A New Variant of PMAC: Beyond the Birthday Bound

Kan Yasuda (NTT, Japan)
CRYPTO 2011
Aug. 17

Summary

- **We present a new MAC which is PMAC-like, highly secure, and highly efficient.**

Outline

- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)

Outline

- **We present a new MAC which is PMAC-like, highly secure, and highly efficient.**
  - **1. MACs (quick review)**
  - **2. PMAC (blockcipher-based MAC)**

Outline

- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)

Outline

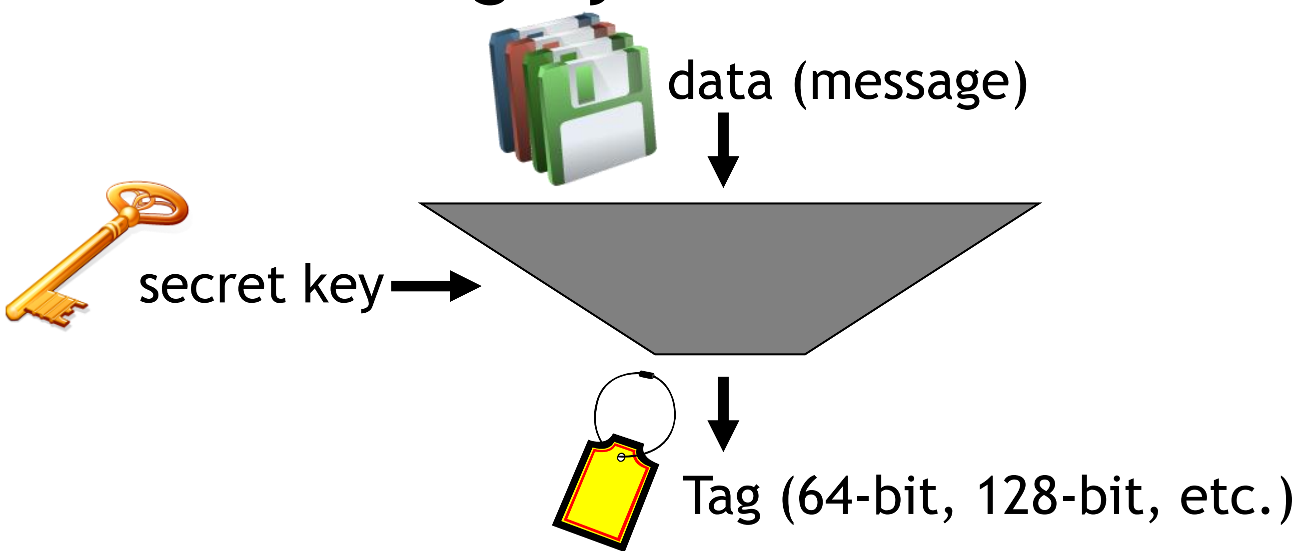
- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)
  - 4. highly efficient (construction details)

Outline

- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)
  - 4. highly efficient (construction details)
  - 5. Some possible refinements

Introduction

- 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

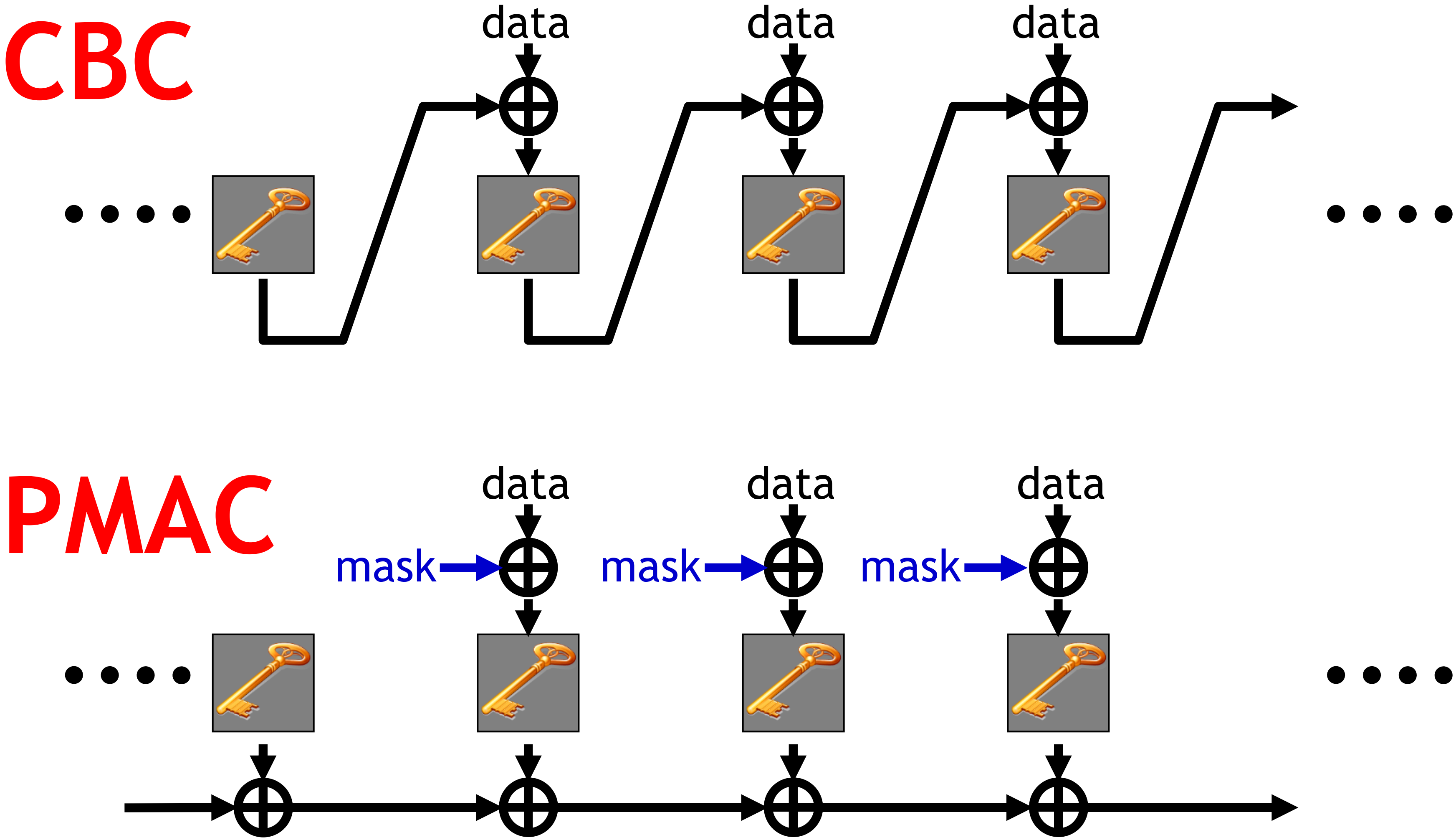
- 1. design from scratch (dedicated MAC)
- 2. use a cryptographic hash function (e.g., HMAC)
- 3. use a universal hash function
- 4. use a blockcipher (e.g., CMAC, PMAC)

