Cryptography
CS 555

Week 10:

- RSA
- Attacks on Plain RSA
- Discrete Log/DDH

Readings: Katz and Lindell Chapter 8.2-8.3,11.5.1

Spring 2021

Recap

- Polynomial time algorithms (in bit lengths $\|\boldsymbol{a}\|$, $\|\boldsymbol{b}\|$ and $\|\mathbf{N}\|$) to do important computations on integers
  - GCD(**a**,**b**)
  - Find multiplicative inverse $\mathbf{a}^{-1}$ of **a** such that $1=[\mathbf{a}\mathbf{a}^{-1} \bmod \mathbf{N}]$ (if it exists)
  - PowerMod: $[\mathbf{a}^{\mathbf{b}} \bmod \mathbf{N}]$
  - Draw uniform sample from $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$
- **Fact:** $[g^x \mathrm{mod}\ \mathrm{N}] = \left[g^{[x\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \mathrm{mod}\ \mathrm{N}\right]$ where $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$
  - Proof: Group Theory
- Chinese Remainder Theorem

CS 555: Week 10: Topic 1
Finding Prime Numbers, RSA

RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p - 1)(q - 1)$

Step 3: …

Question: How do we accomplish step one?

Bertrand's Postulate

Theorem 8.32. For any n > 1 the fraction of n-bit integers that are prime is at least ${}^{1}/_{3n}$.

GenerateRandomPrime(1^n)

For i=1 to $3n^2$:

  p' ← $\{0,1\}^{n-1}$

  p ← $1||p'$

  if isPrime(p) **then**

    return p

return fail

Can we do this in polynomial time?

Bertrand's Postulate

Theorem 8.32. For any n > 1 the fraction of n-bit integers that are prime is at least ${}^{1}/_{3n}$.

GenerateRandomPrime(1^n)

For i=1 to $3n^2$:

  p' ← $\{0,1\}^{n-1}$

  p ← $1||p'$

  if isPrime(p) **then**

    return p

return fail

Assume for now that we can run isPrime(p). What are the odds that the algorithm fails?

On each iteration the probability that p is not a prime is $\left(1-\frac{1}{3n}\right)$

We fail if we pick a non-prime in all $3n^2$ iterations. The probability of failure is at most

$$\left(1-\frac{1}{3n}\right)^{3n^2} = \left(\left(1-\frac{1}{3n}\right)^{3n}\right)^{n} \le e^{-n}$$

isPrime(p): Miller-Rabin Test

- We can check for primality of p in polynomial time in $||p||$.

Theory: Deterministic algorithm to test for primality.

- See breakthrough paper "Primes is in P"
- https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf

Practice: Miller-Rabin Test (randomized algorithm)

- **Guarantee 1:** If p is prime then the test outputs YES
- **Guarantee 2:** If p is not prime then the test outputs NO (except with negligible probability).

The "Almost" Miller-Rabin Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

for i=1 to t:

  a ← {1,…,N-1}

  if $a^{N-1} \neq 1 \text{ mod N}$ then return "composite"

Return "prime"

Claim: If N is prime then algorithm always outputs "prime"

Proof: For any a ∈ {1,…,N−1} we have $a^{N-1} = a^{\phi(N)} = 1 \; mod \; N$

$\phi(N) = N - 1$ for primes N

The “Almost” Miller-Rabin Test

Input: Integer N and parameter 1^t

Output: “prime” or “composite”

for i=1 to t:

  a ← {1,…,N-1} //random

  if $a^{N-1} \neq 1 \bmod \mathrm{N}$ then return “composit

Return “prime”

Need a bit of extra work to handle Carmichael numbers (see textbook).

Fact: If N is composite and not a Carmichael number then the algorithm outputs “composite” with probability

$$1 - 2^{-t}$$

Miller-Rabin Primality Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

If Even(N) or **PerfectPower(N**) return "composite"

Else find u (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 = 2^r u$

for j=1 to t:

  pick a in [2,N-2] randomly

  if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r - 1$

    return "composite"

Return "prime"

Miller-Rabin Primality Test

Lemma: If p is prime and $x^2 = 1 \bmod \mathrm{p}$ then
$$x = \pm 1 \bmod \mathrm{p}$$

Input: Integer N and parameter 1^t

Output: “prime” or “composite”

If Even(N) or **PerfectPower(N**) return “composite”

Else find u (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 = 2^r u$

for j=1 to t:

  pick a in [2,N-2] randomly

  if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq\ i \leq\ r - 1$

    return “composite”

Return “prime”

Miller-Rabin Primality Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

If Even(N) or **PerfectPower(N**) return "composite"

Else find u (odd) and $r \geq 1$ s.t. $\mathrm{N}-1 = 2^r u$

for j=1 to t:

pick a in [2,N-2] randomly

if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r-1$

return "composite"

Return "prime"

If N is prime then:

$$\left(a^{2^{r-1}u}\right)^2 = a^{N-1} \bmod \mathrm{N}$$
$$= 1 \quad \bmod \mathrm{N}$$

$$a^{2^i u} - 1 = \left(a^{2^{i-1}u} - 1\right)\left(a^{2^{i-1}u} + 1\right)$$

Miller-Rabin Primality Test

Input: Integer N and pa

Output: “prime” or “co

If Even(N) or **PerfectPo**

Else find u (odd) and r

for j=1 to t:

pick a in [2,N-2] randomly

if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r-1$

return “composite”

Return “prime”

If N is prime we won’t return composite

$$\mathbf{0} = \left(a^{2^r u} - \mathbf{1}\right) = \left(a^{2^{r-1}u} - \mathbf{1}\right)\left(a^{2^{r-1}u} + \mathbf{1}\right)$$

$$= \cdots = \left(a^{2^{r-2}u} - \mathbf{1}\right)\left(a^{2^{r-2}u} + \mathbf{1}\right)\left(a^{2^{r-1}u} + \mathbf{1}\right)$$

$$a^{2^i u} - \mathbf{1} = \left(a^{2^{i-1}u} - \mathbf{1}\right)\left(a^{2^{i-1}u} + \mathbf{1}\right)$$

Miller-Rabin Primality Test

Input: Integer N and para

Output: “prime” or “com

If Even(N) or **PerfectPow**

Else find u (odd) and $r \geq$

for j=1 to t:

pick a in [2,N-2] randomly

if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r-1$

return “composite”

Return “prime”

If N is prime we won’t return composite

$$0 = \left(a^{2^r u}\right) - 1 = \cdots = (a^u - 1)\prod_{i=0}^{r-1}\left(a^{2^i u} + 1\right)$$

$$a^{2^i u} - 1 = \left(a^{2^{i-1} u} - 1\right)\left(a^{2^{i-1} u} + 1\right)$$

Miller-Rabin P

Input: Integer N and para

Output: “prime” or “comp

If Even(N) or **PerfectPow**

Else find u (odd) and $r \geq$

for j=1 to t:

pick a in [2,N-2] randomly

if $a^u \neq \pm 1$ mod N and $a^{2^i u} \neq -1$ mod N for all $1 \leq i \leq r-1$

return “composite”

Return “prime”

One of these factors must be 0 (mod N)

If N is prime we won’t return composite

$$0 = \left(a^{2^r u}\right) - 1 = \cdots = \left(a^u - 1\right)\prod_{i=0}^{r-1}\left(a^{2^i u} + 1\right)$$

$$a^{2^i u} - 1 = \left(a^{2^{i-1}u} - 1\right)\left(a^{2^{i-1}u} + 1\right)$$

Miller-Rabin Primality Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

If Even(N) or **PerfectPower(N)** return "composite"

Else find u (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 = 2^r u$

for j=1 to t:

    if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r - 1$

        return "composite"

Return "prime"

Claim: If N is composite then *at most* ¼ choices of random value a in [2,n-1] will pass the test

Miller-Rabin Primality Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

If Even(N) or **PerfectPower(N)** return "composite"

Else find u (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 = 2^r u$

for j=1 to t:

  if $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r - 1$

    return "composite"

Return "prime"

Claim: If N is composite then we return prime with probability at most 4^{-t}

Proof: (See textbook ☺)

Back to RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p-1)(q-1)$

Step 3: Pick e > 1 such that gcd(e, $\phi(N)$)=1

Step 4: Set d=[e^{-1} mod $\phi(N)$]   (secret key)

Return: N, e, d

- How do we find d?
- **Answer:** Use extended gcd algorithm to find e^{-1}mod $\phi(N)$.

Back to RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p-1)(q-1)$

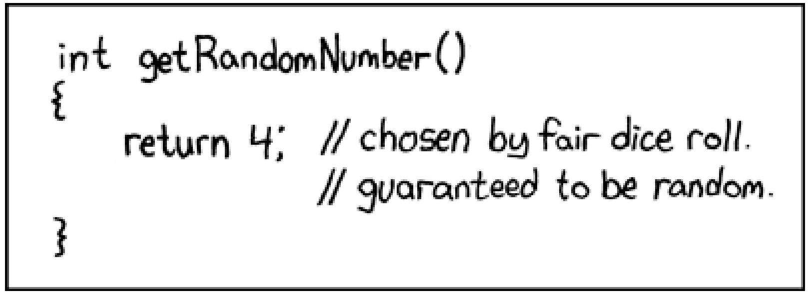
Step 3: Pick e > 1 such that gcd(e, $\phi(N)$)=1

Step 4: Set d=[e^{-1} mod $\phi(N)$]   (secret key)

Return: N, e, d

- What is the probability that e^{-1}mod $\phi(N)$ exists when we pick e randomly?
- **Hint:** $\phi\big(\phi(N)\big)$ choices of e in $\mathbb{Z}_{\phi(N)}$ have a multiplicative inverse mod $\phi(N)$.

Be Careful Where You Get Your "Random Bits!"



- RSA Keys Generated with weak PRG
  - Implementation Flaw
  - Unfortunately Commonplace
- Resulting Keys are Vulnerable
  - Sophisticated Attack
  - Coppersmith's Method


ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS

COMPLETELY BROKEN —

Millions of high-security crypto keys crippled by newly discovered flaw

Factorization weakness lets attackers impersonate key holders and decrypt their data.

DAN GOODIN - 10/16/2017, 7:00 AM

Enlarge / 750,000 Estonian cards that look like this use a 2048-bit RSA key that can be factored in a matter of days.

The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli (CCS 2017)

(Plain) RSA Encryption

- Public Key: PK=(N,e)
- Message $\mathrm{m} \in \mathbb{Z}_{\mathrm{N}}$

$$\textbf{Enc}_{\textbf{PK}}\textbf{(}\mathrm{m}\textbf{)} = [m^e \bmod \mathrm{N}]$$

- **Remark:** Encryption is efficient if we use the power mod algorithm.

