Public-key Cryptography & Public Key Infrastructure

CSIE 7190 Cryptography and Network Security, Spring 2019

https://ceiba.ntu.edu.tw/1072csie_cns

cns@csie.ntu.edu.tw

Hsu-Chun Hsiao

Housekeeping

3/26: Reading critique #5 deadline

Reading critique #5

Write a critique on one of the following:

- A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. "The Tangled Web of Password Reuse," in NDSS, 2014.
- K. C. Wang and M. K. Reiter, "How to End Password Reuse on the Web," in NDSS, 2019.
- M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," in IEEE Transactions on Software Engineering, vol. 22, no. 1, pp. 6-15, Jan. 1996.

Text only, one page

Agenda

Public-Key Cryptography

- RSA cryptosystem
- ElGamal threshold decryption
- Diffie-Hellman key exchange

Public Key Infrastructures (PKI)

- Digital certificates & certificate authorities
- Real-world issues with PKIs/CAs

Public-Key Cryptography

Why public-key cryptography?

對稱式密碼學難以解決：

Key distribution problem

- 對稱式密碼學假設雙方已建立共同的秘鑰
- 秘鑰怎麼安全地建立？
- c.f. Merkle puzzles, key pre-distribution

Digital signatures

- 數位簽章需提供「不可否認性」
- 在對稱式密碼學中，簽名方跟驗證方須有共同的秘鑰
- 雙方知道的都相同，要如何達到不可否認性（non-repudiation）？
- c.f. TESLA, one-time signatures

Computational hardness assumptions

Many public-key cryptographic algorithms rely on **computational hardness assumptions**

- Assumption that a problem is infeasible to solve
- Assumptions, not proofs

Common hardness assumptions:

- Integer factorization
- RSA problem
- Discrete logarithm problem
- The elliptic curve discrete logarithm problem
- Decisional/Computational Diffie-Hellman assumptions
- Shortest vector problem
- …

Q: How about NP-hard problems?

Integer factorization problem

Easy: Multiply two large primes, $n = a \times b$

(Believed) Hard: Factor n

The unique prime factorization of 2016 = ?

The unique prime factorization of

```
123018668453011775513049495838496272077285356959533479219732245
215172640050726365751874520219978646938995647494277406384592519
255732630345373154826850791702612214291346167042921431160222124
047927473779408066535141959745985690214341 3 =?
```

```
33478071698956898786044169848212690817704794983713768568912
43138898288379387800228761471165253174308773781446799948 9
                                  ×
36746043666799590428244633799627952632279158164343087642676
03222838157396665112792333734171433968102700927987363089 17
```

Discrete logarithm problem

Logarithm vs. exponentiation: $b^y = x \Leftrightarrow log_b(x) = y$

Discrete logarithm problem:

- Public values: large prime p, generator g of a group
- Given p, g, and x, find a such that $g^a \ mod \ p = x$

Note

- A generator produces all elements in the group
- For simplicity, we use $\mathbb{Z}_p^*$ group as an example.
  - $\mathbb{Z}_p^*$ = all elements that are relatively prime to p

Discrete logarithm problem

Easy: compute discrete exponentiation

(Believed) Hard: compute discrete logarithm

Number field sieve is fastest algorithm known today to solve discrete logarithm problem

- Running time: $O(e^{(1.923+o(1))(\ln(p))^{1/3}(\ln(\ln(p)))^{2/3}})$

Modular arithmetic

Discrete logarithm is hard

But other operations such as addition, subtraction, multiplication, and division are easy

Divide by d is the same as multiply the inverse of d

- Example: $x \equiv 1/3 \ mod \ 11$ => find x such that $3x \equiv 1 \ mod \ 11$
- x can be found using the extended Euclidean algorithm

Example of a discrete logarithm problem

Example: given $p = 11$, $g = 2$ and $x = 6$, what is a?

a	1	2	3	4	5	6	7	8	9	10
g^a mod p	2	4	8	5	10	9	7	3	6	1

Cost of brute force attack = ?

$O(p)$, given a prime p and a generator g

What if p is not prime or g is not a generator?

RSA cryptosystem

RSA cryptosystem

For public-key encryption, digital signatures, key exchange

The **RSA problem** is no harder than the **Integer factorization problem**, but the opposite may not be true.

Factoring is believed to be neither P nor NP-complete.


Ron Rivest, Adi Shamir, Leonard Adleman

RSA cryptosystem

RSA cryptosysem relies on a **trapdoor permutation**

Trapdoor: a secret allows efficiently inverting a function

One-way trapdoor: without the trapdoor, the function is hard to invert

Such one-way trapdoor functions create *computational gaps* in public-key cryptography

How to use a trapdoor for encryption?

Public-key encryption

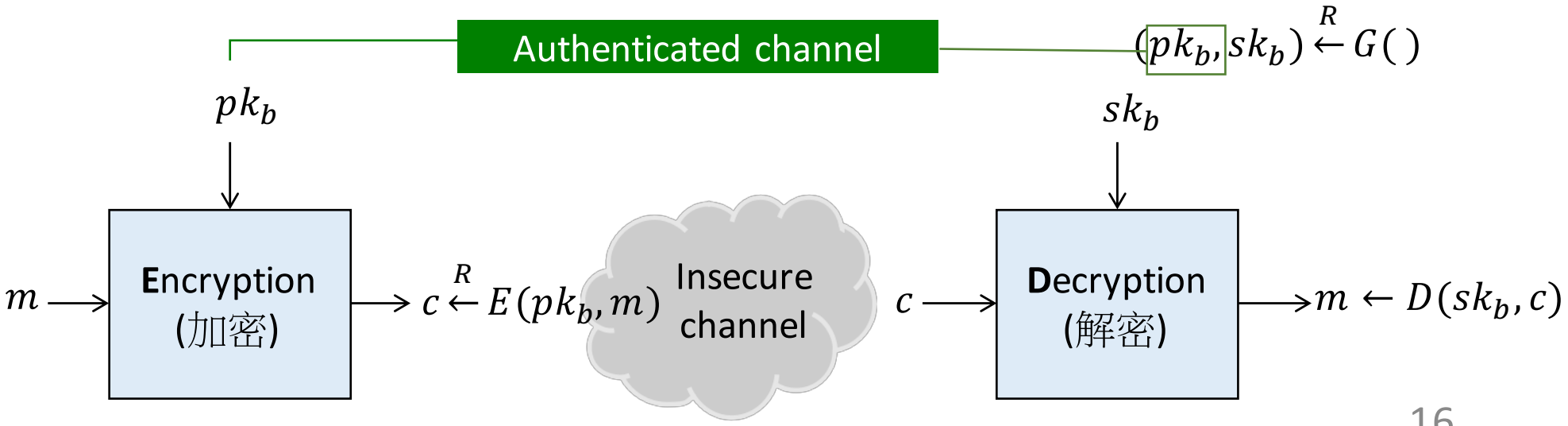
Bob has a public/private key pair (pk_b, sk_b).

Assume an **authenticated channel** to publish pk_b

- Authenticated channel: msgs can't be modified but can be overheard

Bob keeps his own private key sk_b in secret, so only Bob can decrypt c, and only Alice and Bob know m.

Can be combined with symmetric encryption to efficiently encrypt longer messages



(Textbook) RSA encryption

Setup and key generation $G(\,)$:

$n = pq$, where p and q are two large primes

Choose e s.t. e and $\phi(n) = (p-1)(q-1)$ are co-primes

- $\phi(n)$ is Euler's totient function
- $\phi(n) =$ | positive integers $\leq n$ & are co-prime to n |

Compute $d = e^{-1} \pmod{\phi(n)}$ — (*)

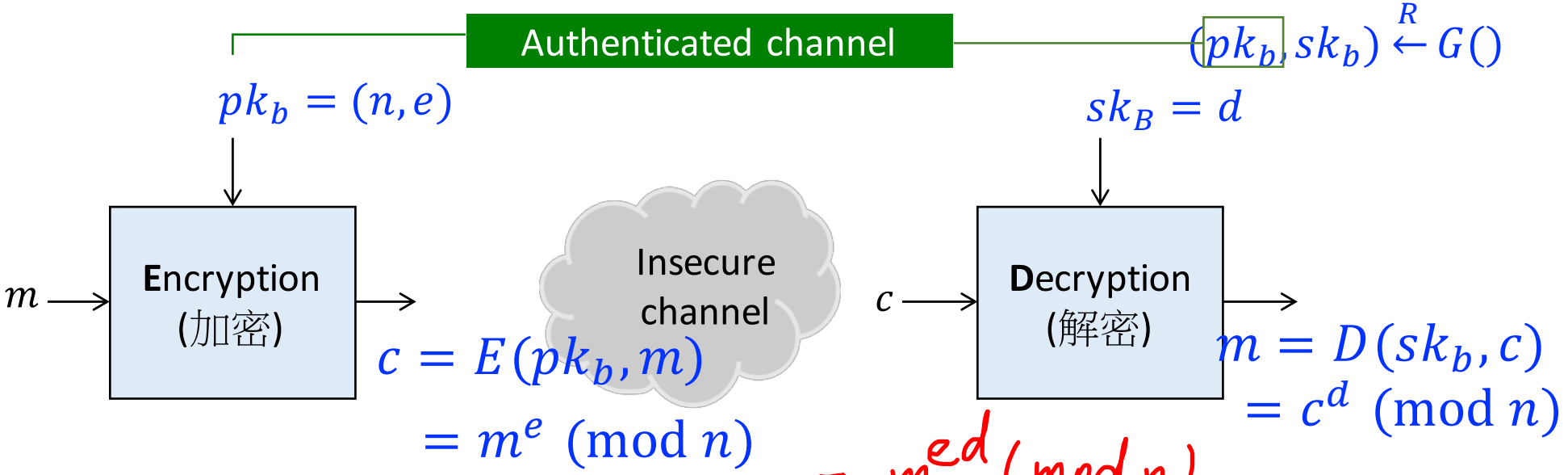
Bob's public key: $pk_b = (n, e)$

Bob's private key: $sk_b = d$

$de \equiv 1 \pmod{\phi(n)}$

$\Rightarrow de = 1 + k\phi(n)$

(Textbook) RSA encryption



Proof of correctness

Euler's theorem: $a^{\phi(n)} \equiv 1 \pmod n$ when a and n are co-primes

$= m^{ed} \pmod n$

by (*) $= m^{1+k\phi(n)} \pmod n$

$= m \cdot (m^{\phi(n)})^k \pmod n$

$= m \cdot 1^k \pmod n$

$= m \pmod n$

Attacking textbook RSA

Can the adversary recover the plaintext?

