Cryptography
CS 555

Topic 31: RSA Attacks + Fixes

Recap

- CPA/CCA Security for Public Key Crypto
- Key Encapsulation Mechanism
- El-Gamal

Recap

- Plain RSA
- Public Key (pk): N = pq, e such that $\mathrm{GCD}\big(\mathrm{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$

$$\mathbf{Enc}_{\boldsymbol{pk}}(\boldsymbol{m}) = \boldsymbol{m^e \ mod\ N}$$
$$\mathbf{Dec}_{\boldsymbol{sk}}(\boldsymbol{c}) = \boldsymbol{c^d \ mod\ N}$$

- Decryption Works because
  $[c^d \mathrm{mod\ N}] = [m^{ed} \mathrm{mod\ N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \mathrm{mod\ N}\right] = [m \ \mathrm{mod\ N}]$

Recap RSA-Assumption

RSA-Experiment: RSA-INV$_{A,n}$

1. **Run KeyGeneration**(1^n) **to obtain (N,e,d)**
2. **Pick uniform** $\mathrm{y} \in \mathbb{Z}_{\mathrm{N}}^*$
3. Attacker A is given N, e, y and outputs $\mathrm{x} \in \mathbb{Z}_{\mathrm{N}}^*$
4. Attacker wins (RSA-INV$_{A,n}$=1) if $x^e = \mathrm{y} \bmod \mathrm{N}$

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}\ \Pr[\text{RSA−INVA}_{,n} = 1] \le \mu(n)$$

(Review) Attacks on Plain RSA

- We have not introduced security models like CPA-Security or CCA-security for Public Key Cryptosystems

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- Plain RSA is also highly vulnerable to chosen-ciphertext attacks
  - Attacker intercepts ciphertext c of secret message m
  - Attacker generates ciphertext c’ for secret message 2m
  - Attacker asks for decryption of c’ to obtain 2m
  - Divide by 2 to recover original message m

(Plain) RSA Discussion

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- In a public key setting the attacker does have access to an encryption oracle

- Encrypted messages with low entropy are vulnerable to a brute-force attack.
  - If $m < B$ then attacker can recover m after at most B queries to encryption oracle (using public key)

Recovering Encrypted Message faster than Brute-Force

Claim: Let m < 2^n be a secret message. For some constant $\alpha = \frac{1}{2} + \varepsilon$.
We can recover m in in time $T = 2^{\alpha n}$ with high probability.

For r=1,…,T

  let $\mathrm{x_r} = [cr^{-e} mod\ N]$, where $r^{-e} = (r^{-1})^e mod\ N$

Sort $\mathbf{L} = \{(\boldsymbol{r}, \boldsymbol{x_r})\}_{\boldsymbol{r=1}}^{\boldsymbol{T}}$ **(by the $\mathbf{x_r}$ values)**

For s=1,…,T

  if $[s^e mod\ N] = \mathrm{x_r}$ for some r **then**

    return $[sr\ mod\ N]$

Recovering Encrypted Message faster than Brute-Force

Claim: Let m < 2^n be a secret message. For some constant $\alpha = \frac{1}{2} + \varepsilon$. We can recover m in in time $T = 2^{\alpha n}$ with high probability.

Roughly $\sqrt{B}$ steps to find a secret message m < B

More Weaknesses: Plain RSA with small e

- (Small Messages) If $m^e < N$ then we can decrypt $c = m^e \bmod N$ directly e.g., $m=c^{(1/e)}$
- (Partially Known Messages) If an attacker knows first 1-(1/e) bits of secret message $m = m_1 \| ??$ then he can recover m given
$$\mathbf{Enc}_{\boldsymbol{pk}}(m) = m^e \bmod \mathrm{N}$$

Theorem[Coppersmith]: If p(x) is a polynomial of degree e then in polynomial time (in log(N), e) we can find all m such that p(m) = 0 mod N and $|m|<N^{(1/e)}$