(Plain) RSA Decryption

- Secret Key: SK=(N,d)
- Ciphertext $c \in \mathbb{Z}_N$

$$\mathbf{Dec_{SK}(c)} = [c^d \bmod \mathrm{N}]$$

- **Remark 1:** Decryption is efficient if we use the power mod algorithm.
- **Remark 2:** Suppose that $\mathrm{m} \in \mathbb{Z}_N^*$ and let c=**Enc**$_{\mathbf{PK}}$(m) $= [m^e \bmod \mathrm{N}]$

$$\begin{aligned}\mathbf{Dec_{SK}(c)} &= [(m^e)^d \bmod \mathrm{N}] = [m^{ed} \bmod \mathrm{N}] \\ &= \left[m^{[ed \bmod \boldsymbol{\phi}(\boldsymbol{N})]} \bmod \mathrm{N}\right] \\ &= [m^1 \bmod \mathrm{N}] = m\end{aligned}$$

Chinese Remainder Theorem

Theorem: Let N = pq (where gcd(p,q)=1) be given and let $f: \mathbb{Z}_{\mathrm{N}} \to \mathbb{Z}_p \times \mathbb{Z}_q$ be defined as follows

$$f(x) = ([x \bmod p], [x \bmod q])$$

then

- f is a bijective mapping (invertible)
- f and its inverse $f^{-1}: \mathbb{Z}_p \times \mathbb{Z}_q \to \mathbb{Z}_{\mathrm{N}}$ can be computed efficiently
- $f(x+y) = f(x) + f(y)$
- The restriction of f to $\mathbb{Z}_N^*$ yields a bijective mapping to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$
- For inputs $x, y \in \mathbb{Z}_N^*$ we have $f(x)f(y) = f(xy)$

RSA Decryption

- Secret Key: SK=(N,d)
- Ciphertext $c \in \mathbb{Z}_N$

$$\textbf{Dec}_{\textbf{SK}}(c) = [c^d \bmod \mathrm{N}]$$

- **Remark 1:** Decryption is efficient if we use the power mod algorithm.
- **Remark 2:** Suppose that $m \in \mathbb{Z}_N^*$ and let c=**Enc$_{\textbf{PK}}$**(m) $= [m^e \bmod \mathrm{N}]$ **then**

$$\textbf{Dec}_{\textbf{SK}}(c) = m$$

- **Remark 3:** Even if $m \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$ and let $c =$ **Enc$_{\textbf{PK}}$**(m) $= [m^e \bmod \mathrm{N}]$ **then**

$$\textbf{Dec}_{\textbf{SK}}(c) = m$$

  - Use Chinese Remainder Theorem to show this

$$ed = 1 + k(p-1)(q-1)$$

$$\rightarrow \mathrm{f}(c^d) = \left([m^{ed} \bmod \mathrm{p}], [m^{ed} \bmod \mathrm{q}]\right) = ([m^1 \bmod \mathrm{p}], [m^1 \bmod \mathrm{q}])$$

$$\rightarrow f^{-1}\left(\mathrm{f}(c^d)\right) = f^{-1}([m^1 \bmod \mathrm{p}], [m^1 \bmod \mathrm{q}]) = m$$

Plain RSA (Summary)

- Public Key (pk): N = pq, e such that $\text{GCD}\big(\text{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$
- **Encrypt(pk=(N,e),m)** $= m^{\boldsymbol{e}}\ mod\ N$
- **Decrypt(sk=(N,d),c)** $= c^{\boldsymbol{d}}\ mod\ N$

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

Factoring Assumption

Let **GenModulus**(1^n) be a randomized algorithm that outputs (N=pq,p,q) where p and q are n-bit primes (except with negligible probability **negl**(n)).

Experiment $\text{FACTOR}_{A,n}$

1. (N=pq,p,q) ← **GenModulus**(1^n)
2. Attacker A is given N as input
3. Attacker A outputs p' > 1 and q' > 1
4. Attacker A wins if N=p'q'.

Factoring Assumption

- Necessary for security of RSA.
- Not known to be sufficient.

Experiment $\text{FACTOR}_{A,n}$

1. $(N=pq,p,q) \leftarrow$ **GenModulus**(1^n)
2. Attacker A is given N as input
3. Attacker A outputs p' > 1 and q' > 1
4. Attacker A wins ($\text{FACTOR}_{A,n} = 1$) if and only if N=p'q'.

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

RSA-Inversion Assumption

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

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

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

RSA-Assumption

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

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

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

- Plain RSA Encryption behaves like a one-way function
- Attacker cannot invert encryption of random message

Discussion of RSA-Assumption

- Plain RSA Encryption behaves like a one-way-function
- Decryption key is a “trapdoor” which allows us to invert the OWF
- RSA-Assumption ➔ OWFs exist

Recap

- Plain RSA
- Public Key (pk): $\mathrm{N} = \mathrm{pq}$, 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 $\phi(N)$
- **Encrypt(pk=(N,e),m)** $= m^{\boldsymbol{e}}\ mod\ N$
- **Decrypt(sk=(N,d),c)** $= c^{\boldsymbol{d}}\ mod\ N$

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

Mathematica Demo

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

http://develop.wolframcloud.com/app/

Note: Online version of mathematica available at https://sandbox.open.wolframcloud.com (reduced functionality, but can be used to solve homework bonus problems)

(Toy) RSA Implementation in Mathematica

```
(* Random Seed 123456 is not secure, but it allows us to repeat the experiment *)
        SeedRandom[123456]
(* Step 1: Generate primes for an RSA key *)
        p = RandomPrime[{10^1000, 10^1050}];
        q = RandomPrime[{10^1000, 10^1050}];
        NN = p q;   (*Symbol N is protected in mathematica *)
        phi = (p - 1) (q - 1);
```

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

(Toy) RSA Implementation in Mathematica

(* Step 1.A: Find e *)

GCD[phi,7]

Output: 7

(* GCD[phi,7] is not 1, so he have to try a different value of e *)

GCD[phi,3]

Output: 1

(* We can set e=3 *)

e=3;

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

(Toy) RSA Implementation in Mathematica

(* Step 1.B find d s.t. ed = 1 mod N by using the extended GCD algorithm *)

(* Mathematica is clever enough to do this automatically *)

Solve[e x == 1, Modulus->phi]

Output:

{{x->36469680590663028301700626132883867272718728905205088...

……………………………………………………………………………………

394069421778610209425624440980084481398131}}

(* We can now set d = x *)

d=364696805…. 8131;

(Toy) RSA Implementation in Mathematica

(* Double Check 1 = [ed mod $\phi(N)$] *)

Mod [e d, (p-1)(q-1)]

Output: 1

(* Encrypt the message 200, c= m^e mod N *)

m = 200;

PowerMod[m,e,NN]

Output: 8 000 000

(Toy) RSA Implementation in Mathematica

(* Encrypt the message 200, c= m^e mod N *)

m = 200;

PowerMod[m,e,NN]

Output: 8 000 000

(* Hm...That doesn't seem too secure *)

CubeRoot[PowerMod[m,e,NN]]

Output: 200

(* Moral: if $m^e < N$ then Plain RSA does not hide the message m. *)

RSA Implementation in Mathematica

(* Encrypt a larger message, c= m^e mod N *)

SeedRandom[1234567];

m2= RandomInteger[{10^1500,10^1501}];

c=PowerMod[m2,e,NN]

Output: 405215834903772786……… 388068292685976133

(* Does it Decrypt Properly? *)

PowerMod[c,d, NN]-m2

Output: 0

(* Yes! *)

CS 555: Week 10: Topic 2
Attacks on Plain RSA

(Plain) RSA Discussion

- 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

- As we will see Plain RSA is also highly vulnerable to chosen-ciphertext attacks

(Plain) RSA Discussion

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

$\rightarrow$Plain RSA is not secure against chosen-plaintext attacks

- **Remark:** In a public key setting the attacker who knows the public key *always* has access to an encryption oracle

- Encrypted messages with low entropy are particularly vulnerable to brute-force attacks
    - **Example:** If $m < B$ then attacker can recover m from $\mathrm{c} = \mathrm{Enc}_{\mathrm{pk}}(m)$ after at most B queries to encryption oracle (using public key)

Chosen Ciphertext Attack on Plain RSA

1. Attacker intercepts ciphertext $c = [m^e \bmod \mathrm{N}]$
2. Attacker generates ciphertext c' for secret message 2m as follows
3. c' $= [(c2^e) \bmod \mathrm{N}]$
4. $= [(m^e 2^e) \bmod \mathrm{N}]$
5. $= [(2m)^e \bmod \mathrm{N}]$
6. Attacker asks for decryption of $[c2^e \bmod \mathrm{N}]$ and receives 2m.
7. Divide by two to recover message

Above Example: Shows plain RSA is highly vulnerable to ciphertext-tampering attacks

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 $\mathrm{m} = \mathrm{m}_1 \| ??$ then he can recover m given
$$\mathbf{Encrypt}(\mathrm{pk}, \mathrm{m}) = m^e \bmod \mathrm{N}$$

Theorem[Coppersmith]: If p(x) is a polynomial of degree e then in polynomial time (in log(N), 2^e) we can find all m such that p(m) = 0 mod N and $|m|<N^{(1/e)}$

More Weaknesses: Plain RSA with small e

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|<$N^{(1/e)}$

Example: e = 3, $m = m_1 \| m_2$ and attacker knows $m_1 (2k\ bits)$ and $\boldsymbol{c} = (m_1 \| m_2)^e \mathrm{mod\ N}$, but not $m_2 (k\ bits)$

$$p(x) = \left(2^k m_1 + x\right)^3 - c$$

Polynomial has a small root mod N at x= m_2 and coppersmith's method will find it!

