Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers

Nicky Mouha[1], Bart Mennink[1], Anthony Van Herrewege[1],
Dai Watanabe[2], Bart Preneel[1], Ingrid Verbauwhede[1]

[1]ESAT/COSIC, KU Leuven and iMinds, Belgium
[2]Yokohama Research Laboratory, Hitachi, Japan

SAC 2014 — August 15, 2014

MAC Algorithm for Microcontrollers

Message Authentication Code (MAC)

- $\text{MAC}_K(m) = \tau$
- Authenticity, no confidentiality
- Same key for MAC generation and verification

MAC Algorithm for Microcontrollers

Message Authentication Code (MAC)

- $\text{MAC}_K(m) = \tau$
- Authenticity, no confidentiality
- Same key for MAC generation and verification

Microcontroller

- Cheap 8/16/32-bit processor: USD 25-50¢
- Applications: home, medical, industrial,...
- Ubiquitous: 30-100 in any recent car

Design

Requirements

- Drop-in replacement for AES-CMAC
  (variant of CBC-MAC for variable-length messages)
- Same functionality and security

Design

Requirements

- Drop-in replacement for AES-CMAC
  (variant of CBC-MAC for variable-length messages)
- Same functionality and security

Speed

- "Ten times faster than AES"

Design

Requirements

- Drop-in replacement for AES-CMAC
  (variant of CBC-MAC for variable-length messages)
- Same functionality and security

Speed

- "Ten times faster than AES"

Approach

- Dedicated design for microcontrollers

Commonly used MACs

Based on (cryptographic) hash function

- **Example:** HMAC, SHA3-MAC
- Large block size, collision resistance unnecessary

Commonly used MACs

Based on (cryptographic) hash function

- **Example:** HMAC, SHA3-MAC
- Large block size, collision resistance unnecessary

Based on universal hashing

- **Examples:** UMAC, GMAC, Poly1305
- **Requires:** nonce, constant-time multiply, long tags

Commonly used MACs

Based on (cryptographic) hash function

- **Example:** HMAC, SHA3-MAC
- Large block size, collision resistance unnecessary

Based on universal hashing

- **Examples:** UMAC, GMAC, Poly1305
- **Requires:** nonce, constant-time multiply, long tags

Based on block cipher

- **Example:** CMAC

Commonly used MACs

Based on (cryptographic) hash function

- **Example:** HMAC, SHA3-MAC
- Large block size, collision resistance unnecessary

Based on universal hashing

- **Examples:** UMAC, GMAC, Poly1305
- **Requires:** nonce, constant-time multiply, long tags

Based on block cipher

- **Example:** CMAC
- **Problem:** ten times too slow!

Our Approach

Every cycle counts!

- Avoid load/store: keep data in registers
- Avoid bit masking
- Make optimal use of instruction set

Our Approach

Every cycle counts!

- Avoid load/store: keep data in registers
- Avoid bit masking
- Make optimal use of instruction set

Bridging the gap

- Cryptanalysis
- Implementation

Primitive

Which primitive?

- Cryptographic hash function ✗

Primitive

Which primitive?

- Cryptographic hash function ✗
- Universal hash function ✗

Primitive

Which primitive?

- Cryptographic hash function ✗
- Universal hash function ✗
- Block cipher ✗

Primitive

Which primitive?

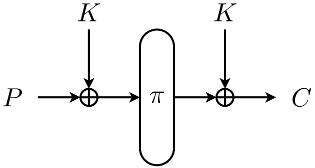
- Cryptographic hash function ✗
- Universal hash function ✗
- Block cipher ✗
- Ideal permutation ✗

Primitive

Which primitive?

- Cryptographic hash function ✗
- Universal hash function ✗
- Block cipher ✗
- Ideal permutation ✗

$\left.\begin{array}{l}\text{Block cipher ✗}\\ \text{Ideal permutation ✗}\end{array}\right\} \rightarrow$ Even-Mansour Block Cipher ✓

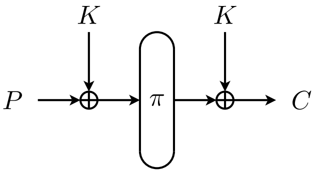


Primitive

Which primitive?

