Euclidean Algorithm & Chinese Reminder Theorem

CS 5158/6058 Data Security and Privacy

Spring 2018

Instructor: Boyang Wang

Textbook RSA

- KeyGen: given a security parameter 1^n, generate two n-bit primes p, q, compute $N = pq$, choose e s.t. $\gcd(e, \phi(N)) = 1$, compute $d = e^{-1} \mod \phi(N)$, output public key $pk = (N, e)$, private key $sk = d$
- Enc: given a message m and a public key $pk = (N, e)$, return $c = m^e \mod N$
- Dec: given a ciphertext c and a private key $sk = d$, return $m = c^d \mod N$

Textbook RSA

- Example: p*q = 11*7 = 77 = N
  - $\phi(N)$ = $|Z_N^*|$ = (p-1)(q-1) = 10*6 = 60
  - choose e, s.t. gcd(e, $\phi(N)$) = 1
  - e = 7,
  - d = 43, ed = 7*43 = 1 mod 60
  - pk = (N, e) = (77, 7), sk = d = 43

  - Given m = 2 and pk, c = m^e = 2^7 = 51 mod 77
  - Given c = 51 and sk, m = c^d = 51^{43} = 2 mod 77

Textbook RSA

- RSA Problem: given N and e, compute d
- (Textbook) RSA is secure if RSA problem is hard

$$m^e = c \mod N, \quad c^d = m \mod N$$

  - easy to compute (encrypt) given N and e
  - hard to invert (decrypt) given N and e
  - but easy to invert (decrypt) given d

  - In practice, N is 1024 or 2048 bits

Textbook RSA

- Factoring is the best (known) way to solve RSA problem
  - There might be better algo to compute RSA problem
    - But we do not know……

- If factoring is easy, then RSA is not secure.
  - If factoring is easy, given N, obtain p and q
  - then obtain $\phi(N)$ = (p-1)(q-1)
  - then obtain d = e^{-1} mod $\phi(N)$
  - d is the private key, can decrypt any ciphertext

Textbook RSA

$$m^e = c \mod N, \quad c^d = m \mod N$$

- Textbook RSA is deterministic, not CPA-secure

- (Padded) RSA in practice (E.g., PKCS #1)
    - Pad a message with a random
    - plaintext m is (N-k) bits
    - m' = r||m, where r is chosen uniformly from 2^k
    - m' has N bits
    - Padding is invertible s.t. decryption is correct

Euclidean Algorithm

Given e and $\phi(N)$, how to compute $d = e^{-1} \mod \phi(N)$

- Need to learn Euclidean Algorithm first
  - Given a and b, Euclidean(a, b) outputs gcd(a, b)
  - Input: a and b

    1) if a < b, switch a and b

    2) Divide a by b, get reminder r

    if r = 0, then b is gcd(a, b)

    else do a = b, b = r, run Euclidean(a, b)

- Euclidean(a, b)

  1) if a < b, switch a and b

  2) Divide a by b, get reminder r

     if r = 0, then b is gcd(a, b)

     else do a = b, b = r, run Euclidean(a, b)

- <u>Example</u>: a = 210, b = 45
  - Divide 210 by 45, 210 = 4*45 + 30, reminder is 30
- a = b = 45, b = r = 30
  - Divide 45 by 30, 45 = 1*30 + 15, reminder is 15
- a = b = 30, b = r = 15
  - Divide 30 by 15, 30 = 2*15 + 0, reminder is 0
  - 15 is gcd(210, 45)

- Euclidean(a, b)

  1) if a < b, switch a and b

  2) Divide a by b, get reminder r

     if r = 0, then b is gcd(a, b)

     else do a = b, b = r, run Euclidean(a, b)

- <u>Practice</u>: a = 215, b = 55
    - Divide 215 by 55, 215 = 3*55 + 50, reminder is 50
  - a = b = 55, b = r = 50
    - Divide 55 by 50, 55 = 1*50 + 5, reminder is 5
  - a = b = 50, b = r = 5
    - Divide 50 by 5, 50 = 10*5 + 0, reminder is 0
    - 5 is gcd(215, 55)

Extended Euclidean Algorithm

- Given a and b, there are m and n, s.t.
  - gcd(a, b) = m*a + n*b
- Extended Euclidean Algorithm outputs m and n

- Example of Euclidean Algorithm:
  - a = 210, b = 45
    - Divide 210 by 45, 210 = 4*45 + 30, reminder is 30
  - a = b = 45, b = r = 30
    - Divide 45 by 30, 45 = 1*30 + 15, reminder is 15
  - a = b = 30, b = r = 15,   15 is gcd(210, 45)