D. Coppersmith (1996). "Finding a Small Root of a Univariate Modular Equation".

More Weaknesses: Plain RSA with small e

Theorem[Coppersmith]: Can also find small roots of bivariate polynomial $\mathrm{p}(\boldsymbol{x_1}, \boldsymbol{x_2})$

- **Similar Approach used to factor weak RSA secret keys N=q_1q_2**
- Weak PRG → Can guess many of the bits of prime factors
    - Obtain $\widetilde{q_1} \approx q_1$ and $\widetilde{q_2} \approx q_2$
- Coppersmith Attack: Define polynomial p(.,.) as follows

$$\mathrm{p}(\boldsymbol{x_1}, \boldsymbol{x_2}) = (\boldsymbol{x_1} + \widetilde{\boldsymbol{q_1}})(\boldsymbol{x_2} + \widetilde{\boldsymbol{q_2}}) - \boldsymbol{N}$$

- **Small Roots of** $\mathrm{p}(\boldsymbol{x_1}, \boldsymbol{x_2})$**:** $x_1 = q_1 - \widetilde{q_1}$ and $x_2 = q_2 - \widetilde{q_2}$

D. Coppersmith (1996). "Finding a Small Root of a Bivariate Integer Equation; Factoring with high bits known" ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS

COMPLETELY BROKEN —

Millions of high-security crypto keys crippled by newly discovered flaw

Factorization weakness lets attackers impersonate key holders and decrypt their data.

DAN GOODIN - 10/16/2017, 7:00 AM

Steve Jurvetson

Enlarge / 750,000 Estonian cards that look like this use a 2048-bit RSA key that can be factored in a matter of days.

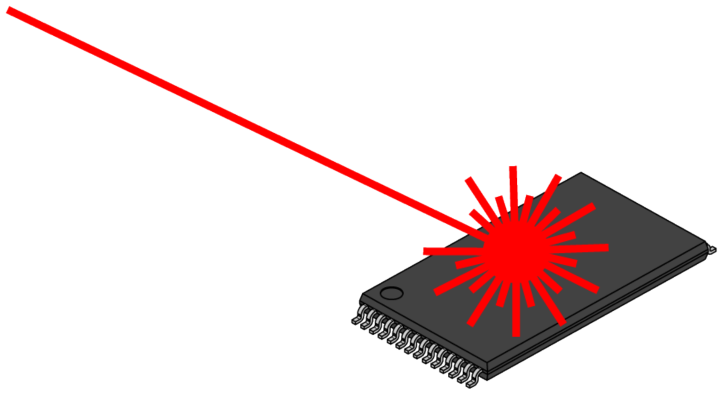
*The Return of **Coppersmith's Attack**: Practical Factorization of Widely Used RSA Moduli (CCS 2017)*

A Side Channel Attack on RSA with CRT

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

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

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



Recovering Encrypted Message faster than Brute-Force

Brute Force Attack: Suppose we know the secret message m < 2^n

We can recover m from ciphertext c=m^e mod N in time 2^n

(Solution: Search from m' < 2^n **s.t. c=m'^e mod N)**

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

Similar to birthday attack

Fixes for Plain RSA

- Approach 1: RSA-OAEP
  - Incorporates random nonce r
  - CCA-Secure (in random oracle model)

- Approach 2: Use RSA to exchange symmetric key for Authenticated Encryption scheme (e.g., AES)
  - Key Encapsulation Mechanism (KEM)
  - Alice has public key (N,e)
  - Bob picks random $r \in \mathbb{Z}_N$ and sends $\mathrm{c} = r^e \; mod \; N$ to Alice
  - Alice and Bob use the symmetric secret key $\mathrm{K} = \mathrm{H}(r)$ for authenticated encryption
  - Intuition:
    - If attacker never queries $\mathrm{H}(r)$ then K can be viewed as truly random secret key (Random Oracle Model)
    - If attacker does query $\mathrm{H}(r)$ with non-negligible probability then we can win RSA-Inversion game using A
- More details in future lectures…stay tuned!

Recap and Announcements

- Plain RSA
  - Primality Tests and Key Generation
  - Encryption/Decryption
  - Factoring/RSA-Inversion
  - Attacks on Plain RSA
  - Fixes: RSA-OAEP, Key-Exchange + Authenticated Encryption (more coming)
- Announcements
  - Quiz 4 released today (Due: Saturday (3/27) at 11:30PM on Brightspace)
  - Homework 4 released (Due: April 8th at 11:59 PM on Gradescope)
    - Q4: Programming Assignment
    - Q2: Programming Assignment or Written Solution (You pick!)

CS 555: Week 10: Topic 3
Discrete Log + DDH Assumption

(Recap) Finite Groups

Definition: A (finite) group is a (finite) set $\mathbb{G}$ with a binary operation $\circ$ (over G) for which we have

- (**Closure**:) For all $\mathrm{g}, \mathrm{h} \in \mathbb{G}$ we have $\mathrm{g} \circ \mathrm{h} \in \mathbb{G}$
- **(Identity**:) There is an element $\mathrm{e} \in \mathbb{G}$ such that for all $\mathrm{g} \in \mathbb{G}$ we have
$$\mathrm{g} \circ \mathrm{e} = \mathrm{g} = \mathrm{e} \circ \mathrm{g}$$
- **(Inverses**:) For each element $\mathrm{g} \in \mathbb{G}$ we can find $\mathrm{h} \in \mathbb{G}$ such that $\mathrm{g} \circ \mathrm{h} = \mathrm{e}$. We say that h is the inverse of g.
- (**Associativity:**) For all $\mathrm{g}_1, \mathrm{g}_2, \mathrm{g}_3 \in \mathbb{G}$ we have
$$(\mathrm{g}_1 \circ \mathrm{g}_2) \circ \mathrm{g}_3 = \mathrm{g}_1 \circ (\mathrm{g}_2 \circ \mathrm{g}_3)$$

We say that the group is **abelian** if

- (**Commutativity:**) For all $\mathrm{g}, \mathrm{h} \in \mathbb{G}$ we have $\mathrm{g} \circ \mathrm{h} = \mathrm{h} \circ \mathrm{g}$

Finite Abelian Groups (Examples)

- **Example 1:** $\mathbb{Z}_N$ when ∘ denotes addition modulo N
- Identity: 0, since 0 ∘ x =[0+x mod N] = [x mod N].
- Inverse of x? Set x^{-1}=N-x so that [x^{-1}+x mod N] = [N-x+x mod N] = 0.

- **Example 2:** $\mathbb{Z}_N^*$ when ∘ denotes multiplication modulo N
- Identity: 1, since 1∘ x =[1(x) mod N] = [x mod N].
- Inverse of x? Run extended GCD to obtain integers a and b such that

$$ax + bN = \gcd(x, N) = 1$$

Observe that: x^{-1} = a. Why?

Cyclic Group

- Let $\mathbb{G}$ be a group with order $\mathrm{m} = |\mathbb{G}|$ with a binary operation $\circ$ (over G) and let g $\in \mathbb{G}$ be given consider the set

$$\langle g \rangle = \{g^0, g^1, g^2, \dots\}$$

Fact: $\langle g \rangle$ defines a subgroup of $\mathbb{G}$.

- Identity: g^0
- Closure: $g^i \circ g^j = g^{i+j} \in \langle g \rangle$
- g is called a "generator" of the subgroup.

Fact: Let $\mathrm{r} = |\langle g \rangle|$ then $g^i = g^j$ if and only if $i = j \ mod \ r$. Also m is divisible by r.

Finite Abelian Groups (Examples)

Fact: Let p be a prime then $\mathbb{Z}_p^*$ is a cyclic group of order p-1.

- **Note**: Number of generators g s.t. of $\langle g \rangle = \mathbb{Z}_p^*$ is $\phi(p-1)$

Example (non-generator): p=7, g=2

<2>={1,2,4}

Example (generator): p=7, g=5

<2>={1,5,4,6,2,3}

Discrete Log Experiment $\mathrm{DLog}_{A,G}(n)$

1. Run $G(1^n)$ to obtain a cyclic group $\mathbb{G}$ of order q (with $\|q\| = n$) and a generator g such that $< \mathrm{g} >= \mathbb{G}$.
2. Select $\mathrm{h} \in \mathbb{G}$ uniformly at random.
3. Attacker A is given $\mathbb{G}$, q, g, h and outputs integer x.
4. Attacker wins ($\mathrm{DLog}_{A,G}(n)=1$) if and only if $g^x=h$.

We say that the discrete log problem is hard relative to generator G if

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}\ \Pr[\mathrm{DLog}_{A,n} = 1] \leq \mu(n)$$

Diffie-Hellman Problems

Computational Diffie-Hellman Problem (CDH)

- Attacker is given $\mathrm{h}_1 = g^{x_1} \in \mathbb{G}$ and $\mathrm{h}_2 = g^{x_2} \in \mathbb{G}$.
- Attackers goal is to find $g^{x_1 x_2} = (\mathrm{h}_1)^{x_2} = (\mathrm{h}_2)^{x_1}$
- **CDH Assumption**: For all PPT A there is a negligible function negl upper bounding the probability that A succeeds with probability at most negl(n).

Decisional Diffie-Hellman Problem (DDH)

- Let $\mathrm{z}_0 = g^{x_1 x_2}$ and let $\mathrm{z}_1 = g^{r}$, where x_1,x_2 and r are random
- Attacker is given g^{x_1}, g^{x_2} and z_b (for a random bit b)
- Attackers goal is to guess b
- **DDH Assumption**: For all PPT A there is a negligible function negl such that A succeeds with probability at most ½ + negl(n).

Secure key-agreement with DDH

1. Alice publishes g^{x_A} and Bob publishes g^{x_B}
2. Alice and Bob can both compute $K_{A,B} = g^{x_B x_A}$ but to Eve this key is indistinguishable from a random group element (by DDH)

Remark: Protocol is vulnerable to Man-In-The-Middle Attacks if Bob cannot validate g^{x_A}.

Can we find a cyclic group where DDH holds?

- **Example 1:** $\mathbb{Z}_p^*$ where p is a random n-bit prime.
  - CDH is believed to be hard
  - DDH is *not* hard (Exercise 13.15)

- **Theorem:** Let p=rq+1 be a random n-bit prime where q is a large λ-bit prime then the set of r^{th} residues modulo p is a cyclic subgroup of order q. Then $\mathbb{G}_r = \{[h^r mod\ p] \mid h \in \mathbb{Z}_p^*\}$ is a cyclic subgroup of $\mathbb{Z}_p^*$ of order q.
  - **Remark 1:** DDH is believed to hold for such a group
  - **Remark 2:** It is easy to generate uniformly random elements of $\mathbb{G}_r$
  - **Remark 3:** Any element (besides 1) is a generator of $\mathbb{G}_r$

Can we find a cyclic group where DDH holds?

