Lightweight MACs from Universal Hash Functions

Sébastien Duval

Gaëtan Leurent

November 13, 2019

Authentication

- Need for lightweight crypto
- Relatively few authentication solutions compared to encryption
- **Our goal**: design a fast MAC, MAC611, for 32-bit micro-controllers

MACs

Message Authentication Code

- Definition: Tag $t = H_k(m)$
- Security: bound on the proba. that an adversary forges a valid tag

MAC constructions:

- Block-Cipher-based (CBC-MAC, PMAC)
- Hash-Function-based (HMAC)
- from scratch (Pelican MAC, Chaskey)
- from Universal Hash Functions (GMAC, Poly1305-AES)

Lightweight MACs

Chaskey, SipHash, TuLP, LightMAC, QUARK, SPONGENT
Aim for 64-bit security

MACs

Message Authentication Code

- Definition: Tag $t = H_k(m)$
- Security: bound on the proba. that an adversary forges a valid tag

MAC constructions:

- Block-Cipher-based (CBC-MAC, PMAC)
- Hash-Function-based (HMAC)
- from scratch (Pelican MAC, Chaskey)
- **from Universal Hash Functions** (GMAC, Poly1305-AES)

Lightweight MACs

Chaskey, SipHash, TuLP, LightMAC, QUARK, SPONGENT
Aim for 64-bit security

[Almost] Universal Hash Functions

A family $H : A \to B$ is:

ε-almost universal (ε-AU)

$$\forall m \neq m' \in A,\ |\{h \in H : h(m) = h(m')\}| \leq \varepsilon |H|$$

ε-almost XOR universal (ε-AXU)

$$\forall m \neq m' \in A,\ \forall d \in B,\ |\{h \in H : h(m) \oplus h(m') = d\}| \leq \varepsilon |H|$$

H ε-AXU $\Rightarrow$ H ε-AU, and $G : A \times B \to B$ ε-AU:

$$G = \{(m_1, m_2) \mapsto h(m_1) \oplus m_2 : h \in H\}$$

Deterministic MACs

- CBC-MAC, PMAC, HMAC, Pelican MAC, Chaskey
- Birthday bound: collision in internal state (Preneel and van Oorschot)
- Collision attack with $2^{n/2}$ data, with n the state size

For lightweight MACs:

- 64-bit security $\Rightarrow \geq$128-bit state size
  or
- Non-deterministic MAC

Wegman-Carter MACs

n: tag/state size q: number of oracle queries

Wegman-Carter

H ε-AXU, F a PRF:

$$\mathrm{WC}[H,F]_{k_1,k_2}(M,N) = H_{k_1}(M) \oplus F_{k_2}(N)\,,$$

$$\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{WC}[H,F]}(q) \leq \mathbf{Adv}^{\mathrm{PRF}}_{F}(q) + \varepsilon + 2^{-n}\,.$$

Wegman-Carter-Shoup (GMAC, Poly1305-AES)

$$\mathrm{WCS}[H,E]_{k_1,k_2}(M,N) = H_{k_1}(M) \oplus E_{k_2}(N)\,,\ E \text{ a BC},$$

$$\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{WCS}[H,E]}(q) \leq \mathbf{Adv}^{\mathrm{PRF}}_{E}(q) + \varepsilon + 2^{-n}\,,$$

PRF-PRP switching lemma: $\mathbf{Adv}^{\mathrm{PRF}}_{E}(q) \leq \mathbf{Adv}^{\mathrm{PRP}}_{E}(q) + \frac{q^2}{2^n}$.

Sub-optimality of WCS

Usual MACs: GMAC and Poly1305-AES are WCS MACs:

$$\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{WCS}[H,E]}(q) \leq \mathbf{Adv}^{\mathrm{PRP}}_{E}(q) + \frac{q^2}{2^n} + \varepsilon + 2^{-n},$$

- Non-deterministic
- Still only $2^{n/2}$ security
- Not better than deterministic MACs

Using Beyond-the-Birthday-Bound-secure MACs (BBB-secure): better cost-security trade-off

Beyond-birthday Secure MACs

WMAC: $F(H(M) \| N)$

N a nonce, H ε-AU, F a $2n$-bit PRF:

$$\mathrm{WMAC}[H,F]_{k_1,k_2}(M,N) = F_{k_2}(H_{k_1}(M) \| N),$$

Nonce-respecting: $\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{WMAC}[H,F]}(q) \leq \mathbf{Adv}^{\mathrm{PRF}}_{F}(q) + \varepsilon + 2^{-n}$.