- <u>Example of Euclidean Algorithm:</u>
  - a = 210, b = 45
    - Divide 210 by 45, 210 = 4*45 + 30, reminder is 30
  - a = b = 45, b = r = 30
    - Divide 45 by 30, 45 = 1*30 + 15, reminder is 15
  - a = b = 30, b = r = 15
    - 15 is gcd(210, 45)

- EEA outputs m and n, s.t., gcd(a, b) = m*a + n*b
  - Based on round1: r = 30 = 210 - 4*45
  - Based on round2: r = 15 = 45 - 1*30
  - 15 = 45-1*(210-4*45) = (-1)*210 + 5*45 = m*a + n*b
  - m = -1 and n = 5

- <u>Example of Euclidean Algorithm:</u>
    - a = 215, b = 55
        - Divide 215 by 55, 215 = 3*55 + 50, reminder is 50
    - a = b = 55, b = r = 50
        - Divide 55 by 50, 55 = 1*50 + 5, reminder is 5
    - a = b = 50, b = r = 5
        - 5 is gcd(215, 55)

- <u>Practice:</u> compute m and n s.t. gcd(a, b) = m*a + n*b
    - Based on round1: r = 50 = 215 - 3*55
    - Based on round2: r = 5 = 55 - 1*50
    - 5 = 55 - 1*(215-3*55) = (-1)*215 + 4*55 = m*a + n*b
    - m = -1 and n = 4

Extended Euclidean Algorithm

- Given a and b, there are m and n, s.t.
  - gcd(a, b) = m*a + n*b
- Extended Euclidean Algorithm outputs m and n
  - if gcd(a, b) = 1, then <u>1 = m*a + n*b</u>
    - 1 = m*a mod b
    - m = a^{-1} mod b
- Given a and b, EEA outputs m, s.t., m = a^{-1} mod b
- Given e and $\phi(N)$, EEA outputs d, s.t.

$$d = e^{-1} \mod \phi(N)$$

- <u>EEA Example:</u> $a = 3$, $b = 20$, $\gcd(a, b) = 1$
  - $a = 3$, $b = 20$ (switch)
    - Divide 20 by 3, $20 = 6*3 + 2$, reminder is 2
  - $a = b = 3$, $b = r = 2$
    - Divide 3 by 2, $3 = 1*2 + 1$, reminder is 1
  - $a = b = 2$, $b = r = 1$
    - 1 is $\gcd(20, 3)$

- Based on round1: $r = 2 = 20 - 6*3$
- Based on round2: $r = 1 = 3 - 1*2$
- $1 = 3 - 1*(20 - 6*3) = 7*3 + (-1)*20 = m*a + n*b$
- $1 = 7*a \bmod b = 7*3 \bmod 20$
- $m = a^{-1} = 7 \bmod b$

Textbook RSA

- <u>Example of RSA</u>: p*q = 11*7 = 77 = N
  - $\phi(N)$ = $|Z_N^*|$ = (p-1)(q-1) = 10*6 = 60
    - choose e, s.t. gcd(e, $\phi(N)$) = 1
    - e = 7, $\phi(N)$ = 60,
    - public key pk = (N, e),
    - private key sk = d

- What is d, s.t. $d = e^{-1} \mod \phi(N)$?
  - Use EEA to compute d

- <u>EEA Example in RSA:</u>
  - a = e = 7, b = $\phi(N)$ = 60, gcd(a, b) = 1

  - Use Euclidean Algorithm compute gcd(a, b)
  - a = 7, b = 60 (switch)
    - Divide 60 by 7, 60 = 8*7 + 4, reminder is 4
  - a = b = 7, b = r = 4
    - Divide 7 by 4, 7 = 1*4 + 3, reminder is 3
  - a = b = 4, b = r = 3
    - Divide 4 by 3, 4 = 1*3 + 1, reminder is 1
  - a = b = 3, b = r = 1,
    - Divide 3 by 1, 3 = 3*1 + 0 , reminder is 0
    - 1 is gcd(7, 60)

- EEA Example in RSA: a = e = 7, b = $\phi(N)$ = 60,
  - a = 7, b = 60 (switch), reminder is 4, 60 = 8*7 + 4
  - a = b = 7, b = r = 4, reminder is 3, 7 = 1*4 + 3
  - a = b = 4, b = r = 3, reminder is 1, 4 = 1*3 + 1
  - a = b = 3, b = r = 1, reminder is 0, 1 is gcd(7, 60)
  - Based on round1: r = 4 = 60 - 8*7
  - Based on round2: r = 3 = 7 - 1*4
  - Based on round3: r = 1 = 4 - 1*3
  - 1 = 4 - 1*(7 - 1*4) = (-1)*7 + 2*4 = (-1)*7 + 2(60 - 8*7) = (-17)*7 + 2*60 = m*a + n*b
  - 1 = m*a mod b = (-17)*7 mod 60
  - m = a^{-1} = -17 = 43 mob 60
  - d = e^{-1} = 43 mod $\phi(N)$