We focus on
blockcipher-based construction

Outline

- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)
  - 4. highly efficient (construction details)
  - 5. Some possible refinements

Blockcipher-based MACs (2 types of iteration)



Mask needs to be updated at each iteration

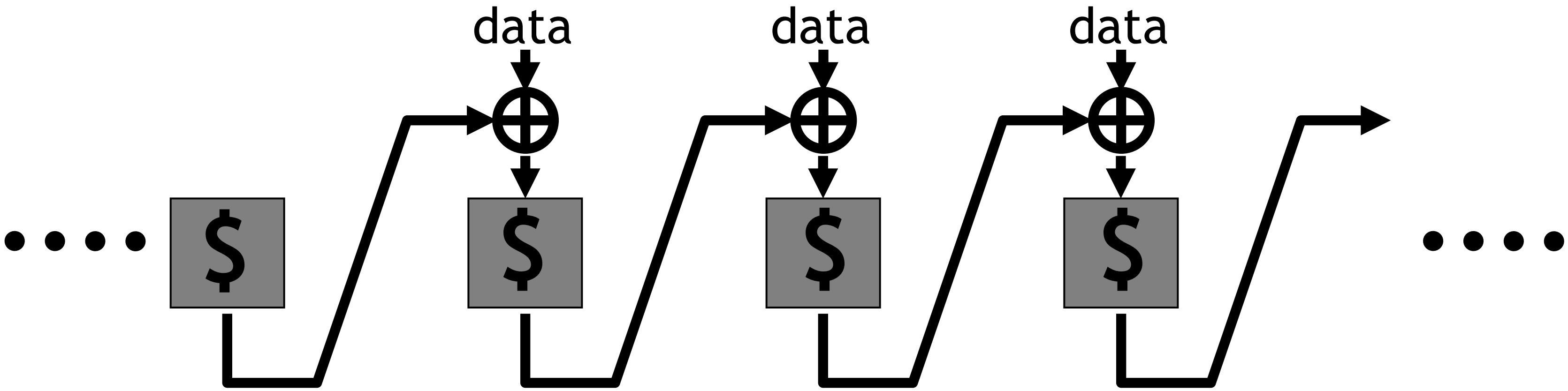
CBC vs. PMAC

CBC	PMAC
Sequential	✔ Parallelizable
✔ Only XOR	Requires mask update and XOR

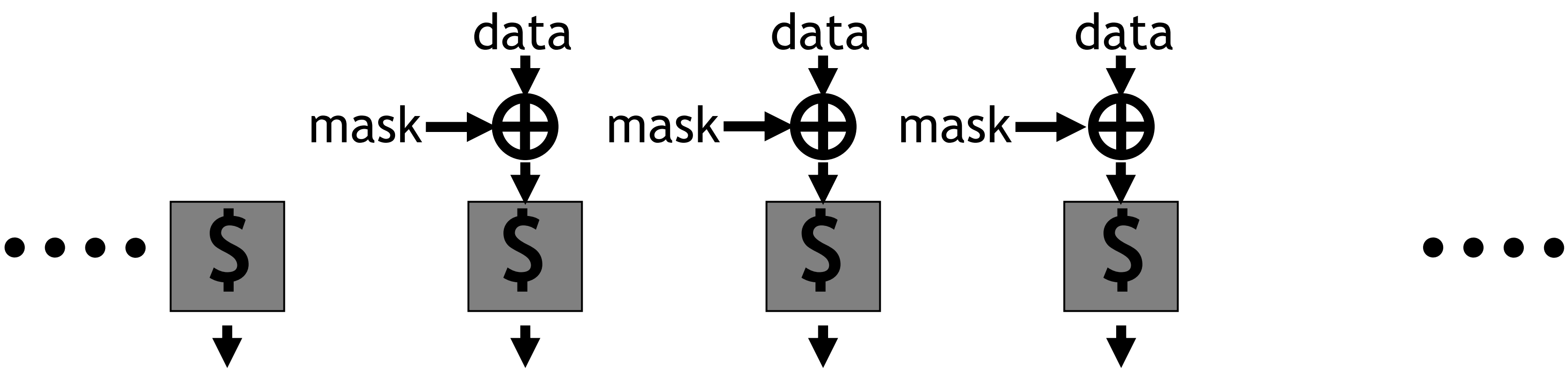
We choose PMAC, because . . .