Known attacks

- Deterministic encryption
- Small n
- Small e
- Malleable
- Repeated p, q
- Hastad's Broadcast Attack
- …

Dan Boneh. Twenty Years of Attacks on the RSA Cryptosystem.
https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf

Deterministic Encryption

Same plaintext produces same ciphertext

It leaks information

The attacker can encrypt a lot of messages by itself (why?) and compare the ciphertext

Solution: randomized padding

- Optimal Asymmetric Encryption Padding (OAEP)

Malleable

In textbook RSA, $E(pk, x) * E(pk, y) = E(pk, x * y)$

(n, e) (n, e)

m_1 m_2

Example:

$$c_1 = x^e \pmod n$$
$$c_2 = y^e \pmod n$$
$$c_1 c_2 = ?$$

$(xy)^e \pmod n$

$= E(pk, xy)$

Pros? Cons?

Solution: randomized padding

- Optimal Asymmetric Encryption Padding (OAEP)

Small n $= p \times q$

Factoring algorithms

- Trial Division
- Pollard's P - 1 Method
- Quadratic Sieve
- Number Field Sieve

Tools

- factordb
- factor
- YAFU

Small e

What happen if $m^e < n$?

$c = m^e \pmod n$

If $m^e < n$, then $m = c^{1/e}$

Easily decrypted by taking the *e*th root of c

Repeated p, q

Simply compute the GCD!

https://factorable.net/

- N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, “Mining your Ps and Qs: detection of widespread weak keys in network devices,” *USENIX Security*, 2012.

Håstad's broadcast attack

Alice A wants to encrypt and send the same message m to three people P1, P2, P3

Their public keys are as follows, and e = 3

- P1: (e, n1)
- P2: (e, n2)
- P3: (e, n3)

So the ciphertexts are:

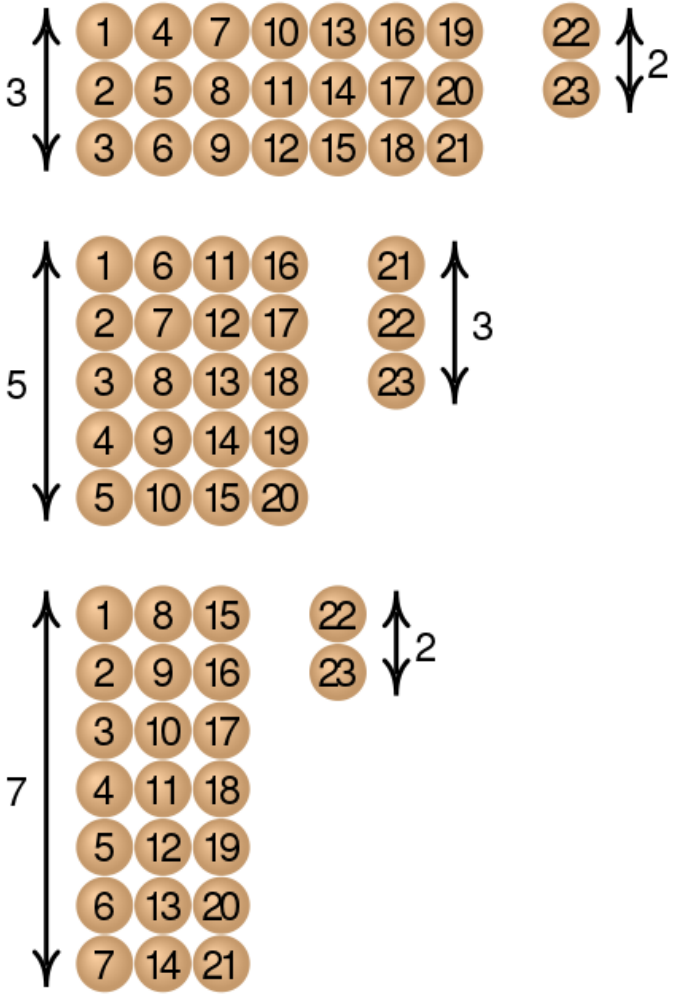
- $c_1 = m^3 \pmod{n_1}$
- $c_2 = m^3 \pmod{n_2}$
- $c_3 = m^3 \pmod{n_3}$

What can go wrong?

Chinese Remainder Theorem

CRT can be used to solve this type of congruence equations

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$



Håstad's broadcast attack

Given the equations:

- $c_1 = m^3 \pmod{n_1}$
- $c_2 = m^3 \pmod{n_2}$
- $c_3 = m^3 \pmod{n_3}$

We can obtain $C \equiv m^3 \pmod{n_1 n_2 n_3}$

Since $m^3 < n_1 n_2 n_3$ (why?), then $m = C^{1/3}$

Real-world vulnerability: Not-so-random primes in RSA

2017/10: ROCA (CVE-2017-15361), can infer private keys from public keys

“若要破解512位元長度的RSA金鑰只需花費CPU兩小時，成本只要0.6美元，而破解1024位元則需要97天，成本介於40到80美元之間，破解2048位元的RSA金鑰則需140.8年的CPU，成本最高，需要花費2萬至4萬美元。CRoCS以AWS執行攻擊為例，顯示計算出1024位元RSA私鑰的價格為76美元，算出2048位元私鑰則需4萬美元。”


新聞

英飛凌TPM晶片爆安全漏洞，Google、微軟忙修補

CRoCS發現英飛凌的TPM韌體有一演算法漏洞，可能產生脆弱的RSA金鑰，若駭客知公鑰，可能因此計算出私鑰，影響英飛凌自2012年以來推出的TPM晶片，包括英飛凌、Google、微軟、聯想、HP、富士通等業者已開始修補。

