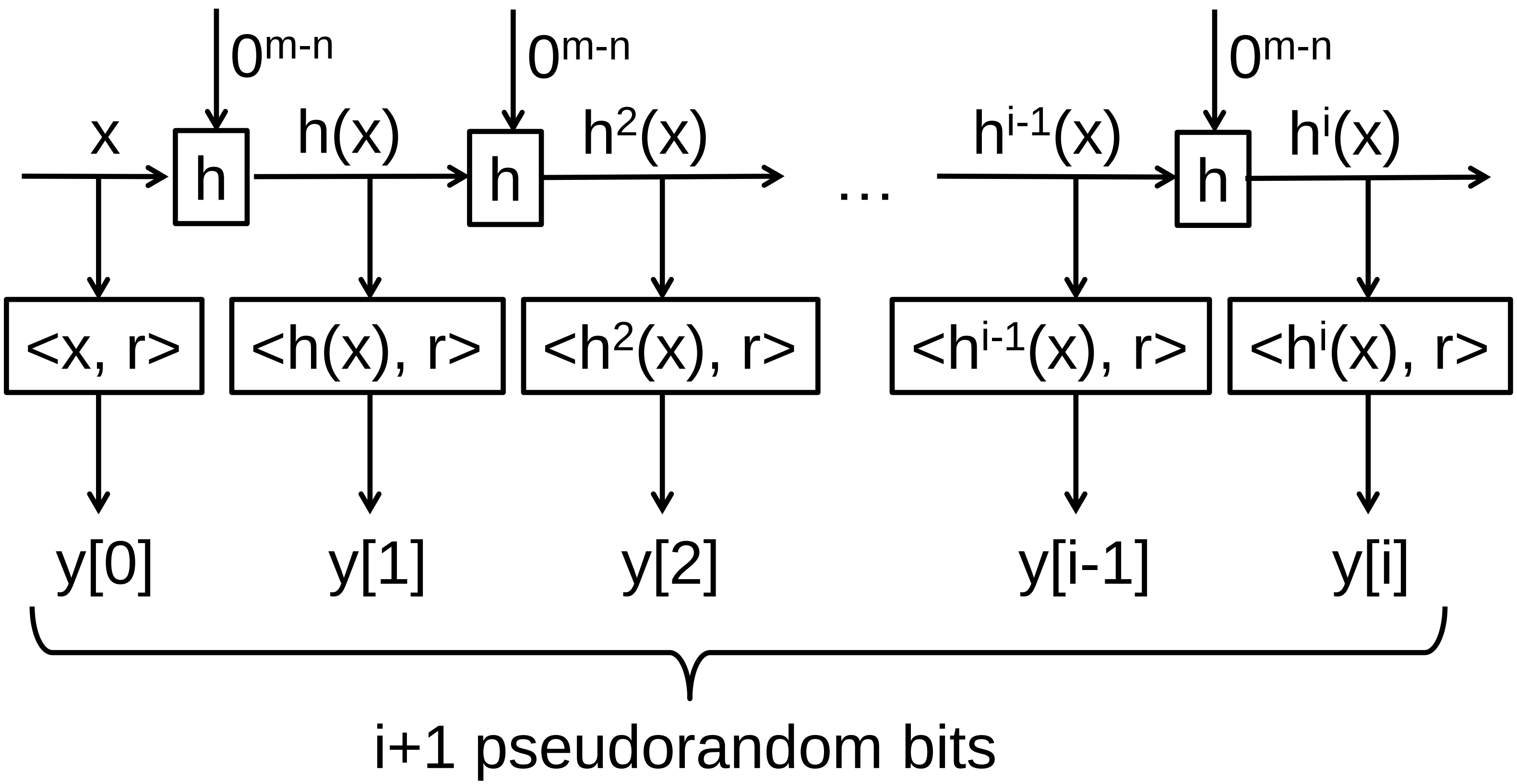
- If x is random and Π is a permutation, then $\Pi(x)$ is also random, and so is $\Pi^i(x)$ for any i.
- Hence, if Π is one-way, then it is also one-way on iterates (OWI), i.e. given $\Pi^i(x)$ for a random x, it's hard to compute $\Pi^{i-1}(x)$.
- Therefore, given $\Pi^i(x)$ for a random x, (y[0], y[1], …, y[i-1]) are HCBs and hence pseudorandom.
- Above proof was given by Goldreich et al. after Levin observed that OWI is sufficient, i.e. don't need OWP.

Our PRG: Construction

RSA CONFERENCE 2012

Our PRG Construction

- Hash function, h: $\{0, 1\}^m \rightarrow \{0, 1\}^n$, seed (x, r)



Highlights of our Construction

An efficient PRG from HFs (m bits → n bits)

- Seed length = 2n, i.e. as efficient as BMY PRG
- Assumptions: Collision-resistance (CR) and regularity
  - CR must be exponential
  - Regularity can be relaxed to worst-case regularity, a new notion introduced in this work
- Improvement from Goldreich et al. and Haitner et al.: We don't need re-randomizing functions, resulting in a much smaller seed

Our PRG: Assumptions and Efficiency Improvement

Exponential Collision-Resistance (CR)

- A hash function family H is ε-CR, if any efficient adversary given a random instance h of H, can return a collision with probability at most ε.
- A collision for h is a pair (x, y) s.t. $x \neq y$ and $h(x) = h(y)$.
- We need $\varepsilon < 2^{-n/2}$, where n is the output size of h.

(Exponential collision-resistance is not a new assumption, but only requires a more strict bound on the probability of finding collisions.)

Worst-Case Regularity

- A function $f: \{0, 1\}^m \rightarrow \{0, 1\}^n$ is regular if the pre-image set of every element in the range is of equal size 2^{m-n}.
- For an $0 < \alpha \leq 1$, f is α-worst-case regular, if the pre-image set of every element in the range is of size at least $\alpha \cdot 2^{m-n}$.

Efficiency Improvement [GKL '88]

- Instead of one hardcore bit, extract log n bits in every iteration
  - Pick r of size (n + log n), and denote
    $r_1 = r[1] || \ldots || r[n]$,
    $r_2 = r[2] || \ldots || r[n+1]$, and so on,
    $r_{\log n} = r[\log n] || \ldots || r[n + \log n - 1]$
  - At every iteration i, output $<\bullet, r_1>, <\bullet, r_2>, \ldots, <\bullet, r_{\log n}>$
  - With almost the same computation, we now have log n times more pseudorandom bits

Our PRG: Proof Idea

RSA CONFERENCE 2012

Proof Idea

- We need to show that HF is OWI, remaining part is similar to the proof of BMY PRG.
- CR alone is not sufficient
  - To show that CR implies OWI, we need to show: ∃ adversary breaking OWI ⇒ ∃ adversary breaking CR
  - Counterexample for above: a CRHF that becomes a permutation after one application
  - If $h^2(x)$ is a permutation over $h^1(x)$, then $h^{-1}(h^2(x))$ is always unique, so the output of OWI adversary can't be used by CR adversary to find collisions in HF.

Proof Idea Contd.

- Instead of re-randomizing functions as used in Goldreich et al. and Haitner et al., we rely on exponential CR
  - We use the exponential CR to lower bound the output size of our hash function iterate
  - HF is clearly not a permutation, so the output size of its subset iterate can potentially become small and thus be easily invertible, i.e. HF may not be OWI
- Proof requires regularity of HF, which we relax with our new notion: worst case regularity.

Conclusion

How our PRG compares with prior PRGs?

- Theoretical ones
  - As efficient as BMY PRG, the most efficient theoretical PRG, furthermore does not need OWP
  - Can be instantiated with hash functions like SHA-1
- Practical and Standardized ones
  - Comparable seed length, but computationally not as efficient, due to hardcore bit computation
  - Active interest by practitioners in finding collisions means collision-resistance is a more reasonable and understood assumption on hash functions than PRF

Conclusion

- We proposed an efficient PRG based on hash functions.
- Our PRG is accompanied with concrete security proofs and relies on standard and reasonable assumptions.
- Our PRG is a step toward making theoretical PRGs practical, but still not as efficient as standardized ones.
- We introduced a new notion relaxing regularity of a function, called worst-case regularity.

Thanks!

PMAC with Parity: Minimizing the Query-Length Influence

Kan Yasuda (NTT, Japan)

CT-RSA 2012

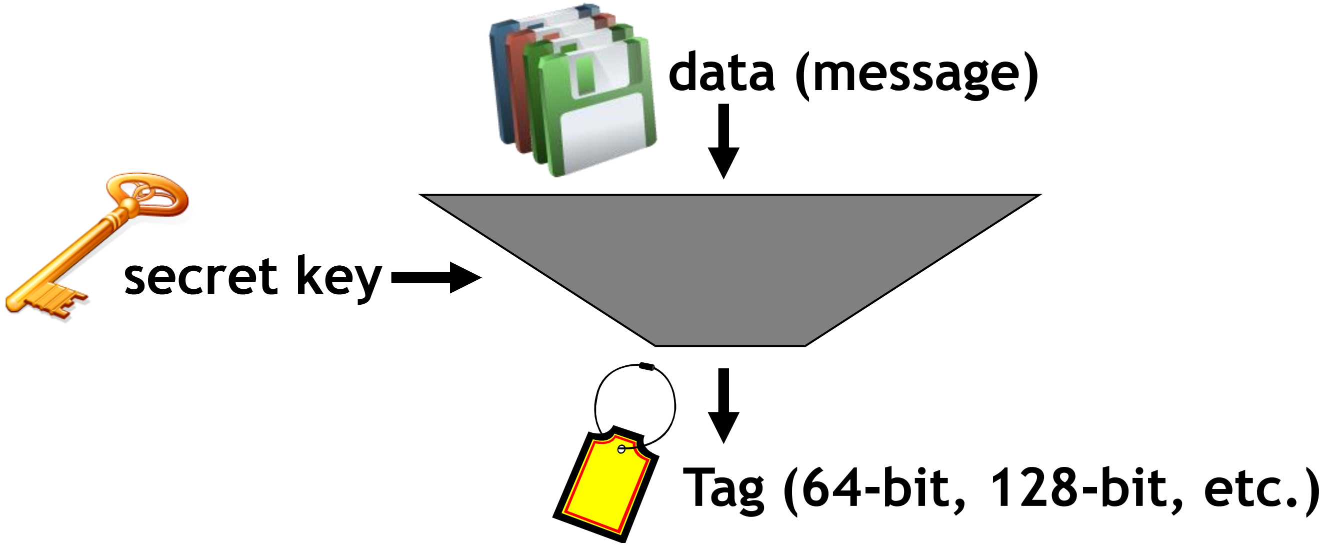
Feb. 27 – March 2

Outline

- Introduction to MACs
- Motivation: Query complexity
- Previous approaches
- New approach: PMAC with parity
- Open problems

(Deterministic) MAC

- MAC (Message Authentication Code)
  - Symmetric-key primitive
  - Input: a secret key and (possibly large) data
  - Output: a fixed-length value (called tag)
  - Used for integrity check of data



4 ways to make a MAC

- 1. Design from scratch (dedicated MAC)
- 2. Use a cryptographic hash function (e.g., HMAC)
- 3. Use a universal hash function
- 4. Use a blockcipher (e.g., CMAC, PMAC)