- PMAC seems to have a structure easier to analyze (<u>for security proofs</u>)
- In fact, some of our proof techniques are not applicable to CBC iteration

Intuition behind the choice



Order of execution does matter



Can be executed in any order

Easier to manipulate events and to evaluate probabilities

Outline

- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)
  - 4. highly efficient (construction details)
  - 5. Some possible refinements

MAC security

- Unforgeability
  - Adversary (without knowing the key) should not be able to produce a valid tag for a new message
- Pseudo-random
  - Randomness implies unforgeability
  - If a MAC is a secure PRF (pseudo-random function), then it is also a secure MAC.

MAC security

- Unforgeability
  - Adversary (without knowing the key) should not be able to produce a valid tag for a new message
- Pseudo-random
  - Randomness implies unforgeability
  - If a MAC is a secure PRF (pseudo-random function), then it is also a secure MAC.

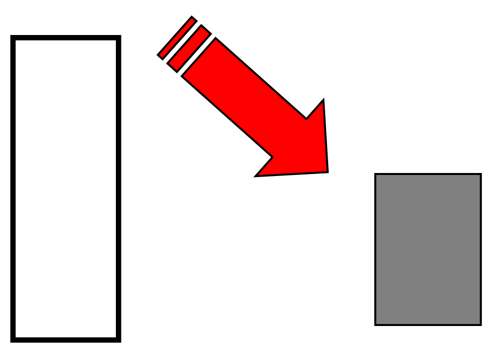
We follow this direction

Birthday problems

- Ordinary MACs usually provide security only half the block size (*n* bit) of the underlying cipher
- For *n*-bit cipher, only 2^(*n*/ 2) security

- For *n* = 64, 2^32 blocks = 32GBytes
- 64-bit blockciphers? Triple-DES, HIGHT, PRESENT, LED, . . .

n-bit security



0.5*n*-bit security

2 diffenent birthday problems exist for blockcipher-based MACs

- Birthday attacks on iterated MACs
  - Existential forgery is possible on any iterated MACs after 2^(*n*/2) queries (*n* the state size)
  - For CBC-type MACs, even universal forgery is possible
- PRP – PRF switching lemma
  - PRP - pseudo-random permutation
  - A (pseudo-random) permutation can be considered as a function only up to 2^(*n*/2) queries

Our security result

- Our construction achieves 2^(2n/3) security
  - For n = 64, 2^42.7 blocks = 51TBytes
- Our MAC is a secure PRF based on the assumption that the underlying blockcipher is a secure PRP
  - We avoid using PRP-PRF switching lemma