文/ 陳曉莉 | 2017-10-17 發表

讚 4.7 萬 按讚加入iThome粉絲團 讚 136 分享

ElGamal Threshold Decryption

ElGamal encryption

Setup and key generation

Public values: large prime p, generator g

Bob picks random b as his private key $sk_B = b$

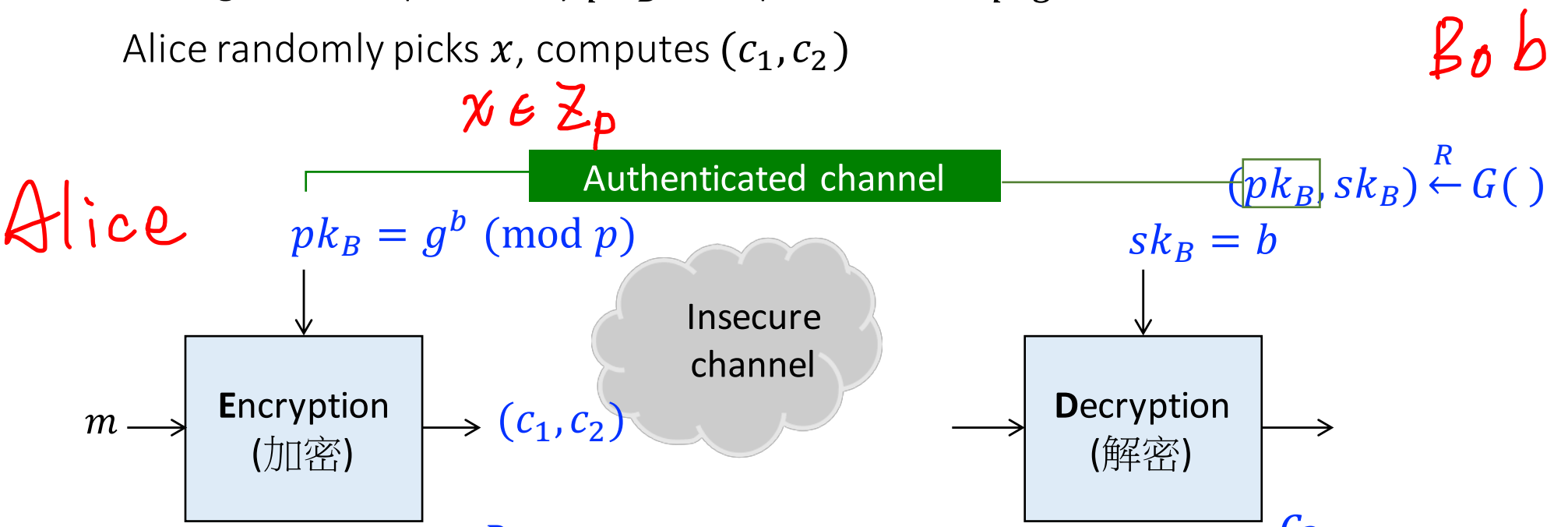
Bob's public key $pk_B = g^b \pmod p$

ElGamal encryption

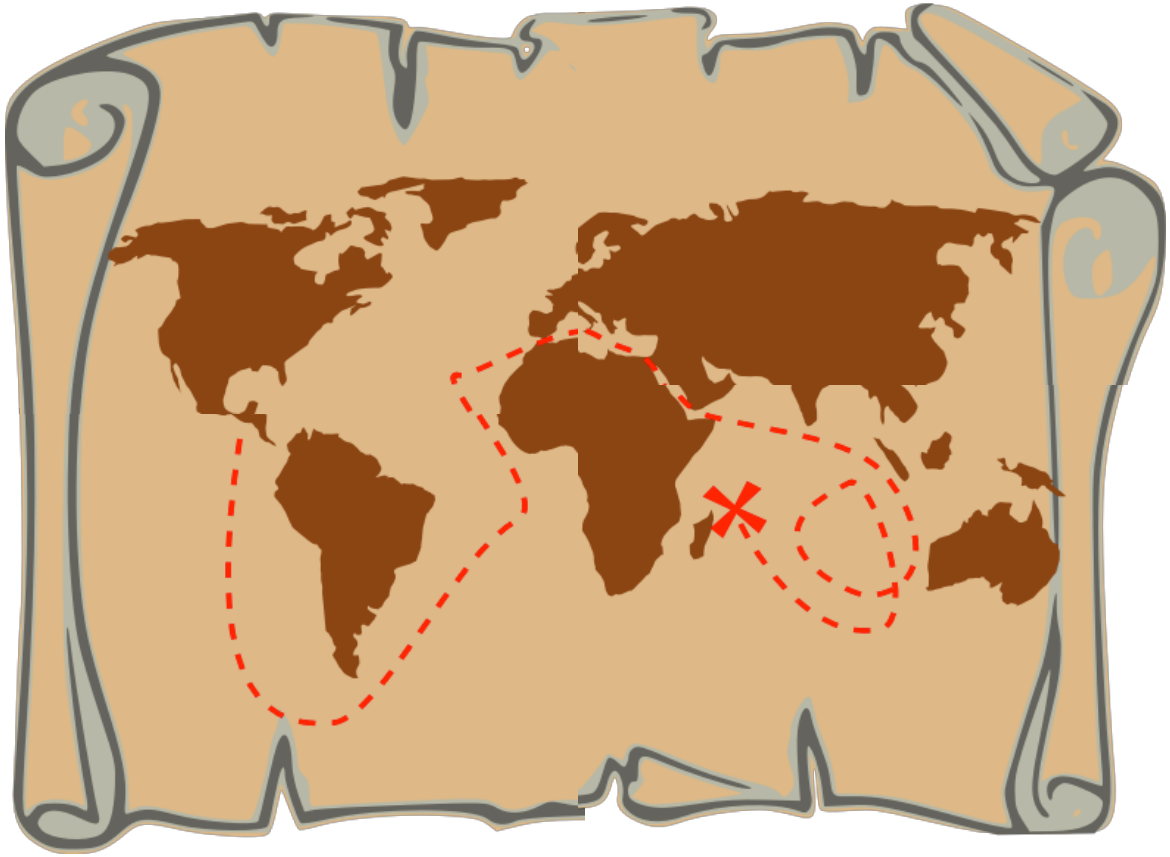
Alice wants to send message m to Bob:

Alice gets Bob's public key pk_B, and public values p, g

Alice randomly picks x, computes (c_1, c_2)



How to share a secret (in ancient times)



Goal: 把藏寶圖分給四個人，只有當四人合作才能重建地圖找到寶藏

直接剪成四份？Not secure!

Secret sharing

Secret = a digital document, a cryptographic key, missile launching code, …

要如何把秘密資料**D** 分給**n**個人（第**i** 個人拿到D_i），只有當**n** 個人合作才能重建這個秘密？

A simple approach:

- Randomly selects D_1, …, D_{n-1}
- Let $D_n = D \oplus D_1 \oplus \ldots \oplus D_{n-1}$

Example: Given D = 0010 1110, n = 3

- D_1 = 1000 1100
- D_2 = 0011 1101
- D_3 = 1001 1111

  奇數個 1 XOR = 1

  偶數個 1 XOR = 0

Threshold secret sharing

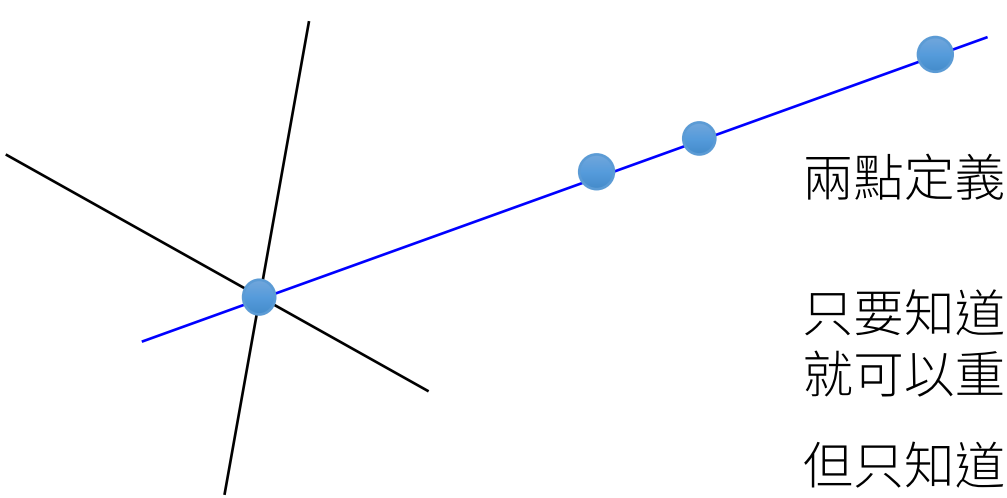
要如何把秘密資料D分給n個人（第i 個人拿到D_i），至少有 t 個人合作才能重建這個秘密？

- A (t, n)-threshold scheme

Any idea? Is it efficient?

Shamir's Secret Sharing

Adi Shamir, "How to share a secret", Communication of ACM 22 (11): 612-613, November 1979.



兩點定義一條直線（一次方程式）

只要知道直線上的任兩點
就可以重建這條線

但只知道一個點是不夠的

Main idea: t points uniquely define a polynomial of degree $t - 1$

Shamir's Secret Sharing

要如何把秘密資料D分給n個人（第i 個人拿到D_i），至少有 t 個人合作才能重建這個秘密？

Shamir's Secret Sharing

- 任意造出一個$t-1$次多項式 $F(x) = D + a_1x + \ldots + a_{t-1}x^{t-1}$
- Let $D_i = (i, F(i) \bmod p)$ for all i = 1, …, n, where p is a prime

Example: Given D = 5, t = 3, n = 4, p = 101

- $F(x) = D + a_1x + a_2x^2$
  $= 5 + 3x + 2x^2$ (a_1, a_2 are selected at random)
- D1 = ?
- D2 = ?
- D3 = ?
- D4 = ?

Shamir's Secret Sharing: Secret reconstruction

Interpolation (內插法) using Lagrange polynomials

1. Suppose we obtain t secret shares in a (t, n)-threshold scheme:

$$(x_1, y_1), \ldots, (x_t, y_t)$$

2. Reconstruct *F(x)*, where $L_j(x) = 1$ if $x = x_j$, and $L_j(x) = 0$ if $x = x_m$, $m \neq j$:

$$F(x) = \sum_{j=1}^{t} y_j L_j(x)$$

3. $L_j(x)$ can be constructed as:

$$L_j(x) = \prod_{\substack{1 \le m \le t \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_1)}{(x_j - x_1)} \cdots \frac{(x - x_{t-1})}{(x_j - x_{t-1})} \frac{(x - x_t)}{(x_j - x_t)}$$

ElGamal threshold decryption

Practice: How to combine ElGamal encryption and Shamir's secret sharing?

Digital Signatures

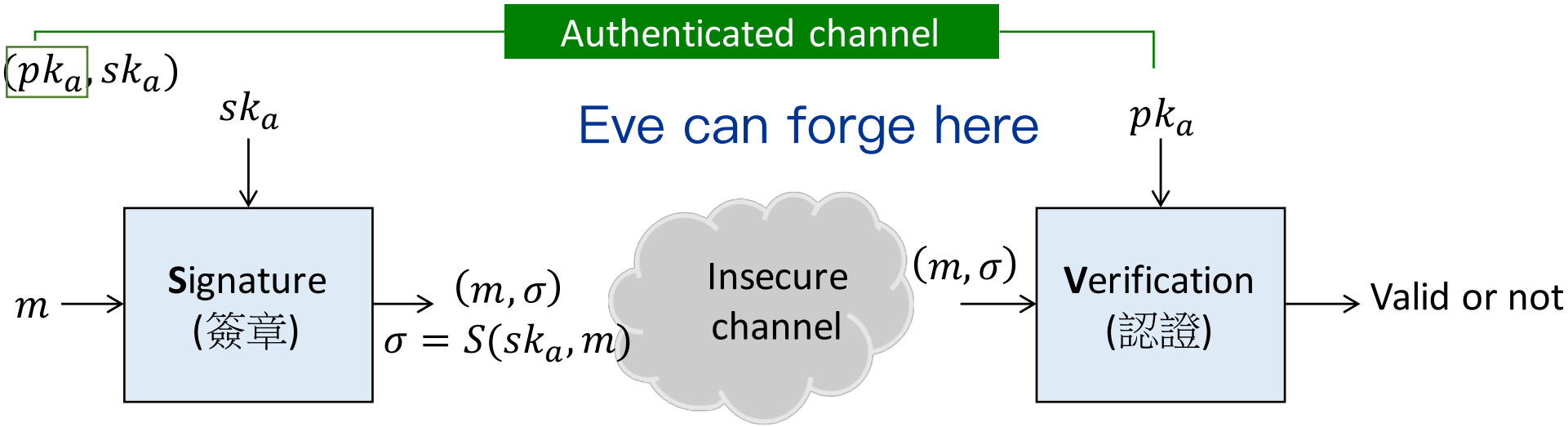
Digital Signatures

Alice has a public/private key pair (pk_a, sk_a)

Only Alice can create this signature (non-repudiation)

Eve can also verify this signature, but any modification will be detected

Can be combined with hash to efficiently sign longer messages



(Textbook) RSA signature

Setup and key generation:

