NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

3010 Lecture Week 12

RSA, PKC & Crypto Failures

Dr Tay Kian Boon

Number 2 on Top 10 OWASP

A02:2021 – Cryptographic Failures

OWASP Top 10 Vulnerabilities

- OWASP a professional org that monitors security failures.
- (www.owasp.org)
- They maintain website: top 10 failures every few years.
- At Number 2:

- Cryptographic Failures

Cryptographic Failures

- Shifting up one position to #2, (previously *Sensitive Data Exposure*)
- Focus is on failures related to cryptography (or lack thereof).
- Often lead to exposure of sensitive data.
- Cryptography must be implemented correctly in order for data to be safe.
- Many points of possible failure in doing it correctly.

Cryptographic Failures

- Notable Common Weakness Enumerations (CWEs) included are
  - *CWE-259: Use of Hard-coded Password,*
  - *CWE-327: Broken or Risky Crypto Algorithm &*
  - *CWE-331 Insufficient Entropy.*
    - (normally link to weak random number generation for keys & IVs etc) – will share this later

Cryptography Overview

Main Crypto Ingredients

- Strong Crypto Algorithms
  - Use to protect data (can be voice, video etc…)
- Secure Hash Functions
  - Use in Digital signatures & ensuring data integrity
- Strong Random Number Generators
  - Use to generate unbiased random keys for crypto algorithm use

Crypto Algorithms

For A to talk to B securely, both need to use

- Same (strong) Crypto Algorithms (private key cryptosystem)
  - Eg AES Encryption algorithm
- Same (strong) Crypto Algorithms (public key cryptosystem)
  - Eg RSA encryption algorithm
- Keys used
  - Need to be generated from crypto secure RNG such as ISAAC, FORTUNA,...
- Protocol to send message must be robust, no leakage, no weakened entropy

Private (symmetric) Key Crypto

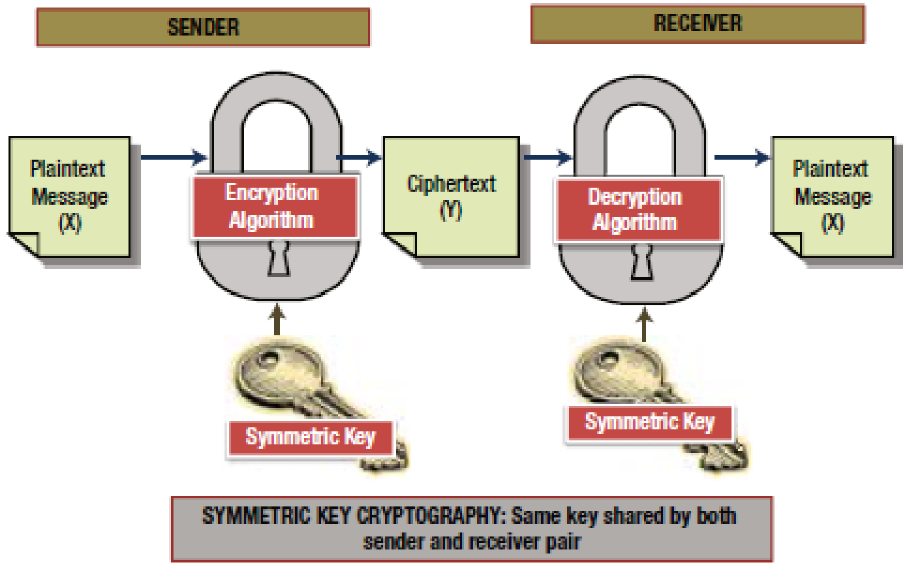


Figure 8.2 Symmetric Key Cryptography

Private (symmetric) Key Crypto

- Private (symmetric) key algorithms such as AES are good.
- Suffer from some serious drawbacks on its own:

1. How to agree on new key change (how to solve this?)
2. How to manage keys – eg storage, expiry date etc
3. How to send encrypted message to someone you don't know?
    1. Basis of ecommerce!

Answer: Public Key Crypto! 2 Keys: Public and Private!