Outline

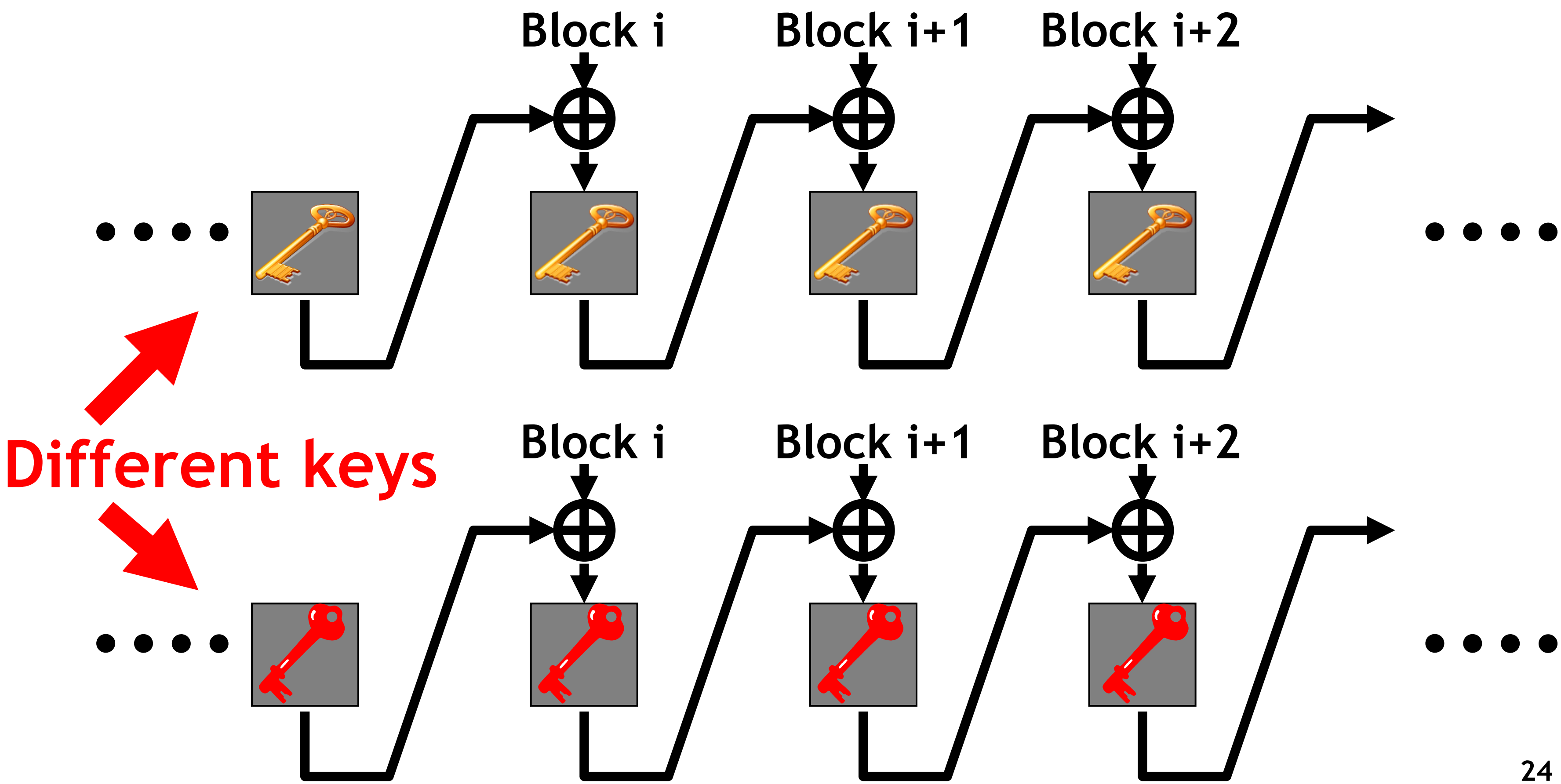
- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)
  - 4. highly efficient (construction details)
  - 5. Some possible refinements

ISO 9797

- (The only) previous construction that achieves security beyond the birthday bound
  - Achieves (Slightly worse than our) 2^(2n/3) security
  - Rate-1/2 construction, twice as slow (as CMAC, PMAC)

ISO 9797 - sum of two CBC MACs

- Requires 2 encryptions to process a block

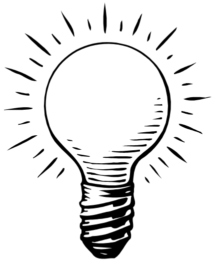


Our solution - basic idea

We want rate-1 construction; only 1 encryption per block . . .