$n = pq$, where p and q are two large primes

Choose e s.t. e and $\phi(n) = (p-1)(q-1)$ are co-primes

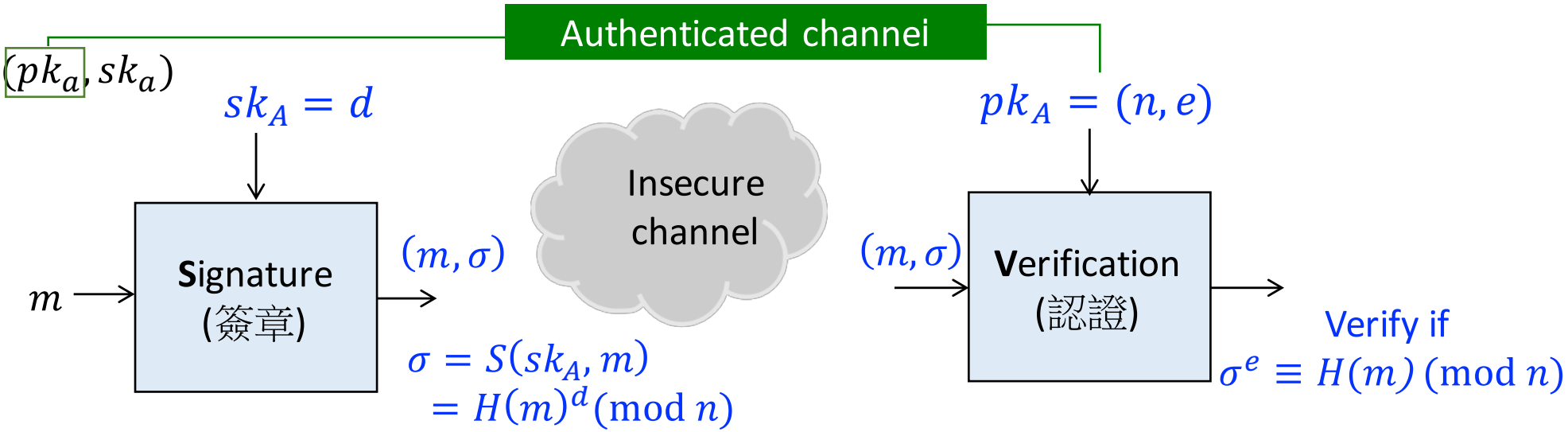
- $\phi(n)$ is Euler's totient function
- $\phi(n) =$ | positive integers $\leq n$ & are co-prime to n |

Compute $d = e^{-1} \pmod{\phi(n)}$

Alice's public key: $pk_A = (n, e)$

Alice's private key: $sk_A = d$

(Textbook) RSA signature



Note: we usually sign the message hash, instead of the message itself!

Proof of correctness

Euler's theorem: $a^{\phi(n)} \equiv 1 \pmod n$ when a and n are co-primes

Attacking textbook RSA Signature

Can the adversary forge signatures?

Known attacks

- Small n
- Malleable
- Bleichenbacher's small exponent attack
  - With vulnerable PKCS #1 v1.5 (padding) implementation
  - Real-world example, Mozilla library NSS: https://www.mozilla.org/en-US/security/advisories/mfsa2014-73/

Other signature schemes

DSA = Digital Signature Algorithm

ECDSA = Elliptic Curve Digital Signature Algorithm

A variant of the ElGamal signature scheme

Key Exchange

Key distribution and management

Many security mechanisms rely on the use of keys and assume keys are properly distributed and managed

- E.g., symmetric encryption, asymmetric encryption

Key distribution: how to deliver keys to parties who wish to exchange data securely

Key management: how to securely manage (including distribute, store, update, revoke, etc.) keys, possibly with a large number of keys and users

The n^2 key distribution problem

金鑰太多很難管理: Consider n parties, how can any pair of them establish a secret key?

- Need $O(n^2)$ keys
- Each user must store $n - 1$ secret keys
- Key management is challenging

建立安全通道不容易、要建一堆很耗資源

- Using symmetric-key crypto requires secure (i.e. secret and authentic) channel to set up shared secret key
- A newly joined user needs to setup a secure channel with every existing user

Solutions to n^2 key distribution problem

We will see two general solutions to the n^2 key distribution problem:

1. Use symmetric crypto: a trusted key distributed center (next week)
2. Use public-key crypto: Diffie-Hellman key agreement and certificate authorities

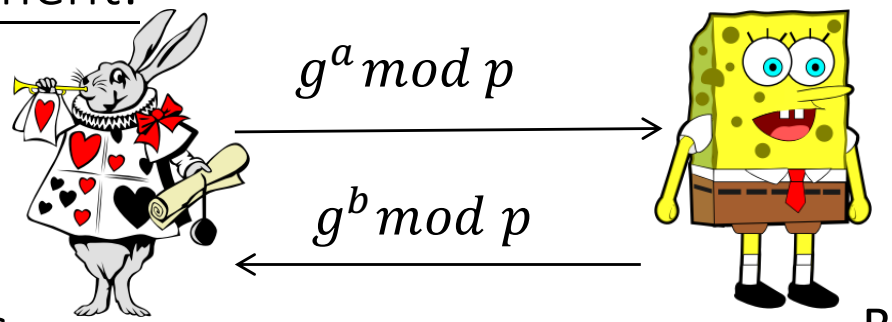
Diffie-Hellman key exchange

Setup:

Public values: large prime p, generator g

Alice picks a secret a, and Bob picks secret b

Key agreement:

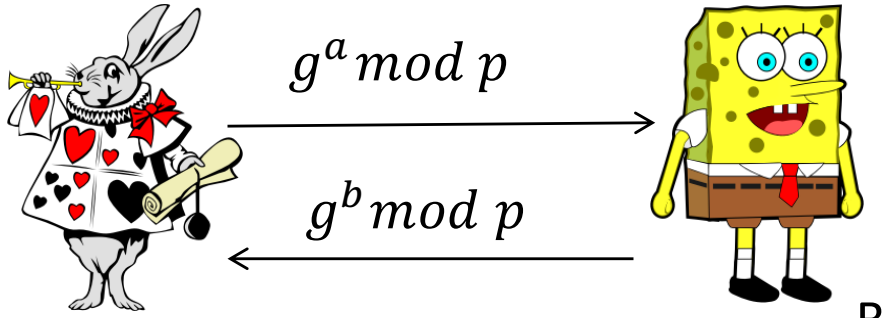


Alice computes
$(g^b)^a = g^{ab} mod\ p$

Bob computes
$(g^a)^b = g^{ab} mod\ p$

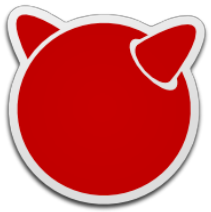
Alice and Bob can then use $g^{ab} mod\ p$ to derive their shared key

Security analysis of Diffie-Hellman key exchange



Alice computes
$(g^b)^a = g^{ab}\ mod\ p$

Bob computes
$(g^a)^b = g^{ab}\ mod\ p$



A passive attacker who only eavesdrops on the communication cannot compute g^{ab} due to the hardness assumption of discrete logarithm problem

But how about an active attacker?

An active attacker who participates in the protocol can perform **man-in-the-middle (MitM) attack**

MitM attack against DH key exchange

Setup:

Public values: large prime p, generator g

Alice picks a secret a, and Bob picks secret b

Man-in-the-Middle attack against DH key agreement:

Eve impersonates Alice to Bob and Bob to Alice

$g^a \bmod p$ →

← $g^m \bmod p$

$g^m \bmod p$ →

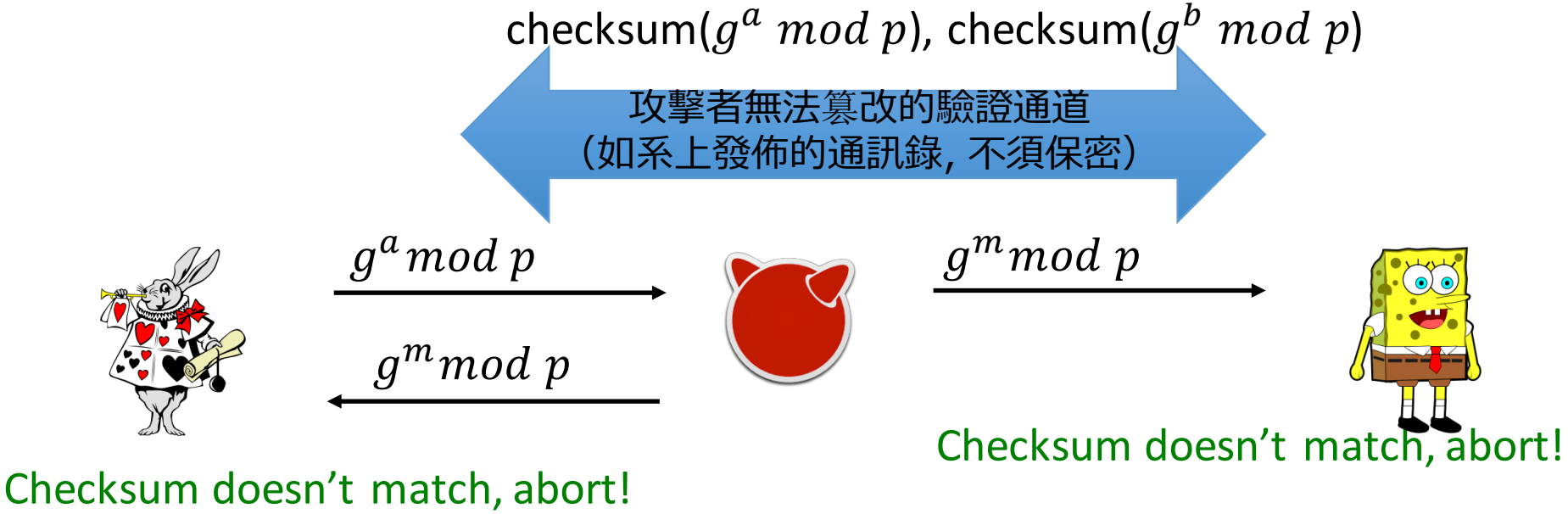
← $g^b \bmod p$

$g^{am} \bmod p \neq g^{bm} \bmod p$

Eve can decrypt the comm. between Alice and Bob

Solution to MitM attack

Distribute "checksum" of $g^a \ mod \ p$ and $g^b \ mod \ p$ via authenticated channels, so that tampering can be detected



Attacking DH in practice: Logjam attack

David Adrian et al. "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice." ACM Computer and Communications Security, 2015.

