CNIT 370

MEA 3

RSA (TOTAL 50 PTS)

PURDUE POLYTECHNIC
COMPUTER AND INFORMATION TECHNOLOGY

NOTE

- You need to include all the group activities in your final MEA report.
- You should zip all the document in a single .zip file and upload it the zip file to Blackboard
- MEA3 report is due by the end of day (11:59pm) on 10/26/17. Blackboard is always slow around 11:59pm, please submit it at least a few minutes, if not a few hours earlier.
- Two upload attempts will be allowed. But only the last attempt will be graded.

Task 1.1 Individual Activity (3 pts)

In 3 minutes, please write down how to use asymmetric key to encrypt and decrypt a message. Use math notations, language, and the diagram to illustrate it.

Task 1.2: Group Activity (2 pts)

- **In 5 minutes, please discuss the following question with the students on your table:**

Diffie-Hellman (DH) Key exchange is often categorized as a public key or asymmetric key system. Can you directly use DH to encrypt and decrypt a message? Why?

Suggested Reading:

- The Secret Story of Nonsecret Encryption

https://www.schneier.com/essays/archives/1998/04/the_secret_story_of.html

- The Open Secret

https://www.wired.com/1999/04/crypto/

Misconceptions on Public Key

- Public-key encryption is more secure from cryptanalysis than symmetric encryption
- Public-key encryption is a general-purpose technique that has made symmetric encryption obsolete
- There is a feeling that key distribution is trivial when using public-key encryption, compared to the cumbersome handshaking involved with key distribution centers for symmetric encryption

Public key Principles

- **It can be used for encryptions**
  - Anything encrypted with public key can be decrypted use its corresponding private key, and vice versa.
  - Why we don't use asymmetric keys directly on encryption?

Generally used in two occasions

- **Key distribution (session set-up)**
  - How to have secure communications in general without having to trust a KDC with your key
- **Digital Signatures (Non-interactive Apps)**
  - How to verify that a message comes intact from the claimed sender

Public Key Requirements

- **A trap-door one-way function is a family of invertible functions f_k, such that**
  - $Y = f_k(X)$ easy, if k and X are known
  - $X = f_k^{-1}(Y)$ easy, if k and Y are known
  - $X = f_k^{-1}(Y)$ infeasible, if Y known but k not known

- **However, do not directly apply the trap-door function as the encryption/description algorithms because the trap-door function is deterministic.**

- ***Some refers to the textbook RSA as RSA trapdoor***

RSA

- **Ronald Rivest, Adi Shamir, Leonard Adelman**
  - 1978 - Communications of the ACM (Feb)
- **Most widely used general-purpose approach to public-key encryption**
- **Currently the "Work Horse" of IT Security**
  - Most PKI products, SSL/TLS, IPSec, PGP, Outlook…
- **Is a cipher in which the plaintext and ciphertext are integers between 0 and $n – 1$ for some n**
  - A typical size for n is 1024 bits, or 309 decimal digits

The Number Theories related to RSA:

- **Prime Factorization**
- **Fermat's little theorem (p is a prime #)**
  - $a^{p-1} \bmod p = 1$

  where p is prime and gcd(a,p)=1
- **Euler Totient Function ø(n)**
  - Number of elements in reduced set of residues
  - for p.q (p,q prime) ø(p.q) = (p-1)(q-1)
- **Euler's Theorem: (**N does not need to be a prime #**)**
  - $a^{ø(N)} \bmod N = 1$ where gcd(a,N)=1, N
- **The Chinese Remainder Theorem (trapdoor)**
  - $x_{mod\ n}=(x_{mod\ p} * x_{mod\ q})$ if $n=pq$

PURDUE POLYTECHNIC INSTITUTE

RSA process

- p and q are two prime numbers.
- $N = pq$
- $t = (p-1)(q-1)$
- e is such that $1 < e < t$ and $\gcd(t,e) = 1$.
- d is such that $(ed) \bmod t = 1$.

- Public key: $P=\{e,N\}$
- Private key: $S=\{d,p,q\}$
- Message: M
- Encrypt => $C = M^e \bmod N$.
- Decrypt => $M = C^d \bmod N$.