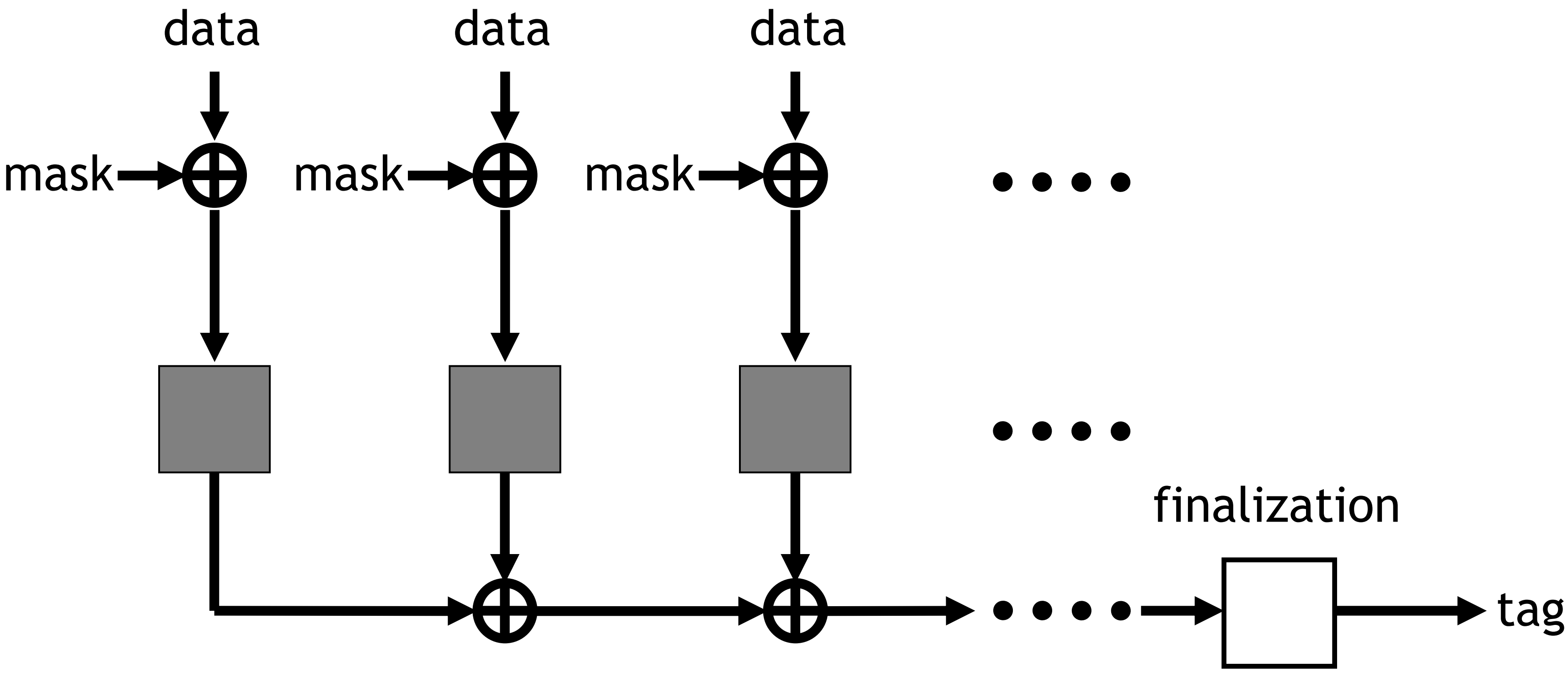
Our solution - basic idea

**We want rate-1 construction,
only 1 encryption per block . . .**

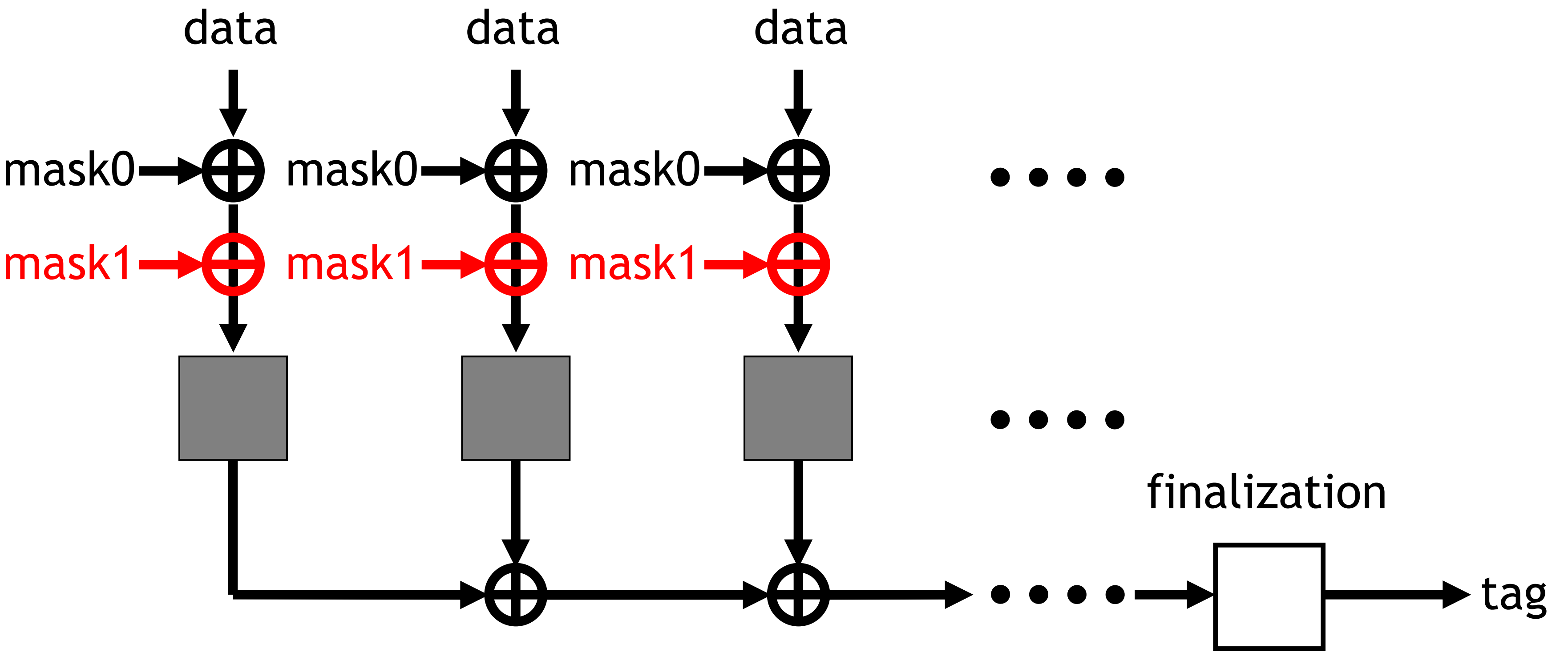


