{ CSCI 6331 · 4331 | Lecture 8 }

Cryptography

Hoeteck Wee · hoeteck@gwu.edu

http://tinyurl.com/cryptogw/

Announcements

- Evaluation:

  10% In-Class/Piazza, 20% Final Presentation / Project

  30% Homework, 40% Final (Apr 25)

- Homework 4 out tonight, due Mar 21 (Wed) in class

Review: Arithmetic mod Primes

- Let p be a prime
- Notation: $\mathbb{Z}_\mathrm{p} = \{0, 1, 2, \ldots, \mathrm{p}-1\}$

  $\mathbb{Z}_\mathrm{p}^* =$ invertible elements in $\mathbb{Z}_\mathrm{p}$

  so $\mathbb{Z}_\mathrm{p}^* = \{1, 2, \ldots, \mathrm{p}-\mathrm{q}\}$

- Facts: $\mathbb{Z}_\mathrm{p}^*$ is cyclic, i.e.

  can write $\mathbb{Z}_\mathrm{p}^* = \{\mathrm{g}^0, \mathrm{g}^1, \mathrm{g}^2, \mathrm{g}^3, \ldots, \mathrm{g}^{\mathrm{p}-2}\}$

  g a generator

- Example. $\mathrm{p} = 5$, then 2 is a generator.

Diffie-Hellman Key Exchange

Q. How to distribute/share secret keys over public channels?

Diffie-Hellman Key Exchange protocol (warm-up).

1. Alice picks prime p and generator g.
2. Alice chooses $\mathrm{x} \leftarrow \mathbb{Z}_{\mathrm{p}}$ and sends g^{x} to Bob.
3. Bob chooses $\mathrm{y} \leftarrow \mathbb{Z}_{\mathrm{p}}$ and sends g^{y} to Alice.
4. shared key is g^{xy}.

Q. How do Alice and Bob compute the shared key?

— Alice sees x and g^{y}

— Bob sees y and g^{x}

Diffie-Hellman Assumption. shared key looks "random" to a passive adversary

— work with a subgroup of $\mathrm{Z}_{\mathrm{p}}^{*}$ of prime order

Public Key Encryption

Private vs Public Key Encryption

— private-key: same key to encrypt and to decrypt (symmetric)

— public-key: one key to encrypt, another to decrypt (asymmetric)

syntax. private-key encryption = three algorithms $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$

— key generation Gen – outputs a key pair $(\mathrm{pk}, \mathrm{sk})$

(public key pk and secret key sk)

— encryption Enc – input pk and a message m; output ciphertext $\mathrm{c} = \mathrm{Enc}_{\mathrm{pk}}(\mathrm{m})$

— decryption Dec – input sk and a ciphertext c; output plaintext $\mathrm{m} = \mathrm{Dec}_{\mathrm{sk}}(\mathrm{c})$

correctness. $\mathrm{Dec}_{\mathrm{sk}}(\mathrm{Enc}_{\mathrm{pk}}(\mathrm{m})) = \mathrm{m}$

security. as before, ciphertexts don't leak information about plaintext, even given multiple ciphertexts

Review: Arithmetic mod Composites

- Let $N = pq$ where p, q are prime
- Notation: $\mathbb{Z}_N = \{0, 1, 2, \ldots, N-1\}$

  $\mathbb{Z}_N^*$ = invertible elements in $\mathbb{Z}_N$

  example: $N = 15 = 3 \cdot 5$, $\mathbb{Z}_N^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$

- Facts:

  $x \in \mathbb{Z}_N$ is in $\mathbb{Z}_N^* \iff \gcd(x, N) = 1$

  number of elements in $\mathbb{Z}_N^*$ is $\phi(N) = (p-1)(q-1)$

  Euler's theorem: $\forall\, x \in \mathbb{Z}_N^* : x^{\phi(N)} = 1$

  if $e \cdot d = 1 \pmod{\phi(N)}$, then $(x^e)^d = x \bmod N$

- Algorithms:

