MACs $\quad Sign_K(x) \rightarrow$ signature

Idea: for every message generate (a,b) by $PRNG_{key}$

Improve: a fixed, random; b by $PRNG_{key}$ — both secret!

Polynomial MAC over field F

Message: $x_1 — x_n \in F$

key : $(a,b) \in F^2$

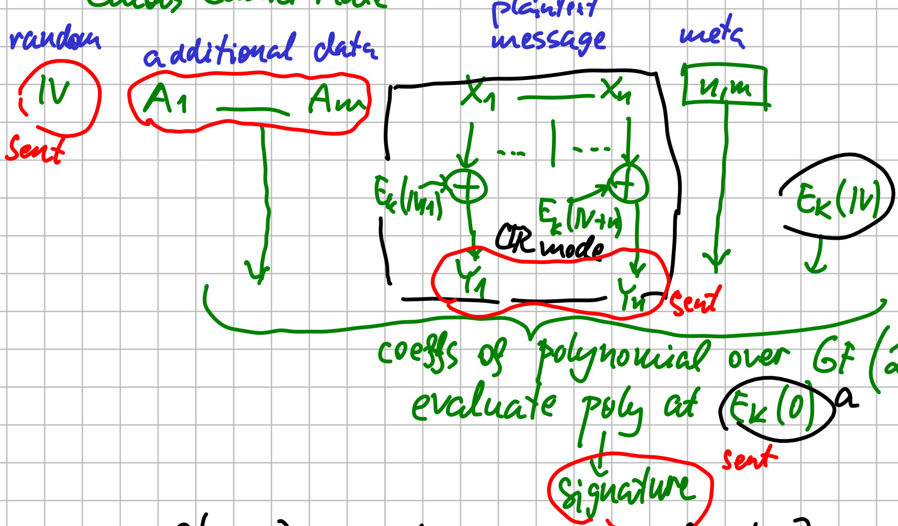
signature $:= x_1 a^n + x_2 a^{n-1} + — + x_n a^1 + b$

$$\Pr_{key}\left[sign_{key}(x') = y' \mid sign_{key}(x) = y \right] \le \frac{n}{|F|} \le \frac{1}{2^{sec.level}}$$

GCM mode of block ciphers

≙ Galois Counter Mode

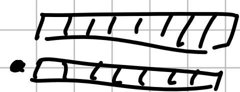
Authenticated Encryption with additional data



for AES ↓ elements are 128-bit strings

addition is XOR

multiplication of polynomials over $\mathbb{Z}_2$



Carry-less multiplication (CLMUL instruction)

mod some irreducible poly

$x^{128}+1$?

$O(m+n)$ operations in the field
$O(n)$ block encryptions } → fast

Poly 1305 [Bernstein 2005]

Poly MAC construction from a block cipher < AES, Chacha 20, any other

computed in $GF(2^{130}-5)$ (prime)

quite tricky & very fast

Random Generators

We want: Even if the attacker knows complete past output, the next bit is unpredictable.

↳ implies statistical uniformity

pseudo-random gen

e.g. block cipher in CTR

$E_k(0), E_k(1), \ldots$

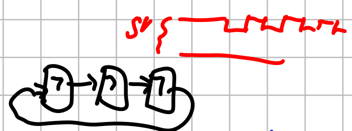
re-key using phys. randomness

$k' \leftarrow h(k \,\|\, rand.\ input)$

Linux: /dev/random

"physical randomness"

- thermal noise at resistor/diode
- radioactive decay
- radio noise
- lava lamp
- ring oscillator
- precise timing of keys, mouse, net packets



The attacker can
- observe
- influence

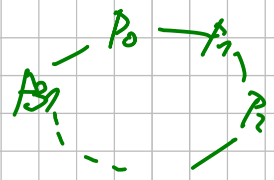
problem: if key (state of PRNG) was compromised

mix 1 bit of entropy to the state

↳ attacker can guess & track the state

↳ need to mix entropy in large batches

Fortuna [Ferguson & Schneier 2003]



Generator based on AES with 256-bit key
encrypts 128-bit counter (never overflows)
after 2^{16} blocks, we re-key: the next 2 blocks become the key
trick: don't reset the counter on re-key (to avoid short cycles)

Accumulator pools P_0 — P_{31} for accum. entropy
external randomness is mixed to the pools in round-robin order
$P_i \leftarrow hash(P_i \,\|\, input)$
after some time, we mix pools to key of the generator
$k' \leftarrow hash(k \,\|\, P_i)$, $P_i \leftarrow \emptyset$

we mix
P_0 every time
P_1 once in 2
P_2 4
P_3 8
⋮

↳ we mix each 100 ms
in j-th mixing we use all P_i with $2^i \backslash j$

b-bit

ρ :: rate of input entropy : bits / 100 ms

$\rho \geq \underbrace{128}_{\hookrightarrow P_0} \cdot 32 \Rightarrow P_0$ is enough for recovery

$\rho \geq 128 \cdot 32 / 2^i \Rightarrow P_i$ is enough

Secure Channel

symmetric

$A \rightleftarrows B$
A: k, B: k — unique & random

If A sends m_1 —— m_n
B receives a sub-sequence & knows which
attackers have no inf. except n

We use:

- random IV for every message (public)
- AES in CTR mode for encryption (ChaCha20)
- MAC (after encrypt)
- sequence numbers (inside MAC) (public)
- key derivation function $hash(\underbrace{k}_{\text{master secret}} \,\|\, id)$ → encr. keys $A \to B$, $B \to A$; MAC keys $A \to B$, $B \to A$

Algorithmic Number Theory

- arithmetics with b-bit numbers

$O(b \log b)$ using FFT
on a RAM with $\log b$-words ... $O(b)$
$n \log n$, $n = \frac{b}{\log b}$

$+, -$ $O(b)$
$*$ $O(b^2)$ or better... $O(b)$ practical: $O(b^{1.\dots})$
$/, \%$ $O(b^2)$

x^k .... by repeated squaring $x^{2k} = (x^k)^2$, $x^{2k+1} = (x^k)^2 \cdot x$
$O(\log k)$ multiplications

$x^k \bmod N$ $O(b^3)$

Next: More number theory
NextNext: RSA cryptosystem & similar asymmetric cipher