RSA works, because

- in RSA have:
  - N=p.q
  - ø(N)=(p-1)(q-1)
  - carefully chosen e & d to be inverses mod ø(N)
  - hence e*d=1+k.ø(N) for some k

- Hence: (all the calculation is mod N)

$$C^{d} = (M^{e})^{d} = M^{ed} = M^{1+k.ø(N)} = M^{1}.(M^{ø(N)})^{k}$$
$$= M^{1}.(1)^{k} = M^{1}$$

Finding e and d

- **Euclid's algorithm**
  - GCD (m,n)=GCD (n, mod n) (m>n). Continue the process until n=0
- **Using Euclid's extended algorithm**
  - `x[0] = (p-1)*(q-1)   y[0] = 0`
  - `x[1] = e          y[1] = 1`
  - `while x[i] > 0 calculate: x[i] = x[i-2] modulo x[i-1]`
  - `y[i] = y[i-2] - floor( x[i-2] / x[i-1] ) * y[i-1]`

Task 2 Group Activity RSA Example, (5pts)

Finish this in 10 minutes

1. Select primes: p=17 & q=11,
2. Compute $N = pq =$ ______________________________
3. Compute $\phi(N)=(p-1)(q-1)=$ ______________________
4. Select e : gcd(e,____)=1; choose e= 7
5. Determine: d= 23 works because __________________
6. Publish public key P= ________________________________
7. Keep secret private key S= ________________________________
8. given message M = 88 (88 <   )
9. encryption: C = __________
10. decryption: M = __________

RSA Keys

- **The public key is the combination of *e* and *N***
  - Made available to everyone
- **The private key is the combination of *p, q,* and *d***
  - You can calculate any of these from any other
    - Therefore many references will state the private key is simply *d*
  - Kept secret

What if you lost either p, q, or d?

Choosing values for RSA variables

- **Values of *e***
  - RSA can be used for both encryption and digital signatures
  - You should always use different values of *e* for each action
    - Ensures that the two applications don't interact
  - Common applications are *e*=3 for signatures and *e*=5 for encryption or *e*=17 for signatures and *e*=65537
- **Values of *n***
  - N should be at least 2048 bits
  - Therefore *p* and *q* should be at least 1024 bits

Task 3: Individual Activity (10 pt)

- Use the ‘RSA key Generator’ and the ‘RSA’ module in Cryptool 2.0, illustrate how to encrypt and decrypt a message. Do this outside classroom.
- A) Encrypt a message (5pt), use random prime generation with a range of 50. Output the message in Hex format (output the byte array to a String Encoder, and choose presentation format Hex)
- A) Decrypt the cipher text produced in Task 3.A (5pt)

Type (or Copy & Paste) the answers in the report, and attach the *.cwm file in the zip file.

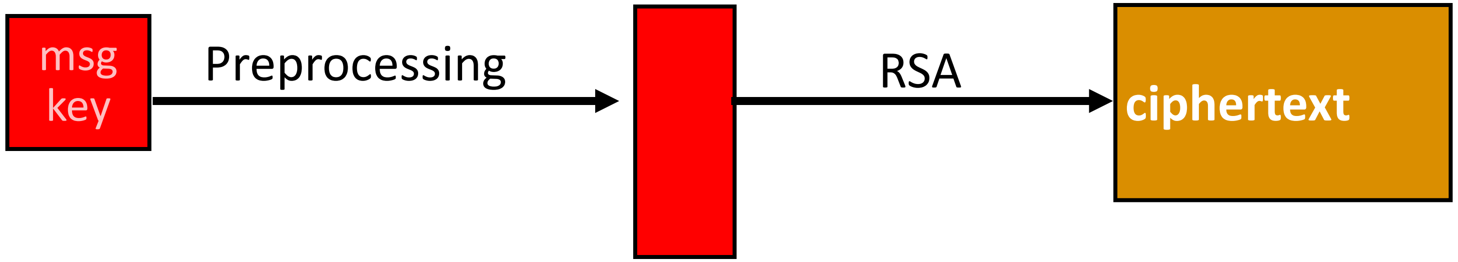
Task 4, Group Activity (10 pt)