4 ways to make a MAC

- 1. Design from scratch (dedicated MAC)
- 2. Use a cryptographic hash function (e.g., HMAC)
- 3. Use a universal hash function
- 4. Use a blockcipher (e.g., CMAC, PMAC)

We focus on
blockcipher-based construction
(how to iterate a blockcipher)

Outline

- Introduction to MACs
- Motivation: Query complexity
- Previous approaches
- New approach: PMAC with parity
- Open problems

Security of MACs

- "Secure" means secure against adversaries having up to certain resources
- Adversarial resources measured in terms of
  - Time (and memory) complexity
    - The running time of adversary
  - Query complexity
    - The amount of queries to the MAC oracle made by adversary

Time vs. Query

- Time complexity
  - Essentially determined by key length
  - "Single-key" MAC construction: key length is equal to that of blockcipher (80-bit, 128-bit, 192-bit, 256-bit, etc.)
- Query complexity
  - Heavily depends on which MAC construction one uses
  - Also depends on block size (64-bit or 128-bit)

Time vs. Query

- Time complexity
  - Essentially determined by key length
  - "Single-key" MAC construction: key length is equal to that of blockcipher (80-bit, 128-bit, 192-bit, 256-bit, etc.)
- Query complexity

Our focus

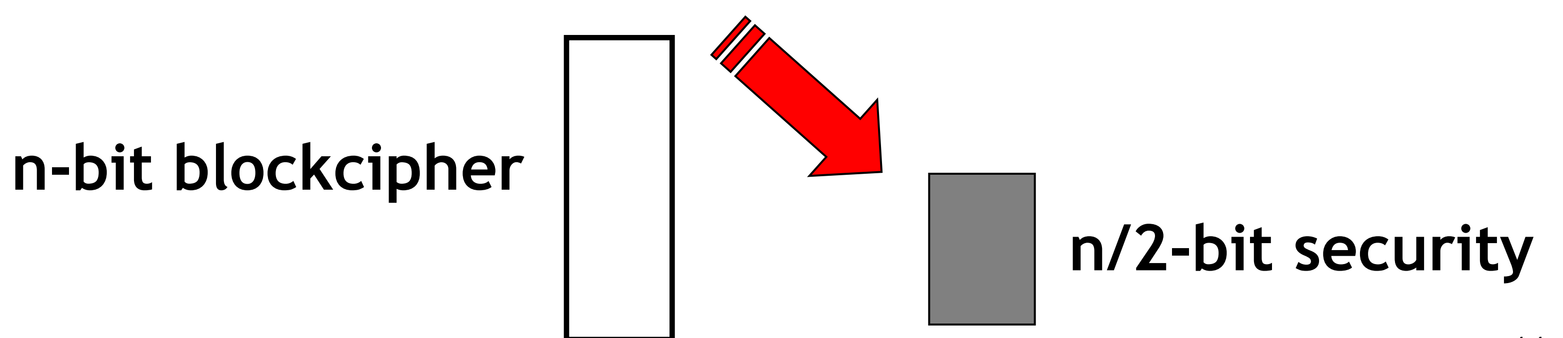
  - Heavily depends on which MAC construction one uses
  - Also depends on block size (64-bit or 128-bit)

Query complexity

- Two factors of query complexity
  - q: The # of times adversary can make queries
  - ℓ: The max length of each query

Birthday bound

- Majority of (blockcipher-based) MACs have "birthday-bound" security
- Using n-bit blockcipher, birthday-bound security offers n/2-bit security
  - AES 128-bit blockcipher, 64-bit security
  - DES 64-bit blockcipher, 32-bit security



32-bit security for 64-bit blockciphers (in terms of q and ℓ)

- 64-bit blockciphers still widely used, or newly devleoped as "lightweight" algorithms
  - Triple-DES, HIGHT, PRESENT, LED, . . .

- 32-bit security comes from $O(\ell^2 q^2/2^{64})$ bound
  - $q = 2^{32}$ corresponds to 136 years if executed every second
  - $\ell = 2^{32}$ corresponds to 32GByte

32-bit security for 64-bit blockciphers (in terms of q and ℓ)

- 64-bit blockciphers still widely used, or newly devleoped as "lightweight" algorithms
  - Triple-DES, HIGHT, PRESENT, LED, . . .

- 32-bit security comes from $O(\ell^2 q^2/2^{64})$ bound
  - $q = 2^{32}$ corresponds to 136 years if executed every second
  - $\ell = 2^{32}$ corresponds to 32GByte

Practical limit
We want to remove this

Our motivation:

Outline

- Introduction to MACs
- Motivation: Query complexity
- Previous approaches
- New approach: PMAC with parity
- Open problems

Previous work

- 1) Some work improved proofs and bounds over $O(\ell^2 q^2/2^n)$ for existing MAC schemes

- 2) Some work provided new constructions (including "beyond-birthday-bound" MACs)

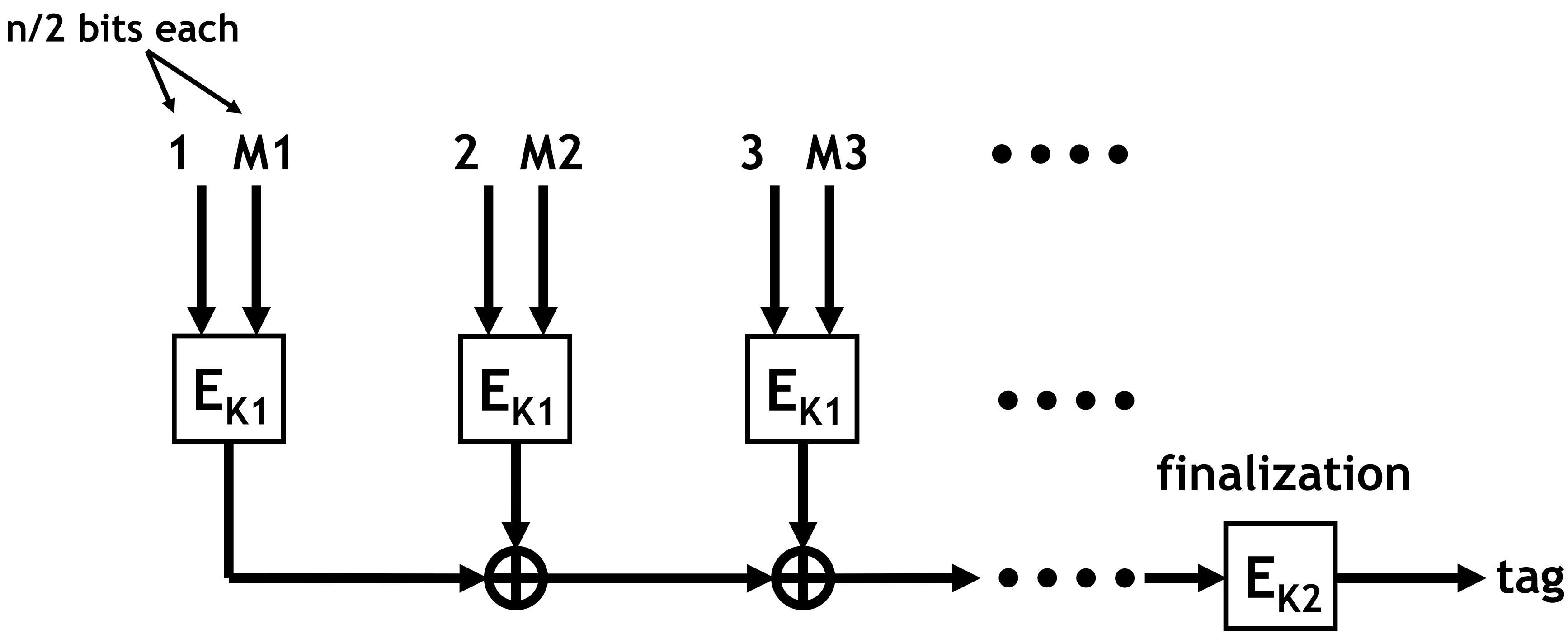
1) Improving over $O(\ell^2 q^2/2^n)$

- Better proofs yield better bounds
- Improves only ℓ-factor (there exit attacks at $q = 2^{n/2}$)
- Previous work:
  - $O(\sigma^2/2^n)$ is possible, where $\sigma \leq \ell q$ is the total query complexity (length)
  - $O(\ell q^2/2^n)$ for CBC MAC [CRYPTO 2005] and for PMAC [FSE 2006]
  - $O(\sigma q/2^n)$ for various MACs [FSE 2010]

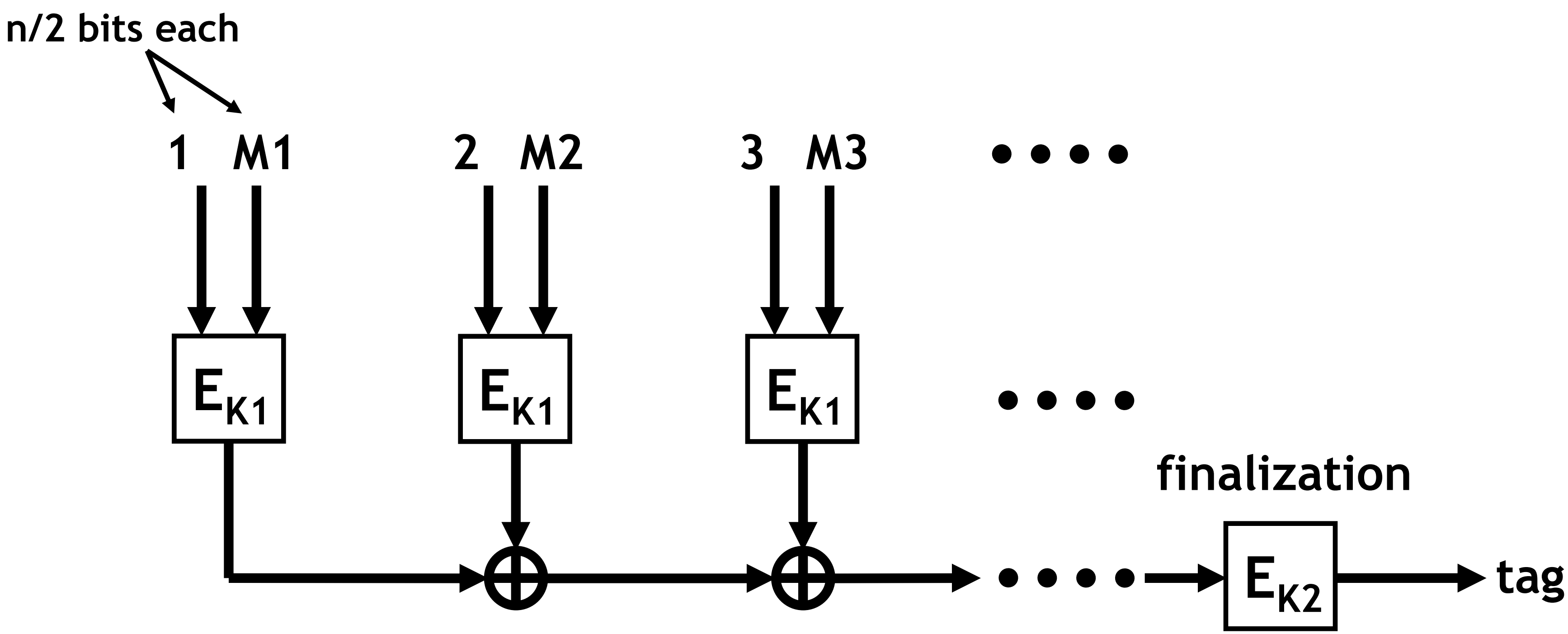
2) Previous work (new constructions)