More Attacks: Encrypting Related Messages

- Sender encrypts m and $m + \delta$, where offset δ is known to attacker

- Attacker intercepts

$$c_1 = \mathrm{Enc}_{pk}(m) = m^e \; mod \; N$$

and

$$c_2 = \mathrm{Enc}_{pk}(m + \delta) = (m + \delta)^e \; mod \; N$$

- Attacker defines polynomials

$$f_1(x) = x^e - c_1 \, mod \; N$$

and

$$f_2(x) = (x + \delta)^e - c_2 \, mod \; N$$

More Attacks: Encrypting Related Messages

$$c_1 = \mathrm{Enc}_{pk}(m) = m^e \; mod \; N$$
$$c_2 = \mathrm{Enc}_{pk}(m+\delta) = (m+\delta)^e \; mod \; N$$

- Attacker defines polynomials

$$f_1(x) = x^e - c_1 \, mod \; N$$

and

$$f_2(x) = (x+\delta)^e - c_2 \, mod \; N$$

- Both polynomials have a root at x=m, thus (x-m) is a factor of both polynomials
- The GCD operation can be extended to operate over polynomials ☺
- GCD($f_1(x)$, $f_2(x)$) reveals the factor (x-m), and hence the message m

Sending the Same Message to Multiple Receivers

- Homework 3 Bonus Question
  - $c_1 = [m^3 \ \ mod \ N_1]$
  - $c_2 = [m^3 \ \ mod \ N_2]$
  - $c_2 = [m^3 \ \ mod \ N_3]$
  - Since $\gcd(N_1, N_2) = \gcd(N_1, N_3) = \gcd(N_2, N_3) = 1$, we can find a unique number $x < N_1 N_2 N_3$ such that $x = m^3 \ mod \ Ni$
  - This, number is $x = m^3$

- Mathematica Demo

Sending the Same Message to Multiple Receivers

- Homework 3 Bonus Question
  - $c_1 = [m^3 \ \ mod \ N_1]$
  - $c_2 = [m^3 \ \ mod \ N_2]$
  - $c_2 = [m^3 \ \ mod \ N_3]$
  - Since $\gcd(N_1, N_2) = \gcd(N_1, N_3) = \gcd(N_2, N_3) = 1$, we can find a unique number $x < N_1 N_2 N_3$ such that $x = m^3 \, mod \ Ni$
  - This, number is $x = m^3$

- **Question**: What if $\gcd(N_2, N_3) > 1$?
  - Either $N_2 = N_3$ or $\gcd(N_2, N_3)$ reveals a shared factor of N_2, N_3

Apply GCD to Pairs of RSA Moduli?

- **Fact**: If we pick two random RSA moduli N_1 and N_2 then except with negligible probability $\gcd(N_1, N_2) = 1$

- In theory the attack shouldn't work, but…

- In practice, many people generated RSA moduli using weak pseudorandom number generators.
  - .5% of TLS hosts
  - .03% of SSH hosts
- See https://factorable.net

Dependent Keys Part 1

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee I subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.

- **Question**: Is this secure?

- **Answer**: No, given $e_i d_i$ employee i can factor N (and then recover everyone else's secret key).
- See Theorem 8.50 in the textbook

Dependent Keys Part 2

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.

- Suppose that each employee is trusted (so it is ok if employee i factors N)