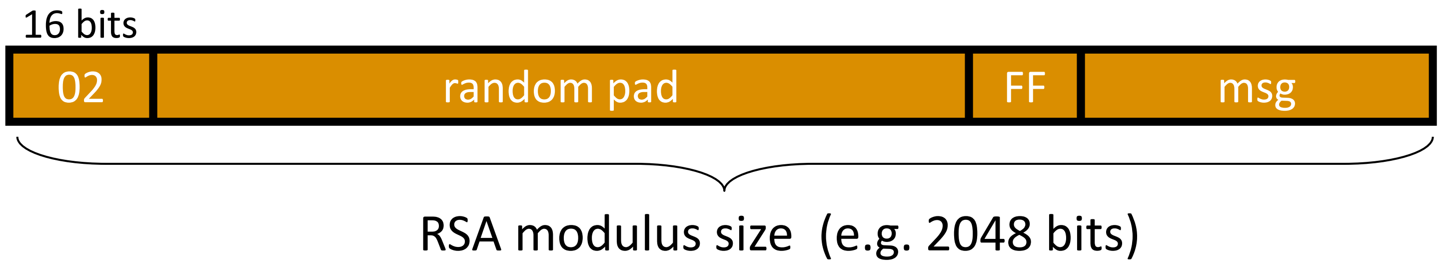
- Public key: (N: <u>56977</u> e: <u>23</u>)
- Cipher Text (HEX)
- AA 12 49 0D EE B0 6B 79 FE BD 93 4E 49 0D D3 8E 5C 43 36 CB 8D 43 49 0D DE D3 99 9D 49 69 93 4E
-
- Use factorizer, RSA key generator and RSA (decryption mode) to break the ciphered text.

RSA Implementation

- **All RSA messages must be larger than the eth root of n**
  - Or else no modulo reduction takes place and you can easily recover the message
    - If e=5 and $m < 5^{th}$ root of n then an attacker can simply take the 5^{th} root of m to recover m
- **This is common with sending AES keys via RSA**
  - Use pre-processing to ensure m is large enough
- **RSA encryption is usually much faster than Decryption (CRT:** Chinese Reminder Theory**)**

RSA encryption in practice

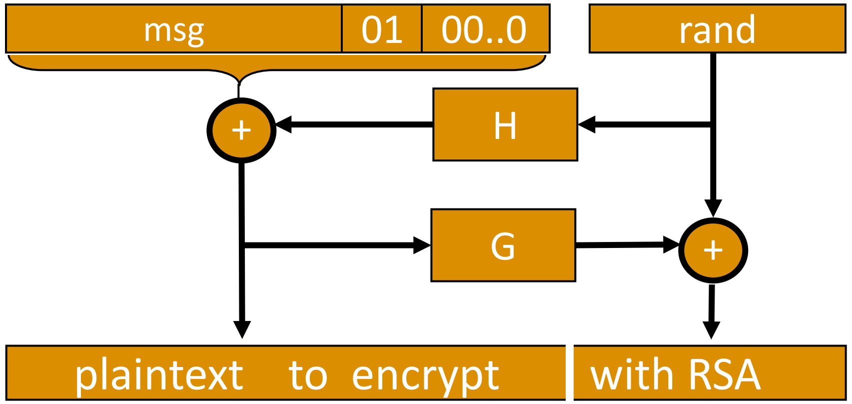




Known as PKCS1 mode 2 (still not very secure), widely used in https [Bleichenbacher attack, 1998]

Slides from Dr. Dan Boenh, Stanford University

PKCS1 v2.0 OAEP (1994)



Slide from Dr. Dan Boenh, Stanford University

Theorem: RSA-OAEP is CCA secure when H,G are *random oracles (ideal hash functions)*

in practice: use SHA-256 for H and G

Subtleties in implementing OAEP [M '00]

OAEP-decrypt(ct):

```
error = 0;
  ........
if ( RSA^-1(ct) > 2^(n-1) )
    { error =1;  goto exit; }
 ........
if ( pad(OAEP^-1(RSA^-1(ct))) != "01000" )
    { error = 1;  goto exit; }
```

Problem: timing information leaks type of error

$\Rightarrow$ Attacker can decrypt any ciphertext

Lesson: Don't implement RSA-OAEP yourself !