- **Theorem:** Let p=rq+1 be a random n-bit prime where q is a large λ-bit prime then the set of rth residues modulo p is a cyclic subgroup of order q. Then $\mathbb{G}_r = \left\{[h^r mod\ p] \middle| h \in \mathbb{Z}_p^*\right\}$ is a cyclic subgroup of $\mathbb{Z}_p^*$ of order q.
  - **Closure**: $h^r g^r = (hg)^r$
  - **Inverse** of h^r is $(h^{-1})^r \in \mathbb{G}_r$
  - **Size** $(h^r)^x = h^{[rx\ mod\ rq]} = (h^r)^x = h^{r[x\ mod\ q]} = (h^r)^{[x\ mod\ q]} mod\ p$

Remark: Two known attacks on Discrete Log Problem for $\mathbb{G}_r$(Section 9.2).

- First runs in time $O\left(\sqrt{q}\right) = O\left(2^{\lambda/2}\right)$
- Second runs in time $2^{O\left(\sqrt[3]{n}(\log n)^{2/3}\right)}$

Can we find a cyclic group where DDH holds?

Remark: Two known attacks (Section 9.2).

- First runs in time $O(\sqrt{q}) = O(2^{\lambda/2})$
- Second runs in time $2^{O(\sqrt[3]{n}(\log n)^{2/3})}$, where n is bit length of p

Goal: Set λ and n to balance attacks

$$\lambda = O\left(\sqrt[3]{n}(\log n)^{2/3}\right)$$

How to sample p=rq+1?

- First sample a random λ-bit prime q and
- Repeatedly check if rq+1 is prime for a random n- λ bit value r

Can we find a cyclic group where DDH holds?

Elliptic Curves Example: Let p be a prime (p > 3) and let A, B be constants. Consider the equation

$$y^2 = x^3 + Ax + B \ mod \ p$$

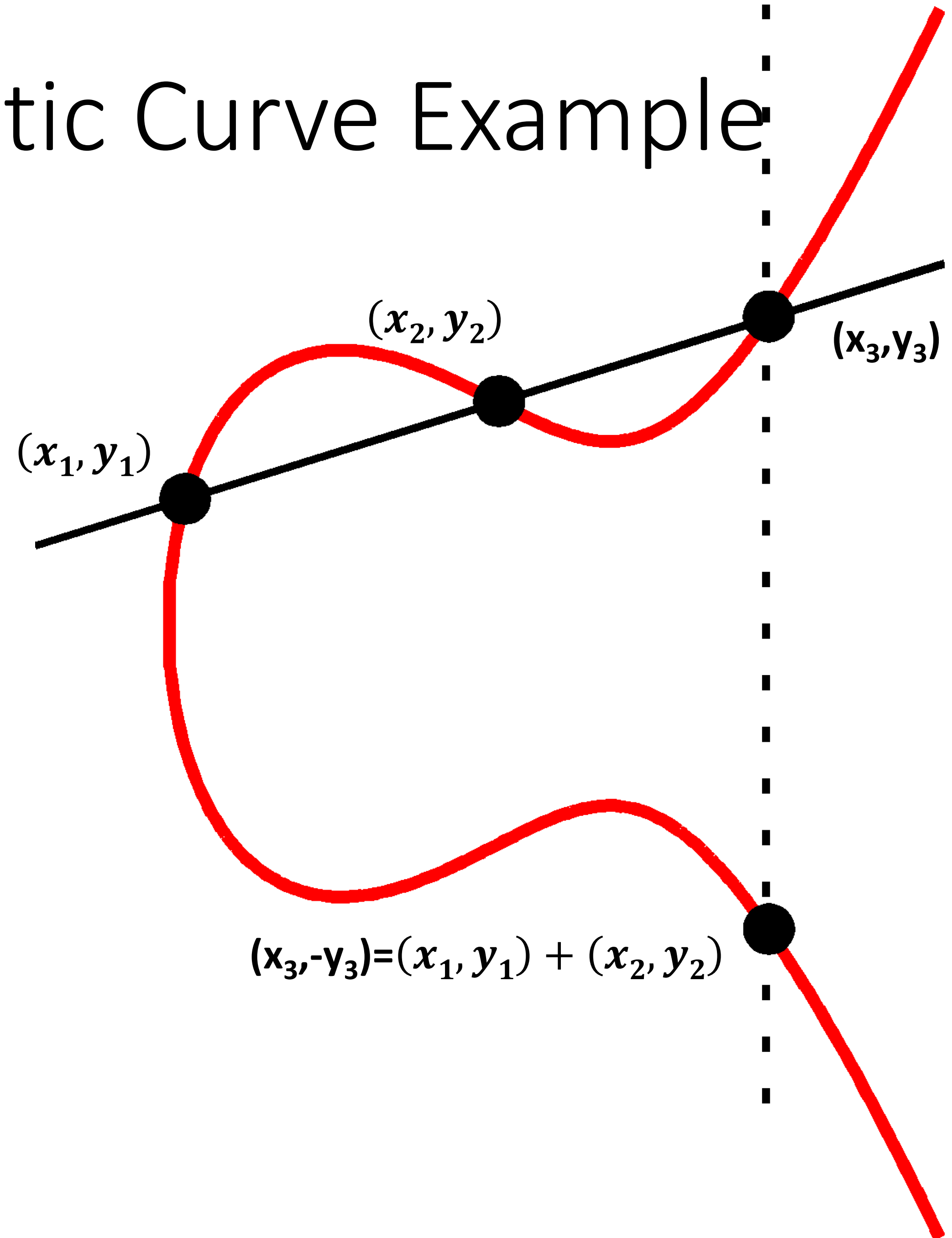
And let

$$E\left(\mathbb{Z}_p\right) = \left\{(x, y) \in \mathbb{Z}_p^2 \middle| y^2 = x^3 + Ax + B \ mod \ p\right\} \cup \{\mathcal{O}\}$$

Note: $\mathcal{O}$ is defined to be an additive identity $(x, y) + \mathcal{O} = (x, y)$

What is $(x_1, y_1) + (x_2, y_2)$?

Elliptic Curve Example



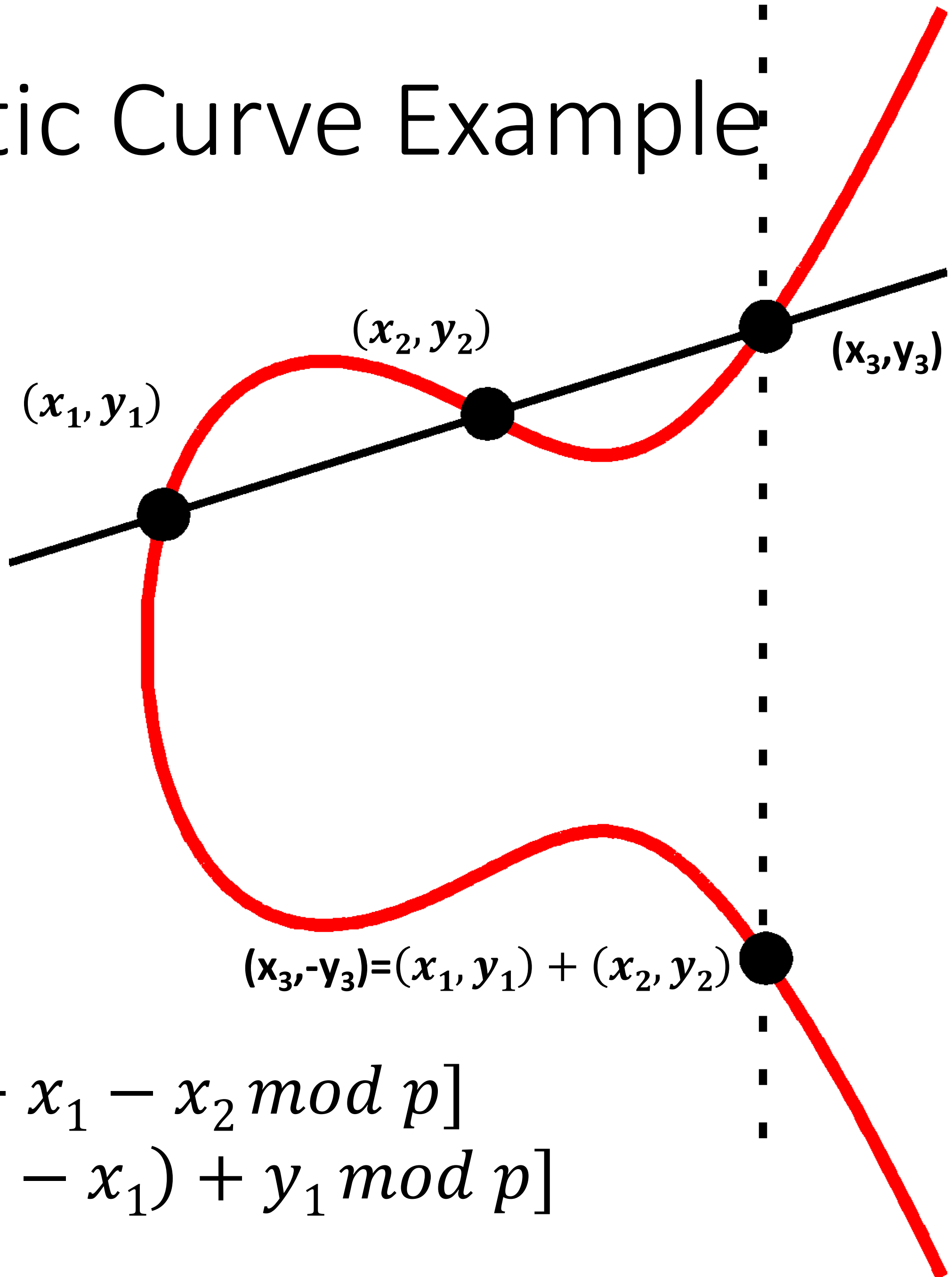
The line passing through $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ has the equation

