CS3235 Part3 Lec1

$_{/}|/|U_Ch@NgRu!$

December 16, 2021

1 Side Channel Attacks:timing attack

Threat model

Learn system's secret by observing how long taken for computation

Attacker's goal

Key recovery

Adversary in side-channel attack

1. Remote
2. Inside its own virtual machine
3. Keys could be in tamper-proof storage or smartcard

1.1 RSA cryptosystem

1.1.1 Classroom RSA setting

- Key generation
    1. Select Two Prime Number p, q
    2. $n = p * q$
    3. $\varphi(n) = (p-1)(q-1)$
    4. find a number e such that $gcd(e, \varphi(n))$
    5. Find a number b (b is larger than e usually and e b should not be close), such that $e * d \equiv 1 \mod (\varphi(n))$

Mu Changrui

  - Public key $K_e = (e, n)$
  - Private key $K_d = d$
- Encryption:
  - $c = Enc(K_e, m) = m^e \mod n$
- Decryption
  - $Dec(K_e, c) = c^d \mod n$

In practice, $d >> e$, therefore the decryption takes very long time usually as the follows

1. Method 1:$c^d \mod n = \underbrace{c * c * ... * c}_{d} \mod n$, here $\underbrace{c * c * ... * c}_{d}$ is a very large number if p,q are 2048 bits, n is 4096 bits, $\underbrace{c * c * ... * c}_{d}$ may be around 4096^{2048}, which is abviously infeasible
2. Method 2: $c^d \mod n = \underbrace{c * c * ... * c}_{d} \mod n = (...((c * \underbrace{c \mod n) * c \mod n) * ...) * c \mod n}_{d-1}$ This even though saves space, but it's still slow, mod is one of the slowest operation in computer
3. Method 3: Here assume d=2^k, then $c^d \mod n = \underbrace{(...((c * c)^2 \mod n)^2 \mod n...)^2}_{k} \mod n$, this reduce the time complexity from O(n) to O(log n)
4. Method 4(time and multiply): Any positive integer d can be expressed as $d = 2^{k1} + 2^{k2} + ...$, this waive the limit that $d = 2^k$ in Method 3
   e.g. $d = 38_{10} = 100110_2 = 100000_2 + 100_2 + 10_2$

   hence $c^{38} = c^{2^5+2^2+2^1} = c^{(((2^3)+1)*2+1)*2} = ((((c^2)^2)^2 * c)^2 * c)^2$

   The above strategy is further simplified as square & multiply algorithm:

```
Input: c, d, n
SqureMultiply:
    d has w bits
    For k=0 upto w-1:
        if (bit k of d) if 1:
            Then Let Rk=(Sk*c) mod n #
        else:
            Let Rk=Sk
        Let Sk+1=Rk^2 mod n
    End For
    Return Rw-1
```

MrChangrui

The running time for line6 is very different from the running time on line 8, which gives adversary chance to get the value bit by bit by counting the running time

1.1.2 Chinese Remainder Theorem

To compute $m = c^d \mod n$, where n=pq, we instead **pre-compute**

- $d_1 \mod (p-1)$
- $d_2 \mod (q-1)$
- $qinv = q^{-1} \mod p$

Then when get c we compute:

- $m_1 = c^{d1} \mod p$
- $m_2 = c^{d2} \mod q$
- $m = m_1 + (qinv * (m_1 - m_2) \mod p) * q$

Proof of Chinese Remainder Theorem

RTBD

Optimization on CRT

e.g. $m_2 = c^{d2} \mod q$
compute with a sliding window(process d_2 in k bit blocks)

1.1.3 Montgomery reduction

To compute $z = x * y \mod n$, we instead introduce a value $R = 2^k, k \in Q$)(Here R and p,q must be co-prime), then transform x,y into Montgomery form:

$x \to xR \mod n$

$y \to yR \mod n$

Then we compute z in Montgomery form:

$xR * yR * R^{-1} \mod n = xyR \mod n$

Rationality of Montgomery reduction

RTBD

Mu Changrui

1.1.4 Schindler's observation

- If $zR > q$, then need to subtract q(called extra reduction)

Rationality of Schindler's observation

RTBD

1.2 Reduction Timing Dependency

1.2.1 Karatsuba and normal multiplication

RTBD

1.3 Summary of Time dependency

1.4 Open SSL workflow

1.4.1 Attack HandShake

1.5 Defence

2 Side-channel attack: Traffic Analysis

RTBD

Mu Changrui