RSA Performance

Try **“`openssl speed rsa`”** on your computer

performance on Mac OS	RSA-2048(Enc)	RSA-2048(Dec)
Operations/ second	12,200	329

- Normally, RSA Enc is faster than RSA Dec
  - e is small, $c = m^e \bmod N$ is faster
  - d is large, $m = c^d \bmod N$ is slower

RSA Performance

Try **`“openssl speed aes”`** on your computer

performance on Mac OS	AES-CBC-256	RSA-2048(Enc)
Operations/ second	7,368,657	12,200

- RSA is much slower than AES
  - AES is over 600 times faster than RSA
  - E.g., Encrypting a movie with RSA is inefficient

Hybrid Encryption

- Hybrid Encryption
  - Encrypt data with a random AES key
  - Then encrypt random AES key with RSA

m

Encryption

Given pk and m

- Generate a random key k
- $c_1 \leftarrow \mathsf{AES.Enc}_k(m)$
- $c_2 \leftarrow \mathsf{RSA.Enc}_{pk}(k)$

c

Decryption

Given sk and $c = \{c_1, c_2\}$

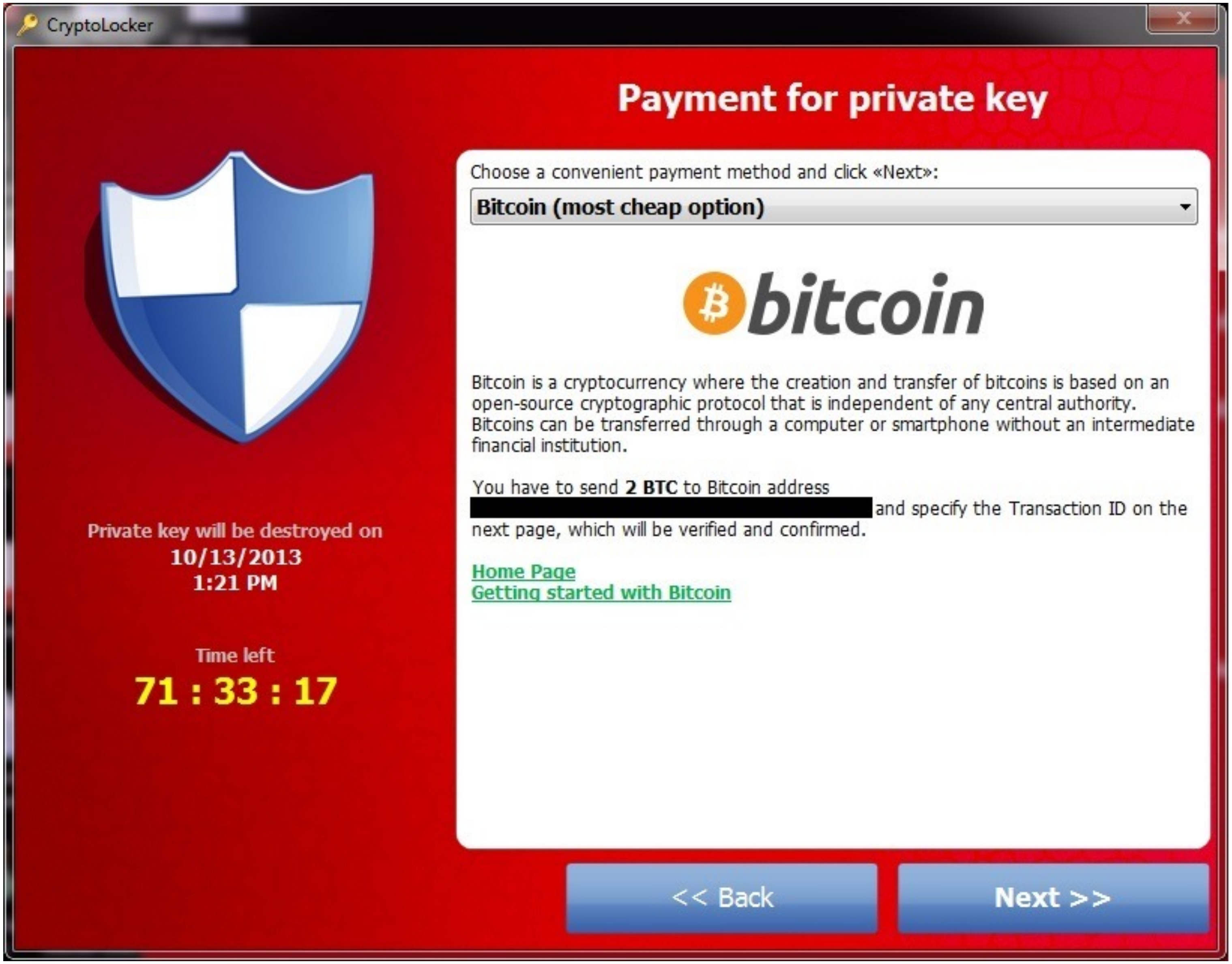
- $k \leftarrow \mathsf{RSA.Dec}_{sk}(c_2)$
- $m \leftarrow \mathsf{AES.Dec}_k(c_1)$