- Cryptographic hash function ✗
- Universal hash function ✗
- Block cipher ✗
- Ideal permutation ✗

$\Big\} \rightarrow$ Even-Mansour Block Cipher ✓

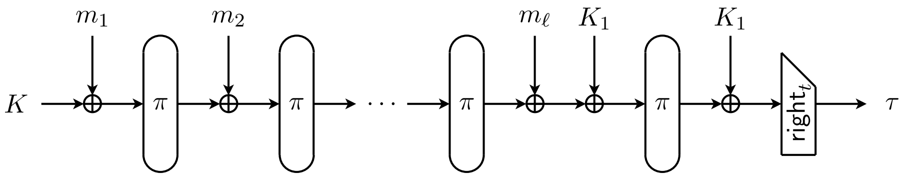


Related-key attacks

- Insecure: choose uniformly random keys!

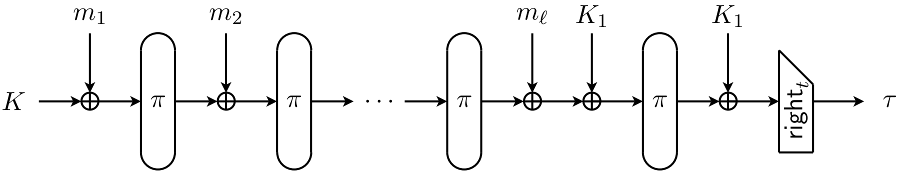
Chaskey: Mode of Operation

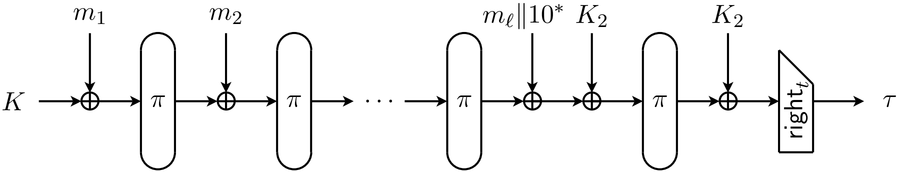
- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$
- $K_1 = 2K$



Chaskey: Mode of Operation

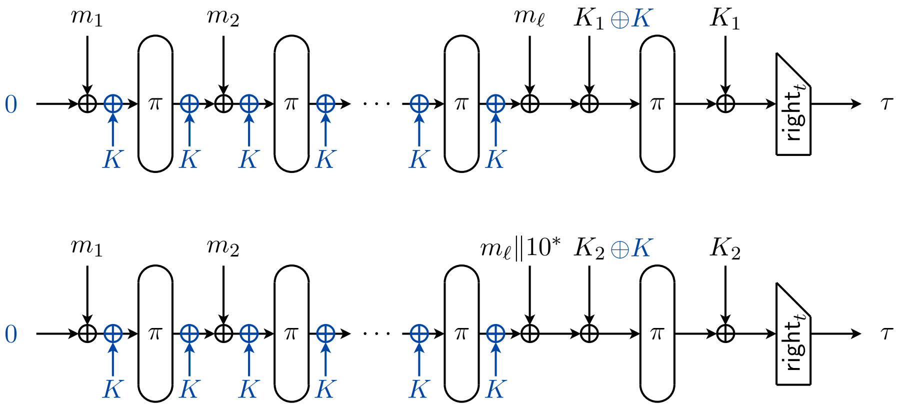
- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$