- Use randomization for improving ℓ-factor [FSE 2007]
- "Beyond-birthday-bound" constructions for improving q-factor, $O(\ell^3 q^3/2^{2n})$ [CT-RSA 2010, CRYPTO 2011]
- Counter method (e.g., XOR MAC, PCS) for improving ℓ-factor

Counter method



Counter method



- Offers $O(q^2/2^n)$ security, no ℓ-factor
- Twice as slow as usual --- rate 1/2
- Data size up to $2^{n/2}$ blocks, cannot handle longer messages

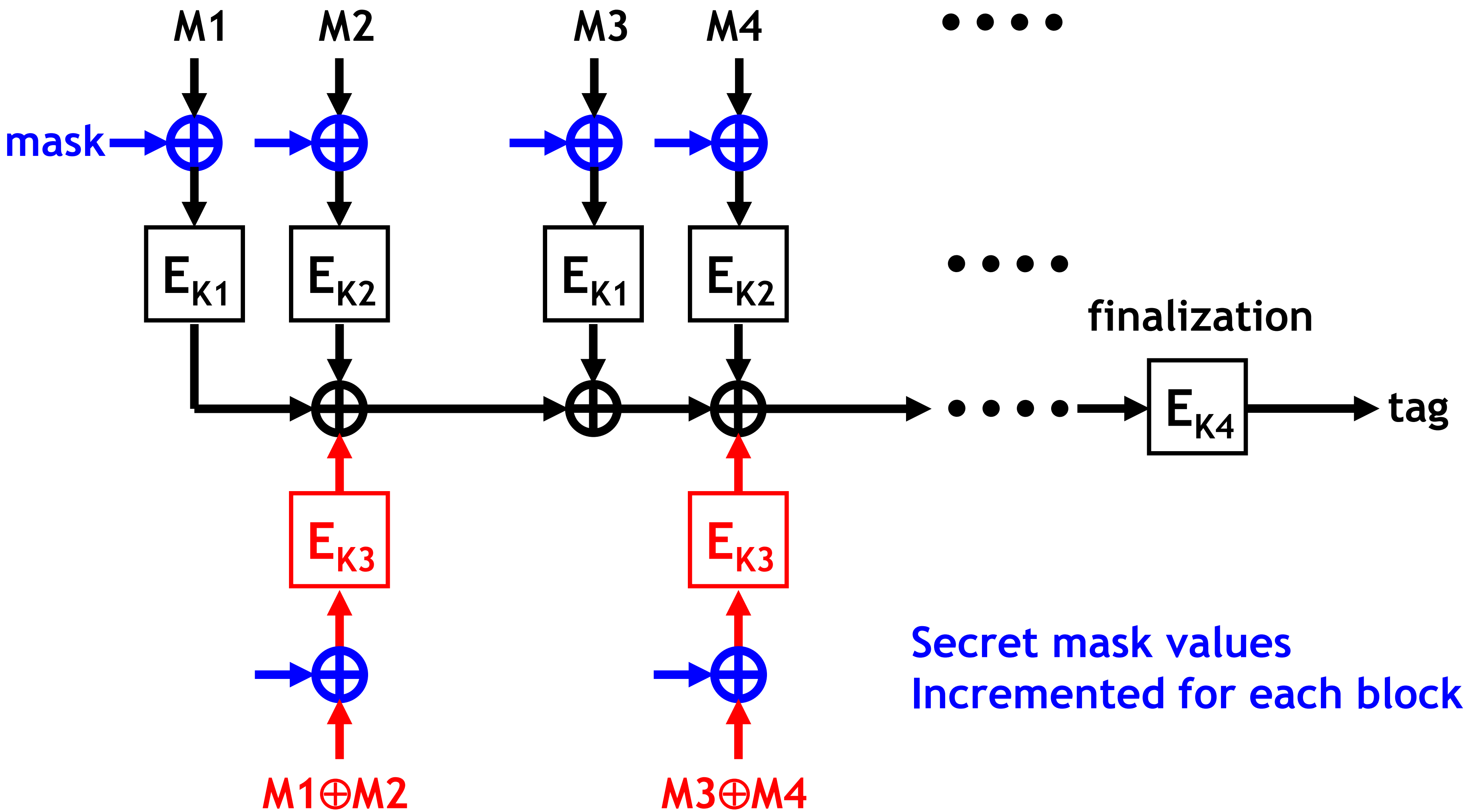
Outline

- Introduction to MACs
- Motivation: Query complexity
- Previous approaches
- New approach: PMAC with parity
- Open problems

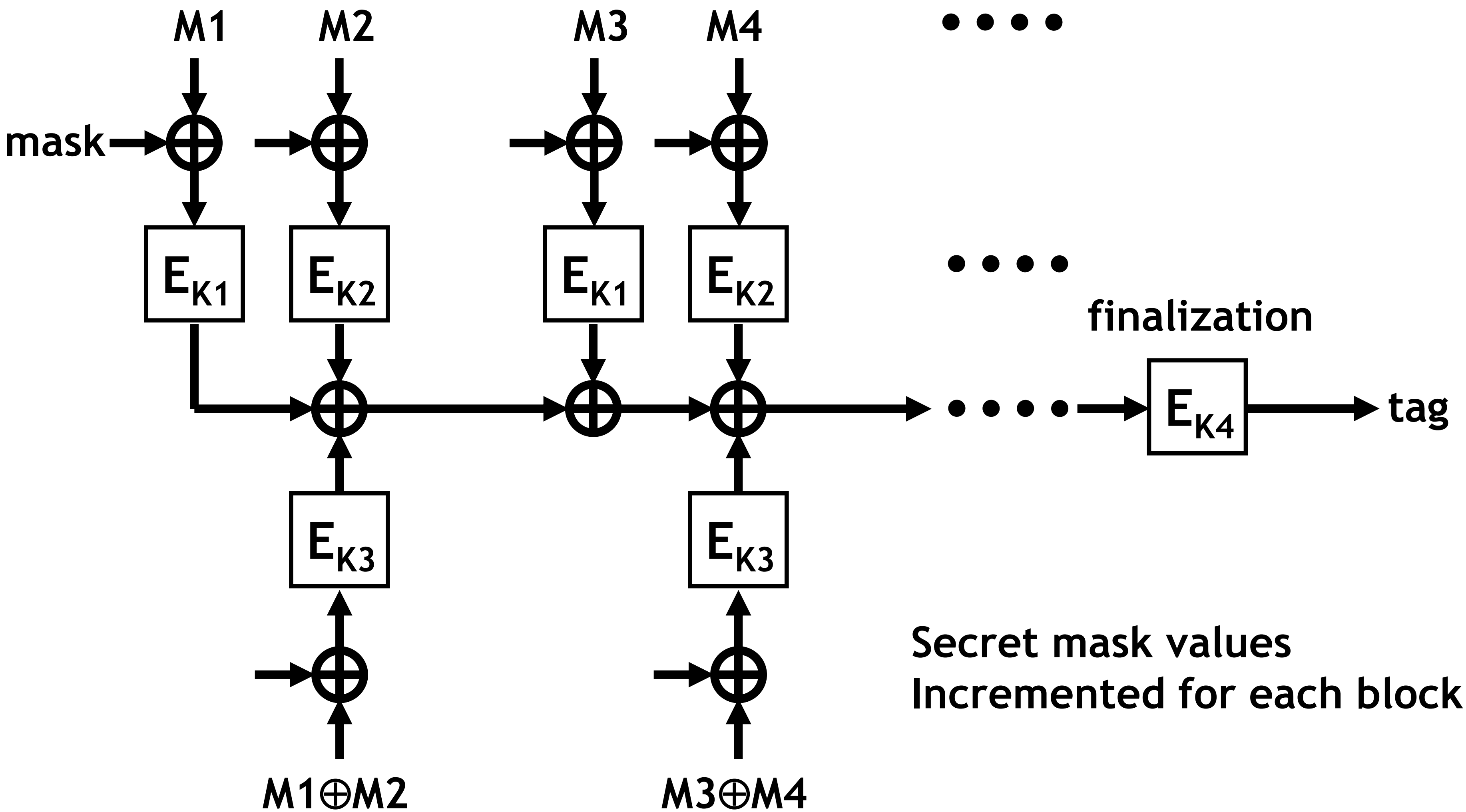
New approach

- We want to do better than the counter method
- We provide a new construction:
  - ✔ Faster than rate 1/2 : rate 2/3 or better
  - ✔ Can handle messages longer than $2^{n/2}$ blocks
  - ✔ Provided with bound $O(q^2/2^n+\ell\sigma q/2^{2n})$
    - Becomes $O(q^2/2^n)$ if $\ell \le 2^{n/2}$
    - Better than $O(\sigma q/2^n)$
    - Even better than $O(\ell^3 q^3/2^{2n})$ for $\ell \ge 2^{n/6}$