- A man-in-the-middle downgrades connections to "export-grade" Diffie-Hellman **DHE_EXPORT**
- Downgraded TLS to legacy 512-bit security
- A week-long precomputation for a 512-bit prime, and then can compute individual discrete logs in a median of 70s
- Can practically break TLS
- https://weakdh.org/

Attacking DH in practice: Logjam attack

Exploit two weaknesses in many DH implementation:

1. Support weak, obsolete export-grade crypto (DHE_EXPORT)
   - DHE_EXPORT is exactly the same as DHE except using a 512-bit group
   - Many libraries and servers support DHE_EXPORT for backward compatibility
   - This **was** considered safe because Modern TLS clients don't accept DH_EXPORT
   - 8.4% of top one million HTTPS sites
2. Use hard-coded, widely shared Diffie-Hellman parameters
   - Two 512-bit DH groups used by > 92% of the vulnerable servers

Source	Popularity	Prime

Apache	82%	`9fdb8b8a004544f0045f1737d0ba2e0b`
		`274cdf1a9f588218fb435316a16e3741`
		`71fd19d8d8f37c39bf863fd60e3e3006`
		`80a3030c6e4c3757d08f70e6aa871033`

mod_ssl	10%	`d4bcd52406f69b35994b88de5db89682`
		`c8157f62d8f33633ee5772f11f05ab22`
		`d6b5145b9f241e5acc31ff090a4bc711`
		`48976f76795094e71e7903529f5a824b`

(*others*)	8%	(463 distinct primes)

Table 1: **Top 512-bit DH primes for TLS.** 8.4% of Alexa Top 1M HTTPS domains allow DHE_EXPORT, of which 92.3% use one of the two most popular primes, shown here.

Quick summary

Cryptography reduces protection of data to protection of keys

Symmetric cryptography

- + Fast
- - Need to ensure secrecy and authenticity of symmetric keys
- Commonly used to protect bulk data

Asymmetric cryptography

- - Slow
- + Only need to ensure authentic public key
- Commonly used to establish symmetric keys or protect small data

Common usage: use asymmetric cryptography to establish a shared secret, then switch to symmetric cryptography.

Public Key Infrastructures

Digital certificates & certificate authorities

X.509 PKI

Real-world issues with PKIs/CAs

CNSMessenger

瞭解到市面上的通訊軟體有被竊聽或被伺服器側錄的疑慮，班上的同學開發了一款能支援點對點的加密通訊軟體。此軟體使用public-key cryptography建立shared secret key。

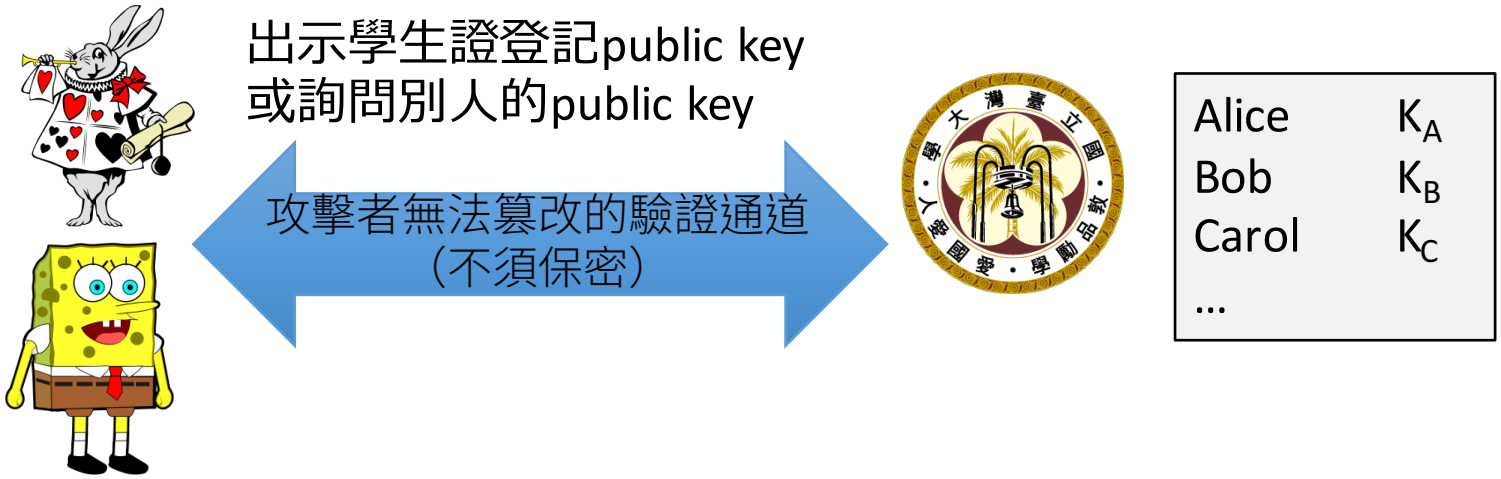
Bob要怎麼確認拿到的public key真的是Alice的？

Alice's public key = 95 E1 A8 F2 6B 07 DA E8 89 1E D0 07 DF C8 A2 75 F0 E9 64 18 25 FF E2 03 C1 16 A7 87 7F CA 88 C7 B9 A5 FD C1 6C FF 16 7B 05 97 63 32 A0 C5 A1 1E 98 7F 12 BD 5C 7B A3 15 04 05 36 BD D5 7E C0 AF 5B EE C0 D5 8C 32 F8 D7 41 24 83 A5 A9 28 96 79 76 2C B6 61 14 72 E1 00 2A 6C 22 1B 85 79 A3 CC 21 AE 52 4D EB 76 8A 15 6E A5 DA 9A 71 0D C7 07 F6 71 AE CD 4D 87 D8 E3 D3 DF CE C9 14 FB 2E 84 C0 3D 1F 64 F7 1C AA 51 2C 81 F0 5F 93 68 71 55 14 BA 59 AC 20 3A 03 E4 50 C3 07 D9 74 4B 5D 20 6E AE 83 FF 7D 6E 39 9C 3F CF BF B6 58 3F 41 EA C4 E8 52 F7 94 3D DA 8C B7 72 C9 3B 79 EC 6E E8 58 1F BB 90 82 BF E0 34 79 B1 D2 B1 33 45 32 0D 96 DD B8 7D 71 EB 51 98 94 76 68 02 EE 68 B4 03 AC 3E 7F 9A 72 B5 F9 59 CA A9 8A E4 E7 A4 5A 8D 7F D5 DA 96 D9 43 CA F4 9B 83 B2 F3 1B 56 25 9F

Distributing PK via online trusted 3rd party

同學必須先去系辦出示學生證認證身份，登錄自己的public key。要知道別人的public key也必須去系辦詢問。

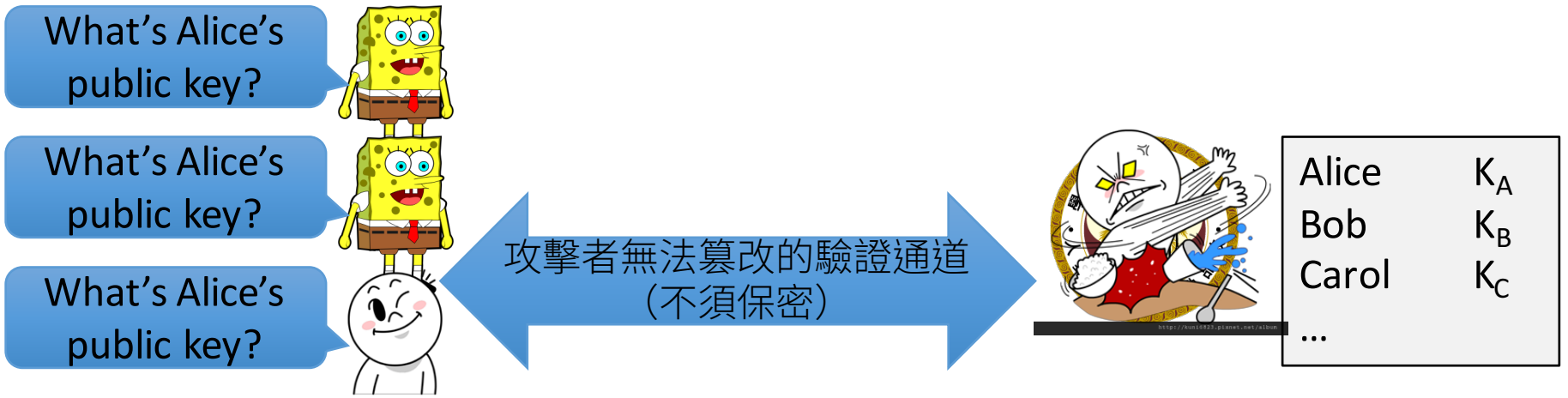
- 假設系辦是trusted 3rd party



Distributing PK via online trusted 3rd party

軟體相當成功，全系的同學都想使用。但每個人每次使用前，都要先去系辦詢問最新的public key，系辦不勝其擾。

有什麼辦法能減輕系辦的負擔？



Distributing PK via digital certificates

A digital certificate **binds a user's ID to its public key.**

Setup:

Assume a trusted Certification Authority (e.g., 系辦)