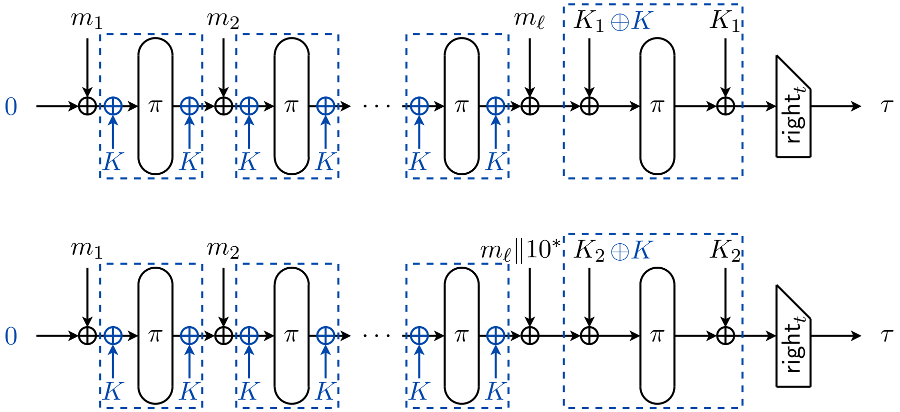
Chaskey: Mode of Operation: Phantom XORs

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \le |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$



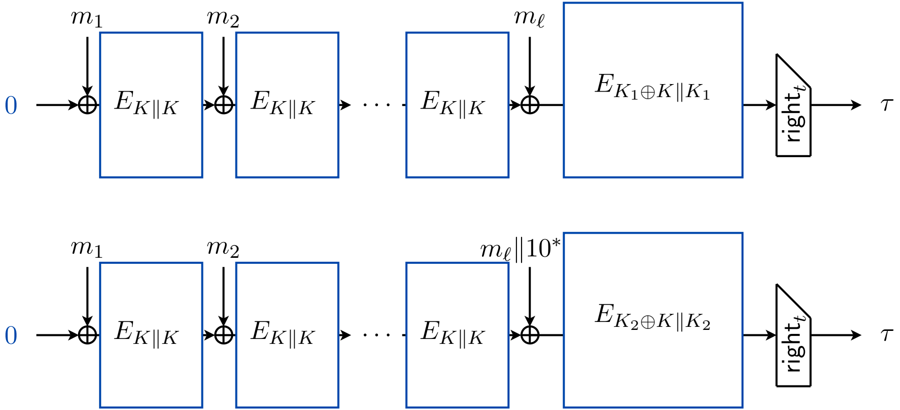
Chaskey: Mode of Operation: Phantom XORs

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$



Chaskey: Mode of Operation: Block-cipher-based

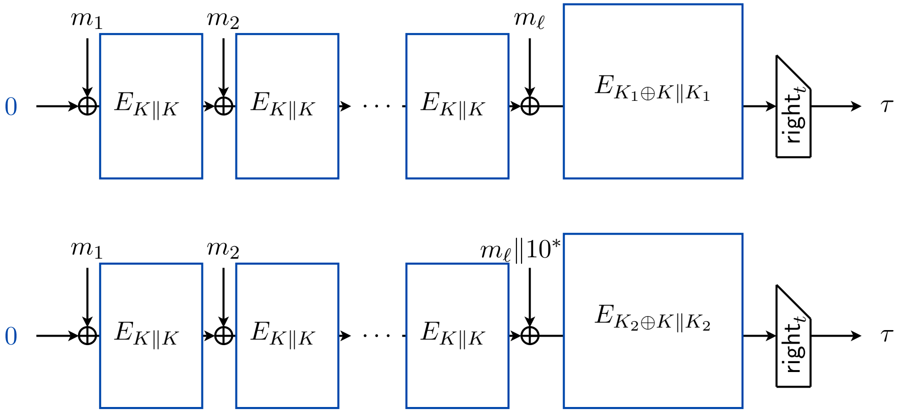
- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$