m

Ransomware

- Ransomware encrypts Alice's data on her laptop
- Alice needs to pay certain amount of Bitcoin to attacker to recover her data

- Ransomware leverages Hybrid Encryption
  - Alice clicks a link in a spam email
  - Malware generates a random AES key
  - Malware encrypts Alice's data with AES
  - Malware encrypts AES key with Attacker's pk
  - Malware sends encrypted AES key to Attacker


CryptoLocker
Payment for private key
Choose a convenient payment method and click «Next»:
Bitcoin (most cheap option)
bitcoin
Bitcoin is a cryptocurrency where the creation and transfer of bitcoins is based on an open-source cryptographic protocol that is independent of any central authority. Bitcoins can be transferred through a computer or smartphone without an intermediate financial institution.
You have to send 2 BTC to Bitcoin address [redacted] and specify the Transaction ID on the next page, which will be verified and confirmed.
Home Page
Getting started with Bitcoin
Private key will be destroyed on
10/13/2013
1:21 PM
Time left
71 : 33 : 17
<< Back
Next >>

Ransomware

- Ransomware leverages Hybrid Encryption
  - Malware sends encrypted AES key to Attacker
  - Malware asks Alice to pay Bitcoin
  - Alice pays Bitcoin, Attacker receives Bitcoin
  - Attacker sends AES key (in plaintext) to Alice
  - Alice decrypts her data with AES key

  - If Attacker receives Bitcoin, but does not send AES key to Alice, Alice………..

Ransomware

- Why Ransomware is successful (why Alice cannot recover her data without paying Bitcoin)?
  - Hybrid encryption is secure

- Some unsuccessful examples of ransomware fail to implemented correctly
  - Use a same seed, e.g., `Rand(0)`, for AES key
  - Alice can recompute AES key with `Rand(0)`
  - Alice recovers her data without paying bitcoin

Chinese Reminder Theorem

- Sun Tzu, Chinese mathematician, 4th century AD

"We have a number of things, but we do not know exactly how many. If we count them by threes we have two left over. If we count them by fives we have three left over. If we count them by sevens we have two left over. How many things are there?"

$$\begin{aligned} x &= 2 \mod 3 \\ x &= 3 \mod 5 \\ x &= 2 \mod 7 \end{aligned}$$

Chinese Reminder Theorem

- CRT indicates an interesting structure of group Z_N^*, which can improve RSA encryption time

- CRT: for $N=p*q$, p and q are primes.
  - for all integers u and v, where
    - u in $Z_p^*=\{1, 2, \ldots, p-1\}$, v in $Z_q^*=\{1, 2, \ldots, q-1\}$
  - there is a unique x in Z_N^*, s.t.
    - $x = u \bmod p$ && $x = v \bmod q$
    - i.e., a mapping x <—> (u, v)

Chinese Reminder Theorem

- CRT indicates a unique mapping from x <—> (u, v)
  - N =pq, p and q are primes
- For any x in Z_N^*, there is a unique pair (u, v) in (Z_p^*, Z_q^*)
  - Isomorphism: Z_N^* <—> (Z_p^*, Z_q^*)

- Example: N=15=5*3=p*q, Z_{15}^* <—> (Z_5^* Z_3^*)
  - $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$
  - $Z_5^* = \{1, 2, 3, 4\}$
  - $Z_3^* = \{1, 2\}$