PMAC with parity

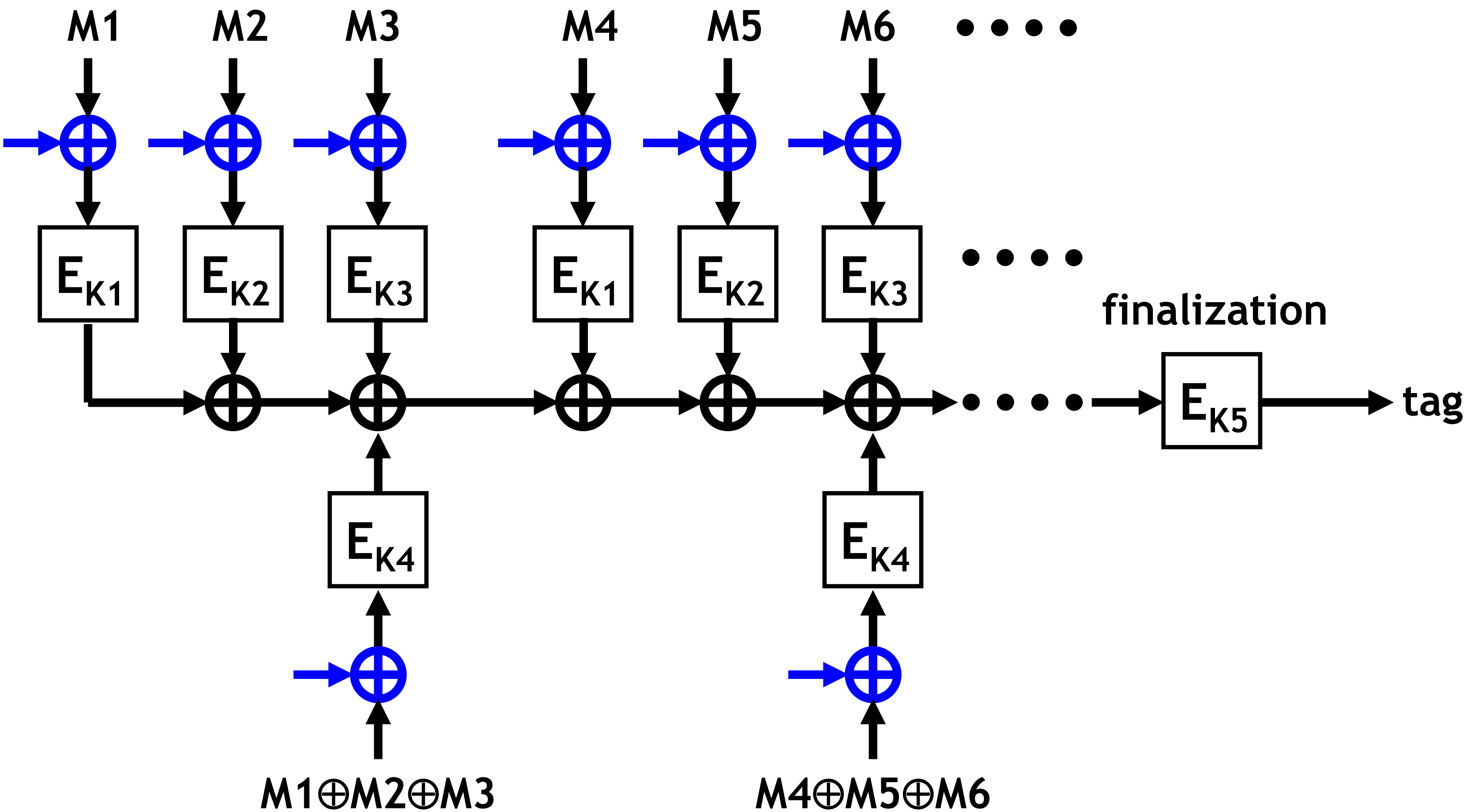


PMAC with parity

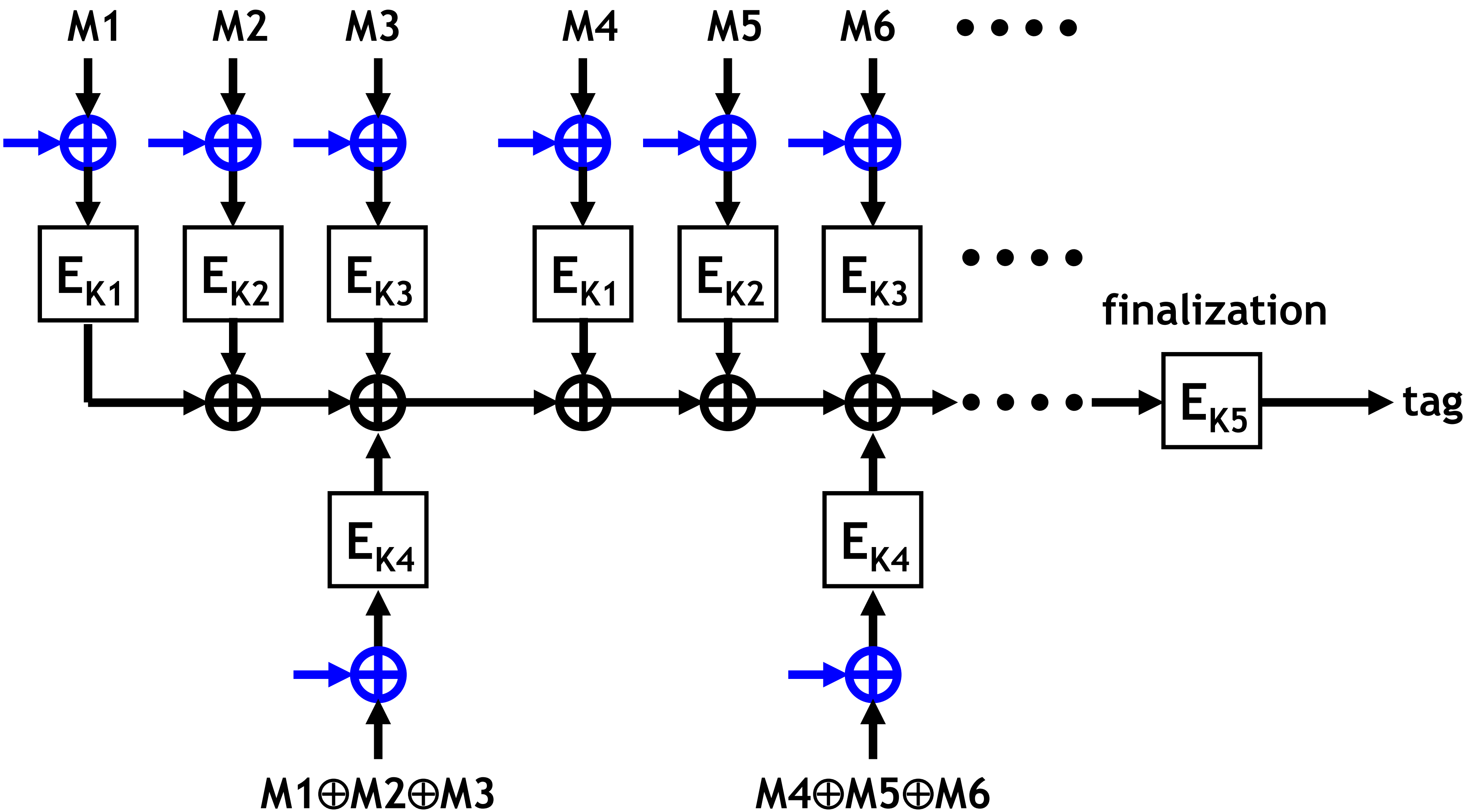


$O(q^2/2^n+\ell\sigma q/2^{2n})$ proof technique:
PMAC and Multilane HMAC [Indocrypt 2007]

Rate 3/4 version



Rate 3/4 version



Same $O(q^2/2^n+\ell\sigma q/2^{2n})$ security (but more keys)

Outline

- Introduction to MACs
- Motivation: Query complexity
- Previous approaches
- New approach: PMAC with parity
- Open problems

A couple of open problems

- Single-key construction?
  - PMAC with parity uses (at least) 4 different blockcipher keys
  - Running a blockcipher with different keys may be costly
- Rate-1 construction?
  - PMAC with parity is not rate-1 but rate 2/3, 3/4, 4/5, …
  - Desirable to be rate-1, i.e., one blockciper call per message block

Thank you

Boomerang Attacks against ARX Hash Functions

Gaëtan Leurent & Arnab Roy

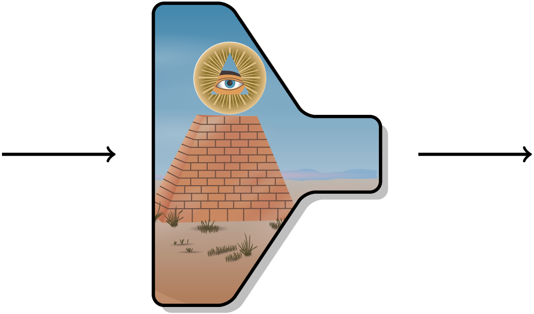
Gaëtan Leurent
University of Luxembourg

Session ID: CRYP-301
Session Classification: Advanced

RSACONFERENCE2012

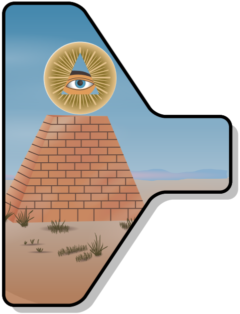
Introduction to Hash Functions

An Ideal Hash Function: the Random Oracle



- Public Random Oracle
- The output can be used as a fingerprint of the document

An Ideal Hash Function: the Random Oracle



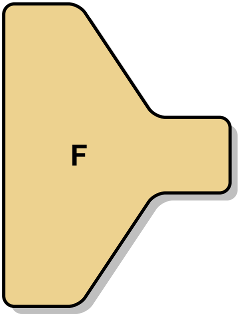
0x1d66ca77ab361c6f

- Public Random Oracle
- The output can be used as a fingerprint of the document

A Concrete Hash Function

- A public function with no structural property.
  - Should behave like a random function.
  - Cryptographic strength without any key!

- $F : \{0,1\}^* \to \{0,1\}^n$

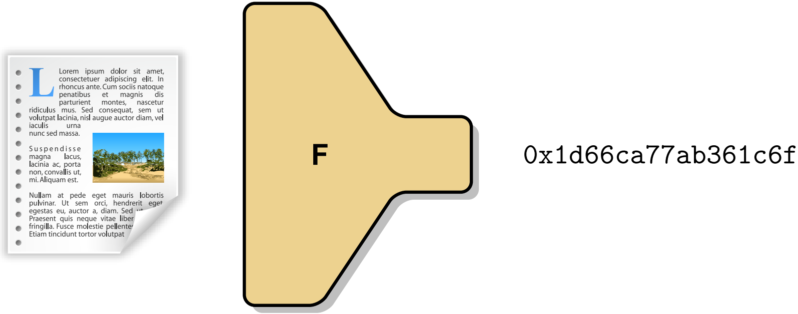


0x1d66ca77ab361c6f

A Concrete Hash Function

- A public function with no structural property.
  - Should behave like a random function.
  - Cryptographic strength without any key!

- $F : \{0,1\}^* \to \{0,1\}^n$



Using Hash Functions

Hash functions are used in many different contexts:

- To generate unique identifiers
  - Hash-and-sign signatures
  - Commitment schemes
- As a one-way function
  - One-Time-Passwords
  - Forward security
- To break the structure of the input
  - Entropy extractors
  - Key derivation
  - Pseudo-random number generator
- To build MACs
  - HMAC
  - Challenge/response authentication

The SHA-3 Competition

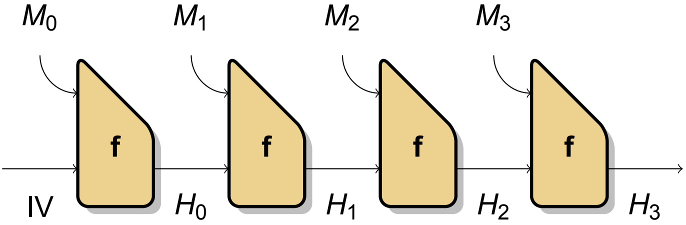
After Wang *et al.*'s attacks on the MD/SHA family,
we need new hash functions

The SHA-3 competition

- Organized by NIST
- Similar to the AES competition
- Submission deadline was October 2008: 64 candidiates
- 51 valid submissions
- 14 in the second round (July 2009)
- 5 finalists in December 2010:
  - Blake, Grøstl, JH, Keccak, Skein
- Winner in 2012?