CA's public/private key pair = (pk_{CA}, sk_{CA})

pk_{CA} is published via an authenticated channel

Upon a public-key certification request $(pk_A, Alice)$:

CA verifies Alice's identity and then generates $cert_A$ by signing $(pk_A, Alice)$ using its private key

$$cert_A = [S(sk_{CA}, (pk_A, Alice)), (pk_A, Alice)]$$

$$cert_B = [S(sk_{CA}, (pk_B, Bob)), (pk_B, Bob)]$$

Distributing PK via digital certificates

A digital certificate **binds a user's ID to its public key.**

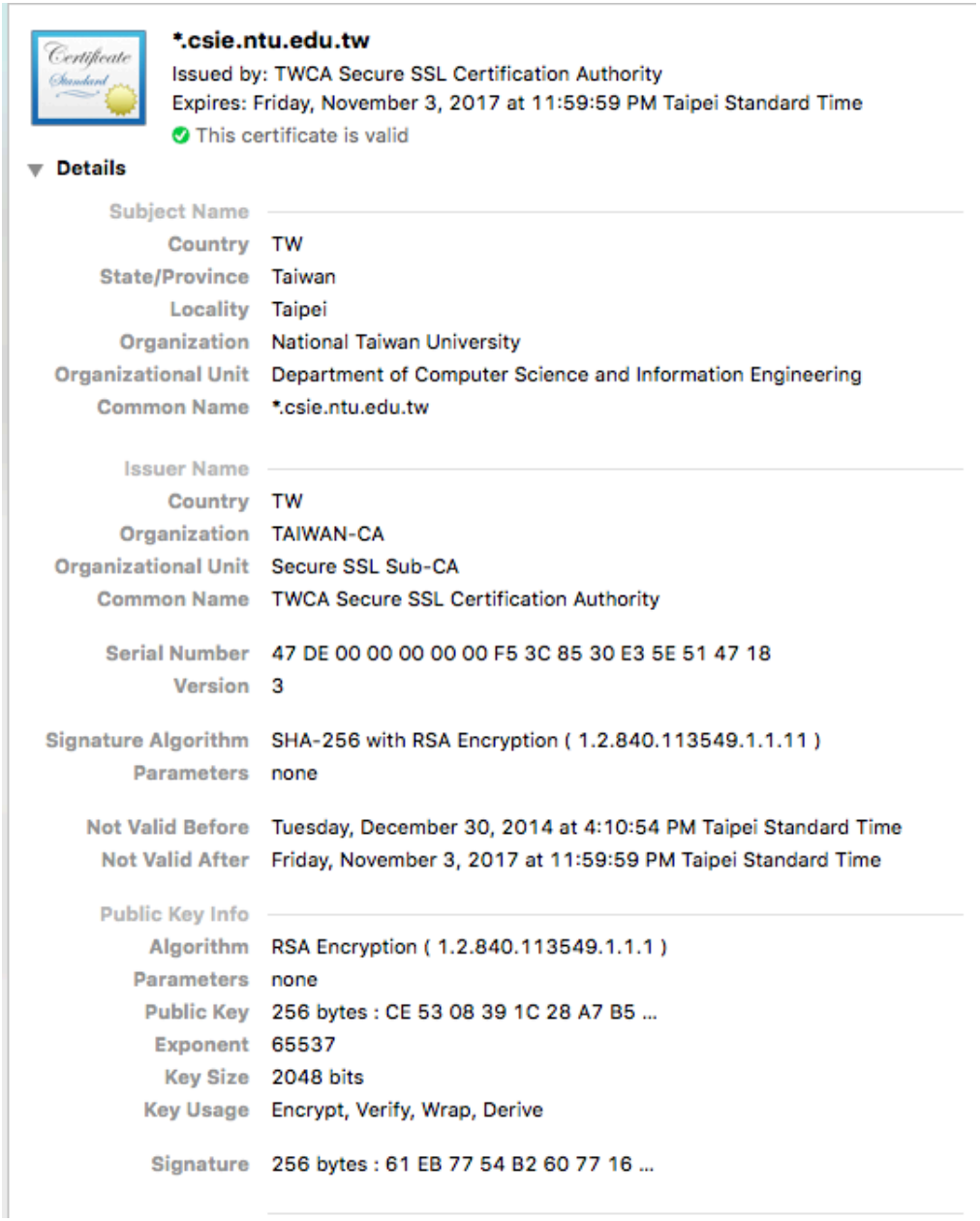
$$Data = (pk_A, Alice)$$
$$cert_A = [S(sk_{CA}, Data), Data]$$

K_A is Alice's public key, and here is a proof.

I trust the CA correctly validates Alice's identity and generates the certificate and I trust the CA's public key is pk_{CA}, so I trust that Alice's public key is pk_A.

X509v3 Certificate

Version	Serial no.	Sig. algo.
Issuer		
Validity	Not Before	Not After
Subject		
Subject Public Key Info		
	Algorithm	Public Key
X509 v3 Extensions		
	CA Flag, EV, CRL, etc.	
Signature		



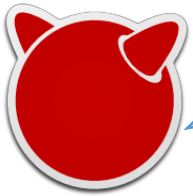
R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL landscape: a thorough analysis of the x. 509 PKI using active and passive measurements," in *Proceedings of ACM IMC*, 2011.

Distributing PK via digital certificates

軟體紅到全世界，國內外的網友和網站都想使用，但每個人願意相信的Certification Authority不見得都一樣。這下子使用者（人/網站）要怎麼確認彼此的public key為真？

I don't know these CAs… Which one is mail.google.com's ***real*** public key?

mail.google.com's public key is pk_G, and $cert_G$ is certified by CA1



mail.google.com's public key is pk_E, and $cert_E$ is certified by CA2

Public Key Infrastructures (PKI)

A PKI = a secure system to manage **digital certificates**

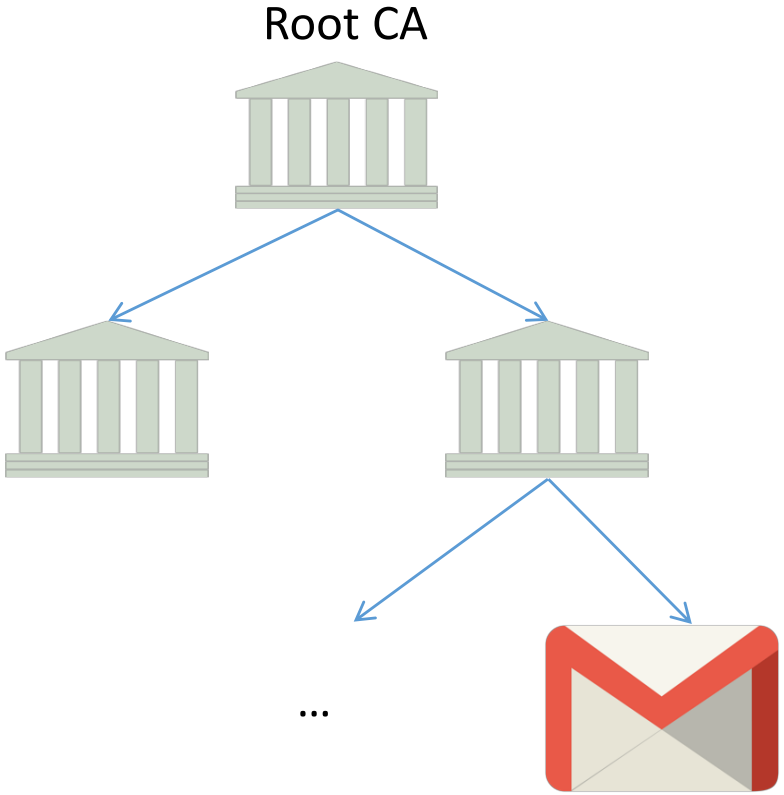
- Certificate issuance
- Certificate revocation

Many security protocols and applications rely on PKI: TLS, IPSec, S/MIME, DNSSEC, code signing…

X.509 is an ITU-T standard for a public key infrastructure (PKI)

How it is done today: X.509 Public Key Infrastructure (PKI)