Public Key Cryptography

Public key cryptography uses a pair of keys for encryption and decryption. A ***public key*** is used to encrypt the data and a ***private key*** is used to decrypt the data. Using the public key, anyone can encrypt the data, but they cannot decrypt the data. In this approach, both sender and receiver have the ability to generate both keys (using a computer system) together. However, only the public key is made known to the other party, who can download this key even from a web server; the private key is not known to anyone. It is not sent to the other party, hence the problem of distribution of the key never arises. In case of intrusion or any other problems, the system can generate a private key, and a corresponding public key that can be published again. The algorithms that generate keys are related to each other mathematically in such a way that knowledge of one key does not permit anyone to determine the other key easily.

Figure 8-5 illustrates how the confidentiality of a message is ensured through asymmetric key cryptography (alternatively known as public key cryptography).

Public Key crypto

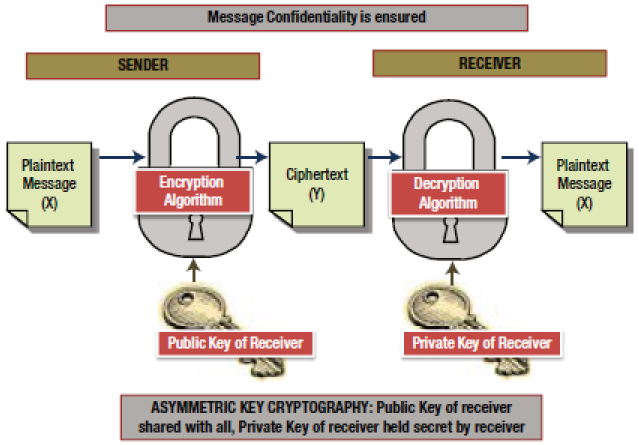
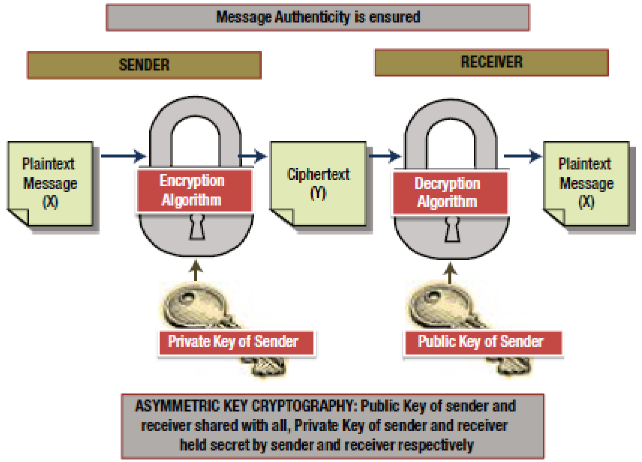


Figure 8-5. *Public Key Cryptography – How Confidentiality is ensured*

Figure 8-6 illustrates how the authenticity of the message is ensured through asymmetric key cryptography (i.e., public key cryptography).

Public Key crypto-ensure authentication



Message Authenticity is ensured
SENDER
RECEIVER
Plaintext Message (X)
Encryption Algorithm
Ciphertext (Y)
Decryption Algorithm
Plaintext Message (X)
Private Key of Sender
Public Key of Sender
ASYMMETRIC KEY CRYPTOGRAPHY: Public Key of sender and receiver shared with all, Private Key of sender and receiver held secret by sender and receiver respectively

Figure 8-6. *Public Key Cryptography – How Authenticity is ensured*

Public Key crypto-ensure authentication

- If A wants to let B know he has sent the message using PKC.
- A uses his private key to encrypt msg, and send to B.
- B want to verify if message is really sent by A.
- She will look up A’s public key to decrypt the message.
- Note Public-private key pairs are inverse operations of each other.
- If B manage to read the decrypted msg, then B knows msg has been sent by A (assuming A has not lost his private key!)