Hash Function Design

- Build a small compression function, and iterate.
  - Cut the message in chunks $M_0, \ldots M_k$
  - $H_i = f(M_i, H_{-1})$
  - $F(M) = H_k$

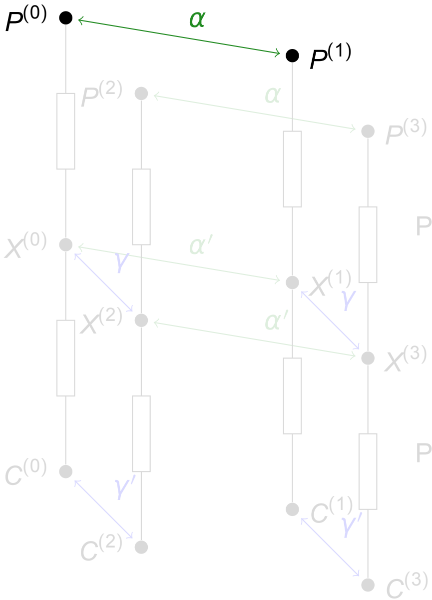


Boomerang Attacks

Boomerang Attacks

- Introduced by Wagner, many later improvements
- Combine two short differentials instead of using a long one.
  - $f = f_b \circ f_a$
  - for f_a, $\alpha \to \alpha'$ with probability p_a
  - for f_b, $\gamma \to \gamma'$ with probability p_b
  - Interesting when we don't know how to build iterative differentials.
- Uses an encryption oracle together with a decryption oracle
  - Adaptive attack

Boomerang Attacks

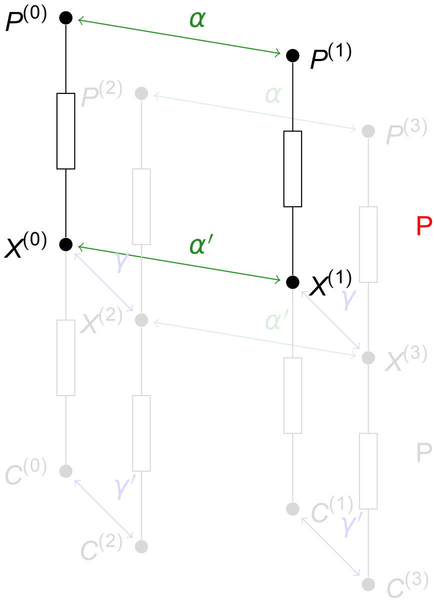


1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$\Pr[\alpha \to \alpha'] = p_a$

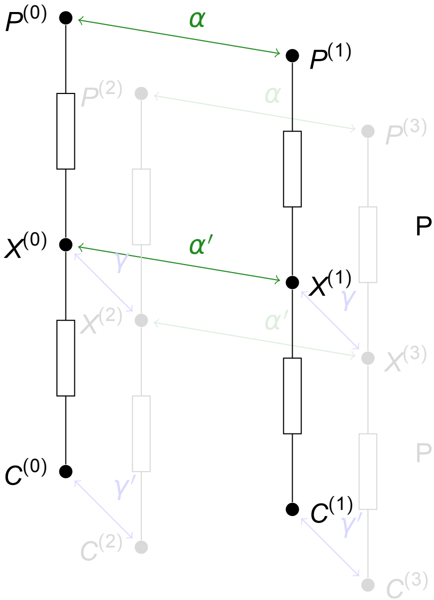
$\Pr[\gamma \to \gamma'] = p_b$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

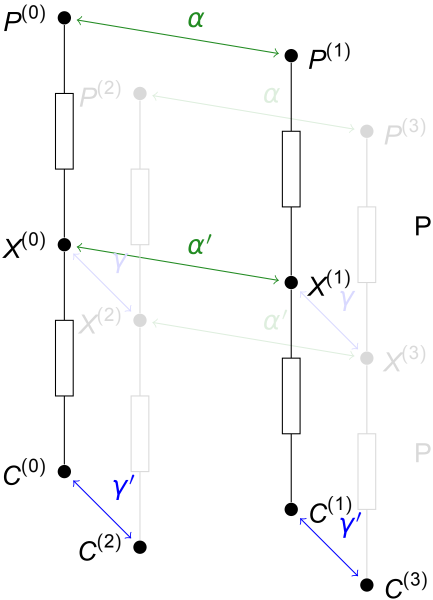
$$\Pr[\alpha \to \alpha'] = p_a$$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$\Pr[\gamma \to \gamma'] = p_b$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \to \alpha'] = p_a$$

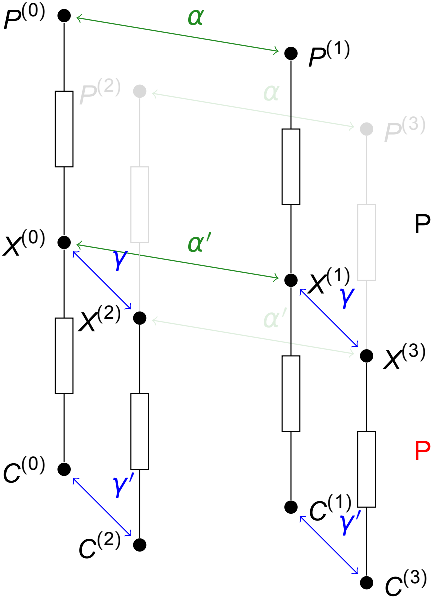
$$\Pr[\gamma \to \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$\Pr[\alpha \to \alpha'] = p_a$

$\Pr[\gamma \to \gamma'] = p_b$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a}\frac{1}{p_b^2}\frac{1}{p_a}$$

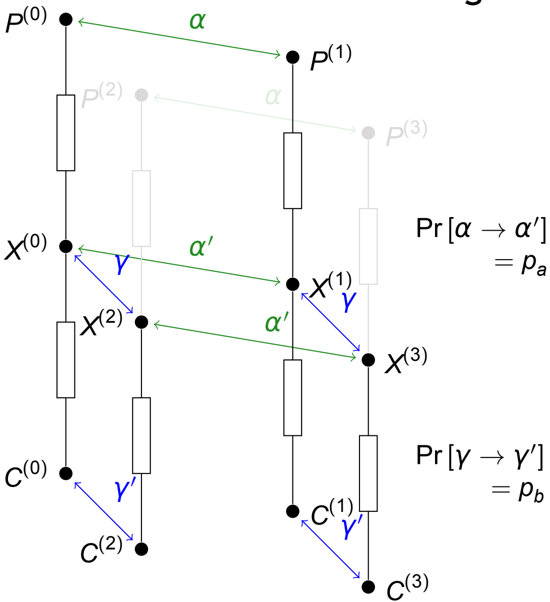
$P^{(0)} \oplus P^{(1)} = \alpha$

$P^{(2)} \oplus P^{(3)} = \alpha$

$C^{(0)} \oplus C^{(1)} = \gamma'$

$C^{(2)} \oplus C^{(3)} = \gamma'$

Boomerang Attacks

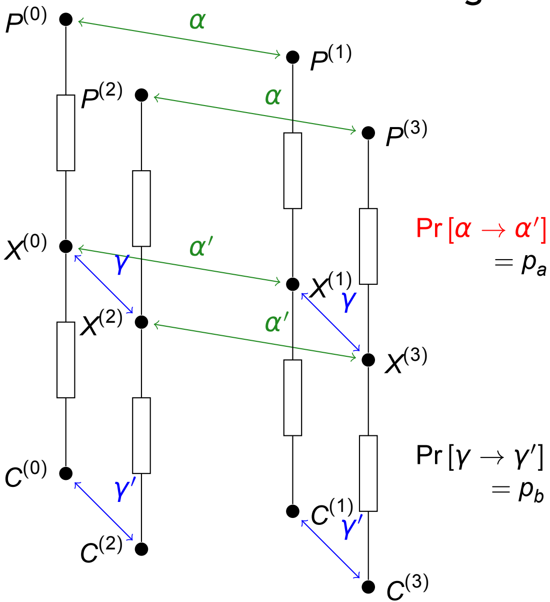


1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks

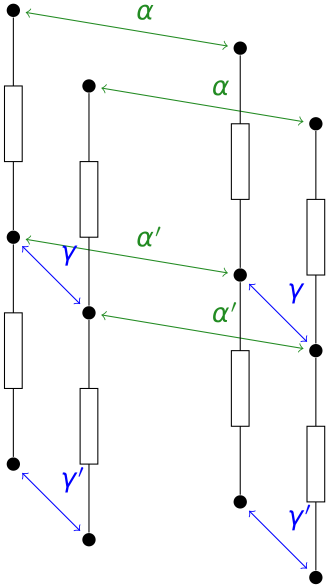


1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a}\frac{1}{p_b^2}\frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

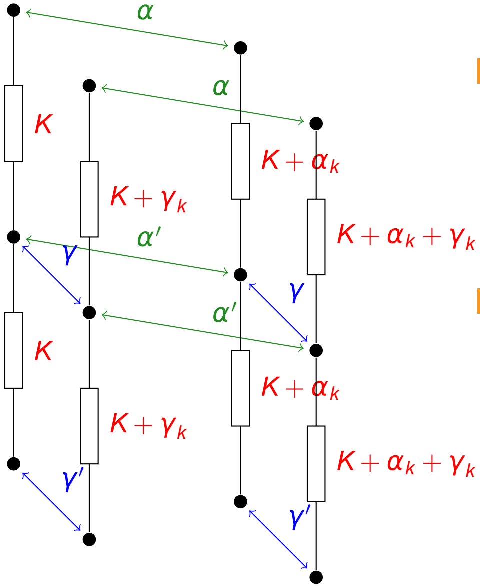
Improvements to the Boomerang Attack



1. Amplified probabilities
   - Do not specify α' and γ
   - $\hat{p}_a = \sqrt{\sum_{\alpha'} \Pr[\alpha \to \alpha']}$
     $\hat{p}_b = \sqrt{\sum_{\gamma} \Pr[\gamma \to \gamma']}$

2. Related-key
   - $p_a = \Pr\left[\alpha \xrightarrow{\alpha_k} \alpha'\right]$
     $p_b = \Pr\left[\gamma \xrightarrow{\gamma_k} \gamma'\right]$

Improvements to the Boomerang Attack



1. Amplified probabilities
   - Do not specify α' and γ
   - $\hat{p}_a = \sqrt{\sum_{\alpha'} \Pr[\alpha \to \alpha']}$
     $\hat{p}_b = \sqrt{\sum_{\gamma} \Pr[\gamma \to \gamma']}$

2. Related-key
   - $p_a = \Pr\left[\alpha \xrightarrow{\alpha_k} \alpha'\right]$
     $p_b = \Pr\left[\gamma \xrightarrow{\gamma_k} \gamma'\right]$

Boomerang Attacks on Hash Functions

- Most hash functions are based on a block cipher:

  Davies-Meyer $f(h, m) = E_m(h) \oplus h$

  Matyas-Meyer-Oseas $f(h, m) = E_h(m) \oplus m$

- A (related-key) boomerang attack gives a quartet:

$$\sum P^{(i)} = 0 \qquad \sum C^{(i)} = 0 \qquad \sum K^{(i)} = 0$$

- This is a zero-sum for the compression function:

$$\sum h^{(i)} = 0 \qquad \sum m^{(i)} = 0 \qquad \sum f(h^{(i)}, m^{(i)}) = 0$$

- In general this is hard:
  - $\sum f(h, m) = 0$, best attack $2^{n/3}$, lower bound $2^{n/4}$
  - $\sum f(h, m) = \sum h = \sum m = 0$, best attack $2^{n/2}$, lower bound $2^{n/3}$

- With a known key, one can start from the middle
  - Message modification

New Technique:
Better Use of Degrees of Freedom in a Hash Function Setting.