A hierarchy of Certification Authorities



GeoTrust Global CA

Root certificate authority

Expires: Saturday, May 21, 2022 at 12:00:00 PM Taipei Standard Time

This certificate is valid

Google Internet Authority G2

Intermediate certificate authority

Expires: Sunday, January 1, 2017 at 7:59:59 AM Taipei Standard Time

This certificate is valid

mail.google.com

Issued by: Google Internet Authority G2

Expires: Tuesday, May 31, 2016 at 8:00:00 AM Taipei Standard Time

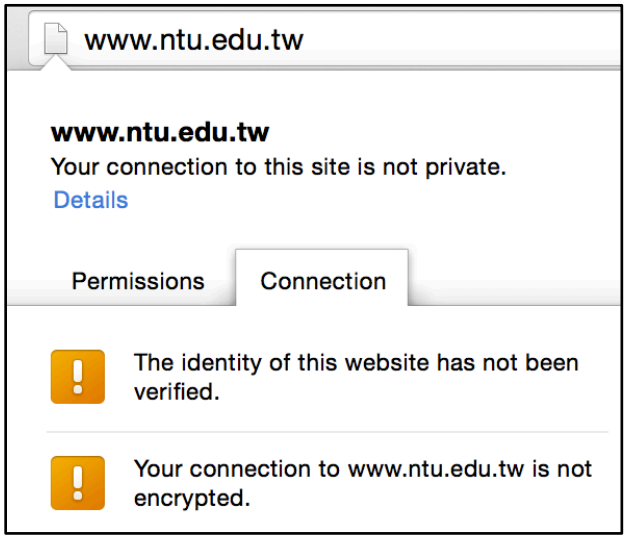
This certificate is valid

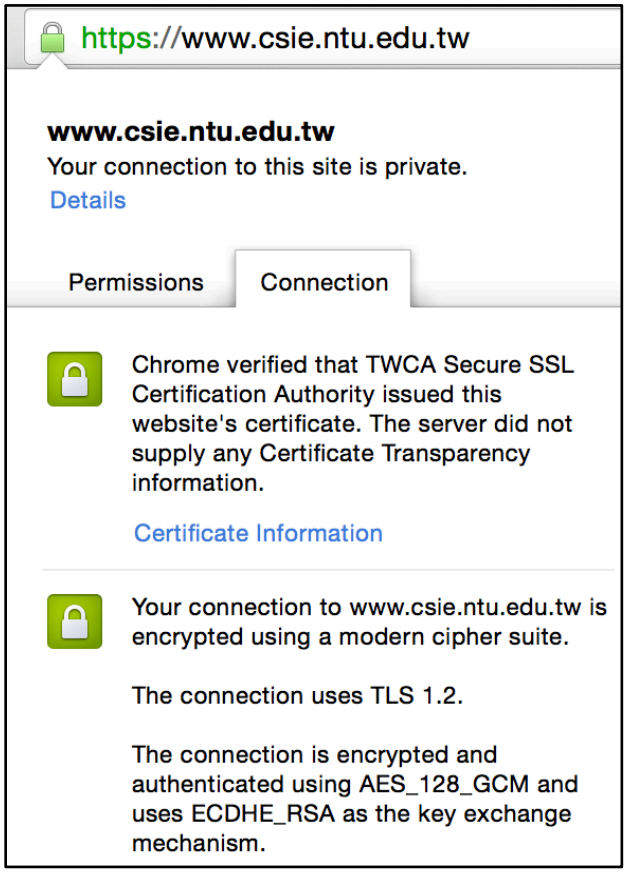
How it is done today: X.509 Public Key Infrastructure (PKI)

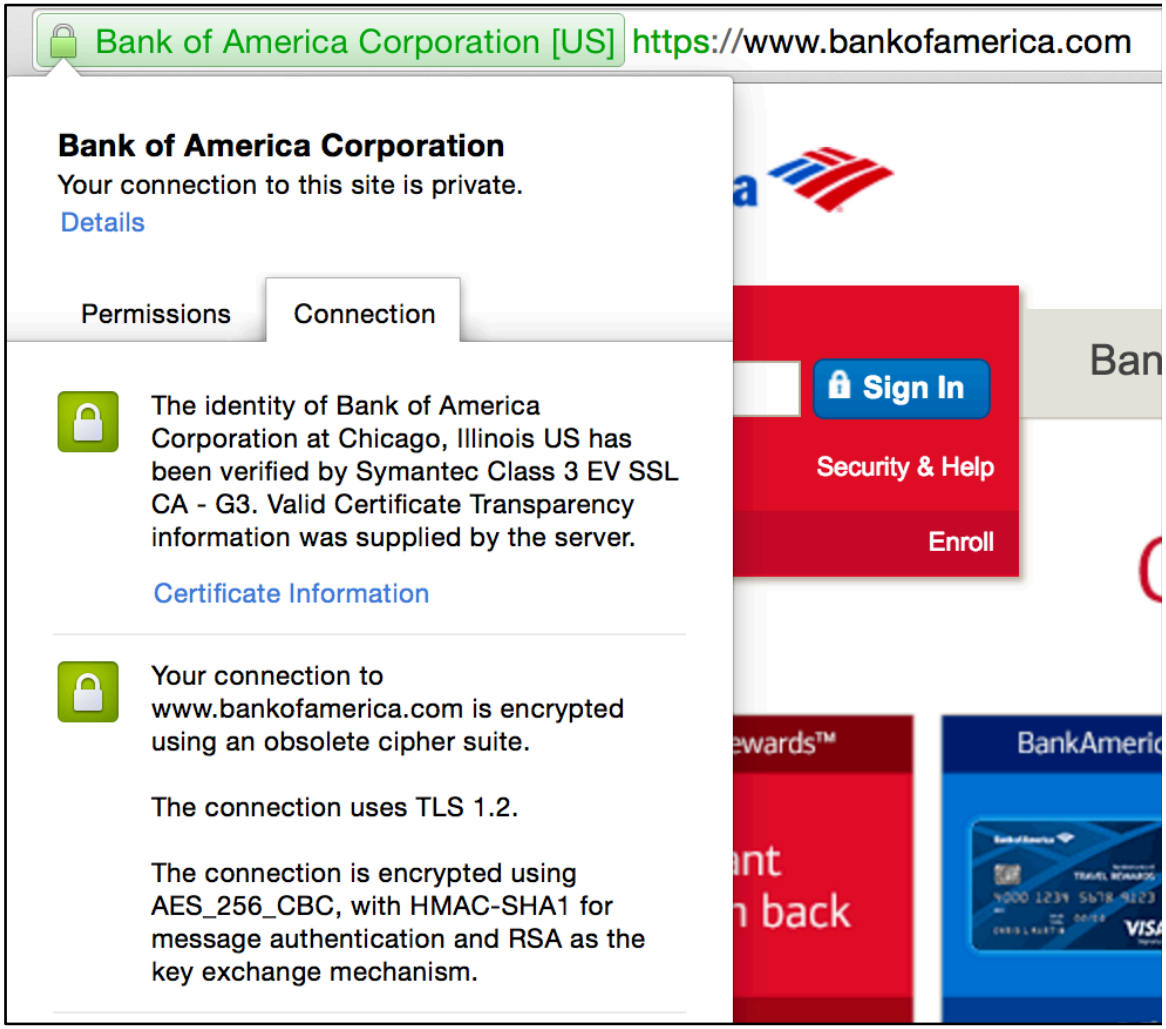
Root certificates: All your trust relationships online are reduced to trusting this list of root certificates

ePKI Root Certification Authority	certificate
Equifax Secure Certificate Authority	certificate
Equifax Secure eBusiness CA-1	certificate
Equifax Secure eBusiness CA-2	certificate
Equifax Secure Global eBusiness CA-1	certificate
Federal Common Policy CA	certificate
GeoTrust Global CA	certificate
GeoTrust Primary Certification Authority	certificate
GeoTrust Primary Certification Authority - G2	certificate
GeoTrust Primary Certification Authority - G3	certificate
Global Chambersign Root	certificate
Global Chambersign Root - 2008	certificate
GlobalSign	certificate
GlobalSign	certificate
GlobalSign	certificate
GlobalSign	certificate
GlobalSign Root CA	certificate
Go Daddy Class 2 Certification Authority	certificate
Go Daddy Root Certificate Authority - G2	certificate
Government Root Certification Authority	certificate

OS X Keychain


www.ntu.edu.tw
www.ntu.edu.tw
Your connection to this site is not private.
Details
Permissions
Connection
The identity of this website has not been verified.
Your connection to www.ntu.edu.tw is not encrypted.


https://www.csie.ntu.edu.tw
www.csie.ntu.edu.tw
Your connection to this site is private.
Details
Permissions
Connection
Chrome verified that TWCA Secure SSL Certification Authority issued this website's certificate. The server did not supply any Certificate Transparency information.
Certificate Information
Your connection to www.csie.ntu.edu.tw is encrypted using a modern cipher suite.
The connection uses TLS 1.2.
The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.


Bank of America Corporation [US]
https://www.bankofamerica.com
Bank of America Corporation
Your connection to this site is private.
Details
Permissions
Connection
The identity of Bank of America Corporation at Chicago, Illinois US has been verified by Symantec Class 3 EV SSL CA - G3. Valid Certificate Transparency information was supplied by the server.
Certificate Information
Your connection to www.bankofamerica.com is encrypted using an obsolete cipher suite.
The connection uses TLS 1.2.
The connection is encrypted using AES_256_CBC, with HMAC-SHA1 for message authentication and RSA as the key exchange mechanism.
Sign In
Security & Help
Enroll

Real-world issues with PKIs/CAs

Too many CAs

Rouge CAs or stolen certificates

- "Two code-signing certificates, stolen from two separate chip manufacturers (JMicron and Realtek) in Taiwan…"
- March 2011: "in an attack on a Comodo reseller, fake certificates were issued for mail.google.com, www.google.com, login.yahoo.com, login.skype.com, login.live.com, and addons.mozilla.org"

Certificate revocation

Too many CAs & root certificates

Pre-loaded into browser and/or OS

~150 root certificates from ~50 organizations

Roots certificates can authorize intermediate CAs

EFF's SSL Observatory reports that Microsoft IE and Mozilla Firefox trust 1482 different CA public keys held by 651 organizations located throughout the world

Any CA can issue an acceptable certificate for any site

Fake Certificates

Attacker uses a fake certificate that passes browser check

Who wants to do this?

- Organizations monitoring traffic
- Governments doing censorship
- Cloud providers
- …

How?