$$y = m(x - x_1) + y_1 \; mod \; P$$

Where the slope

$$m = \left[\frac{y_1 - y_2}{x_1 - x_2} \; mod \; p\right]$$

Elliptic Curve Example



Formally, let

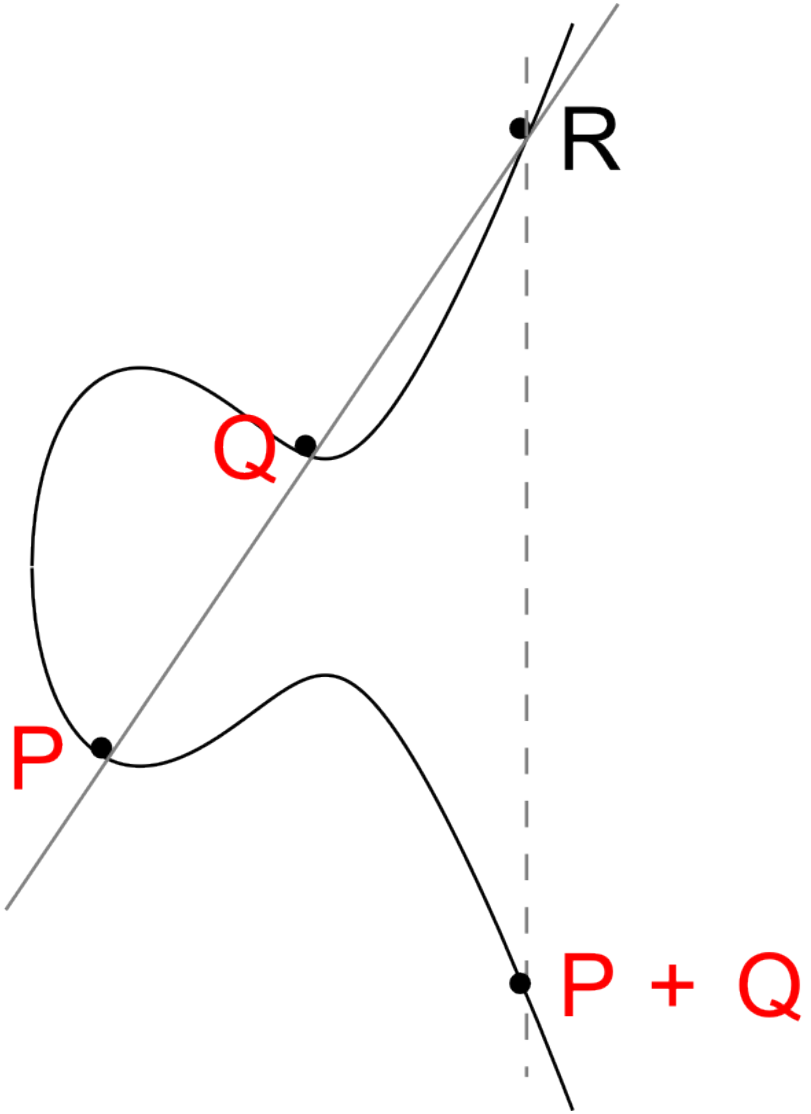
$$m = \left[\frac{y_1 - y_2}{x_1 - x_2} \ mod\ p\right]$$

Be the slope. Then the line passing through $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ has the equation

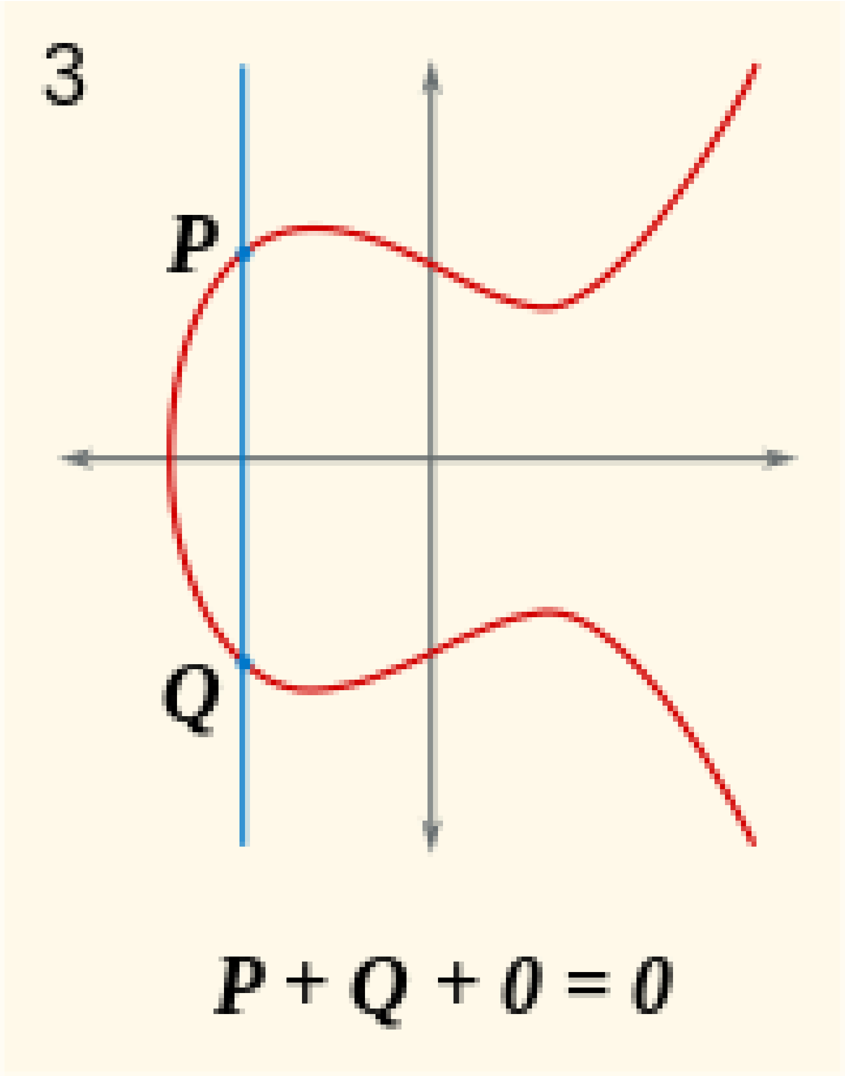
$$y = m(x - x_1) + y_1 \ mod\ P$$

$$(m(x - x_1) + y_1)^2$$
$$= x^3 + Ax + B \ mod\ p$$

$$x_3 = [m^2 - x_1 - x_2 \ mod\ p]$$
$$y_3 = [m(x_3 - x_1) + y_1 \ mod\ p]$$


R
Q
P
P + Q

Elliptic Curve Example



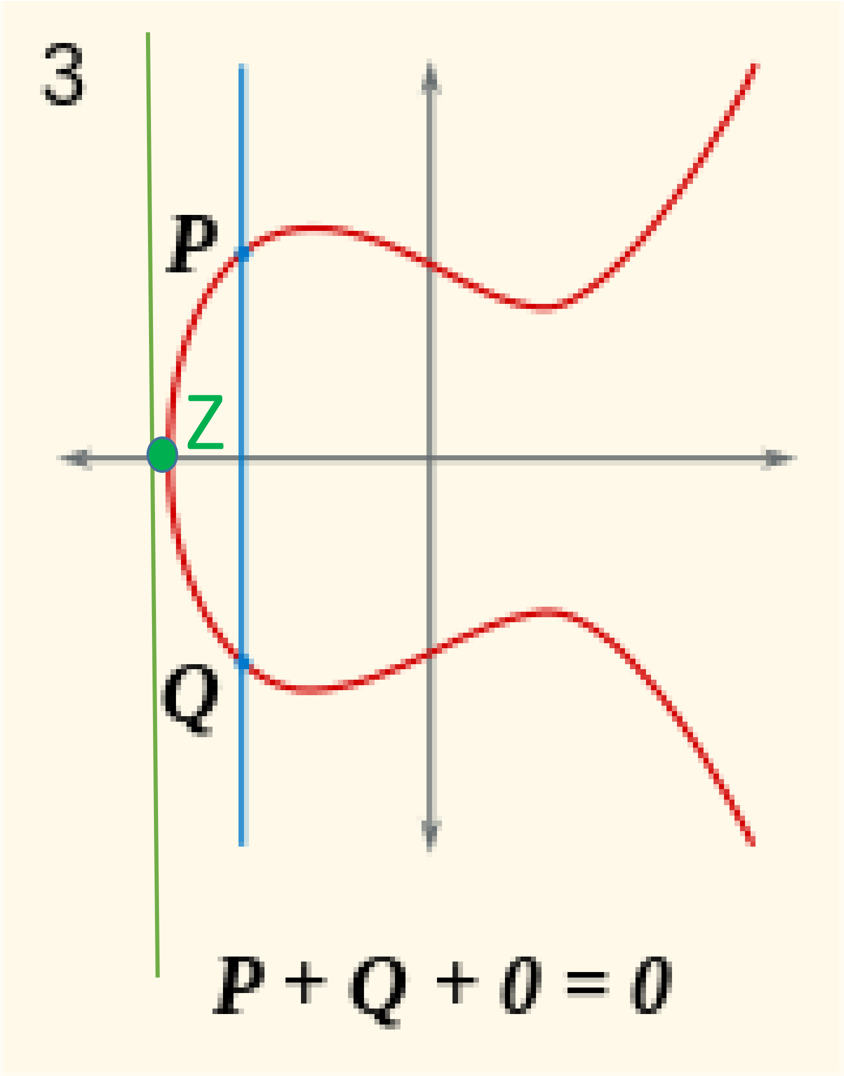
No third point R on the elliptic curve.

P+Q = 0

(Inverse)