Using Auxiliary Paths

- Divide f in three sub-functions: $f = f_c \circ f_b \circ f_a$
  - for f_a, $\alpha \to \alpha'$ with probability p_a
  - for f_b, $\beta_j \to \beta_j'$ with probability p_b
  - for f_c, $\gamma \to \gamma'$ with probability p_c

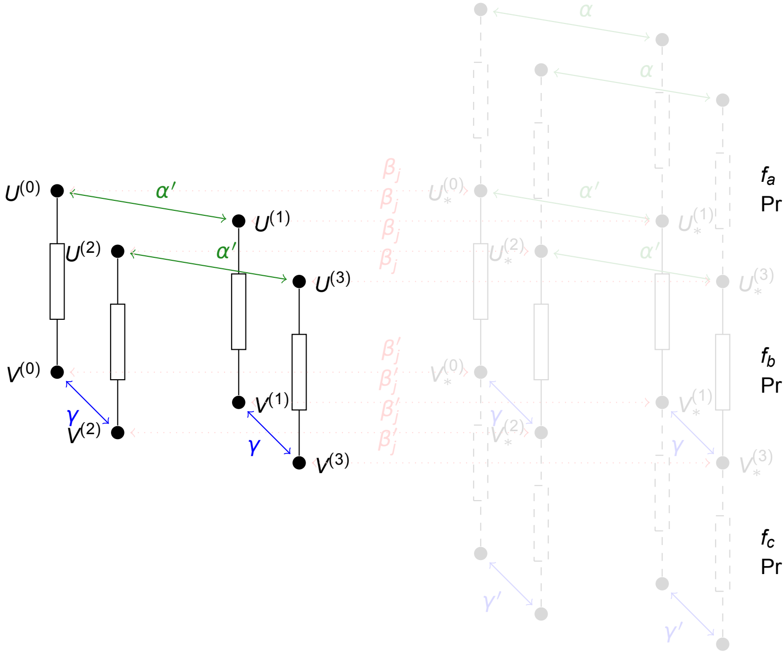
1. Start with a boomerang quartet for f_b:

$$U^{(1)} = U^{(0)} + \alpha' \qquad U^{(3)} = U^{(2)} + \alpha'$$
$$V^{(2)} = V^{(0)} + \gamma \qquad V^{(2)} = V^{(1)} + \gamma$$

2. For each auxiliary path, construct $U_*^{(i)} = U^{(i)} + \beta_j$.
   With probability p_b^4, $V_*^{(i)} = V^{(i)} + \beta_j'$, and we have a new quartet:

$$U_*^{(1)} = U_*^{(0)} + \alpha' \qquad U_*^{(3)} = U_*^{(2)} + \alpha'$$
$$V_*^{(2)} = V_*^{(0)} + \gamma \qquad V_*^{(2)} = V_*^{(1)} + \gamma$$

3. Check if the f_a and f_b paths are satisfied.



f_a

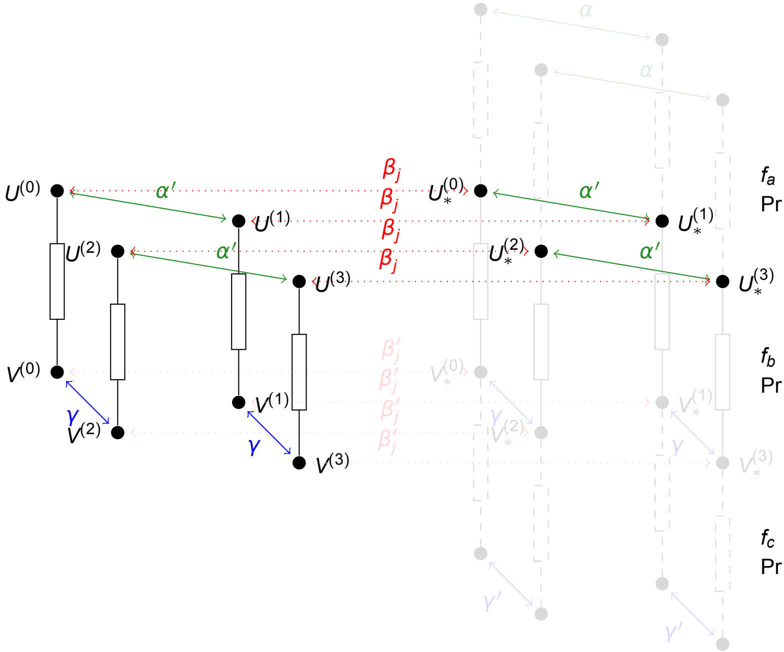
$\Pr[\alpha \to \alpha'] = p_a$

f_b

$\Pr\left[\beta_j \to \beta'_j\right] = p_b$

f_c

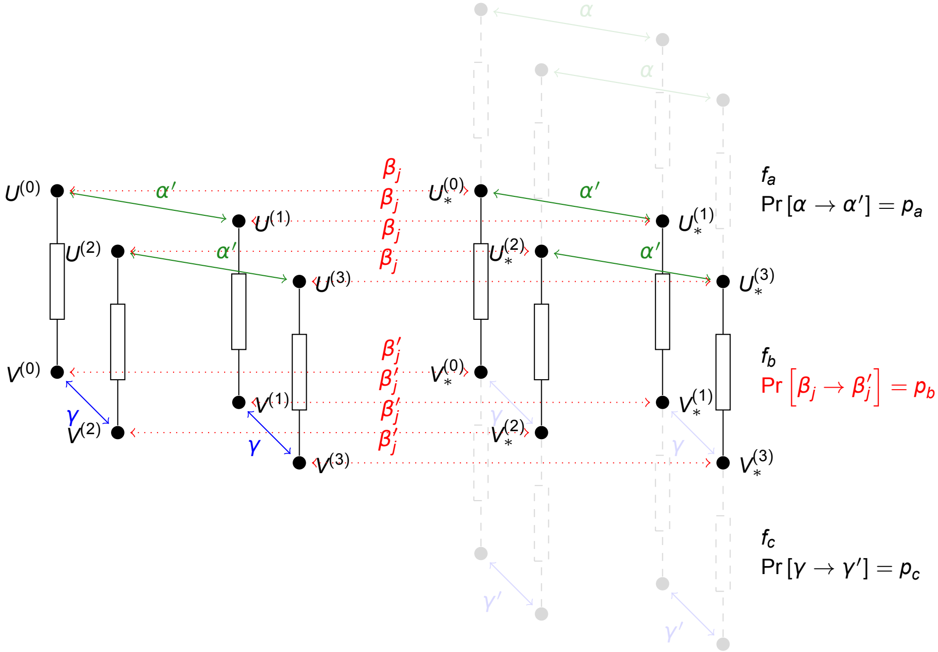
$\Pr[\gamma \to \gamma'] = p_c$

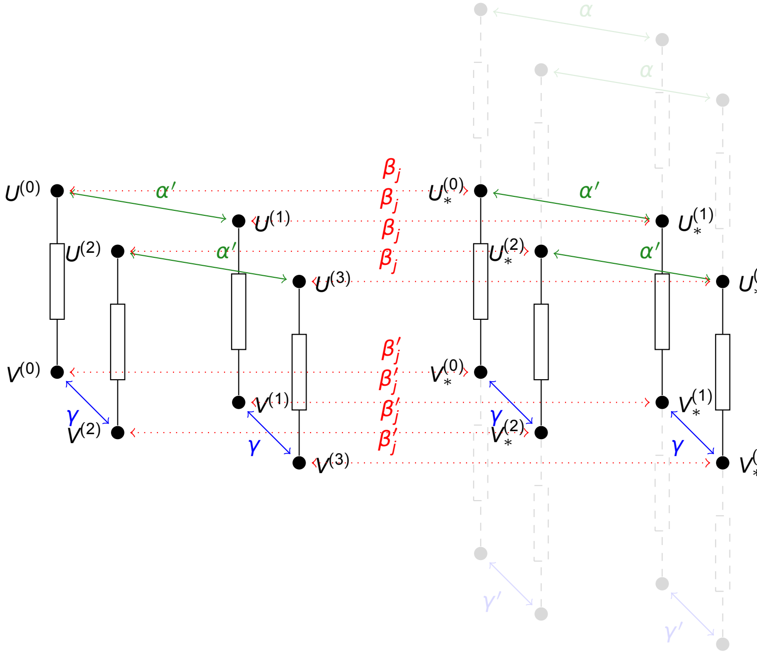


f_a
$\Pr[\alpha \to \alpha'] = p_a$

f_b
$\Pr\left[\beta_j \to \beta'_j\right] = p_b$

f_c
$\Pr[\gamma \to \gamma'] = p_c$


f_a
$\Pr[\alpha \to \alpha'] = p_a$
f_b
$\Pr\left[\beta_j \to \beta_j'\right] = p_b$
f_c
$\Pr[\gamma \to \gamma'] = p_c$
$U^{(0)}$ $U^{(1)}$ $U^{(2)}$ $U^{(3)}$
$U_*^{(0)}$ $U_*^{(1)}$ $U_*^{(2)}$ $U_*^{(3)}$
$V^{(0)}$ $V^{(1)}$ $V^{(2)}$ $V^{(3)}$
$V_*^{(0)}$ $V_*^{(1)}$ $V_*^{(2)}$ $V_*^{(3)}$
α α' β_j β_j' γ γ'



f_a

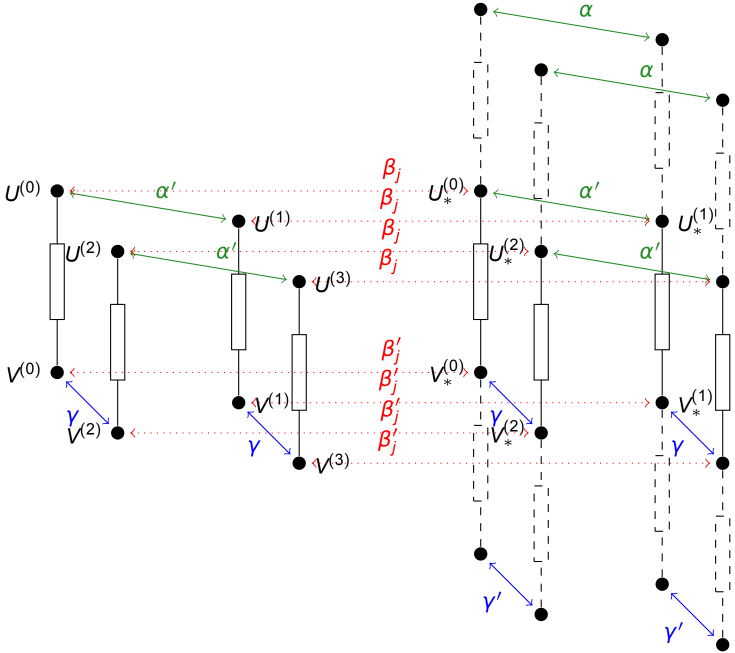
$\Pr[\alpha \to \alpha'] = p_a$

f_b

$\Pr\left[\beta_j \to \beta_j'\right] = p_b$

f_c

$\Pr[\gamma \to \gamma'] = p_c$



f_a

$\Pr[\alpha \to \alpha'] = p_a$

f_b

$\Pr\left[\beta_j \to \beta'_j\right] = p_b$

f_c

$\Pr[\gamma \to \gamma'] = p_c$

Using Auxiliary Paths

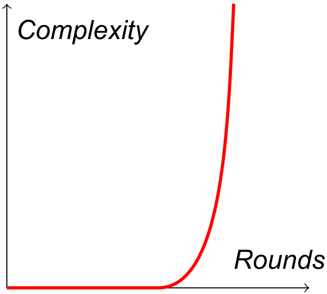
- Hash function setting allows to start from the middle and to build related quartets (instead of related pairs)
- Complexity: $$\frac{1}{p_a^2 p_c^2}\left(\frac{C}{b \cdot p_b^4} + 1\right)$$
  - Cost C to build an initial quartet
  - b paths with probability p_b for f_b
- Also works with related-key paths
  - New quartet with a different key
- Very efficient with a large family of probability 1 paths
  - We can combine three paths instead of two