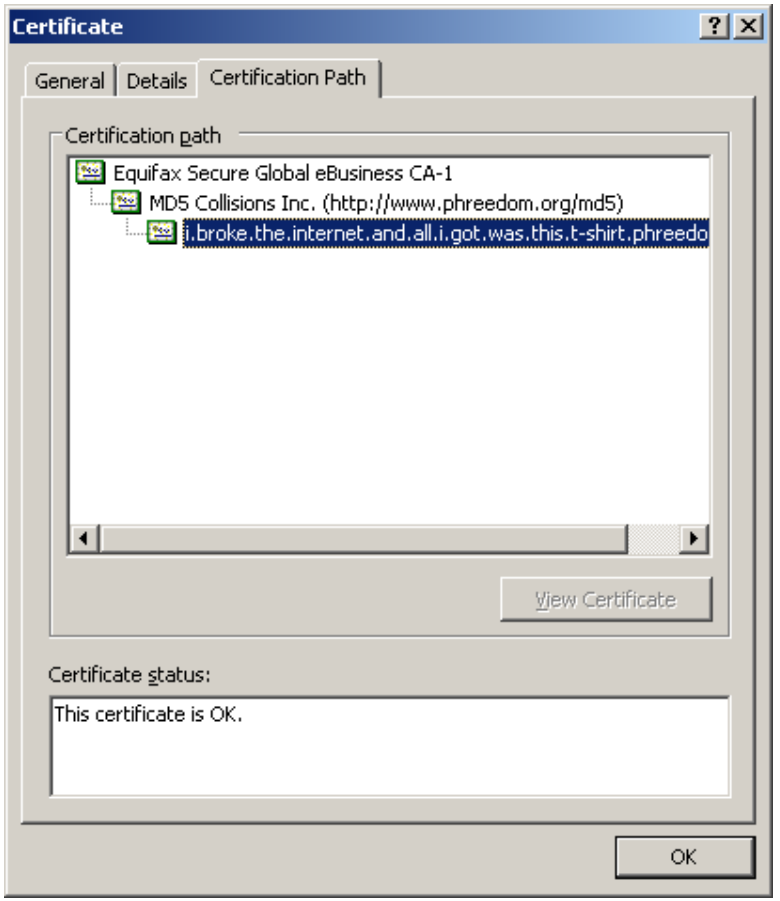
- Compromised CAs
- Compelled certificates
- Browser not checking correctly (perhaps on purpose)
- …

Faked certificates via hash collision

MD5 collisions can be found easily (complexity ~2^{18})

Example of real-world exploits: create rogue certificates

- Digital signatures sign the message digest, not the message itself
- Register a certificate for www.something.com and use this certificate for www.bank.com if H(Cert of something.com) = H(Cert of bank.com)

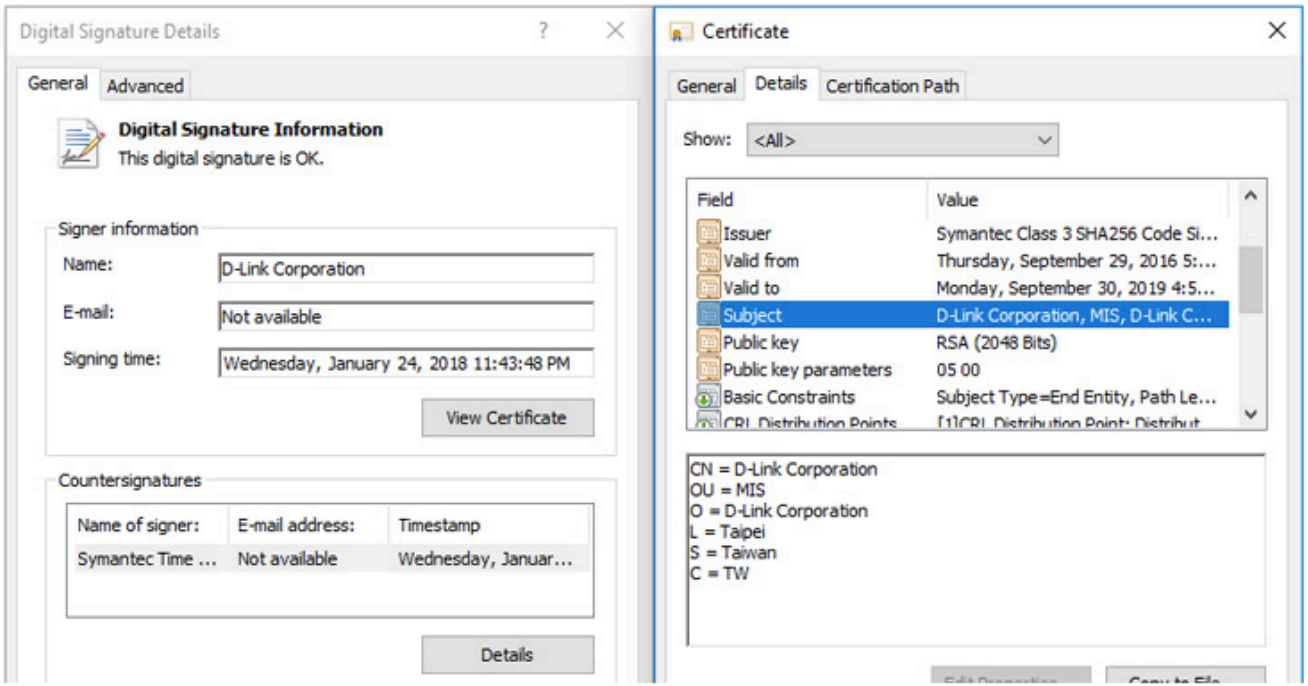


MD5 considered harmful today. http://www.win.tue.nl/hashclash/rogue-ca/
http://www.md5.net/md5-cracker/

Fake certificates via stolen keys

Stolen D-Link Certificate Used to Digitally Sign Spying Malware

July 09, 2018 Swati Khandelwal



https://thehackernews.com/2018/07/digital-certificate-malware.html

Fake certificates via compelled CAs

Google抨擊中國濫發數位憑證，假冒的憑證可攔截加密內容

http://www.ithome.com.tw/news/94784

Google指中國互聯網絡信息中心（CCNIC）發給埃及MCS公司的數位憑證，被安裝在執行SSL中間人流量管理的裝置，可攔截連至Google的加密流量。

由埃及的MCS中間憑證機構（intermediate Certificate Authority）發行，假冒Google網域的數位憑證

MCS的中間憑證則是由中國的根憑證機構CNNIC（中國互聯網絡信息中心）所發行

由CNNIC發行的憑證幾乎都會被所有的瀏覽器與作業系統視為可信賴憑證

CNNIC ROOT
Root certificate authority
Expires: Friday, April 16, 2027 3:09:14 PM Taipei Standard Time
This certificate is valid

Fake certificates via installing root CAs

Lenovo PCs ship with man-in-the-middle adware that breaks HTTPS connections [Updated]

Superfish may make it trivial for attackers to spoof any HTTPS website.

by **Dan Goodin** - Feb 20, 2015 12:36am CST

How the Nokia Browser Decrypts SSL Traffic: A "Man in the Client"

JANUARY 11, 2013 BY STEVE SCHULTZE

Efforts to handle CA trust issues

Certificate Transparency (and its variants)

- Chrome requires this for newly issued EV Certificates in 2015

HTTP Public Key Pinning (HPKP)

- RFC 4769
- Currently, HPKP is supported in Firefox and Chrome

DNS Certificate Authority Authorization (CAA)

- RFC 6844

Online Certificate Status Protocol (OCSP), OCSP stapling

- RFC 6960
- Supported by major browsers
- Chrome disables OCSP by default

Certificate Transparency

"An open framework for monitoring and auditing SSL certificates in nearly real time"

Maintain publicly verifiable and accessible logs that can be queried by users and domain owners

Transparency eases the problem of detecting fraudulent certificates

http://www.certificate-transparency.org/

HTTP Public Key Pinning (HPKP)

Allows a domain (server-side) to specify a whitelist of public keys for validating certificate chains

- The certificate chain must contain at least one public key in the whitelist

How can the domain tell the browser that it enables HPKP?

- Very similar to HSTS
- Option 1: Return HPKP header over a HTTPS connection
- Option 2: Add itself to the browser preload list

Option 1: Return HPKP header over an HTTPS connection
(Trust-On-First-Use: assume the first request is secured)

```
Public-Key-Pins:  max-age=2592000;
pin-
sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
pin-
sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
report-uri="http://example.com/pkp-report"
```

Option 2: Add itself to the browser preload list
(can protect the first request too)

```
{
  "name": "google",
  "static_spki_hashes": [
    "GoogleBackup2048",
    "GoogleG2",
    "GeoTrustGlobal"
  ],
  "report_uri": "http://clients3.google.com/cert_upload_json"
},
```

Issues with HPKP adoption

Very low adoption rate: Only 375 sites deploy HPKP as of August 2016 [1]

Might be disabled by major browsers

Why?

- Hard to do it right
- HPKP misconfiguration can be more devastating than HSTS

In case of privacy key lost or certificate change, sites may be unavailable for a long time

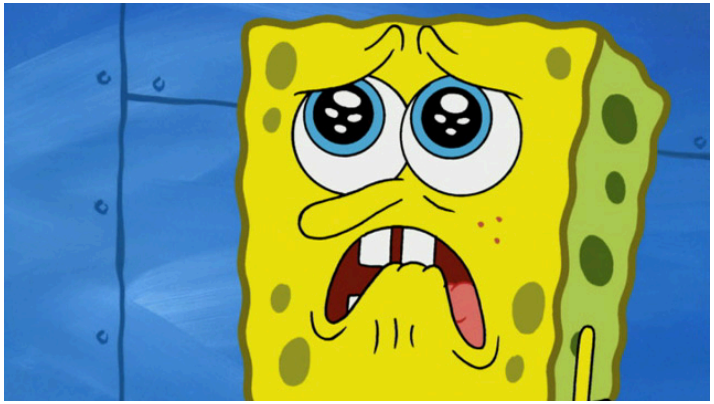
- Can be abused by attackers
- Workaround: HPKP requires at least one backup key

[1] https://scotthelme.co.uk/alexa-top-1-million-crawl-aug-2016/

鑰匙掉了怎麼辦

Bob的private key外洩了，因此他的數位憑證CertB也應該要即刻被廢止（revoke）。

要如何做到？



Certificate revocation

Certificate revocation list (CRL)

- A list of certificates that should be revoked
- Signed by the CA
- Challenge: how to distribute the list to users in an efficient and timely manner?
  - Query for a new CRL whenever receiving a certificate?
  - Periodic update?
  - Using short-lived certificates?

Certificate revocation

Online Certificate Status Protocol (OCSP)

- Bob requests Alice's CA to verify her certificate online
- Challenge:
  - CAs need to be online all the time
  - Increase communication latency

OCSP stapling

- Alice obtains a timestamped OCSP response and present it to Bob
- Challenge:
  - Low implementation rate