Elliptic Curve Special Cases

Z+Z=0

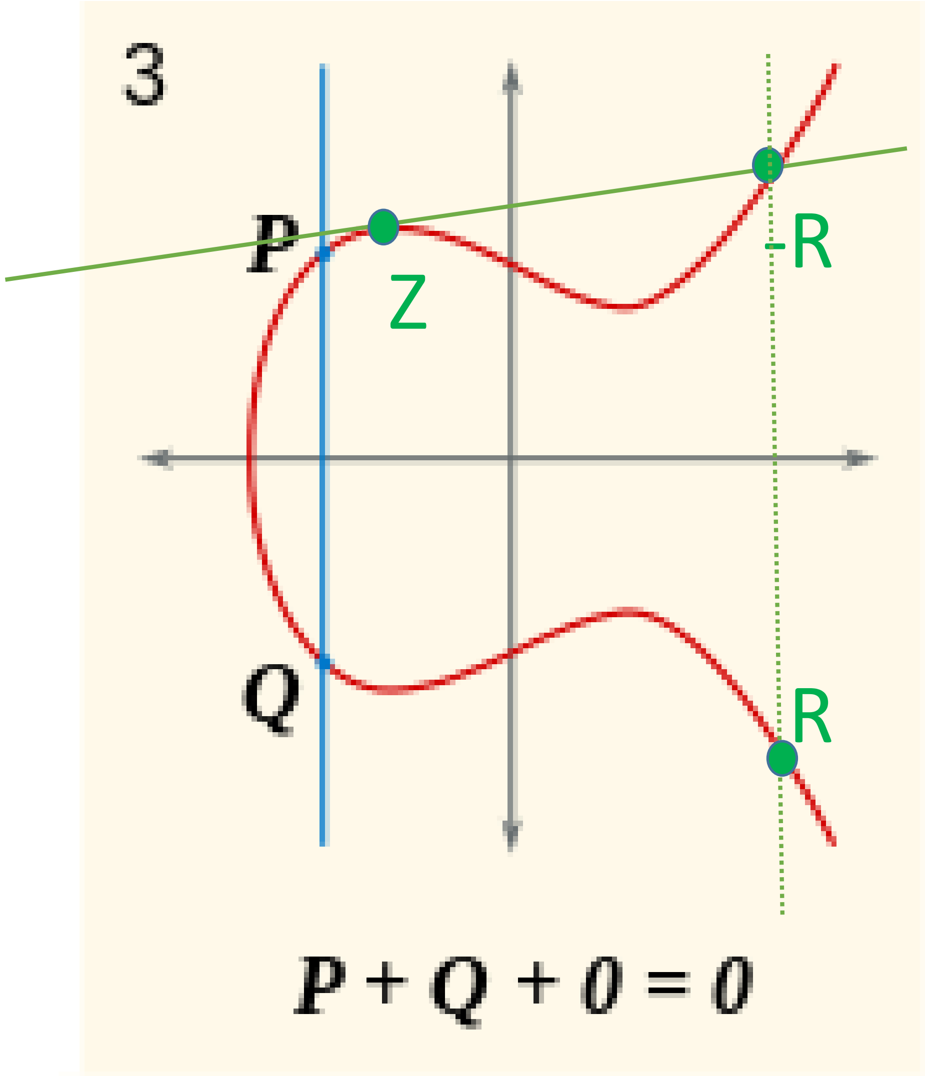


No third point R on the elliptic curve.

P+Q = 0

(Inverse)

Elliptic Curve Special Cases



Z+Z=R

How to find R?

Can we find a cyclic group where DDH holds?

Elliptic Curves Example: Let p be a prime (p > 3) and let A, B be constants. Consider the equation

$$y^2 = x^3 + Ax + B \bmod p$$

And let

$$E\left(\mathbb{Z}_p\right) = \left\{(x, y) \in \mathbb{Z}_p^2 \middle| y^2 = x^3 + Ax + B \bmod p\right\} \cup \{\mathcal{O}\}$$

Fact: $E\left(\mathbb{Z}_p\right)$ defines an abelian group

- For *appropriate curves* the DDH assumption is believed to hold
- If you make up your own curve there is a good chance it is broken…
- NIST has a list of recommendations
- Bad Elliptic Curves:
  - Order is p, p+1, order divides $p^k - 1$ for “small” k,…

Generic Group Model

- Suppose $p < 2^n$ is a prime
  - Fact: Every prime order group is isomorphic to $\mathbb{Z}_p$, +
- Random (injective mapping) $\tau: \mathbb{Z}_p \to \{0,1\}^n$
- Access to Group via Two Oracles
  - $\mathrm{Mult}\big(\tau(x), \tau(y)\big) = \tau(x + y \text{ mod p})$
  - Inverse$\big(\tau(x)\big) = \tau(p - x)$

- **Discrete Log Problem:** Attacker is given $\mathrm{g} = \tau(1)$ and $\mathrm{g} = \tau(x)$ for a random $0 \leq \mathrm{x} < p$.
- **Attacker Goal:** Find x
- **DDH Problem:** Challenger picks random bit b and random values $0 \leq \mathrm{x, y, r} < p$
  - Attacker is given $\mathrm{g} = \tau(1), \mathrm{g} = \tau(x), \mathrm{g} = \tau(y)$, and
    - $\mathrm{g} = \tau(r)$ if b=0
    - $\mathrm{g} = \tau(xy)$ if b=1

Generic Group Model

- Suppose $p < 2^n$ is a prime
  - Fact: Every prime order group is isomorphic to $\mathbb{Z}_p$, +
- Random (injective mapping) $\tau: \mathbb{Z}_p \to \{0,1\}^n$
- Access to Group via Two Oracles
  - $\mathrm{Mult}\big(\tau(x), \tau(y)\big) = \tau(x + y \bmod \mathrm{p})$
  - Inverse$\big(\tau(x)\big) = \tau(p - x)$

- **Discrete Log Problem:** Attacker is given $\mathrm{g} = \tau(1)$ and $\mathrm{g} = \tau(x)$ for a random $0 \leq \mathrm{x} < p$.
- **Attacker Goal:** Find x
- **Fact:** Any attacker A making at most q queries to group oracles finds x with probability at most $\mathrm{O}(q/2^{n/2})$
- **Matching Attack:** Birthday Bound
- **Intuition:** Suppose we know i input/output pairs $(x_1, \tau(x_1))$, … $(x_i, \tau(x_i))$ but $\mathrm{x} \neq x_1, \ldots x_i$
  - Can view x as a yet to be sampled element from $\mathbb{Z}_p \backslash \{x_1, \ldots x_i\}$

Generic Group Model

- Suppose $p < 2^n$ is a prime
  - Fact: Every prime order group is isomorphic to $\mathbb{Z}_p$, +
- Random (injective mapping) $\tau: \mathbb{Z}_p \rightarrow \{0,1\}^n$
- Access to Group via Two Oracles
  - $\mathrm{Mult}\big(\tau(x), \tau(y)\big) = \tau(x + y \mathrm{\ mod\ p})$
  - Inverse$\big(\tau(x)\big) = \tau(p - x)$

- **DDH Problem:** Challenger picks random bit b and random values $0 \leq \mathrm{x}, \mathrm{y}, \mathrm{r} < p$
  - Attacker is given $\mathrm{g} = \tau(1), \mathrm{g} = \tau(x), \mathrm{g} = \tau(y)$, and
    - $\mathrm{g} = \tau(r)$ if b=0
    - $\mathrm{g} = \tau(xy)$ if b=1
- **Fact:** Any attacker A making at most q queries to group oracles guesses b with probability at most $\frac{1}{2} + \mathrm{O}(q/2^{n/2})$

Week 12 Topic 2: Formalizing Public Key Cryptography

Public Key Encryption: Basic Terminology

- Plaintext/Plaintext Space
  - A message $\mathrm{m} \in \mathcal{M}$
- Ciphertext $\mathrm{c} \in \mathcal{C}$
- **Public/Private Key Pair $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$**

Public Key Encryption Syntax

- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$
  - $\mathrm{Enc}_{\mathrm{pk}}(m) \in \mathcal{C}$ (Encryption algorithm)
  - $\mathrm{Dec}_{\mathrm{sk}}(c)$ (Decryption algorithm)
    - Input: Secret key sk and a ciphertex c
    - Output: a plaintext message $\mathrm{m} \in \mathcal{M}$

- **Invariant**: $\mathrm{Dec}_{\mathrm{sk}}(\mathrm{Enc}_{\mathrm{pk}}(\mathrm{m}))=\mathrm{m}$

Alice must run key generation algorithm in advance an publishes the public key: pk

Assumption: Adversary only gets to see pk (not sk)

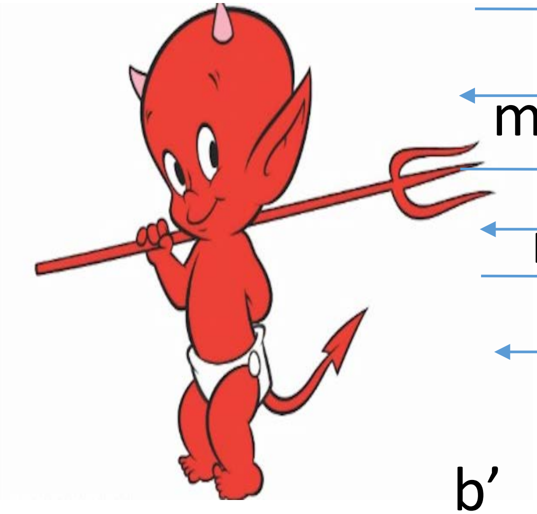
Chosen-Plaintext Attacks

- Model ability of adversary to control or influence what the honest parties encrypt.

- Historical Example: Battle of Midway (WWII).
  - US Navy cryptanalysts were able to break Japanese code by tricking Japanese navy into encrypting a particular message

- Private Key Cryptography

Recap CPA-Security (Symmetric Key Crypto)



$m_{0,1}, m_{1,1}$

$c_1 = Enc_K(m_{b,1})$

$m_{0,2}, m_{1,2}$

$c_2 = Enc_K(m_{b,2})$

$m_{0,3}, m_{1,3}$

$c_3 = Enc_K(m_{b,3})$

⋮

b'

Random bit b

K = Gen(.)

$$\forall PPT\ A\ \exists \mu\ \text{(negligible) s.t}$$

$$\Pr[A\ \ Guesses\ b' = b] \leq \frac{1}{2} + \mu(n)$$

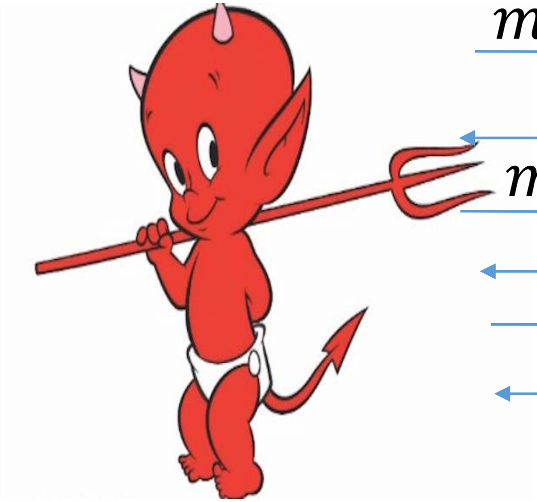
Chosen-Plaintext Attacks

- Model ability of adversary to control or influence what the honest parties encrypt.

- Private Key Crypto
  - Attacker tricks victim into encrypting particular messages

- Public Key Cryptography
  - The attacker already has the public key pk
  - Can encrypt any message s/he wants!
  - CPA Security is critical!