Double everything but blockcipher calls

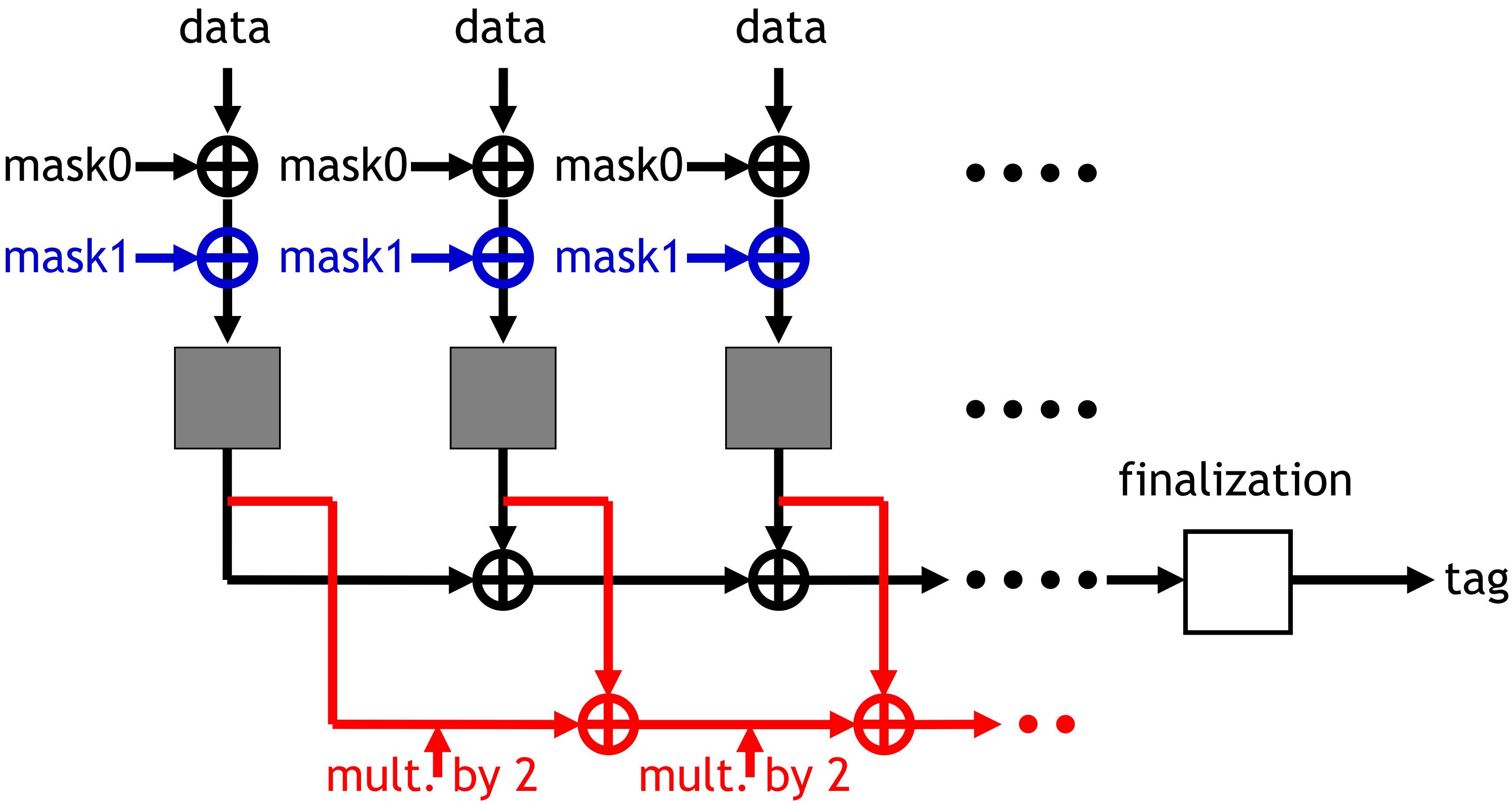
Original PMAC



Doubling the masking

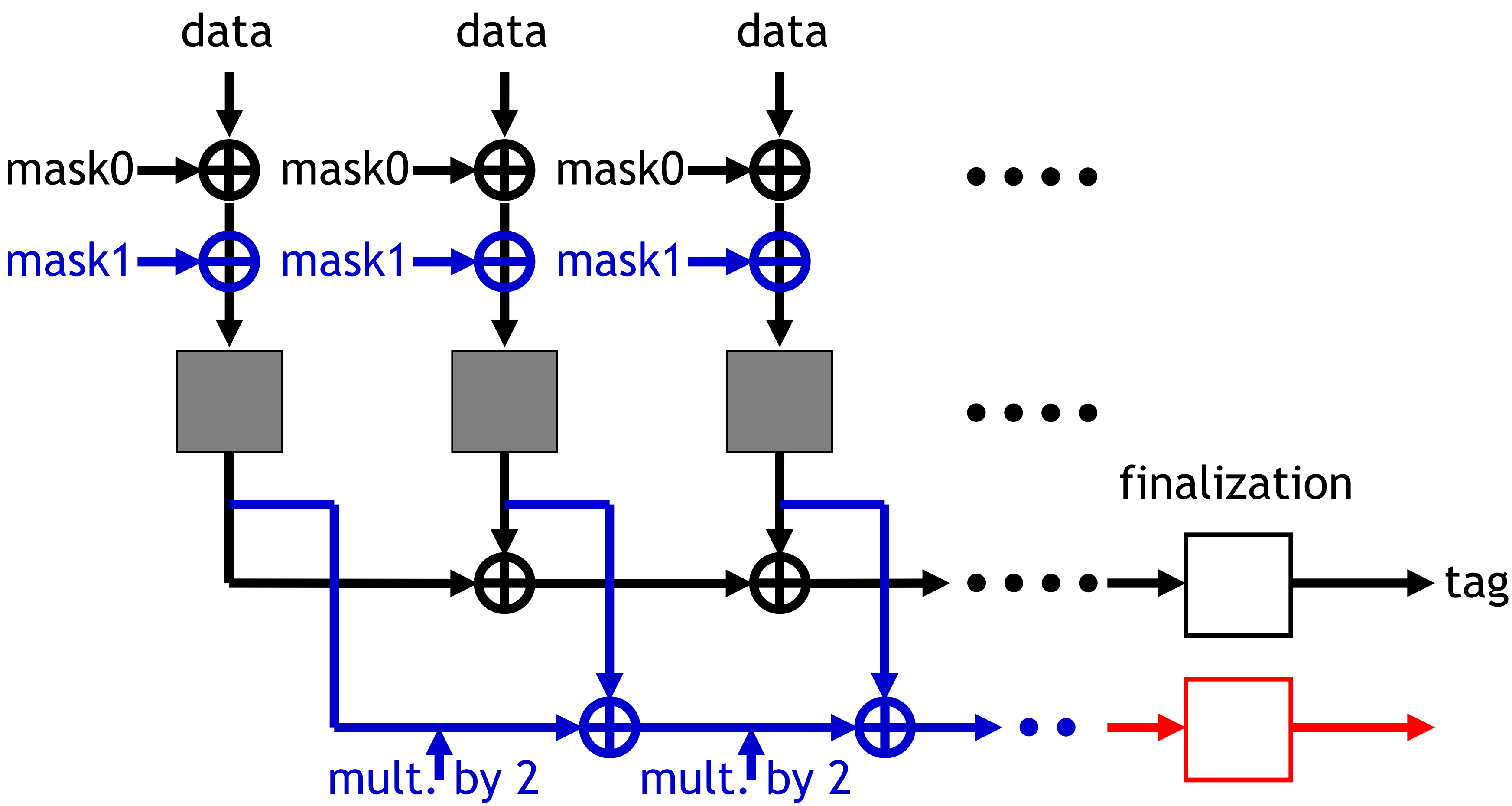