Making Sense of Public Keys

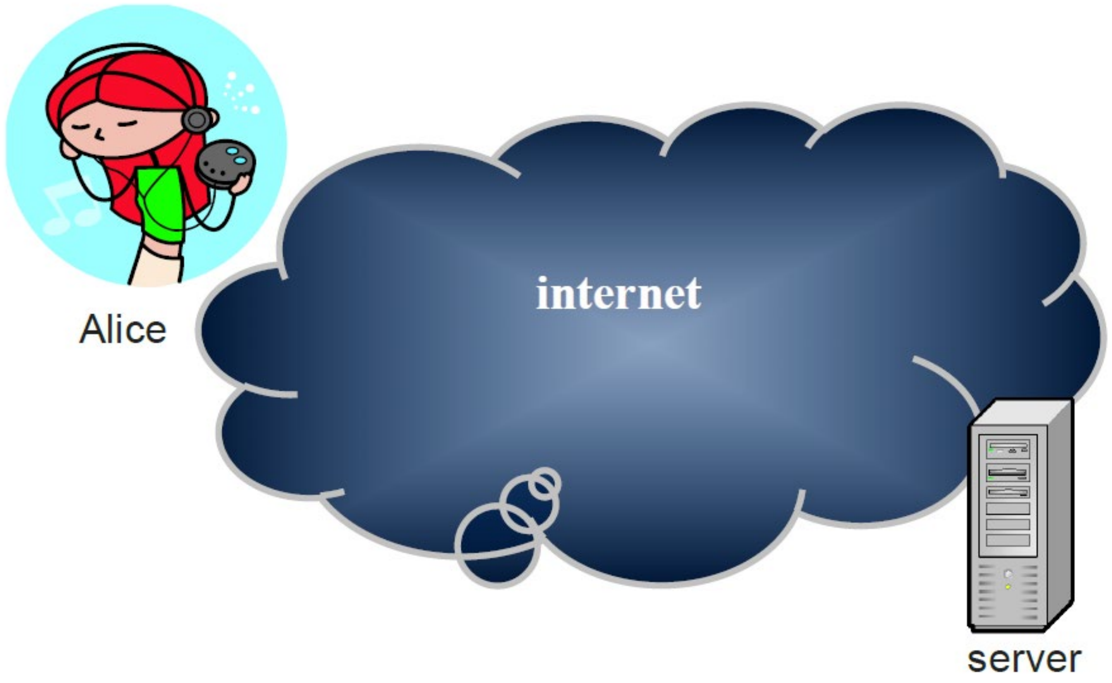
- If you want to write to a party known to you by name, you need a binding between name and public key
  - This binding makes sense of the cryptic sequence of bits

- Hence, a procedure is required for *A* to get an authentic copy of *B*'s public key
  - Need not be easier than getting a shared secret key

- You can use the same key with all the parties that you want to receive encrypted messages from

The Challenge –Integrity

How does Alice recognize tampering with data from server?



Integrity

- Data transmitted over an insecure channel might have been modified in transit or come from a spoofed source (active attacks)

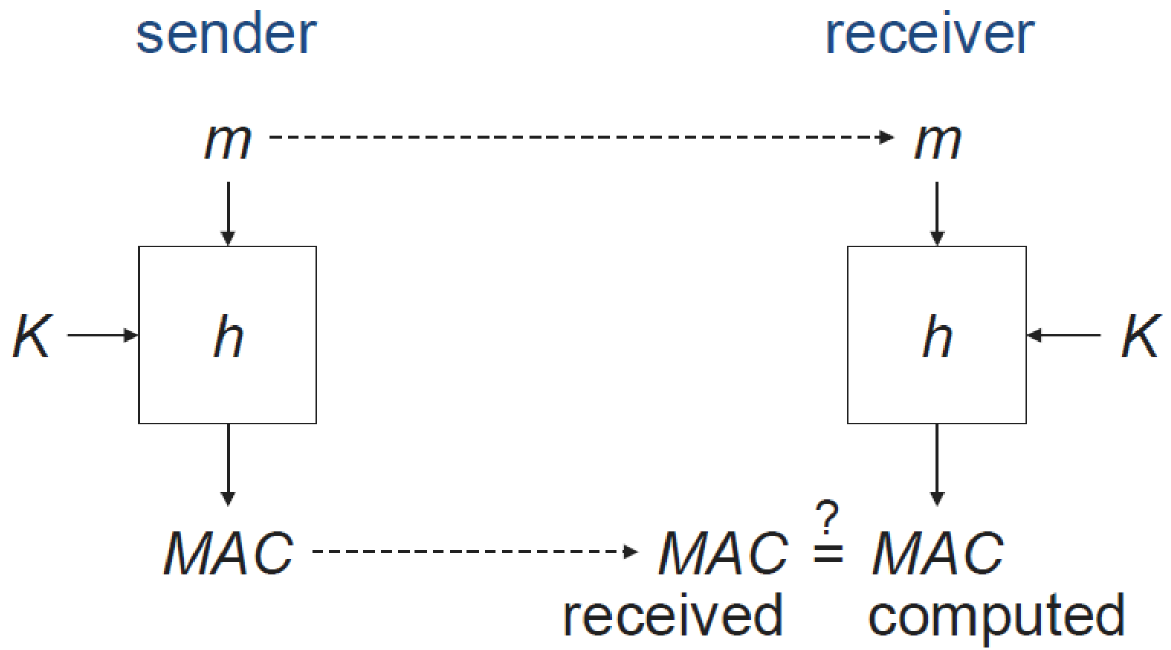
- Protection (symmetric cryptography):
  - message authentication codes (MAC)