Slide from Dr. Dan Boenh, Stanford University

Attacks on RSA Implementations

- **Timing attack: (1997)**
  - The time it takes to compute $C^d \pmod{N}$ can expose d.
- **Power attack: (1999)**
  - The power consumption of a smartcard while it is computing $C^d \pmod{N}$ can expose d.
- **Faults attack: (1997)**
  - A computer error during $C^d \pmod{N}$ one error can expose d

OpenSSL defense: check output. 10% slowdown.

RSA Key Generation problems

OpenSSL RSA key generation (abstract):

```
prng.seed(seed)
p = prng.generate_random_prime()
prng.add_randomness(bits)
q = prng.generate_random_prime()
N = p*q
```

Poor entropy at startup:

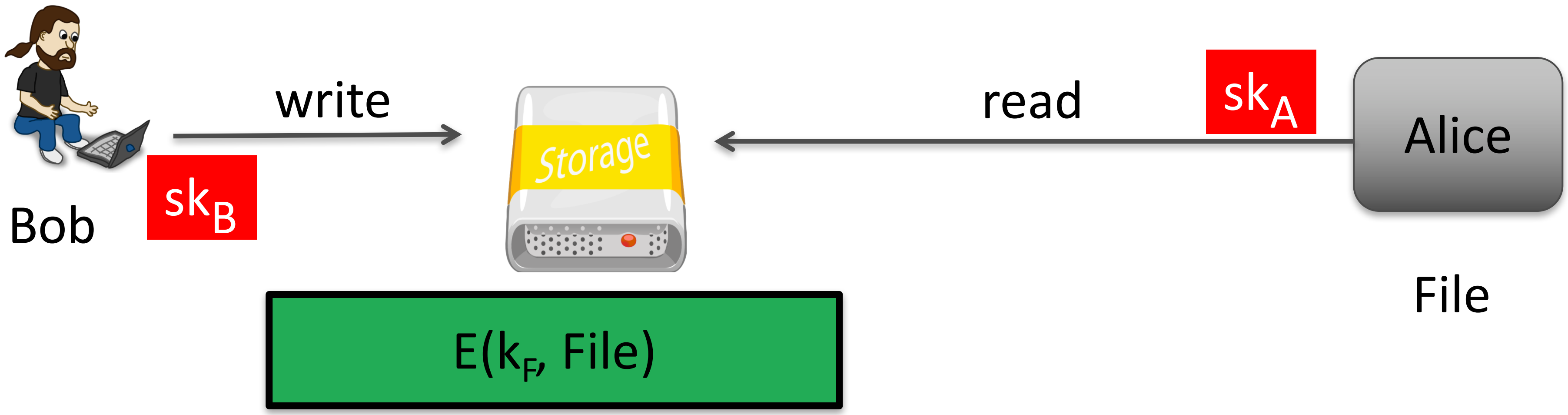
- Same p will be generated by multiple devices, but different q
- N_1 , N_2 : RSA keys from different devices $\Rightarrow$ $\gcd(N_1,N_2) = p$

Slide from Dr. Dan Boenh, Stanford University

Task 5. Individual Activity (5pt)

How do we use public-key encryption to encrypt disk? (EFS)

Hint: You really want to encrypt the file using symmetric key encryption, such as AES. In the example, it is E(K_p, $File$) . So the question is: how do you allow both Alice and Both know K_p ? Use language, diagram and math notation to describe it.



Adapted from Dr. Dan Boenh's course, Stanford University

Task 6. Group Activity (10 pt)

- Cryptool V1, ‘Analysis’, → ‘Asymmetric Encryption’ → ‘Side Channel Attack on Textbook RSA’
- Click ‘Show Information Dialogs’ on the bottom right, then following the instruction to complete the demo.
- Explain in diagram, math notations, and language, how the normal encryption and decryption is carried in this example (5pt)
- Explain in more than two different representations, (two out of language, diagram, and math notations) how the attack is conducted.

Task 7. Individual Activity: 5pt

- Suppose someone finds a way to easily factor large prime numbers. This makes RSA no longer secure. When searching for alternatives, someone suggested that Diffie-Hellman algorithms can be revised to replace RSA for public key and private key encryption.

- If it works, does it make the revised DH safe to use? Put it differently, does large prime factorization a threat to DH?
- If it works, illustrate in language and diagram/math notation, how it works.   (HINT: El Gamal)