Doubling the state

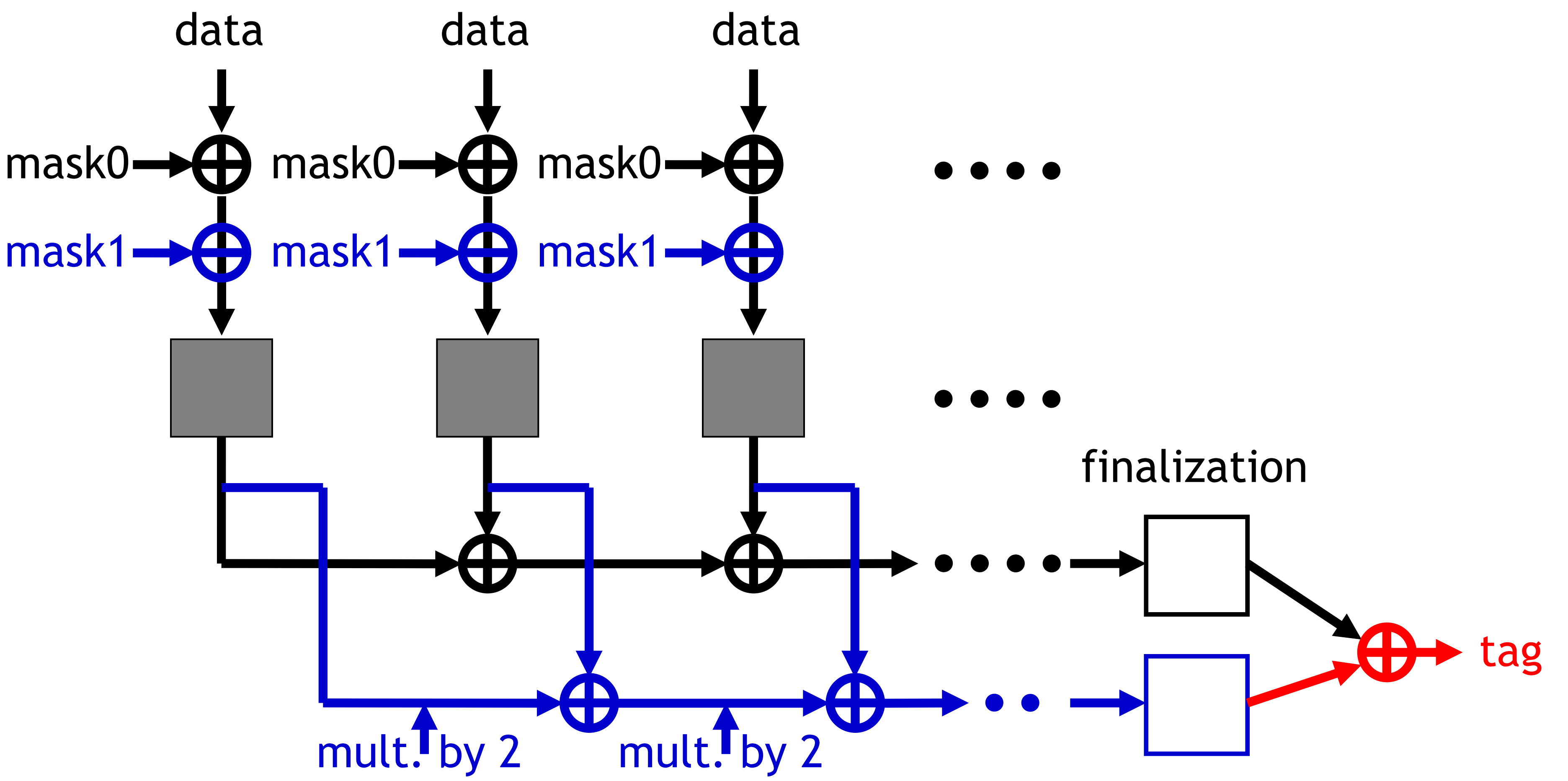


Errata: These finite-field mult. by 2 are written wrongly in the proceddings Fig. 1 page 601