CPA-Security $\left(\mathrm{PubK}_{A,\Pi}^{\mathrm{LR-cpa}}(\mathrm{n})\right)$



Public Key: pk

m_0^1, m_1^1

$\boldsymbol{c_1 = \mathrm{Enc}_{pk}\left(m_b^1\right)}$

m_0^2, m_1^2

$\boldsymbol{c_2 = \mathrm{Enc}_{pk}\left(m_b^2\right)}$

m_0^3, m_1^3

$\boldsymbol{c_3 = \mathrm{Enc}_{pk}\left(m_b^3\right)}$

⋮

b'

Random bit b

(pk,sk) = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr\left[\mathrm{PubK}_{A,\Pi}^{\mathrm{LR-cpa}}(\mathrm{n}) = 1\right] \leq \frac{1}{2} + \mu(n)$$

CPA-Security (Single Message)

$Formally, let\ \Pi = \ (Gen, Enc, Dec)\ denote\ the\ encryption\ scheme,$
$call\ the\ experiment\ PubK_{A,\Pi}^{LR-cpa}(n)\ and\ define\ a\ random\ variable$

$$\mathrm{PubK}_{\mathrm{A},\Pi}^{\mathrm{LR-cpa}}(n) = \begin{cases} 1 & \text{if } b = b' \\ 0 & otherwise \end{cases}$$

$\Pi\ has\ indistinguishable\ encryptions\ under\ a\ chosen\ plaintext\ attack$
$if\ for\ all\ PPT\ adversaries\ A, there\ is\ a$ negligible function μ such that

$$\Pr[\mathrm{PubK}_{\mathrm{A},\Pi}^{\mathrm{LR-cpa}}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

Private Key Crypto

- CPA Security was stronger than eavesdropping security

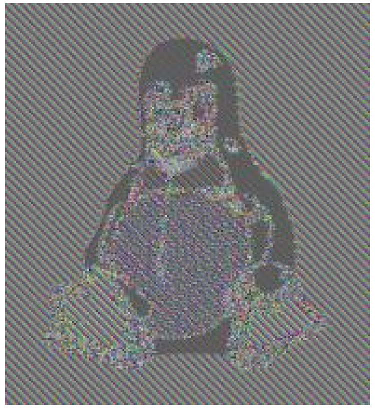
$$\mathrm{Enc_K(m)} = \mathrm{G(K)} \oplus m$$

Vs.

$$\mathrm{Enc_K(m)} = \langle r, F_k(r) \oplus m \rangle$$

Public Key Crypto

- **Fact 1:** CPA Security and Eavesdropping Security are Equivalent
  - Key Insight: The attacker has the public key so he doesn't gain anything from being able to query the encryption oracle!
- **Fact 2:** Any deterministic encryption scheme is not CPA-Secure
  - Historically overlooked in many real world public key crypto systems
- **Fact 3:** Plain RSA is **not** CPA-Secure
- **Fact 4:** No Public Key Cryptosystem can achieve Perfect Secrecy!
  - Exercise 11.1
  - **Hint:** Unbounded attacker can keep encrypting the message m using the public key to recover all possible encryptions of m.



Encrypting Longer Messages

Claim 11.7: Let $\Pi = (Gen, Enc, Dec)$ denote a CPA-Secure public key encryption scheme and let $\Pi' = (Gen, Enc', Dec')$ be defined such that

$$\mathbf{Enc}'_{\mathbf{pk}}(\boldsymbol{m_1} \parallel \boldsymbol{m_2} \parallel \cdots \parallel \boldsymbol{m_\ell}) = \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_1}) \parallel \cdots \parallel \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_\ell})$$

Then Π' is also CPA-Secure.

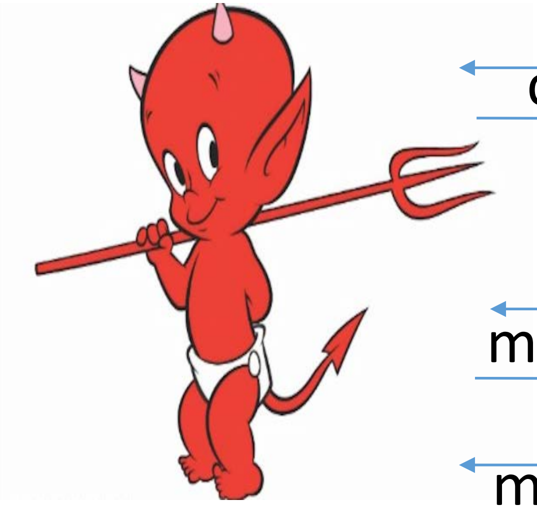
Chosen Ciphertext Attacks

- Models ability of attacker to obtain (partial) decryption of selected ciphertexts

- Attacker might intercept ciphertext c (sent from S to R) and send c' instead.
  - After that attacker can observe receiver's behavior (abort, reply etc…)
- Attacker might send a modified ciphertext c' to receiver R in his own name.
  - E-mail response: Receiver might decrypt c' to obtain m' and include m' in the response to the attacker

Recap CCA-Security (Symmetri…)

We could set $m_0 = m_{-1}$ or $m_1 = m_{-2}$



m_{-1}

$c_{-1} = Enc_K(\ldots$

c_{-2}

$\ldots_{-2} = Dec_K(c_{-2})$

⋮

m_0, m_1

$c = Enc_K(m_b)$

However, we could still flip 1 bit
of c and ask challenger to decrypt

m_3

$\ldots Enc_K(m_2)$

c_3

$m_3 = Dec_K(m_3)$

$c_4 = c$

⋮

"No Way!"

b'

Random bit b

K = Gen(.)

Recap CCA-Security $\left(PrivK_{A,\Pi}^{cca}(n)\right)$

1. Challenger generates a secret key k and a bit b
2. Adversary (A) is given oracle access to Enc_k and Dec_k
3. Adversary outputs m_0, m_1
4. Challenger sends the adversary $c = Enc_k(m_b)$.
5. Adversary maintains oracle access to Enc_k and Dec_k ,however the adversary is not allowed to query $Dec_k(c)$.
6. Eventually, Adversary outputs b'.

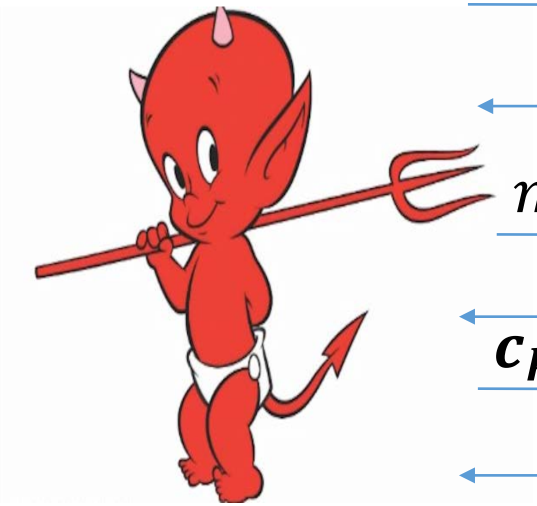
$$PrivK_{A,\Pi}^{cca}(n) = 1 \text{ if } \mathrm{b} = \mathrm{b}'; \text{otherwise } 0.$$

CCA-Security: For all PPT A exists a negligible function negl(n) s.t.

$$\Pr\left[PrivK_{A,\Pi}^{cca}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

CCA-Security $\left(\mathrm{PubK}_{\mathrm{A},\Pi}^{\mathrm{cca}}(\mathrm{n})\right)$

Public Key: pk



$\boldsymbol{c_{-1}}$

$\boldsymbol{m_{-1} = \mathrm{Dec}_{sk}(c_{-1})}$

$\vdots$

m_0 , m_1

$\boldsymbol{c_b = \mathrm{Enc}_{pk}(m_b\ )}$

$\boldsymbol{c_k}$

$\boldsymbol{m_k = \mathrm{Dec}_{sk}(c_k)}$

b’

$\vdots$

Random bit b

(pk,sk) = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\mathrm{negligible})\ \mathrm{s.t}$$

$$\Pr\left[\mathrm{PubK}_{\mathrm{A},\Pi}^{\mathrm{cca}}(\mathrm{n}) = 1\right] \le \frac{1}{2} + \mu(n)$$

Encrypting Longer Messages

Claim 11.7: Let $\Pi = (Gen, Enc, Dec)$ denote a CPA-Secure public key encryption scheme and let $\Pi' = (Gen, Enc', Dec')$ be defined such that

$$\mathbf{Enc}'_{\mathbf{pk}}(\boldsymbol{m_1} \parallel \boldsymbol{m_2} \parallel \cdots \parallel \boldsymbol{m_\ell}) = \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_1}) \parallel \cdots \parallel \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_\ell})$$

Then Π' is also CPA-Secure.

Claim? Let $\Pi = (Gen, Enc, Dec)$ denote a CCA-Secure public key encryption scheme and let $\Pi' = (Gen, Enc', Dec')$ be defined such that

$$\mathbf{Enc}'_{\mathbf{pk}}(\boldsymbol{m_1} \parallel \boldsymbol{m_2} \parallel \cdots \parallel \boldsymbol{m_\ell}) = \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_1}) \parallel \cdots \parallel \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_\ell})$$

Then Π' is also CCA-Secure.

Is this second claim true?

Encrypting Longer Messages

Claim? Let $\Pi = (Gen, Enc, Dec)$ denote a CCA-Secure public key encryption scheme and let $\Pi' = (Gen, Enc', Dec')$ be defined such that

$$\mathbf{Enc}'_{\mathbf{pk}}(\boldsymbol{m_1} \parallel \boldsymbol{m_2} \parallel \cdots \parallel \boldsymbol{m_\ell}) = \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_1}) \parallel \cdots \parallel \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_\ell})$$

Then Π' is also CCA-Secure.

Is this second claim true?

Answer: No!

Encrypting Longer Messages

Fact: Let $\Pi = (Gen, Enc, Dec)$ denote a CCA-Secure public key encryption scheme and let $\Pi' = (Gen, Enc', Dec')$ be defined such that

$$\mathbf{Enc}'_{\mathbf{pk}}(\boldsymbol{m_1} \parallel \boldsymbol{m_2} \parallel \cdots \parallel \boldsymbol{m_\ell}) = \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_1}) \parallel \cdots \parallel \mathbf{Enc}_{\mathbf{pk}}(\boldsymbol{m_\ell})$$

Then Π' is **Provably Not** CCA-Secure.