Chaskey: Mode of Operation: Block-cipher-based

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$

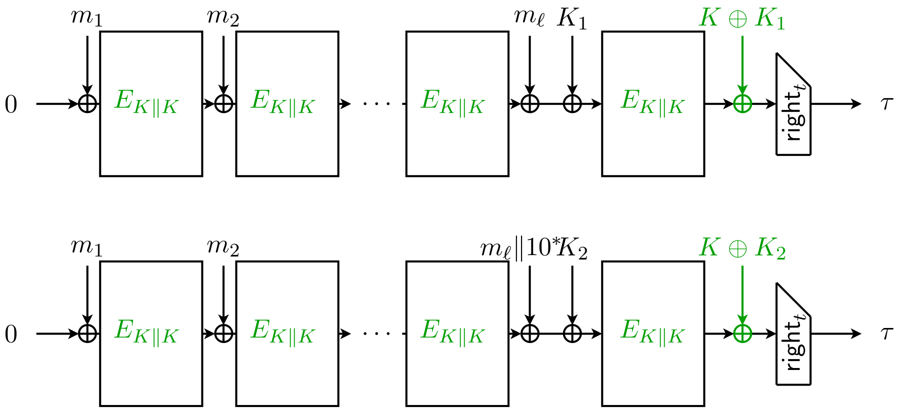
variant of FCBC [BR'00]



Chaskey: Mode of Operation: Compared to CMAC

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$

variant of CMAC [IK'03]

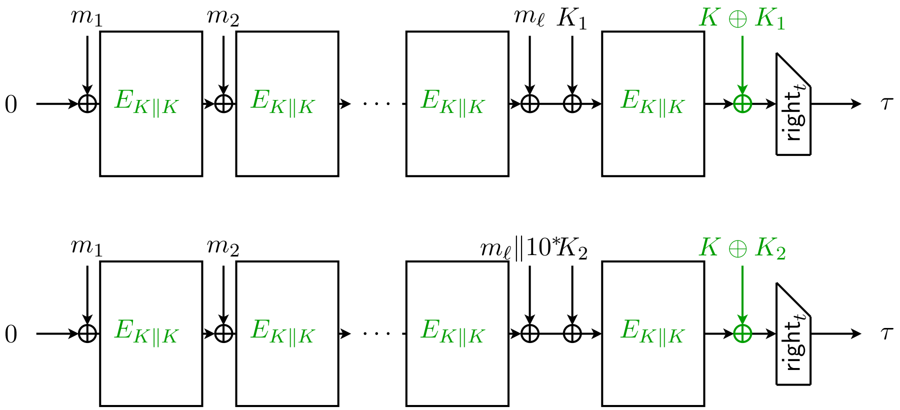


Chaskey: Mode of Operation: Compared to CMAC

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$

variant of CMAC [IK'03]

1 $E_K(0^n) \to K$

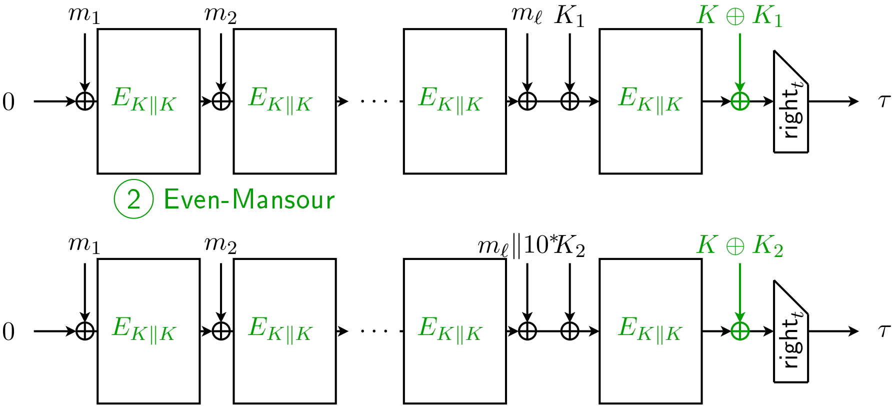


Chaskey: Mode of Operation: Compared to CMAC

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$

variant of CMAC [IK'03]