Doubling the finalization



Our construction



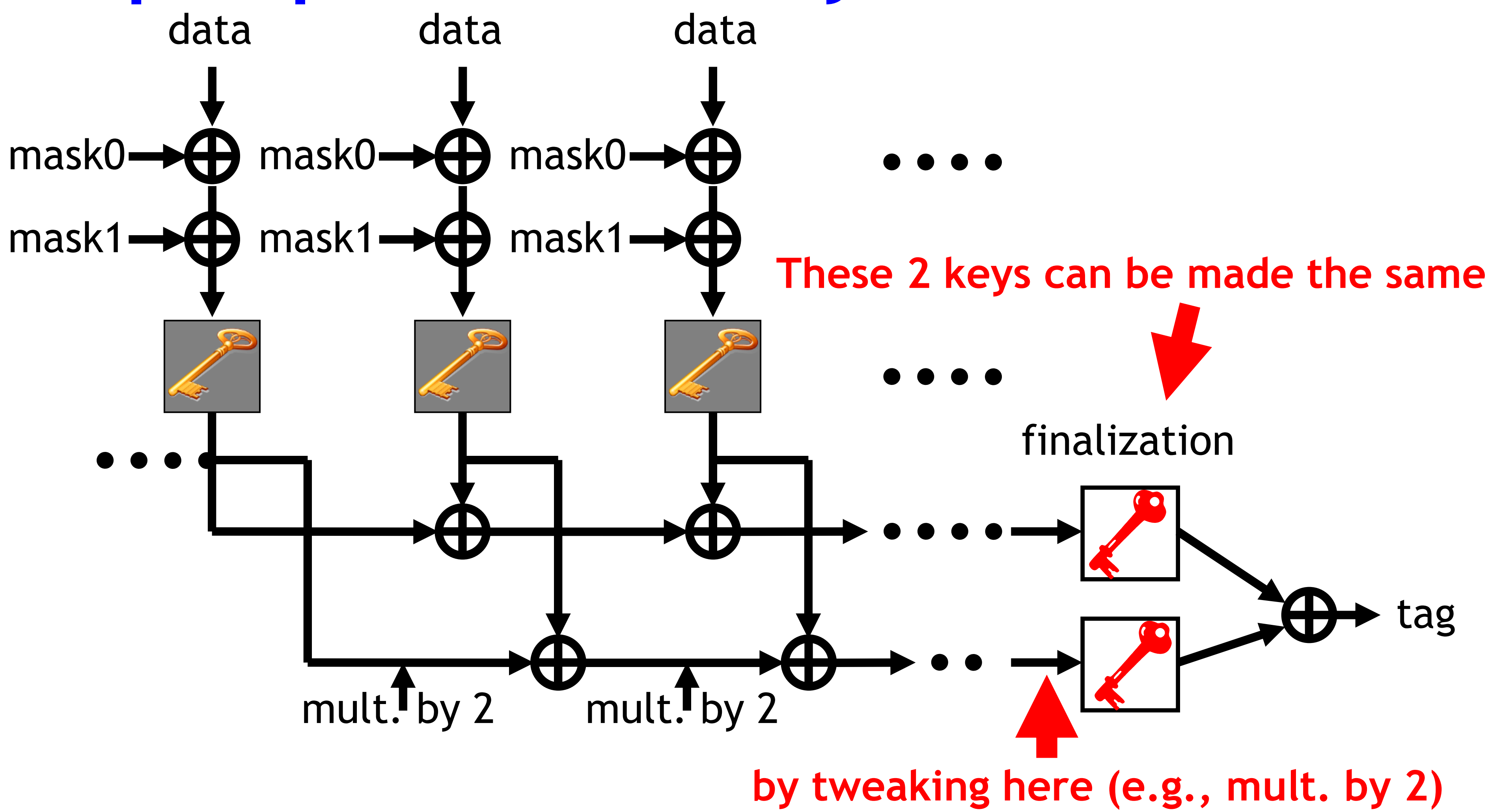
More details

- Mask generation and update
  - mask0 is encryption of 0, mask 1 is encryption of 1
  - mask0 is updated via mult. by 2
  - mask1 is updated via mult. by 4
- Uses 3 keys
  - Use different keys for (each of the) finalization
- Finite-field mult. by 2
  - Can be implemented 1-bit shit + 1 conditional XOR

Outline

- We present a new MAC which is PMAC-like, highly secure, and highly efficient.
  - 1. MACs (quick review)
  - 2. PMAC (blockcipher-based MAC)
  - 3. highly secure (beyond birthday bound)
  - 4. highly efficient (construction details)
  - 5. Some possible refinements

Open problem: 1-key construction



. . . But still a 2-key construction

Open problem: Full 2^n security

- Tripling everything instead of doubling
  - Possibly 2^(3n/4) security, but not 2^n
  - 4 times, 5 times, . . . would result in 2^(4n/5), 2^(5n/6) security (at best)
  - May call them still rate-1, but more and more inefficient
- The 2^(2n/3) bound may not be tight
  - No attacks (of this complexity) known
  - The proofs may be improved

Thank you