Application

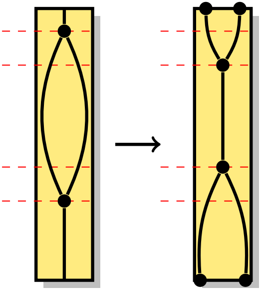
Application to ARX Designs

- Several recent design are based on the ARX design
  - Use only Addition, Rotation, Xor
  - Skein, Blake are SHA-3 finalists

- Short RK paths with high probability



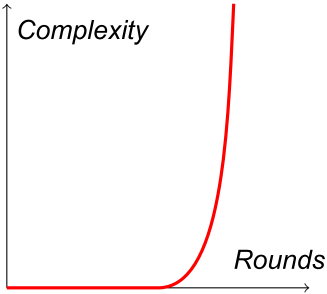
- Hard to build controlled characteristics



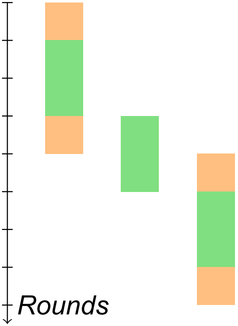
Application to ARX Designs

- Several recent design are based on the ARX design
  - Use only Addition, Rotation, Xor
  - Skein, Blake are SHA-3 finalists

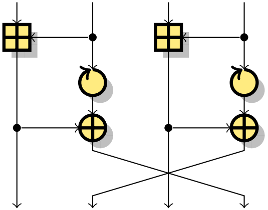
- Short RK paths with high probability



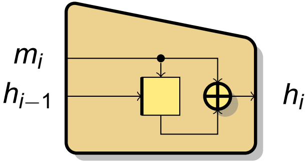
- Using auxiliary paths



Skein



Threefish-256 round



MMO mode

- SHA-3 finalist
- ARX design
  - 64-bit words
  - $\texttt{MIX}_r(a, b) := ((a \boxplus b), (b \lll r) \oplus c)$
  - Word permutations
  - Key addition every four rounds
- Threefish-256:
  - 256-bit key: K_0, K_1, K_2, K_3
  - 128-bit tweak: T_0, T_1
  - 256-bit text

Skein: Differential Trails

Key schedule (Threefish-256):

- 256-bit key: K_0, K_1, K_2, K_3
- 128-bit tweak: T_0, T_1
- $K_4 := K_0 \oplus K_1 \oplus K_2 \oplus K_3 \oplus C$
- $T_2 := T_0 \oplus T_1$

Round				
0	K_0	$K_1 + T_0$	$K_2 + T_1$	$K_3 + 0$
4	K_1	$K_2 + T_1$	$K_3 + T_2$	$K_4 + 1$
8	K_2	$K_3 + T_2$	$K_4 + T_0$	$K_0 + 2$
12	K_3	$K_4 + T_0$	$K_0 + T_1$	$K_1 + 3$
16	K_4	$K_0 + T_1$	$K_1 + T_2$	$K_2 + 4$

- Use a difference in the tweak and in the key so that they cancel out
- One key addition without any difference

Skein: Differential Trails

Key schedule (Threefish-256):

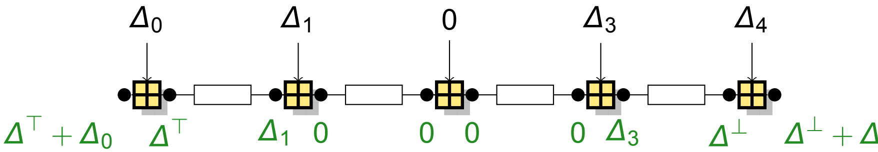
- 256-bit key: K_0, K_1, K_2, K_3
- 128-bit tweak: T_0, T_1
- $K_4 := K_0 \oplus K_1 \oplus K_2 \oplus K_3 \oplus C$
- $T_2 := T_0 \oplus T_1$

Round				
0	K_0	$K_1 + T_0$	$K_2 + T_1$	$K_3 + 0$
4	K_1	$K_2 + T_1$	$K_3 + T_2$	$K_4 + 1$
8	K_2	$K_3 + T_2$	$K_4 + T_0$	$K_0 + 2$
12	K_3	$K_4 + T_0$	$K_0 + T_1$	$K_1 + 3$
16	K_4	$K_0 + T_1$	$K_1 + T_2$	$K_2 + 4$

- Use a difference in the tweak and in the key so that they cancel out
- One key addition without any difference

Skein: Differential Trails

- 16-round trail:

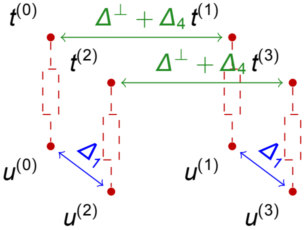


$p = 1$ $\qquad$ $p = 1$

$\Pr\left[\Delta^\top \leftarrow \Delta_1\right] = 2^{-33}$ $\qquad$ $\Pr\left[\Delta_3 \rightarrow \Delta^\perp\right] = 2^{-6}$

- Use a MSB difference for best probability
- Use any difference for auxiliary paths
  - 2^{64} 8-round paths with probability 1

Skein: Description of the Attack



Top path (0–12)

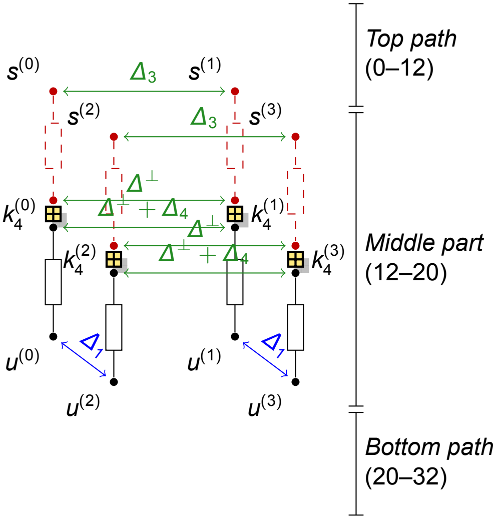
Middle part (12–20)

Bottom path (20–32)

1. Build a quartet for rounds 16—20.
   cost: 2^{18}
2. Extend to rounds 12—20 using random keys.
   cost: 2^{18}
3. Use auxiliary paths to generate quartets.

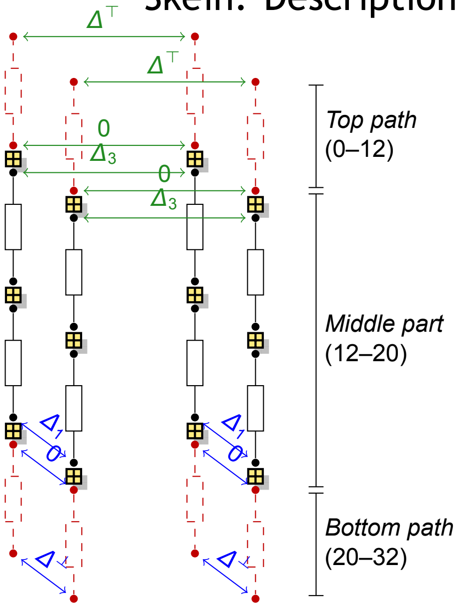
amortized cost: 2^{0}

Skein: Description of the Attack



1. Build a quartet for rounds 16—20.
   cost: 2^{18}
2. Extend to rounds 12—20 using random keys.
   cost: 2^{18}
3. Use auxiliary paths to generate quartets.
   amortized cost: 2^{0}

Skein: Description of the Attack



1. Build a quartet for rounds 16—20.

   cost: 2^{18}

2. Extend to rounds 12—20 using random keys.

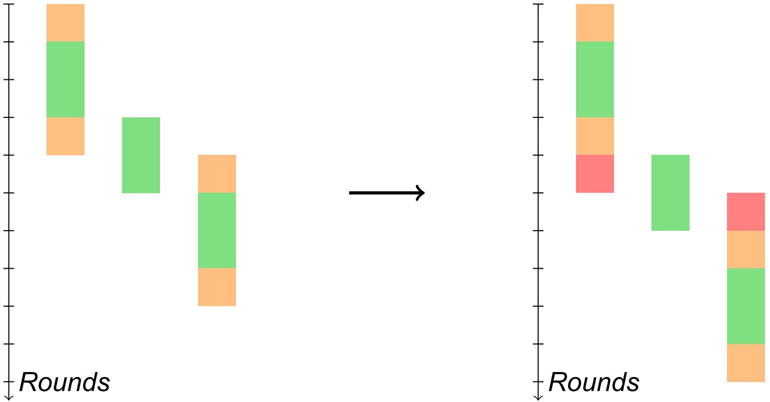
   cost: 2^{18}

3. Use auxiliary paths to generate quartets.

   amortized cost: 2^0

Limitations of the Technique

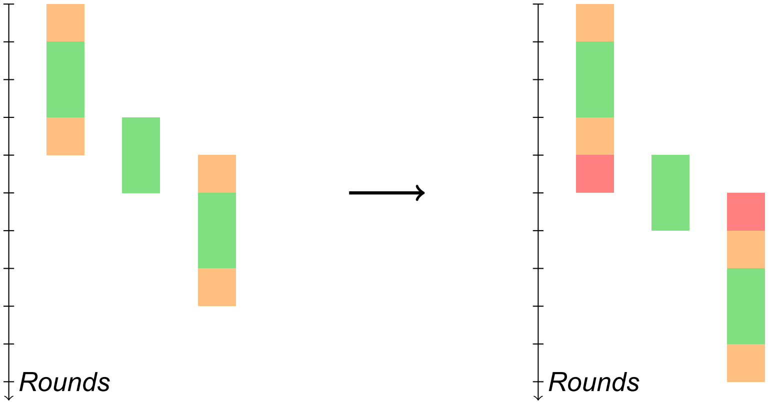
Why not attack more rounds?



Paths are incompatible!

Limitations of the Technique

Why not attack more rounds?



Paths are incompatible!