- Protection (asymmetric cryptography):
  - digital signatures

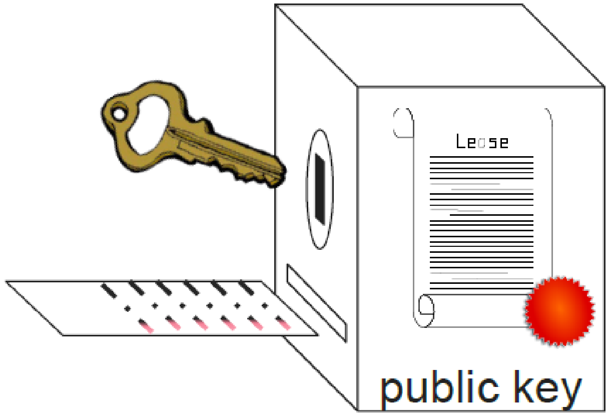
MAC

- Sender and receiver share a secret key *K*
- Compute MAC *h*(*m*,*K*)from message *m* and secret key *K* using a suitable function *h*
- To authenticate a message, the receiver needs the secret key used by the sender for computing the MAC (see next slide)
- A third party that does not know this key cannot validate the MAC

MAC



Digital Signatures



sign: place document in a lockable transparent filing cabinet; document gets tagged on insertion

verify: check whether document is in filing cabinet identified by public key

Digital Signatures

- Public verification key and private signature key
- Signature on document *m* computed with private key
- Public verification key used to check signature on *m*
  - Public key identifies a document repository
  - Private key used for inserting document *m* into the repository (unlock insertion slot of repository)
  - Signature: tag on *m* proving that *m* belongs into repository
- Technically, digital signatures are a cryptographic mechanism associating documents with public keys
- There is no ‘signer’ and ‘verifier’ in this explanation

MACs & Digital Signatures

- MACs and digital signatures are authentication mechanisms
- MAC verification needs secret used for computing the MAC; MAC unsuitable as evidence with a third party
  - Third party would need the secret
  - Third party cannot distinguish between the parties knowing the secret
- Digital signatures, however, can be used as evidence with a third party

Hash Functions

- Basically a fast one-way function to create a fixed length message digest
  - Famous example: SHA-256, SHA-512, KECCAK

- Used in data integrity, digital signatures

- Frequently when you want to download software on website, website will contain SHA(software) for integrity check
  - Apply SHA hash to software downloaded & see if both values agree

Random Number Generators (RNGs)

- Crypto algorithms need keys, and they are n-bit numbers generated by RNGs
- Good keys generated are unbiased and equally likely & unpredictable
- Need crypto-secure RNG to generate secure crypto keys
  - FORTUNE
  - ISAAC

Simple Overview SSL/TLS Protocol

- Client & Server say hello to each other
- Client tell server what algos it has
- Server (normally) picks strongest algo & hash offered by Client
- They exchange info leading to establish common key
- They can start talking (securely!)

2 Types Crypto Algorithms

- Private Key (symmetric)
  - Famous eg AES (128,192, 256)
    - index number refers to key length

- Public Key (asymmetric)
  - Famous eg RSA(1024, 2048)
    - RSA 1024 considered weak nowadays (RSA 829) has been cracked

Strength of Private key Algo

The strength of Strong Algorithms like AES depends on

- Algo Design
- Keys generated (quality of RNG)
- Key length
- Secure implementation!