BC-based: $\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{WMAC}[H,E]}(q) \leq \mathbf{Adv}^{\mathrm{PRP}}_{E}(q) + \frac{q}{2^{3n/2}} + \varepsilon + 2^{-n}$.

Nonce-misuse: Birthday bound

EWCDM

N a nonce, H ε-AXU, E a block-cipher:

$$\mathrm{EWCDM}[H,E]_{k_1,k_2,k_3}(M,N) = E_{k_3}\big(H_{k_1}(M) \oplus E_{k_2}(N) \oplus N\big),$$

$$\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{EWCDM}[H,E]}(q) \leq \mathbf{Adv}^{\mathrm{PRP}}_{F}(q) + \frac{q}{2^n} + \frac{q^2\varepsilon}{2^n} + \frac{1}{2^n}.$$

Our Objective

MAC611:

- Fast on 32-bit micro-controllers
- Based on ε-AXU: lightweight + security based on combinatorial arguments
- Comparison/benchmarks of different ε-AXU schemes
- Nonce-based and BBB-secure
- Roughly 64-bit security (2^{61})
- Small field for faster operations
- Same structure as GMAC and Poly1305-AES, but faster

Choice of Micro-controllers

ARM Cortex-M4

- Efficient integer multiplier
- Large amount of RAM
- Efficient built-in timer

ARM Cortex-M0+

- Not always efficient integer multiplier
- Small amount of RAM
- No built-in timer
- Limited set of ASM instructions

Choice of MAC Mode: WMAC

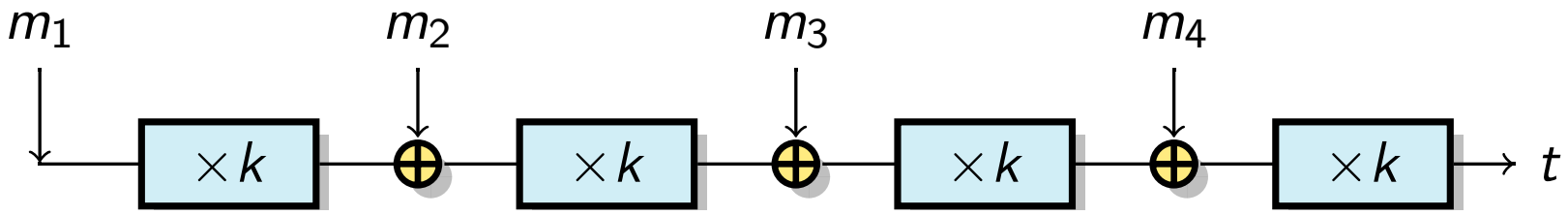
WMAC or EWCDM?
$\Rightarrow$ **WMAC**

Why?

- No good publicly available implementation of a 64-bit BC for 32-bit micro-controllers
- Better security