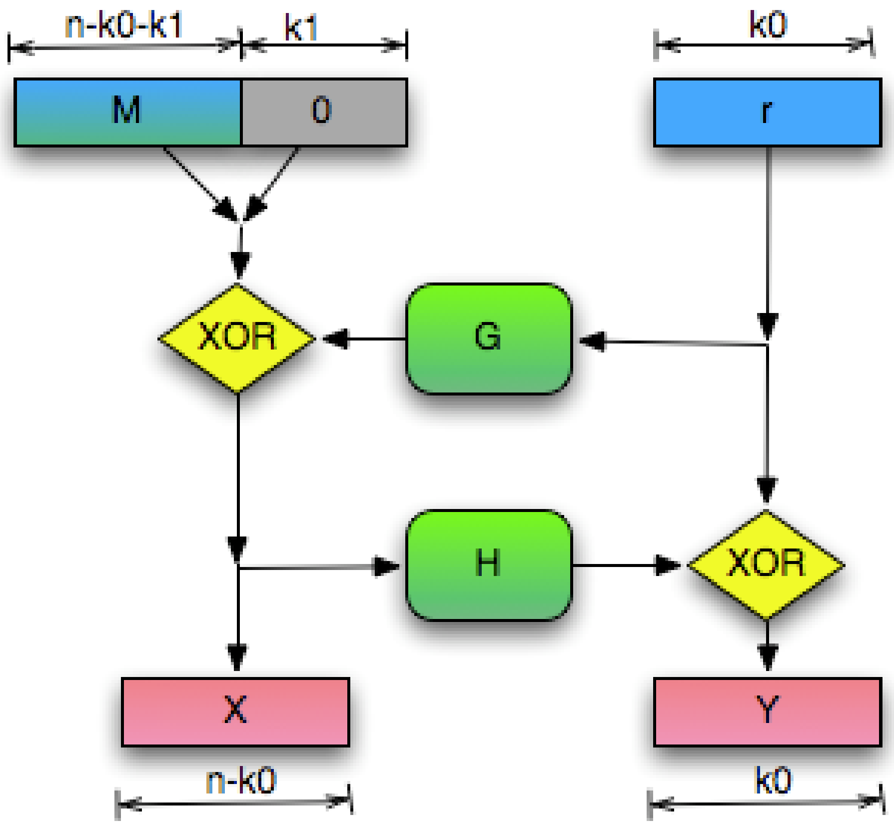
- Suppose that a message m is encrypted and sent to employee 1 and 2.
- Attacker intercepts $c_1 = [m^{e_1}\ mod\ N]$ and $c_2 = [m^{e_2}\ mod\ N_2]$

Dependent Keys Part 2

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.
- Suppose that a message m is encrypted and sent to employee 1 and 2.
- Attacker intercepts $c_1 = [m^{e_1}\ mod\ N]$ and $c_2 = [m^{e_2}\ mod\ N_2]$
- If **gcd**$(e_1,e_2)=1$ (which is reasonably likely) then attacker can run extended GCD algorithm to find X,Y such that $Xe_1+Ye_2=1$.
  $[c_1{}^X c_2{}^Y mod\ N_2] = [m^{Xe_1} m^{Ye_2} mod\ N_2] = [m^{Xe_1+Ye_2}\ mod\ N_2] = m$

RSA-OAEP (Optimal Asymmetric Encryption Padding)

