3/4/20

Today:

RSA: Public key encryption

Digital signature scheme

Diffie & Hellman's vision: ("New Directions in Cryptography", 1976)

Construct public-key cryptography from trapdoor functions

Def: A trapdoor permutation family $\{f_k\}$ is a family of permutations $f_k : D_k \to D_k$ s.t.

$\exists$ PPT key generation alg Gen s.t. on input 1^λ outputs (k, τ) s.t.

trapdoor

- $k, x \in D_k \xrightarrow{\text{easy}} f_k(x)$

- $\tau, f_k(x) \xrightarrow{\text{easy}} x$

- $k, f_k(x) \xrightarrow{\text{hard}} x$

Given such family one can construct a (det.) public key enc. scheme (Gen, Enc, Dec):

Gen(1^λ): Generate $(pk, sk) = (k, \tau)$ by running key gen alg of the trapdoor family.

$\underline{Enc(k, m)} = f_k(m)$

$\underline{Dec(\tau, c)} = f_k^{-1}(f_k(m))$.

This is clearly not CPA secure!

Was suggested as a heuristic (w.o. formal security def).

Moreover using such a family one can digitally sign!

Def: A digital signature scheme consists of PPT alg (Gen, Sign, Ver):

$Gen(1^n)$ generates (PK, SK) (PK often denoted by VK for verification key)

$Sign(SK, m)$ outputs signature σ

$Ver(PK, m, \sigma)$ outputs $0/1$.

Correctness: $\Pr_{(PK,SK) \leftarrow Gen(1^n)}[Ver(PK, m, Sign(SK, m)) = 1] = 1$

Signature scheme based on trapdoor permutations:

$\mathrm{Gen}(1^\lambda)$: generates $(pk, sk) = (k, \tau)$ corresponding to the trapdoor family

$\mathrm{Sign}(\tau, m) = f_k^{-1}(m)$

$\mathrm{Ver}(k, m, \sigma) = 1$ iff $f_k(\sigma) = m$.

(We will talk about security later).

Rivest-Shamir-Adleman (RSA 1977)

First (and only) construction of trapdoor permutation family:

except recent construction from obfuscation

$\mathrm{KeyGen}(1^\lambda)$: Choose at random primes $p, q \in \{0,1\}^\lambda$

Let $n = p \cdot q$. Choose e s.t. $\gcd(e, \varphi(n)) = 1$

$\varphi(n) = |\mathbb{Z}_n^*| = (p-1)(q-1)$

$k = (n, e)$ $\quad d = e^{-1} \bmod \varphi(n)$

$\tau = (n, d)$

$f_k : \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_n^*$

$f_k(x) = x^e \bmod n$.

$f_k^{-1}(y) = y^d \bmod n$

<u>Note</u>: $(x^e)^d \bmod n = x$ ✓

For efficiency, often $e=3$

Need : $\gcd(3, (p-1)\cdot(q-1)) = 1$ $\quad (p, q = 2 \bmod 3)$

<u>Note</u> : $d = e^{-1} \bmod \varphi(n)$ is computed using the

extended GCD alg.

(can also be computed by
$d = e^{|\mathbb{Z}^*_{\varphi(n)}|-1} \bmod \varphi(n)$
not clear how to compute this)

Given $a, b \in \mathbb{Z}$ outputs $x, y \in \mathbb{Z}$ s.t. $a\cdot x + b\cdot y = \gcd(a,b)$.

If $\gcd(a,b) = 1$ then
$x = a^{-1} \bmod b$
$y = b^{-1} \bmod a$.

<u>Example</u>: $a=5$, $b=7$

$5 = 1\cdot 5 + 0\cdot 7$

$7 = 0\cdot 5 + 1\cdot 7$

$\Downarrow$

$2 = (-1)\cdot 5 + 1\cdot 7$

$\Downarrow$

$1 = 3\cdot 5 + (-2)\cdot 7 \implies 3 = 5^{-1} \bmod 7$.

<u>Claim</u>: $\{f_k\}$ is a trapdoor permutation family assuming

<u>RSA Assumption</u>

$\downarrow$

$$n, e, x^e \bmod n \xrightarrow{\text{HARD}} x$$

$n = p \cdot q$

$x \leftarrow \mathbb{Z}_n^*$.

RSA scheme based on trapdoor permution (presented above) is called "textbook RSA".

