

Shallow MAC: MAC with precomputation



Juan Garay

AT&T Labs

Vladimir Kolesnikov

Alcatel-Lucent Bell Labs

Rae McLellan

Alcatel-Lucent Bell Labs

Outline

- Intro
 - Fast MAC uses in secure hardware
- Background
 - MAC
 - PRFG
 - Differential Uniformity
- ShMAC construction
- Related work

The Challenge

- Hardware operation in hostile environment



Protection of

- software and other IP
- device entrusted secrets
- integrity of operation of device

Enforcement of

- DRM
- business model

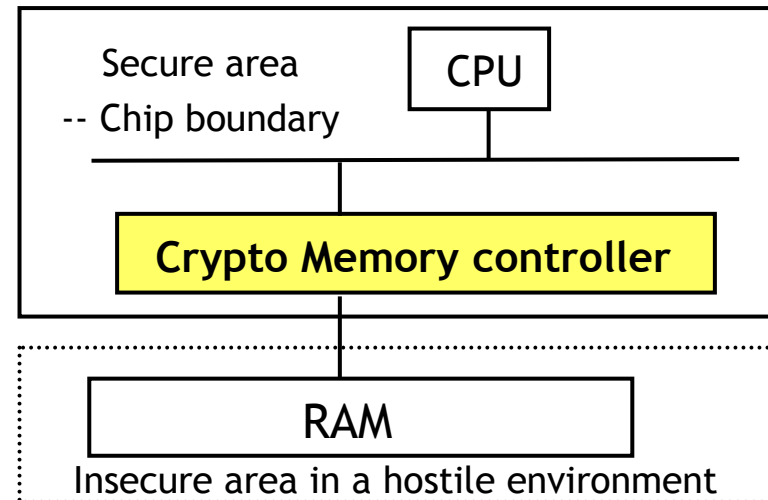
Challenge:

Protect against attacker who has full physical control of the device
-- can examine and modify state

Our solution

System On Chip (SoC)

- Assumed infeasible to break inside standard SoC
- Remains to protect interfaces



Sufficient hardware support to provide:

- Secure boot
- Secure execution even with insecure RAM

Light-weight crypto-based memory controller

- Protects against reverse engineering and tampering
- Strong security with minimal HW overhead
- Rigorous cryptographic design and analysis

Message Authentication Code (MAC)

- Symmetric key equivalent of public key signatures
 - Need secret key to sign and to verify
 - Cannot sign without secret key

We use nonce-based MAC:

MAC: $\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \rightarrow \text{TAG}$

Verification: recompute and compare the tag

Security:

$k \in_R \{0,1\}^n$, Adv can access oracle $O(r,m) = \text{MAC}_k(r, m)$

Adv cannot forge MAC:

Output (m', r', t') , where he never queried $t' = O(r', m')$

Pseudo-random Function Generator

- Models a randomly chosen function
 - Indexed by a secret key
 - Hard to predict, even knowing values in other locations
 - Example: AES

$$F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$$

Security:

$k \in_R \{0,1\}^n$, Adv can access oracle $O(m) = F_k(m)$

Adv cannot distinguish values of $F_k(m')$ from random:

- Can be used to produce (pseudo)randomness: $F(0)$, $F(1)$, ...

MAC from PRFG

$$\text{MAC}_k(m) = F_k(m)$$

Thm: if F is a PRFG, then MAC_k is a secure MAC

Intuition: MAC forgery is a PRFG forgery

Suppose A breaks MAC by constructing forgery $(m, t = \text{MAC}_k(m) = F_k(m))$. Then clearly can distinguish $F_k(m)$ from random.

Faster MAC

But we want more efficient than PRFG
Very few more efficient primitives

Intuition of MAC:

- Mix the bits really well; every input bit affects every output bit in a key-dependent manner
- Must maintain resistance against multiple queries

Idea: use precomputation

Start with precomputed randomness, then mix bits using weaker mixing primitives.

ϵ - Differential Uniformity (ϵ -DU)

- Notion from differential cryptanalysis

Given G (or G_k) , $\forall a, b$:

$$\Pr_X[G(X) \oplus G(X \oplus a) = b]$$

is small ($< \epsilon$)

- Intuition
 - any pair of offsets (a,b) has low probability of satisfying above
 - assures good “bit mixing” by G
 - But:
 - X, k must be random; if they are not, could choose (a,b)
- Very efficient ϵ -DU functions exist:
 - AES2 $\max \Pr = 2^{-28}$
 - AES4 $\max \Pr = 2^{-110}$

ShMAC from ϵ -DU

ϵ -DU G is the core of ShMAC

- Must randomize the input to G
 - precomputation: offset m by $F(r)$ (r is a nonce)
- $MAC = G (F(r) \oplus m)$

Intuition:

Random $F(r)$ offsets m , so multiple queries to G leak no info (G is a permutation).