Use a 128-bit BC: Noekeon

Choice of Almost Universal Hash Function



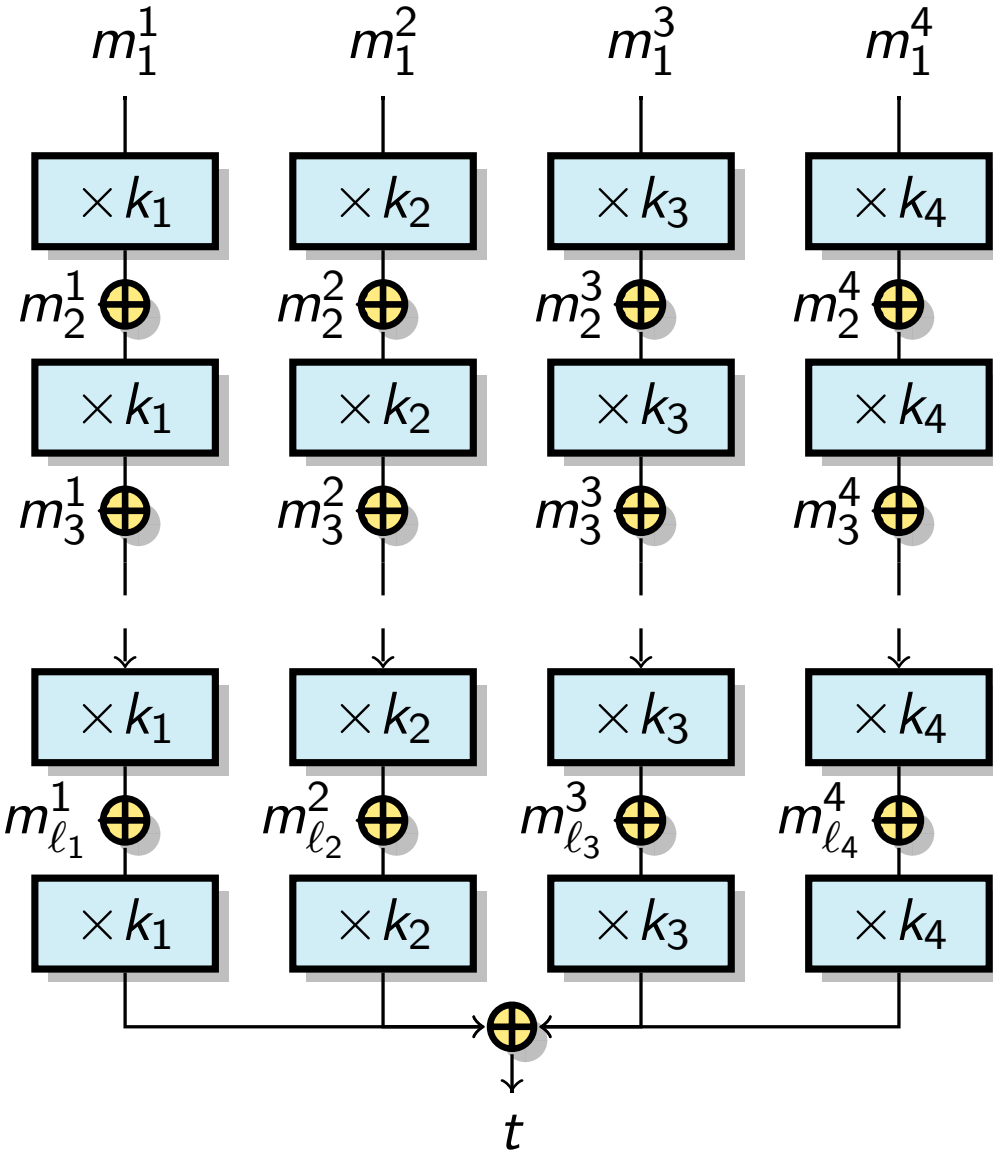
Polynomial Hashing

Hash long message with 1 key.

$$H : \mathbb{F}^{\ell} \to \mathbb{F} = \{m_1, m_2, \ldots m_{\ell} \mapsto \sum_{i=1}^{\ell} m_i \times k^i : k \in \mathbb{F}\}.$$

H is $\ell\varepsilon$-AXU, with ℓ the number of message blocks
How to avoid security loss with message length?

Choice of Almost Universal Hash Function (Sum)



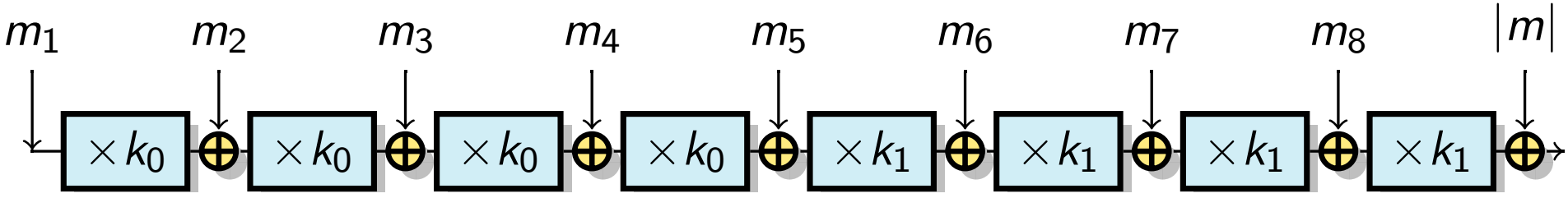
XOR of polynomials of length λ: $2\lambda\varepsilon$-AXU.

Advantage: Parallel (not useful on micro-controllers)

Disadvantage: State of $2n$ bits

Choice of Almost Universal Hash Function (Composition)

XPoly:



Composition of polynomials of length λ: $\lambda\varepsilon$-AU.
($\sim$ Rekeying)
Addition of message length at the end.