<u>CPA secure version</u> (in <u>Random Oracle Model</u>) (ROM)

Gen(1^λ): Same ✓ $pk = (n, e)$ $sk = d$

Enc(pk, m): Choose $r \leftarrow \mathbb{Z}_n^*$ outpute

Hash function

$$\left(r^e \bmod n, \; H(r) \oplus m\right)$$

Dec(d, (c_1, c_2)): Compute $r = c_1^d \bmod n$

Outpute $m = H(r) \oplus c_2$.

<u>Security</u>: The only way to distinguish

$$(PK, \underset{\substack{\| \\ (r^e \bmod n,\ H(r) \oplus m)}}{Enc(PK, m)})$$ from random is to query H

on input r, but this breaks RSA assumption.

Making RSA CCA secure (in the ROM)

<u>Def</u>: An encryption scheme (Gen, Enc, Dec) is CCA secure

if $\forall$ PPT adv $A^{DEC(SK,\cdot)}(PK)$ that chooses $m_0, m_1 \in M$

s.t. $|m_0| = |m_1|$

$$\Pr_{\substack{(PK,SK) \leftarrow Gen(1^\lambda) \\ b \leftarrow \{0,1\} \\ \text{randomness of } Enc}}[A^{Dec(SK,\cdot)}(PK, \underbrace{Enc(PK, m_b)}_{= c_b}) = b] = \frac{1}{2} + negl(\lambda)$$

assuming A does not send c_b as an oracle query.

<u>Note</u>: RSA & El-Gamal are <u>not</u> CCA secure.

Making RSA CCA secure

the goal of this pad is to make x random & dec. oracle useless

$Gen(1^\lambda)$: Same

$Enc((n,e), m)$: $x^e \bmod n$ $\quad x = Pad_r(m)$

where $Pad_r(m) \xrightarrow{easy} m$

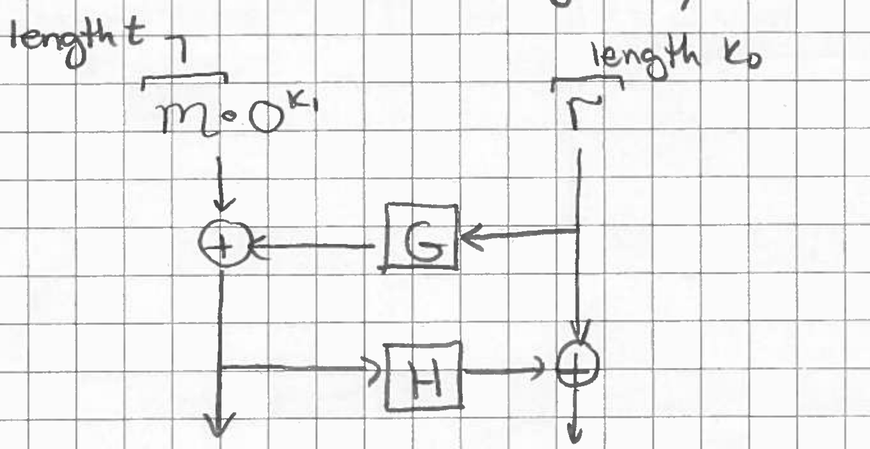
$Dec((n,d), c)$: Compute $x = c^d \bmod n$

Compute

$Pad^{-1}(x) = m$.

PAD: Optimal Asymmetric Encryption Padding (OAEP)

[Bellare-Rogaway 94]



$x_0 = G(r) \oplus (m \circ 0^{k_1})$ $\quad x_1 = r \oplus H(x_0)$

Intuition: To make dec oracle useless let

$pad_r(m) = (r, \overbrace{G(r) \oplus (m \circ 0^{k_1})}^{x_0})$

↑ RO

Intuitively, to generate x^e for valid encoding x must know r.
But may not know m.
To fix this, replace r w. $r \oplus H(x_0)$

CPA security (in the ROM)

Partial information about m is leaked only if r is leaked entirely, which happens only if x_0 is leaked entirely
$\Rightarrow$ (x_0, x_1) leaked entirely.

$\Rightarrow$ CPA security

CCA security follows since $A^{Dec(sk,\cdot)}((x_0 \circ x_1)^e \bmod n)$

cannot use its decryption oracle on CTs that are a func. of $(x_0 \circ x_1)^e \bmod n$.

Suppose A generates $(x_0' \circ x_1')^e \bmod n$ without knowing $(x_0' \circ x_1')$.

$\Rightarrow$ A does not know $x_1' \oplus H(x_0')$ $\Rightarrow$ w.h.p. $x_0' \neq G(\underbrace{r}_{x_1' \oplus H(x_0')}) \oplus (\cdot \circ 0^{k_1})$