RSA

- RSA public parameter N is the product of Two k-bit prime numbers (independently generated)
  - Need good RNG to generate LARGE random odd number first
  - Need robust prime generation routines.
    - Basically find nearest prime to it, call it p
    - Generate another large random odd q. Then find nearest prime to it.
    - Form N=p*q

- Security of RSA is based on difficulty of factoring large numbers

- However there are certain RSA parameters that are weak. Such weakened RSA system can be broken easily without the need to factor the Large number N.

RSA in 1 Page

1. Choose 2 random k-bit indep primes p & q and form n =p*q.
2. Compute $\Phi(n) = \Phi(pq) = (p-1)(q-1)$.
3. Choose e, encryption exponent s.t. gcd (e, $\Phi(n)$) =1.

   (Eg gcd (4,6) = 2, gcd –greatest common divisor)
4. Public parameters is {n,e}.
5. Decryption exponent $d = e^{-1} (\text{mod } \Phi(n))$
6. Encrypt msg M (<n) by M^e (mod n) = C
7. Decrypt cipher C by C^d =(mod n) = M!

Some Weak RSA Parameters & Implementations

- Size Primes p & q $\leq$ 512 bits.
- p & q are not independently generated
- p & q are not randomly generated by crypto-secure RNG
- Decryption key d $\leq$ [N^(0.292)] (recall N=p*q) (“short d”)
- p-1 is a product of only “small” prime factors (best scenario:p-1=2R,R prime)
- Common use of same N for users in same company, although they use different encryption and decryption exponents.

Intro DH Key exchange

Now we cover key exchange by Diffie and Hellman!

Diffie-Hellman Key Exchange

Let p be prime, let g be a generator

- i.e. For any $x \in \{1,2,\ldots,p\text{-}1\}$ there is n s.t. $x = g^n \bmod p$

Alice selects randomly her private value a

Bob selects randomly his private value b

Alice sends $g^a \bmod p$ to Bob

Bob sends $g^b \bmod p$ to Alice

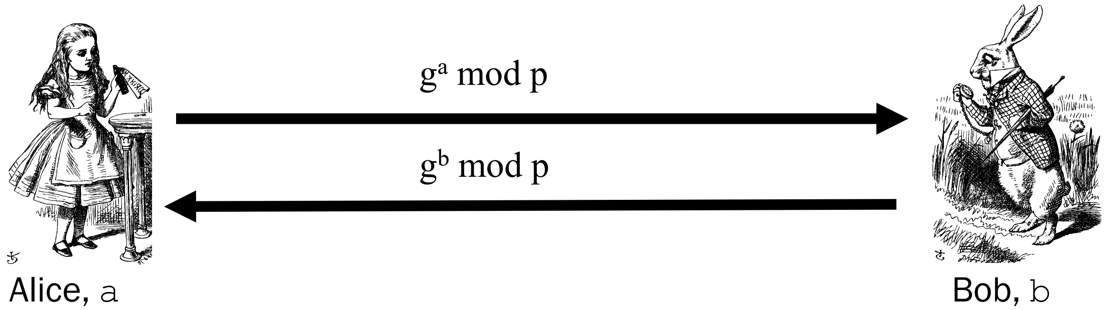
Both compute shared secret, $g^{ab} \bmod p$

Shared secret can be used as symmetric key!