1. Attacker sets $\boldsymbol{m_0} = \mathbf{0}^n \parallel \mathbf{1}^n \parallel \mathbf{1}^n$ and $\boldsymbol{m_1} = \mathbf{0}^n \parallel \mathbf{0}^n \parallel \mathbf{1}^n$ and gets $\boldsymbol{c_b} = \mathbf{Enc}'_{\mathbf{pk}}(\boldsymbol{m_b}) = \boldsymbol{c_{b,1}} \parallel \boldsymbol{c_{b,2}} \parallel \boldsymbol{c_{b,3}}$
2. Attacker sets $\boldsymbol{c'} = \boldsymbol{c_{b,2}} \parallel \boldsymbol{c_{b,3}} \parallel \boldsymbol{c_{b,1}}$, queries the decryption oracle and gets

$$\mathbf{Dec}'_{\mathbf{sk}}(\boldsymbol{c'}) = \begin{cases} \mathbf{1}^n \parallel \mathbf{1}^n \parallel \mathbf{0}^n & \textbf{if b=0} \\ \mathbf{0}^n \parallel \mathbf{1}^n \parallel \mathbf{0}^n & \boldsymbol{otherwise} \end{cases}$$

Achieving CPA and CCA-Security

- Plain RSA is not CPA Secure (therefore, not CCA-Secure)

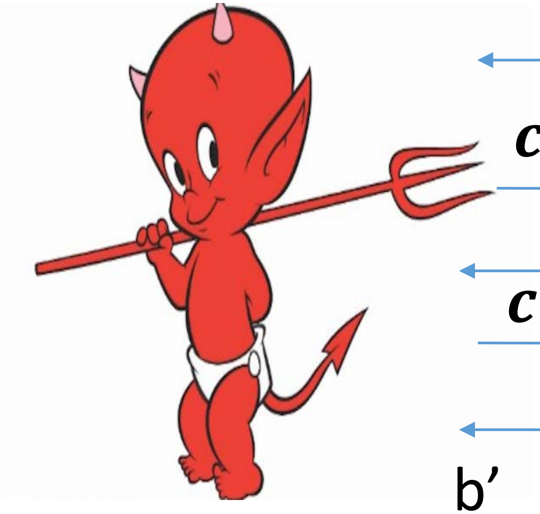
- El-Gamal (future) is CPA-Secure, but not CCA-Secure

- Tools to obtain CCA-Security in Public Key Setting
    - RSA-OAEP, Cramer-Shoup
    - Key Encapsulation Mechanism

Key Encapsulation Mechanism (KEM)

- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$
  - $\mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$
    - Input: security parameter, random bits R
    - Output: Symmetric key $\mathrm{k} \in \{0,1\}^{\ell(n)}$ and a ciphertext c
  - $\mathrm{Decaps}_{\mathrm{sk}}(c)$ (Deterministic algorithm)
    - Input: Secret key $\mathrm{sk} \in \mathcal{K}$ and a ciphertex c
    - Output: a symmetric key$\{0,1\}^{\ell(n)}$ or ⊥ (fail)
- **Invariant**: $\mathrm{Decaps}_{\mathrm{sk}}$(c)=k whenever (c,k) = $\mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$

KEM CCA-Security ($\mathrm{KEM}^{\mathrm{cca}}_{A,\Pi}(n)$)



$(\boldsymbol{pk}, \boldsymbol{c}, \boldsymbol{k_b})$

⋮

$\boldsymbol{c_1} \neq \boldsymbol{c}$

$\mathbf{Decaps}_{\boldsymbol{sk}}(\boldsymbol{c_1})$

$\boldsymbol{c_2} \neq \boldsymbol{c}$

$\mathbf{Decaps}_{\boldsymbol{sk}}(\boldsymbol{c_2})$

b’

⋮

Random bit b

(pk,sk) = Gen(.)

$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$

$$\Pr\left[\mathrm{KEM}^{\mathrm{cca}}_{A,\Pi} = 1\right] \leq \frac{1}{2} + \mu(n)$$

$(\boldsymbol{c}, \boldsymbol{k_0}) = \mathbf{Encaps}_{\boldsymbol{pk}}(.)$

$\boldsymbol{k_1} \leftarrow \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}}$

CCA-Secure Encryption from CCA-Secure KEM

$$\mathbf{Enc_{pk}}(\boldsymbol{m}; \boldsymbol{R}) = \langle \boldsymbol{c}, \mathbf{Enc_k^*}(\boldsymbol{m}) \rangle$$

Where

- $(\boldsymbol{c}, \boldsymbol{k}) \leftarrow \mathbf{Encaps_{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$,
- $\mathbf{Enc_k^*}$ is a CCA-Secure symmetric key encryption algorithm, and
- $\mathbf{Encaps_{pk}}$ is a CCA-Secure KEM.

Theorem 11.14: $\mathbf{Enc_{pk}}$ is CCA-Secure public key encryption scheme.

CCA-Secure KEM in the Random Oracle Model

- Let (N,e,d) be an RSA key (pk =(N,e), sk=(N,d)).

$$\mathrm{Encaps}_{\mathrm{pk}}(1^n, R) = \left(r^e \bmod N, k = H(r)\right)$$

- Remark 1: k is completely random string unless the adversary can query random oracle H on input r.
- Remark 2: If Plain-RSA is hard to invert for a random input then PPT attacker finds r with negligible probability.

Using a CCA-Secure KEM

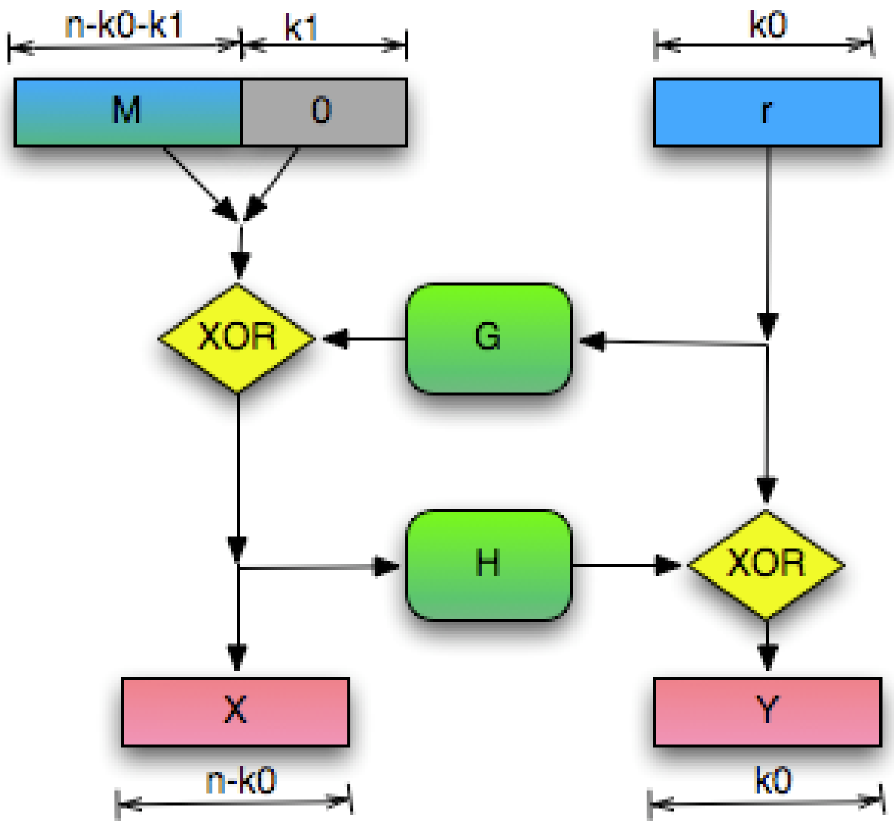
- Let (N,e,d) be an RSA key (pk =(N,e), sk=(N,d)).

$$\mathrm{Enc}_{\mathrm{pk}}(m;R) = (c, \mathrm{AEnc}_{\mathrm{k}}(m))\ \mathit{where}$$
$$c = \mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$$

- Remark 1: k is completely random string unless the adversary can query random oracle H on input r.
- Remark 2: If Plain-RSA is hard to invert for a random input then PPT attacker finds r with negligible probability.

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



Recap RSA-Assumption

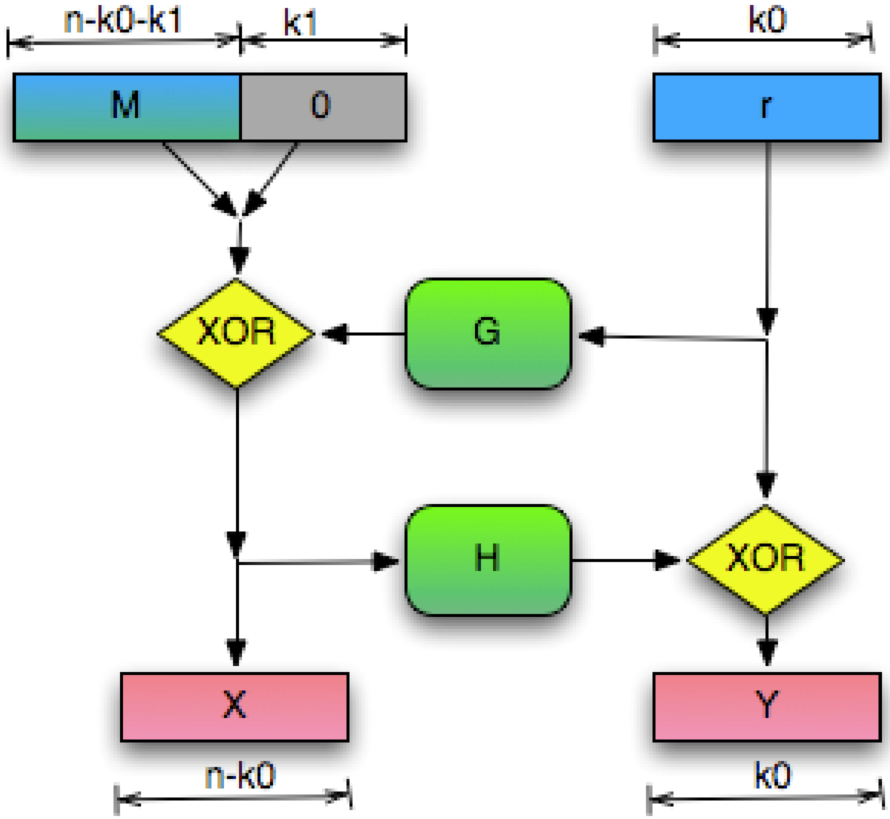
RSA-Experiment: RSA-INV$_{A,n}$

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 $x^e = \mathrm{y} \bmod \mathrm{N}$

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

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 (given RSA-Inversion assumption).



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

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

PKCS #1 v2.0 (Attack)

- Manger's CCA-Attack recovers secret message
  - **Step 1:** Use decryption oracle to check if $2\widetilde{m} \geq \ 2^n$
  - $c = [(\widetilde{m})^e \, mod \; N]$ →$2^e c = [(2\widetilde{m})^e \, mod \; N]$

- 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.
- **Fix:** Implementation should return same error message and should make sure that the time to return each error is the same.