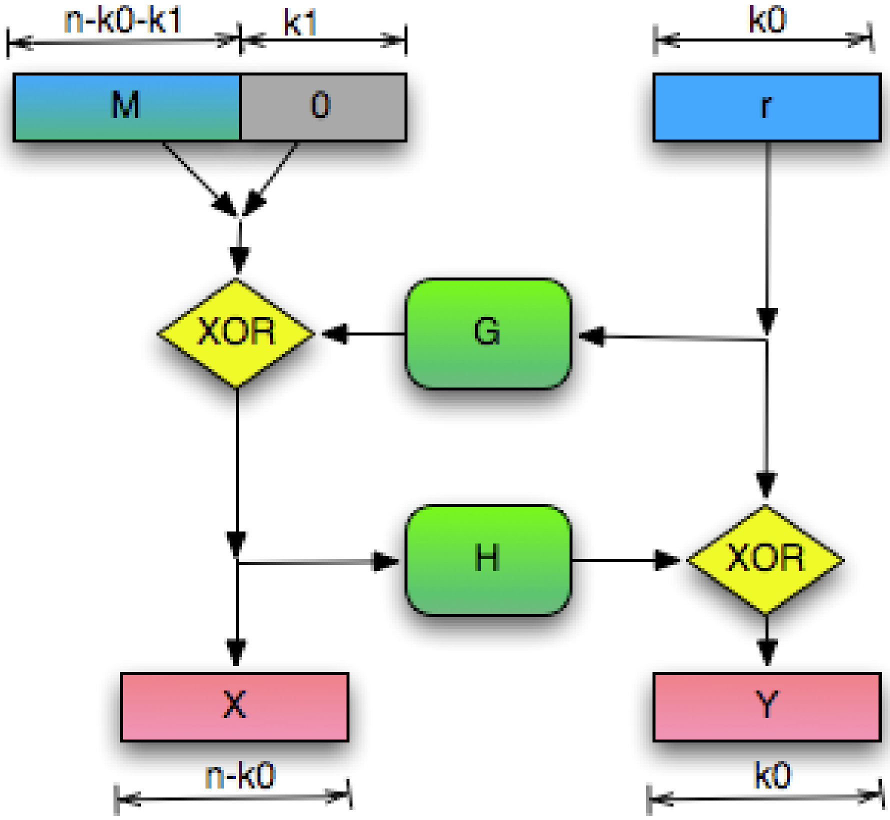
- $\mathbf{Enc}_{\boldsymbol{pk}}(m; r) = [(x \parallel y)^e \ mod\ N]$
- Where $x \parallel y \leftarrow \mathrm{OAEP}(m \parallel 0^{k_1} \parallel r)$
- $\mathbf{Dec}_{\boldsymbol{sk}}(c) =$
- $\tilde{m} \leftarrow [(c)^d \, mod\ N]$
- If $\|\tilde{m}\| > n$ return fail
- $m \parallel z \parallel r \leftarrow \mathrm{OAEP}^{-1}(\tilde{m})$
- If $z \neq 0^{k_1}$ then output fail
- Otherwise output m



RSA-OAEP
(Optimal Asymmetric Encryption Padding)

Theorem: If we model G and H as Random oracles then RSA-OAEP is a CCA-Secure public key encryption scheme.

Bonus: One of the fastest in practice!