To fake MAC

- on a new nonce r , need to break F
- on an old nonce r , need to guess (a,b) that breaks G

ShMAC from ϵ -DU take 2

$$\text{MAC} = G(F(r) \oplus d)$$

ϵ -DU G guarantees input/output unpredictability on *random* inputs

But Adv knows MAC of some strings (e.g. of $G(F(r) \oplus m)$)

Thus he may predict value of $G(F(r) \oplus m \oplus a)$ and use it for MAC forgery

Solution: offset G by a random value

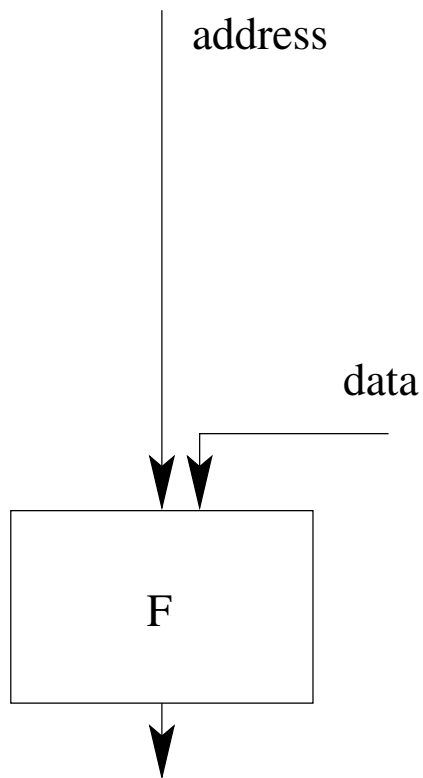
$$\text{MAC}_k = G(F(r) \oplus m) + k$$

- new use of the notion of ϵ -DU
 - previous uses in blockcipher design

Application of ShMAC to Memory authentication

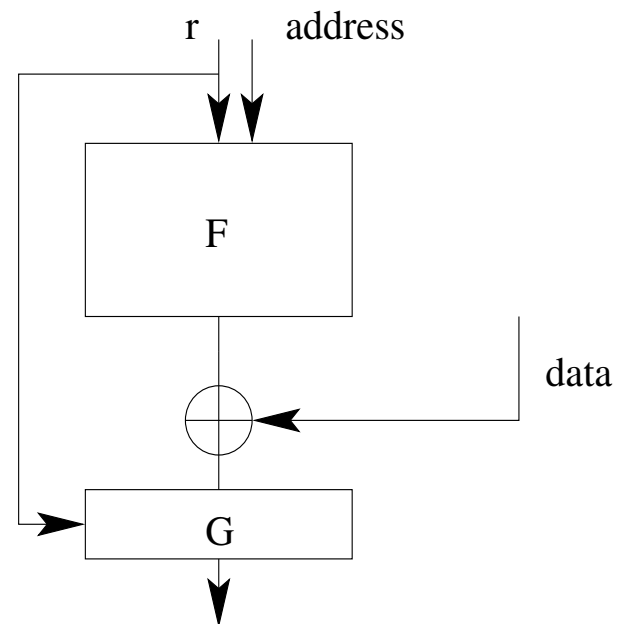
Standard MAC scheme:

$$m = F_k(\text{addr}, d)$$



Shallow MAC scheme:

$$m = G_k(d \oplus F_k(r, \text{addr}), r)$$



G is much faster than F.

Advantage: use idle time that is spent waiting for the data

Related Work on MAC precomputation

Carter-Wegman (CW) MAC:

$$H_k(m) \oplus r$$

where H is Strongly Universal (SU), r is random pad
(need to know k , r to verify MAC)

Lots of work on SU, mostly algebraic solutions – not good for us

Krawczyk (Crypto94): Almost XOR-Universal (AXU) functions sufficient
AXU functions are related to, but stronger than ε -DU.

Jakimoski, Subbalakshmi (AC07) convert ε -DU to AXU, but pay 1 PRF eval
Our work: avoid this cost.

Comparing to other hardware-assisted solutions

- Smart-card type approach
 - Inexpensive,
 - limited – only basic operations, e.g. signing

- Comprehensive SoC approach
 - Tamper-resistant physical protection
 - Expensive (\$\$)
 - Slows down CPU

- Our approach
 - An industry-targeted compromise, fits between the two