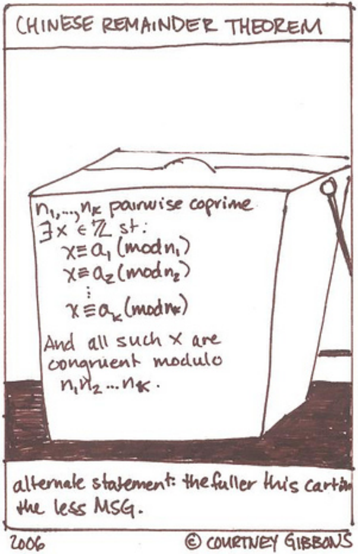


CS 553
CRYPTOGRAPHY

Lecture 26
CRT + DH

Instructor
Dr. Dhiman Saha

Image Source: Google

- Smaller Public Exponent
- In practice $e = 65537$ (fourth Fermat number)
- Speeds up encryption/signature verification
- Larger $e \implies$ slower computation of $x^e \bmod n$

Can e be even smaller

- Low exponent attack

What about Private Exponent

- d is secret
- Implication: Must be unpredictable
- Cannot be restricted to a small value
- Generally, size of the order of modulus
- E.g. close to 2048 for 2048-bit RSA

- Decryption much slower than encryption
- Signing much slower than verification

- Speed up decryption and signing

The Chinese remainder theorem allows for faster decryption by computing two exponentiations, modulo p and modulo q, rather than simply modulo n.

- Because p and q are much smaller than n, its faster to perform two "small" exponentiations than a single "big" one.

- A general result

If $n = n_1 n_2 n_3 \cdots$,

- where the n_i's are **pairwise co-prime**
- (that is, $GCD(n_i, n_j) = 1$ for any distinct i and j)

then the value $x \bmod n$ can be computed from the values

- $x \bmod n_1$,
- $x \bmod n_2$,
- $x \bmod n_3, \cdots$

Applying the CRT to RSA

- Only two factors for each n (p and q)
- Given a ciphertext y to decrypt
- Instead of computing $y^d \bmod n$, use the CRT to compute

- $x_p = y^s \bmod p$, where $s = d \bmod (p-1)$
- $x_q = y^t \bmod q$, where $t = d \bmod (q-1)$

- Combine these two expressions

$$x = x_p \times q \times (1/q \bmod p) + x_q \times p \times (1/p \bmod q) \bmod n$$

- This is **faster** than square-and-multiply. **How**?

More Speed-up

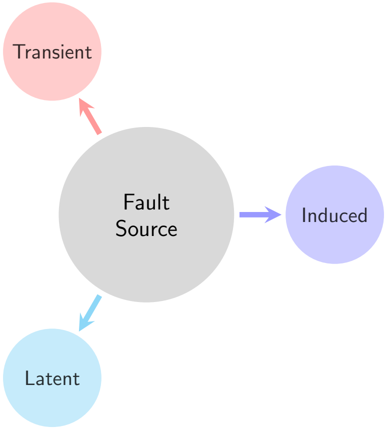
$$x = x_p \times q \times (1/q \bmod p) + x_q \times p \times (1/p \bmod q) \bmod n$$

- **Pre-compute** $q \times (1/q \bmod p)$ and $p \times (1/p \bmod q)$
- Final Overhead of combining:
  - Two multiplications and
  - An addition modulo n

A Side-Channel Attack On CRT

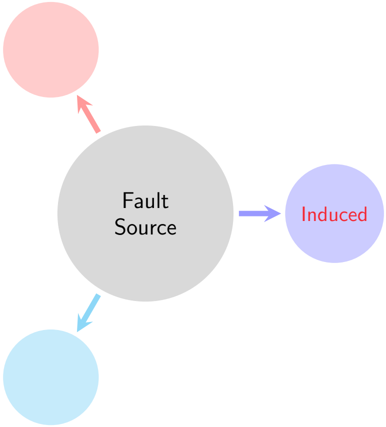
Wh@t i$ a Fau!t?

- An error in execution
- (Un)Intentional



Wh@t i$ a Fau!t?

- An error in execution
- Intentional



Why Should I Care?

- If you are cryptographer you must worry!
- Faults can be fatal.

Basic Idea

Malicious modifications of a cryptographic device might leak cryptanalytically useful information leading to possibly a complete break.

- So, first inject faults in a cryptosystem
- Then exploit information leaked by faulty output

- Referred to as intrusive Side Channel Analysis
- Also branded as Physical Attacks
- Early reference 'The Belcore Attack' - 1996

First reported attack on RSA-CRT - 1997

- Due to Boneh-DeMillo-Lipton
- Initial attack requires both faulty and fault-free signatures

- Improvement suggested by Lenstra
- Requires the faulty signature only

Lets have a look!

Bellcore Attack

Correct Signature

$x = x_p \times q \times (1/q \bmod p) + x_q \times p \times (1/p \bmod q) \bmod n$

- Attacker induces a fault in x_q computation
- Modifies it to some x_q'

Faulty Signature

$x' = x_p \times q \times (1/q \bmod p) + x_q' \times p \times (1/p \bmod q) \bmod n$

The attacker can then subtract the **incorrect** signature x' from the **correct** signature x to factor n!!!

$$x - x' = (x_q - x'_q) \times p \times (1/p \bmod q) \bmod n$$

- $(x - x')$ is therefore a multiple of p
- So $p | (x - x')$
- Recall $p | n$

$$gcd(x - x', n) = p$$

- Then compute $q = n/p$ and d
- Total break of RSA Signatures

The Diffie-Hellman Function

DHF

- Works with groups Z_p* where $p \in \mathbb{P}$
- Another public parameter is the base number, g.
- All arithmetic operations are performed modulo p.
- Two **private** values $a, b \in Z_p*$ chosen randomly by the two communicating parties
- Public value $A = g^a \bmod p$
- Public value $B = g^b \bmod p$

Shared secret: g^{ab}

- $A^b = (g^a)^b = g^{ab}$
- $B^a = (g^b)^a = g^{ab}$

- Shared secret input of **Key Derivation Function (KDF)**

DLP

The DLP consists of finding the y for which $g^y = x$,

- given a base number g within Z_p*
- where p is prime and
- given a group element x

- The DLP is called discrete because we are dealing with integers as opposed to real numbers (continuous)
- Its called a logarithm because were looking for the logarithm of x in base g.

How does the hardness compare with factoring?

CDH The Computational Diffie-Hellman Problem

Computing the shared secret g^{ab} given only the public values g^a and g^b, and not any of the secret values a or b.

- If you can solve DLP, then you can also solve CDH
- DLP is at least as hard as CDH
- Is the converse true?
- We **dont know for sure** whether CDH is at least as hard as DLP

- Note the similarity of the above relation of DLP and CDH with factoring and RSA
- Note DH offers same security level as RSA for a given modulus size

- What if the attacker learns some bits of g^{ab}?
- Still cannot break CDH
- But learns something about g^{ab}

DDH — **The Decisional Diffie-Hellman Problem**

Given g^a, g^b, and a value that is either g^{ab} or g^c for some random c (each of the two with a chance of $1/2$),

- The DDH problem consists of determining whether g^{ab} (the shared secret corresponding to g^a and g^b) was chosen.