Diffie-Hellman Key Exchange

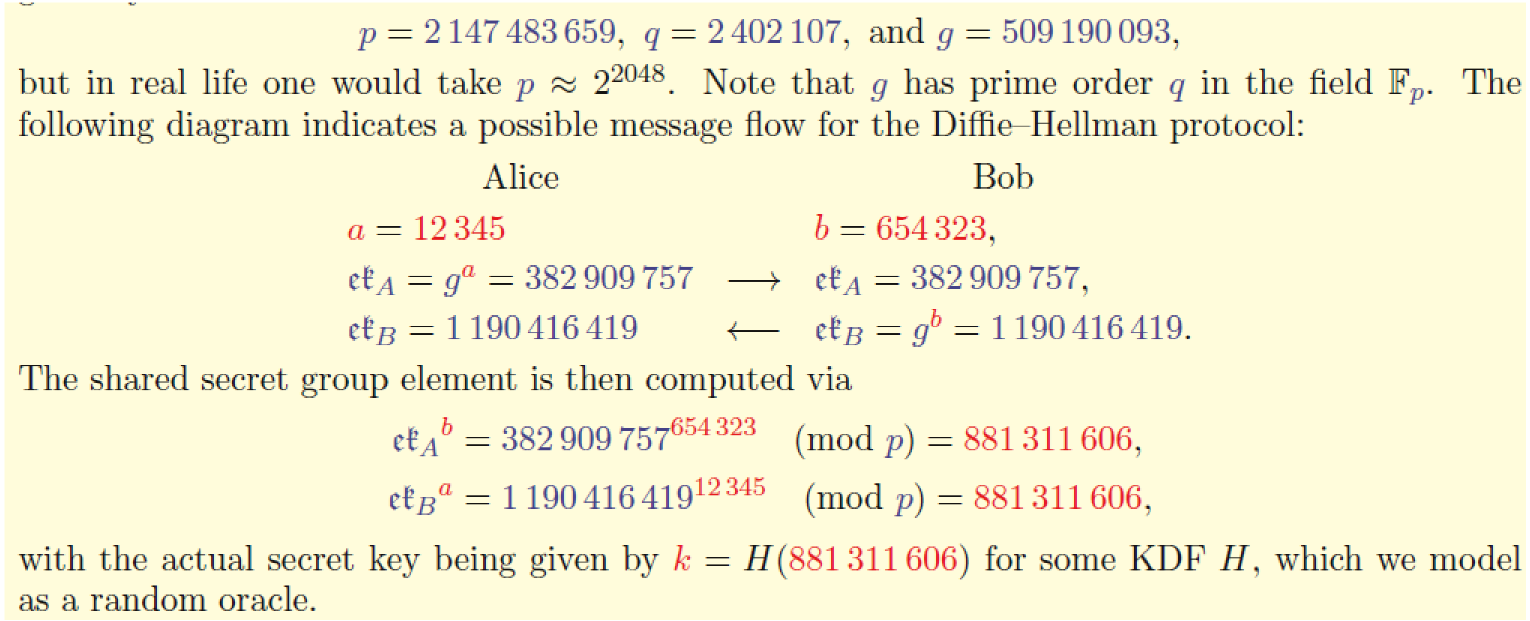
Public: g and p

Private: Alice's exponent a, Bob's exponent b



- Alice computes $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- Bob computes $(g^a)^b = g^{ab} \bmod p$
- They can use $\mathrm{K} = g^{ab} \bmod p$ as symmetric key

Diffie-Hellman Key Exchange - Example



$$p = 2\,147\,483\,659, \quad q = 2\,402\,107, \quad \text{and } g = 509\,190\,093,$$

but in real life one would take $p \approx 2^{2048}$. Note that g has prime order q in the field $\mathbb{F}_p$. The following diagram indicates a possible message flow for the Diffie–Hellman protocol:

$$\begin{array}{lcl}
\text{Alice} & & \text{Bob} \\
a = 12\,345 & & b = 654\,323, \\
\mathfrak{ct}_A = g^a = 382\,909\,757 & \longrightarrow & \mathfrak{ct}_A = 382\,909\,757, \\
\mathfrak{ct}_B = 1\,190\,416\,419 & \longleftarrow & \mathfrak{ct}_B = g^b = 1\,190\,416\,419.
\end{array}$$

The shared secret group element is then computed via

$$\mathfrak{ct}_A{}^b = 382\,909\,757^{654\,323} \pmod{p} = 881\,311\,606,$$

$$\mathfrak{ct}_B{}^a = 1\,190\,416\,419^{12\,345} \pmod{p} = 881\,311\,606,$$

with the actual secret key being given by $k = H(881\,311\,606)$ for some KDF H, which we model as a random oracle.

Crypto Advice

- DO NOT TRUST DO NOT TRUST VENDORS!
- Always verify their algorithms used if you have to purchase their products
- Verify their RNG used produces truly (close to) random bits
  - NIST Suite tests randomness
    - https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic
  - If in doubt, use your own RNG, FORTUNA and ISAAC are good so far.