Incompatible Characteristics

Incompatibilities in Boomerang Paths

- For a Boomerang attack, we usually assume that the path are independent

- We are building a quartet $X^{(0)}, X^{(1)}, X^{(2)}, X^{(3)}$:

$$X^{(1)} = X^{(0)} + \alpha' \qquad X^{(3)} = X^{(2)} + \alpha'$$
$$X^{(2)} = X^{(0)} + \gamma \qquad X^{(2)} = X^{(1)} + \gamma$$

We expect:

$$(X^{(0)}, X^{(1)}) \xleftarrow{f_a} \alpha \qquad (X^{(2)}, X^{(3)}) \xleftarrow{f_a} \alpha$$
$$(X^{(0)}, X^{(2)}) \xrightarrow{f_b} \gamma' \qquad (X^{(1)}, X^{(3)}) \xrightarrow{f_b} \gamma'$$

- But these events are not independent! [Murphy 2011]

Boomerang Incompatibility

$\delta a = \texttt{-x-} \quad \delta b = \texttt{---}$

Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

$\delta a = \texttt{-x-} \quad \delta b = \texttt{-x-}$

Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$\delta u = \texttt{---}$

$\boxed{u = a + b}$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang Incompatibility

$\delta a = \texttt{-x-}$ $\quad \delta b = \texttt{---}$

Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

$\delta a = \texttt{-x-}$ $\quad \delta b = \texttt{-x-}$

Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$\delta u = \texttt{---}$

$\boxed{u = a + b}$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang Incompatibility

$\delta a = \texttt{-x-} \quad \delta b = \texttt{---}$

Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})$ $(a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

$\delta a = \texttt{-x-} \quad \delta b = \texttt{-x-}$

Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})$ $(a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$\delta u = \texttt{---}$

$\boxed{u = a + b}$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang Incompatibility

$\delta a = \texttt{-x-}$ $\quad \delta b = \texttt{---}$

Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

$\delta a = \texttt{-x-}$ $\quad \delta b = \texttt{-x-}$

Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$\delta u = \texttt{---}$

$\boxed{u = a + b}$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang Incompatibility

$\delta a = \texttt{-x-}$ $\quad \delta b = \texttt{---}$ Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})$ $(a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

$\delta a = \texttt{-x-}$ $\quad \delta b = \texttt{-x-}$ Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})$ $(a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$\delta u = \texttt{---}$

$\boxed{u = a + b}$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Other Incompatible Paths

$\delta a = \texttt{-x}$ $\quad \delta b = \texttt{-x}$ $\quad \delta c = \texttt{-x}$

$\boxplus$

$\delta u = \texttt{-x}$

$\boxed{u = a + b + c}$

$\delta a = \texttt{--xxxxx-}$ $\quad \delta b = \texttt{---xx---}$

$\boxplus$

$\delta u = \texttt{-xxxx-x-}$

$\boxed{u = a + b}$

Many “natural” characteristics are in fact incompatible.

- Previous boomerang attacks on Skein-512 do not work
- Works on Skein-256

Results on Skein

Attack	CF/KP	Rounds	CF/KP calls	Ref.
Unknown Key				
Near collisions (Skein-256)	CF	24	2^{60}	[CANS '10]
~~Boomerang dist. (Threefish-512)~~	~~KP~~	~~32~~	$\cancel{2^{189}}$	[ISPEC '10]
~~Key Recovery (Threefish-512)~~	~~KP~~	~~34~~	$\cancel{2^{474.4}}$	[ISPEC '10]
Key Recovery (Threefish-512)	KP	32	2^{312}	[AC '09]
Open key				
Boomerang dist. (Threefish-512)	KP	35	2^{478}	[AC '09]
~~Near collisions (Skein-256)~~	~~CF~~	~~32~~	$\cancel{2^{105}}$	[ePrint '11]
Boomerang dist. (Skein-256)	CF and KP	24	2^{18}	
Boomerang dist. (Threefish-256)	KP	28	2^{21}	
Boomerang dist. (Skein-256)	CF	28	2^{24}	
Boomerang dist. (Threefish-256)	KP	32	2^{57}	
Boomerang dist. (Skein-256)	CF	32	2^{114}	

Conclusion

1. Boomerang attack on hash functions
   - Start from the middle
   - Use auxiliary path to avoid middle rounds
   - Significant improvement over previous results
   - New result: also works on Blake

see details

2. Analysis of differentials paths
   - Problems found in several previous works

Appendix

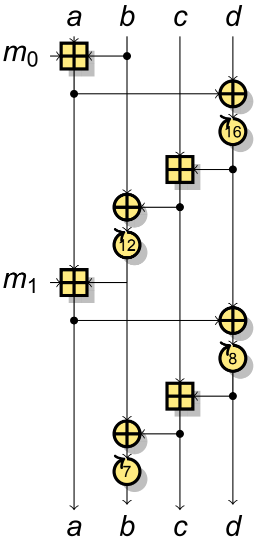
Related work

- Similar to “Boomerang” of Joux and Peyrin (auxiliary paths)
  - In the context of collision attacks
- Similar to message modifications for Boomerang attacks
  - Blake [BNR ’11]
  - SHA-2 [ML ’11]
  - HAVAL [Sasaki ’11]
  - Skein/Threefish [ACMPV ’09, Chen & Jia ’10]
  - Auxiliary paths allow to skip more rounds

New Result: Application to Blake

- The same technique can be applied to Blake
  - Another ARX SHA-3 finalist
- Significant improvement over previous results [FSE '11]
- Compression function attack:
  - 6.5 rounds: 2^{140} (vs. $\cancel{2^{184}}$)
  - 7 rounds: 2^{183} (vs. $\cancel{2^{232}}$)
- Keyed-permutation attacks (Open-key vs. Unknown-key)
  - 7 rounds: 2^{32} (vs. $\cancel{2^{122}}$)
  - 8 rounds: 2^{1xx} (vs. $\cancel{2^{242}}$)

Blake



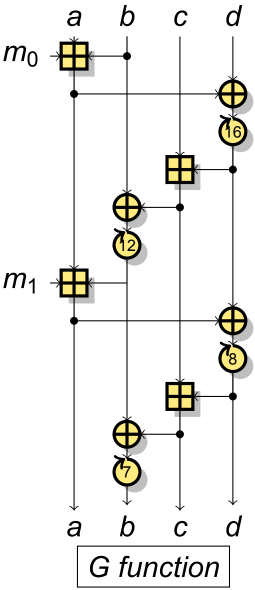
G function

- State is 4×4 matrix:

a_0	a_1	a_2	a_3
b_0	b_1	b_2	b_3
c_0	c_1	c_2	c_3
d_0	d_1	d_2	d_3

- Column step:
  $G(a_0, b_0, c_0, d_0)$
  $G(a_1, b_1, c_1, d_1)$
  $G(a_2, b_2, c_2, d_2)$
  $G(a_3, b_3, c_3, d_3)$
- Diagonal step:
  $G(a_0, b_1, c_2, d_3)$
  $G(a_1, b_2, c_3, d_0)$
  $G(a_2, b_3, c_0, d_1)$
  $G(a_3, b_0, c_1, d_2)$

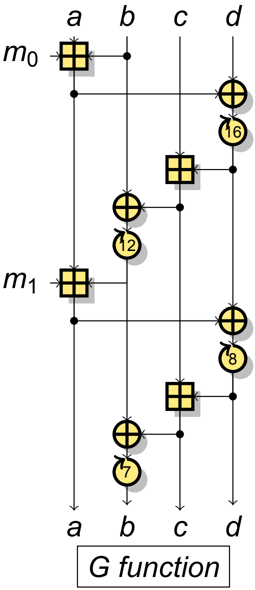
Blake



- State is 4×4 matrix:

a_0	a_1	a_2	a_3
b_0	b_1	b_2	b_3
c_0	c_1	c_2	c_3
d_0	d_1	d_2	d_3

- Column step:
  $G(a_0, b_0, c_0, d_0)$
  $G(a_1, b_1, c_1, d_1)$
  $G(a_2, b_2, c_2, d_2)$
  $G(a_3, b_3, c_3, d_3)$
- Diagonal step:
  $G(a_0, b_1, c_2, d_3)$
  $G(a_1, b_2, c_3, d_0)$
  $G(a_2, b_3, c_0, d_1)$
  $G(a_3, b_0, c_1, d_2)$



Blake

- State is 4×4 matrix:

a_0	a_1	a_2	a_3
b_0	b_1	b_2	b_3
c_0	c_1	c_2	c_3
d_0	d_1	d_2	d_3

- Column step:
  $G(a_0, b_0, c_0, d_0)$
  $G(a_1, b_1, c_1, d_1)$
  $G(a_2, b_2, c_2, d_2)$
  $G(a_3, b_3, c_3, d_3)$
- Diagonal step:
  $G(a_0, b_1, c_2, d_3)$
  $G(a_1, b_2, c_3, d_0)$
  $G(a_2, b_3, c_0, d_1)$
  $G(a_3, b_0, c_1, d_2)$

Blake: Differential Trails

- Key schedule: permutation based

σ_3 :	7	3	13	11	9	1	12	14	2	5	4	15	6	10	0	8
σ_4 :	9	5	2	10	0	7	4	15	14	11	6	3	1	12	8	13

- Choose a message word used
  - at the beginning of a round
  - at the end of the next round
- 4-round trail:

$\Pr\left[\Delta^\top \leftarrow \Delta_1\right] = 2^{-6} \cdot 2^{-42}$ $\quad$ $\Pr\left[\Delta_3 \rightarrow \Delta^\perp\right] = 2^{-24}$

Blake: Differential Trails

- Key schedule: permutation based

σ_3 :	7	3	13	11	9	1	12	14	2	5	4	15	6	10	0	8
σ_4 :	9	5	2	10	0	7	4	15	14	11	6	3	1	12	8	13

- Choose a message word used
  - at the beginning of a round
  - at the end of the next round
- 4-round trail:

$p = 1$ $\quad$ $p = 1/2$

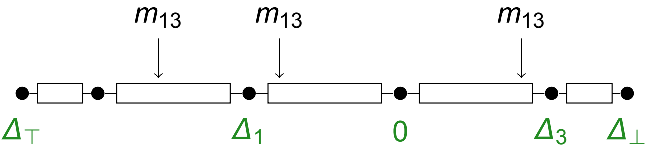
$\Pr\left[\Delta^\top \leftarrow \Delta_1\right] = 2^{-6} \cdot 2^{-42}$ $\qquad$ $\Pr\left[\Delta_3 \rightarrow \Delta^\perp\right] = 2^{-24}$

Blake: Differential Trails

- Key schedule: permutation based

σ_3 :	7	3	13	11	9	1	12	14	2	5	4	15	6	10	0	8
σ_4 :	9	5	2	10	0	7	4	15	14	11	6	3	1	12	8	13

- Choose a message word used
  - at the beginning of a round
  - at the end of the next round
- 4-round trail:



$p = 1$ $\quad$ $p = 1/2$

$\Pr\left[\Delta^\top \leftarrow \Delta_1\right] = 2^{-6} \cdot 2^{-42}$ $\quad$ $\Pr\left[\Delta_3 \rightarrow \Delta^\bot\right] = 2^{-24}$

Blake: Description of the Attack

The hard part is the middle round

- Column step is part of the top path
- Diagonal step is part of the bottom path

1. Find (state, message) candidates for each diagonal G function
   - Start with middle quartets with all differences fixed

2. Look for combinations of candidates that follow the first part of the diagonal step
   - Use the message to randomize

3. Look for candidates that follow the full diagonal step
   - Use the message to randomize

Blake-256: Results

Attack	CF/KP	Rounds	CF/KP calls	Ref.
Unknown Key				
Boomerang dist.	KP	7	2^{122}	[FSE '11]
~~Boomerang dist.~~	~~KP~~	~~8~~	$\cancel{2^{242}}$	[FSE '11]
Open Key				
~~Boomerang dist.~~	~~CF w/ Init~~	~~7~~	$\cancel{2^{232}}$	[FSE '11]
Boomerang dist.	CF w/ Init	7	2^{183}	
Boomerang dist.	KP	7	2^{32}	
Boomerang dist.	KP	8	2^{1xx}	