Disadvantage: not parallel (not a problem on micro-controllers)

Advantage: State of n bits

Choice of Field and Multiplication

Relevant options:

- binary field $\mathbb{F}_{2^{64}}$
- prime field $\mathbb{F}_p$, with $p \sim 2^{64}$

Using the multiplier

If on a prime field:

1. multiplication in $\mathbb{N}$
2. reduction

Can use built-in integer multiplier (if efficient)

Choice of Field and Multiplication (Table)

Table-based implementation

Multiplication by a fixed element (key) is linear:

$$x \in \mathbb{F} \text{ as } x = \sum_{0 \leq i < n/u} x_i \times 2^{ui},$$

then

$$x \times k = \sum_{0 \leq i < n/u} x_i \times 2^{ui} \times k.$$

Pre-compute and store all $x_i \times 2^{ui} \times k$.
The larger u:

- the faster the multiplication (online)
- the larger the table ($\frac{n^2 2^u}{u}$ bits)

Choice of Field and Multiplication

Field	Implem.	Mem.	Cortex-M0+		Cortex-M4	
			mul (c/B)	table (c)	mul (c/B)	table (c)
$\mathbb{F}_{2^{128}}$	1-bit chunks	4KB	148	3984	128	2756
	4-bit chunks	8KB	48	16992	35	10918
	8-bit chunks	64KB	-	-	19	104922
$\mathbb{F}_{2^{64}}$	1-bit chunks	1KB	91	1440	85	1131
	4-bit chunks	2KB	21	6144	19	3769
	8-bit chunks	16KB	12	53184	11	40142
$\mathbb{F}_{2^{128}}$	GMAC	256B	95	?	53	?
$\mathbb{F}_{\mathbf{2^{61}-1}}$	MAC611	-	**19**	-	**3.7**	-
$\mathbb{F}_{2^{130}-5}$	Poly1305-AES	-	94	-	5	-

- Choice: $\mathbb{F}_{2^{61}-1}$ $\rightarrow$ table-based or multiplier-based implementations.

MAC611

λ: number of message blocks

- Polynomial hashing over $\mathbb{F}_{2^{61}-1}$ of length $\lambda = 1024$
- Composition of polynomial hashing
- Finalization by concatenating a 64-bit nonce and encrypting with Noekeon

Implementation strategies:

- When a good multiplier is available, use it (M4, some M0+), otherwise tables
- Elements in $\mathbb{F}_{2^{61}-1}$ encoded on 64 bits: can do several operations before reduction
- Pre-compute the first table (if table-based)

Benchmarks (M0+)

Algorithm	Implem.	Code size (bytes)			Speed (cycles)			
		MAC	Noekeon	Total	56B	896B	7168B	Long
MAC611	Small	542	636	1178	7.7k	42.6k	307k	42 c/B
	Fast	1948	736	2682	5.0k	20.8k	142k	19 c/B
	Tables (16MB)	1507	692	1923	4.4k	22.2k	228k	27 c/B
Poly1305	OpenSSL (-Os)	1478	636	2114	12.3k	93.8k	709k	99 c/B
	OpenSSL (-O3)	3356	692	4048	9.7k	87.6k	676k	94 c/B
GMAC	OpenSSL (-Os)	2212	636	2848	14.9k	110 k	827k	115 c/B
	OpenSSL (-O3)	3440	692	4132	11.5k	89.8k	681k	95 c/B
Chaskey-12	B. Haase	916	-	916	1.6k	12.6k	97k	14 c/B
CBC-MAC	OpenSSL (-Os)	388	636	1024	24.4k	27.2k	211k	293 c/B
	OpenSSL (-O3)	820	692	1512	14.1k	154 k	118k	165 c/B

Benchmarks (M4)

Algorithm	Implem.	Code size (bytes)			Speed (cycles)			
		MAC	Noekeon	Total	56B	896B	7168B	Long
MAC611	Small	842	348	1190	1247	4247	27k	3.7 c/B
	Fast	1168	3724	4892	1029	4029	27k	3.7 c/B
Poly1305	OpenSSL	918	348	1266	1761	5501	35k	5.0 c/B
GMAC	OpenSSL	2168	348	2516	5824	49684	382k	53 c/B
Chaskey-12	C Ref (-Os)	1058	-	1058	905	8305	65k	9.1 c/B
	C Ref (-O3)	1436	-	1436	823	7468	58k	8.2 c/B
CBC-MAC	OpenSSL (-Os)	380	348	728	4863	52398	392k	54 c/B
	OpenSSL (-O3)	828	3724	4552	3996	42881	319k	44 c/B

Conclusion

MAC611

- Unlike mainstream MACs Poly1305-AES and GMAC, we used a BBB-secure mode
- Smaller field
- Faster on 32-bit micro-controllers with comparable security

In the paper:

- Theoretical results on MAC modes using permutations
- Improved security bound for PC-MAC (by Minematsu and Tsunoo)