Crypto recap

$\mathbb{Z}_n^* = \{a \in \{0,1,\dots,n-1\} \mid \gcd(a,n) = 1\}$ — a & n are whole numbers

$\varphi(n = p_1^{e_1} \cdot \dots \cdot p_\ell^{e_\ell}) = (p_1^{e_1} - p_1^{e_1-1}) \cdot \dots \cdot (p_\ell^{e_\ell} - p_\ell^{e_\ell-1})$

fact: there is an efficient algorithm $EEA(a,b) = (u,v,c)$ ($a, b \in \mathbb{N}$) such that $u \cdot a + v \cdot b = c = \gcd(a,b)$ — Extended Euclidean algorithm

$u, v \in \mathbb{Z}$

$a^x \bmod n$ — square & multiply

example: $a^\ell \bmod n = 3^{13} \bmod 7$

$13 = 1101$ in binary

$3^{13} \bmod 7 = ((3^2 \cdot 3^1)^2 \cdot 3^0)^2 \cdot 3^1 \bmod 7 = (((3^2 \cdot 2)^2)^2 \cdot 3 \bmod 7$

fact: the square & multiply algorithm computes $a^x \bmod n$ efficiently iterating between squarings and multiplications

fact: using the EEA we can compute $a/b \bmod n = a \cdot b^{-1} \bmod n$ in $\mathbb{Z}_n^*$ — $b \cdot b^{-1} = 1 \bmod n$

given (b,n) → EEA gives u,v,c such that $u \cdot b + v \cdot n = 1 \Rightarrow u = b^{-1} \bmod n$

fact: (Fermat's Little Theorem)

$$\forall a \in \mathbb{Z}_n^* : a^x \bmod n = a^{x \bmod \varphi(n)} \bmod n$$

$x \in \mathbb{Z}$

fact: given $\varphi(n)$, e with $\gcd(e, \varphi(n)) = 1$, we can compute $a^{\frac{1}{e}} \bmod n$ efficiently, $a \in \mathbb{Z}_n^*$

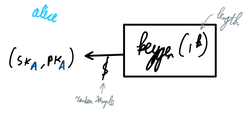
1) compute $e^{-1} \bmod \varphi(n) = d$ via EEA
2) $a^d \bmod n$ via square & multiply

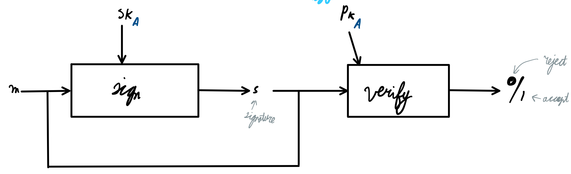
RSA assumption: if $\varphi(n)$ is unknown, computing $a^{\frac{1}{e}} \bmod n$ is believed to be hard

computing $\varphi(n)$ is believed to be hard unless the prime factorization is known

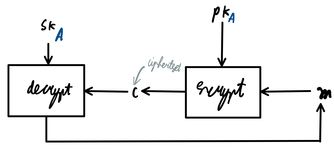
Building blocks

digital signatures

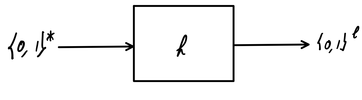




public key encryption



hash function



idealized hash function: fill table on the fly — random oracle

input	output
m	$h(m) \leftarrow \$ \{0,1\}^\ell$

if new input m
output uniformly random in $\{0,1\}^\ell$
store $(m, h(m))$
else
output $h(m)$

Lecture 12 - part 2

Tuesday, 10 January 2023 11:49

encapsulation problems { side-channel attacks → timing & power consumption ; fault injection attacks

error handling — Bleichenbacher oracle

dependence on environment — bad randomness

or problems when using security features/tools

information obtained from implementation not conceptual algorithm

cache attacks : frequency of data use determines whether data is stored in cache or only in memory

↓

faster, so can leak info on secret keys (in principle)

suggestion : avoid data-dependent table lookups

countermeasures : suppress extra information

disturb correlation between extra information and data

e.g. apply randomization, blinding/masking

RSA-FDH ; Alice hashes message, then 'decrypts' hash to sign

$$s = (h(m))^d \bmod n$$

$$s_p = (h(m))^d \bmod p = (h(m))^{d_p} \bmod p, \text{ where } d_p = d \bmod p-1 = d \bmod \varphi(p)$$

similar for q

use EEA to find u, v such that $u \cdot p + v \cdot q = 1$

then $s = s_q \cdot u \cdot p + s_p \cdot v \cdot q$

d_p and d_q can be precomputed

fault injection : assume we can make the device incorrectly compute s_p, while s_q is correctly computed

↓

leaks q ...

countermeasures :

blind d

blind modulus n

blind group element

check signature outcome/validity