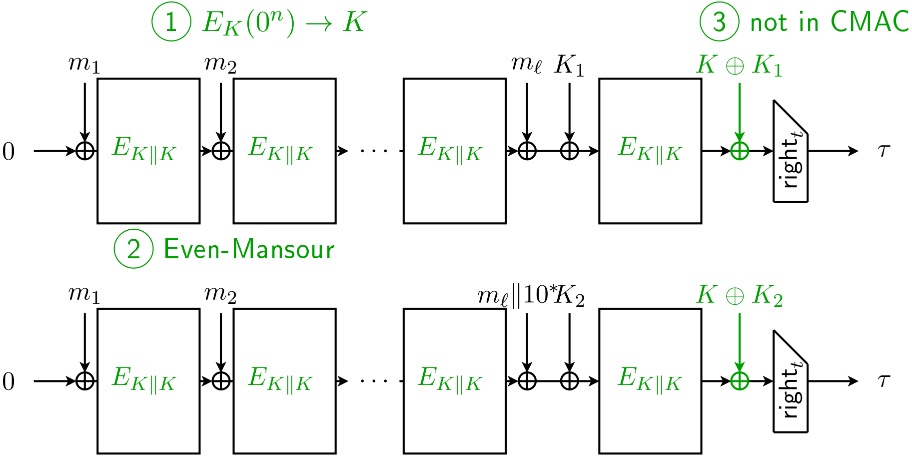
1 $E_K(0^n) \to K$



Chaskey: Mode of Operation: Compared to CMAC

- Split m into ℓ blocks of n bits
- Top: $|m_\ell| = n$, bottom: $0 \leq |m_\ell| < n$
- $K_1 = 2K$, $K_2 = 4K$

variant of CMAC [IK'03]



Cryptanalysis

MAC forgery: find new valid (m, τ)

- D: data complexity (# chosen plaintexts)
- T: time complexity (# permutation eval.)

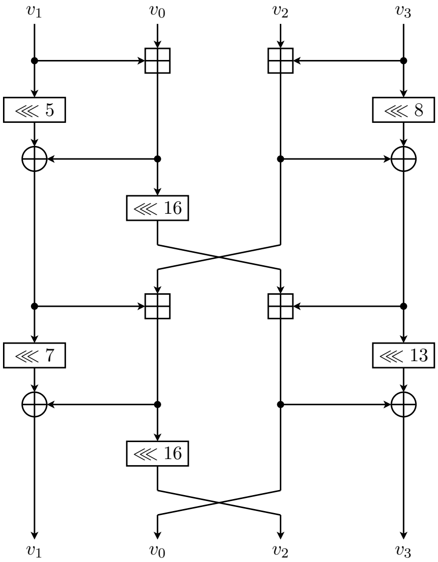
Attacks

- Internal collision: $D \approx 2^{n/2}$
- Key recovery: $T \approx 2^n / D$
- Tag guessing: $\approx 2^t$ guesses

Chaskey parameters

- Key size, block size: $n = 128$, tag length: $t \geq 64$

Permutation



Design

- Add-Rot-XOR (ARX)
- Inspired by SipHash
- 32-bit words
- 8 rounds

Properties

- Rotations by 8, 16:
  faster on 8-bit μC
- Fixed point: $0 \to 0$
- Cryptanalysis: rotational, (truncated) differential, MitM, slide,... see paper!

Chaskey: Speed Optimized (gcc -O2)

Microcontroller	Algorithm	Data [byte]	ROM [byte]	Speed [cycles/byte]
Cortex-M0	AES-128-CMAC	16	13 492	173.4
		128	13 492	136.5
	Chaskey	16	1 308	21.3
		128	1 308	18.3
Cortex-M4	AES-128-CMAC	16	28 524	118.3
		128	28 524	105.0
	Chaskey	16	908	10.6
		128	908	7.0

Chaskey: Size Optimized (gcc -Os)

Microcontroller	Algorithm	Data [byte]	ROM [byte]	Speed [cycles/byte]
Cortex-M0	AES-128-CMAC	16	11 664	176.4
		128	11 664	140.0
	Chaskey	16	414	21.8
		128	414	16.9
Cortex-M4	AES-128-CMAC	16	10 925	127.5
		128	10 925	89.4
	Chaskey	16	402	16.1
		128	402	11.2

Summary

Chaskey:

MAC algorithm for 32-bit microcontrollers

- Addition-Rotation-XOR (ARX)
- Even-Mansour block cipher
- ARM Cortex-M: 7-15$\times$ faster than AES-128-CMAC

Questions?