PKCS #1 v2.0

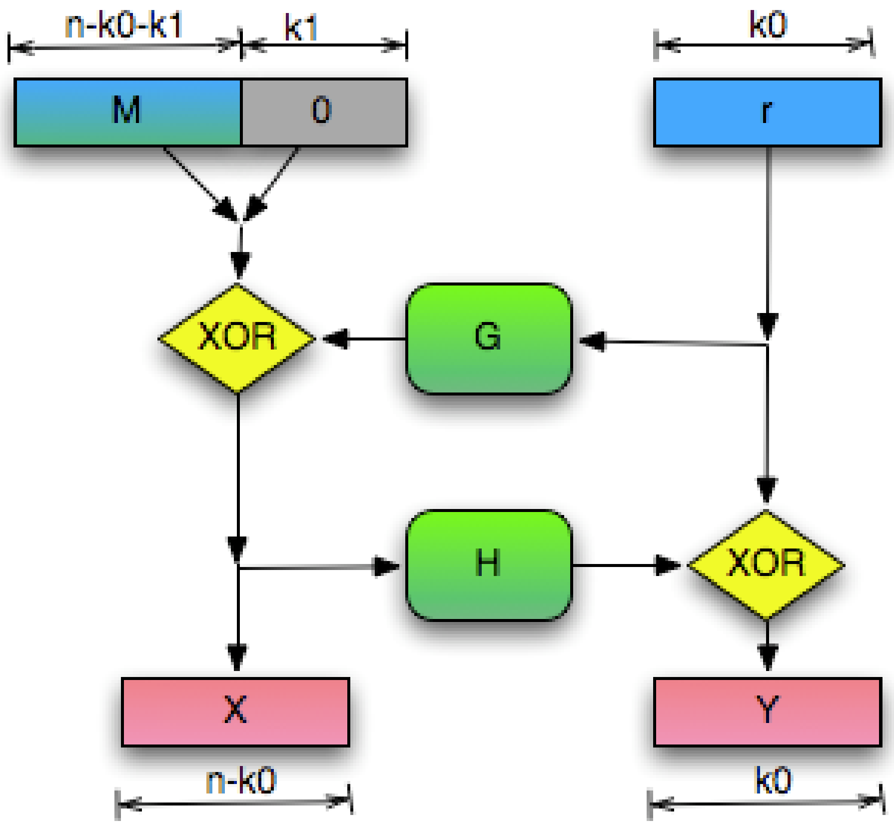
- Implementation of RSA-OAEP

- James Manger found a chosen-ciphertext attack.

- What gives?

PKCS #1 v2.0 (Bad Implementation)

- $\mathbf{Enc}_{\boldsymbol{pk}}\,(m;r) = [(x \parallel y)^e \;\; mod\; N]$
- Where $x \parallel y \;\leftarrow \mathrm{OAEP}(m \parallel 0^{k_1} \parallel r)$
- $\mathbf{Dec}_{\boldsymbol{sk}}\,(c) =$
- $\tilde{m} \leftarrow [(c)^d\, mod\; N]$
- **If $\|\tilde{\boldsymbol{m}}\| > \boldsymbol{n}$ return Error Message 1**
- $m \parallel z \parallel r \leftarrow \mathrm{OAEP}^{-1}(\tilde{m})$
- **If $\boldsymbol{z} \neq \mathbf{0}^{\boldsymbol{k}_1}$ then output Error Message 2**
- Otherwise output



PKCS #1 v2.0 (Attack)

- Manger's CCA-Attack recovers secret key

- Requires $\|\boldsymbol{N}\|$ $queries\ to\ decryption\ oracle$.

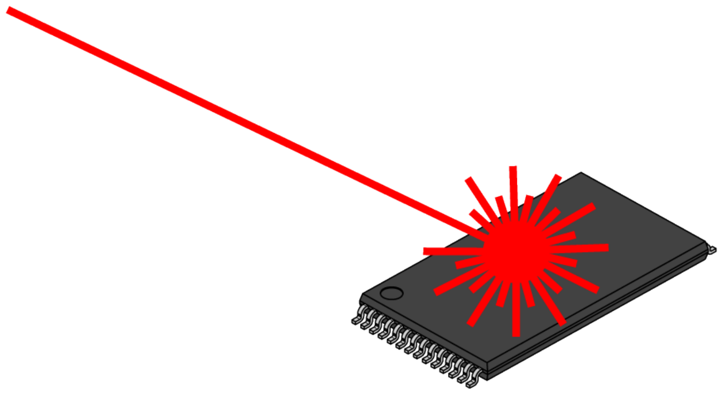
- Attack also works as a side channel attack
  - Even if error messages are the same the time to respond could be different in each case.
- Implementations should return same error message and should make sure that the time to return each error is the same.

Another Side Channel Attack on RSA

- Suppose that decryption is done via Chinese Remainder Theorem for speed.

$$\mathbf{Dec}_{\boldsymbol{sk}}(\boldsymbol{c}) = \boldsymbol{c^d \, mod \, N} \leftrightarrow \left(\boldsymbol{c^d \, mod \, p}, \boldsymbol{c^d \, mod \, q}\right)$$

- Attacker has physical access to smartcard
  - Can mess up computation of $\boldsymbol{c^d \, mod \, p}$
  - Response is $\mathrm{r} \leftrightarrow \left(\boldsymbol{r}, \boldsymbol{c^d \, mod \, q}\right)$
  - $\mathrm{r} - \mathrm{m} \leftrightarrow (\boldsymbol{r - m \, mod \, p}, \boldsymbol{0 \, mod \, q})$
  - GCD(R-m,N)=q



Next Class: Digital Signatures Part 1

- Read Katz and Lindell: 12.1-12.3