- Example: $N = 15 = 5*3 = p*q$, $Z_{15}^* \longleftrightarrow (Z_5^*\ Z_3^*)$
  - $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$
  - $Z_5^* = \{1, 2, 3, 4\}$, $Z_3^* = \{1, 2\}$
  - E.g., $x = 11 \bmod N$
    - $x = 11 = 1 = u \bmod p$, $x = 11 = 2 \bmod q$
    - $11 \longleftrightarrow (1, 2)$
  - E.g., $x = 13 \bmod N$
    - $x = 13 = 3 \bmod p$, $x = 13 = 1 \bmod q$

$1 \longleftrightarrow (1,1)$	$2 \longleftrightarrow (2,2)$	$4 \longleftrightarrow (4,1)$	$7 \longleftrightarrow (2,1)$
$8 \longleftrightarrow (3,2)$	$11 \longleftrightarrow (1,2)$	$13 \longleftrightarrow (3,1)$	$14 \longleftrightarrow (4,2)$

Properties of CRT

- CRT indicates a unique mapping from $x \longleftrightarrow (u, v)$
- Isomorphism: $Z_N^* \longleftrightarrow (Z_p^*, Z_q^*)$
  - Z_N^*, Z_p^*, and Z_q^* are multiplicative groups
- Given g_1, g_2 in Z_N^*
  - if $g_1 \longleftrightarrow (u_1, v_1)$, $g_2 \longleftrightarrow (u_2, v_2)$
  - u_1, u_2 in Z_p^*, v_1, v_2 in Z_q^*

  - $g_1 * g_2 \longleftrightarrow (u_1 * u_2, v_1 * v_2)$
  - For any integer d, $g_1^d \longleftrightarrow (u_1^d, v_1^d)$

1<—>(1,1)	2<—>(2,2)	4<—>(4,1)	7<—>(2,1)
8<—>(3,2)	11<—>(1,2)	13<—>(3,1)	14<—>(4,2)

- N = 15 = 5*3 = p*q, Z_{15}^* <—> (Z_5^*, Z_3^*)
  - Z_{15}^* = {1, 2, 4, 7, 8, 11, 13, 14}
  - Z_5^* = {1, 2, 3, 4}, Z_3^* = {1, 2}
- Example: 11 <—> (1, 2), 4 <—> (4, 1)
  - 11*4 = 44 = 14 mod N
  - 1*4 = 4 mod p
  - 2*1 = 2 mod q
  - 14 <—> (4, 2)

1<—>(1,1)	2<—>(2,2)	4<—>(4,1)	7<—>(2,1)
8<—>(3,2)	11<—>(1,2)	13<—>(3,1)	14<—>(4,2)

- $N = 15 = 5*3 = p*q$, $Z_{15}^* \longleftrightarrow (Z_5^* \, Z_3^*)$
  - $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$
  - $Z_5^* = \{1, 2, 3, 4\}$, $Z_3^* = \{1, 2\}$
- Example: 11 <—> (1, 2), integer m = 4, 11^m = ? mod N
  - $11^m = 11^4 = 1 \bmod N$
  - $1^m = 1^4 = 1 \bmod p$
  - $2^m = 2^4 = 16 = 1 \bmod q$
  - 1 <—> (1, 1)

Chinese Reminder Theorem

- Isomorphism: Z_N^* <—> (Z_p^*, Z_q^*)
- CRT shows a (faster) way of computing exponentiation
  - For any integer d, g^d <—> (u^d, v^d)
  - Instead of computing g^d, compute u^d and v^d

$$\begin{tikzcd}[column sep=huge, row sep=huge]
c \arrow[r, "\text{convert representation}"] \arrow[d, "\text{exponentiate in } \mathbb{Z}_{pq}"'] & (u, v) \arrow[d, "\text{exponentiate in } \mathbb{Z}_p \times \mathbb{Z}_q"] \\
c^d & (u^d, v^d) \arrow[l, "\text{convert representation}"]
\end{tikzcd}$$

Why CRT is Faster?

- For an n-bit element, one exp costs n^3 steps
- Given $N = pq$, N is 2n-bit, p, q are n-bits
  - Given $g \longleftrightarrow (u, v)$ and integer d, compute g^d
  - Without CRT: compute $g^d \bmod N$
    - $(2n)^3 = 8n^3$ steps
  - With CRT: compute $u^d \bmod p$, $v^d \bmod q$
    - $n^3 + n^3 = 2n^3$ steps
  - Mapping between $g^d \longleftrightarrow (u^d, v^d)$ is fast
- Overall, using CRT is (about) 4 times faster

Additional Reading

Chapter 8, *Introduction to Modern Cryptography, Drs. J. Katz and Y. Lindell, 2nd edition*