  Can add, mutiply, compute gcd, exponentiations and inverses mod N efficiently

Trapdoor Permutations

Three algorithms $(\mathrm{G}, \mathrm{F}, \mathrm{F}^{-1})$:

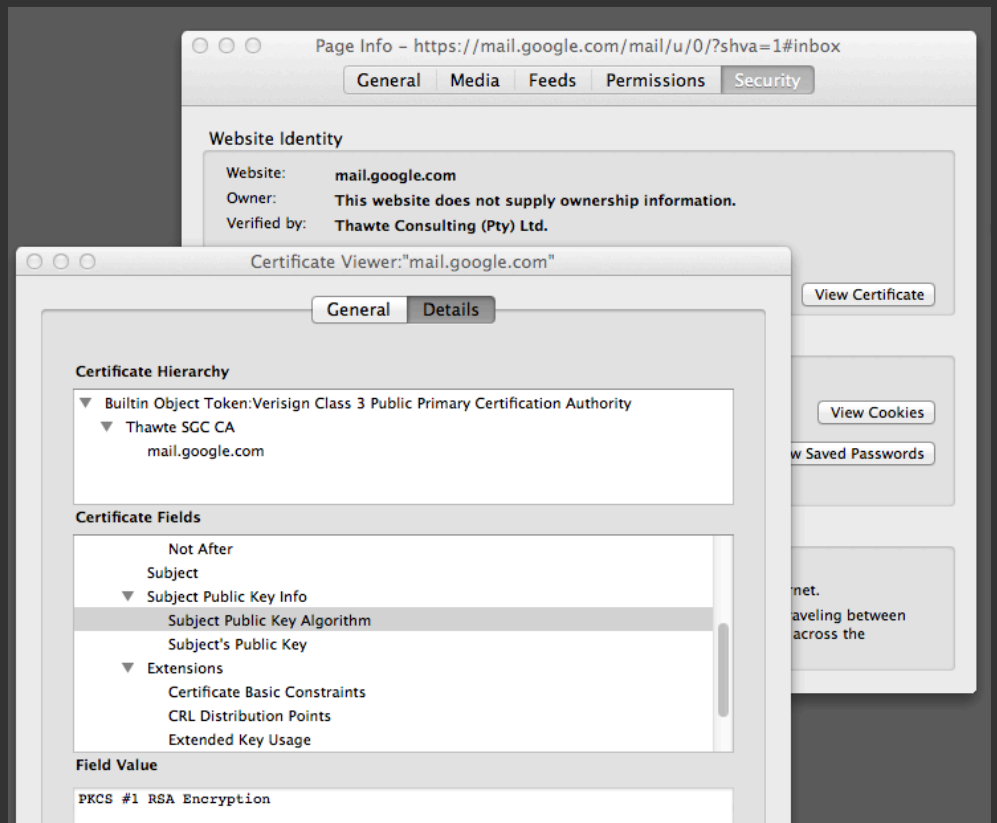
- G outputs $(\mathrm{pk}, \mathrm{sk})$, pk defines a permutation $\mathrm{F}(\mathrm{pk}, \cdot) : \mathrm{X} \to \mathrm{X}$
- $\mathrm{F}(\mathrm{pk}, \mathrm{x})$ evaluates the function at x
- $\mathrm{F}^{-1}(\mathrm{sk}, \mathrm{y})$ inverts the function at y using sk (the trapdoor)

- correctness: $\mathrm{F}^{-1}(\mathrm{sk},\ \mathrm{F}(\mathrm{pk}, \mathrm{x})) = \mathrm{x}$
- security: given random pk, y, hard to compute pre-image of y

  i.e. the function $\mathrm{F}(\mathrm{pk}, \cdot)$ is *one-way* without the trapdoor sk.

RSA Trapdoor Permutation

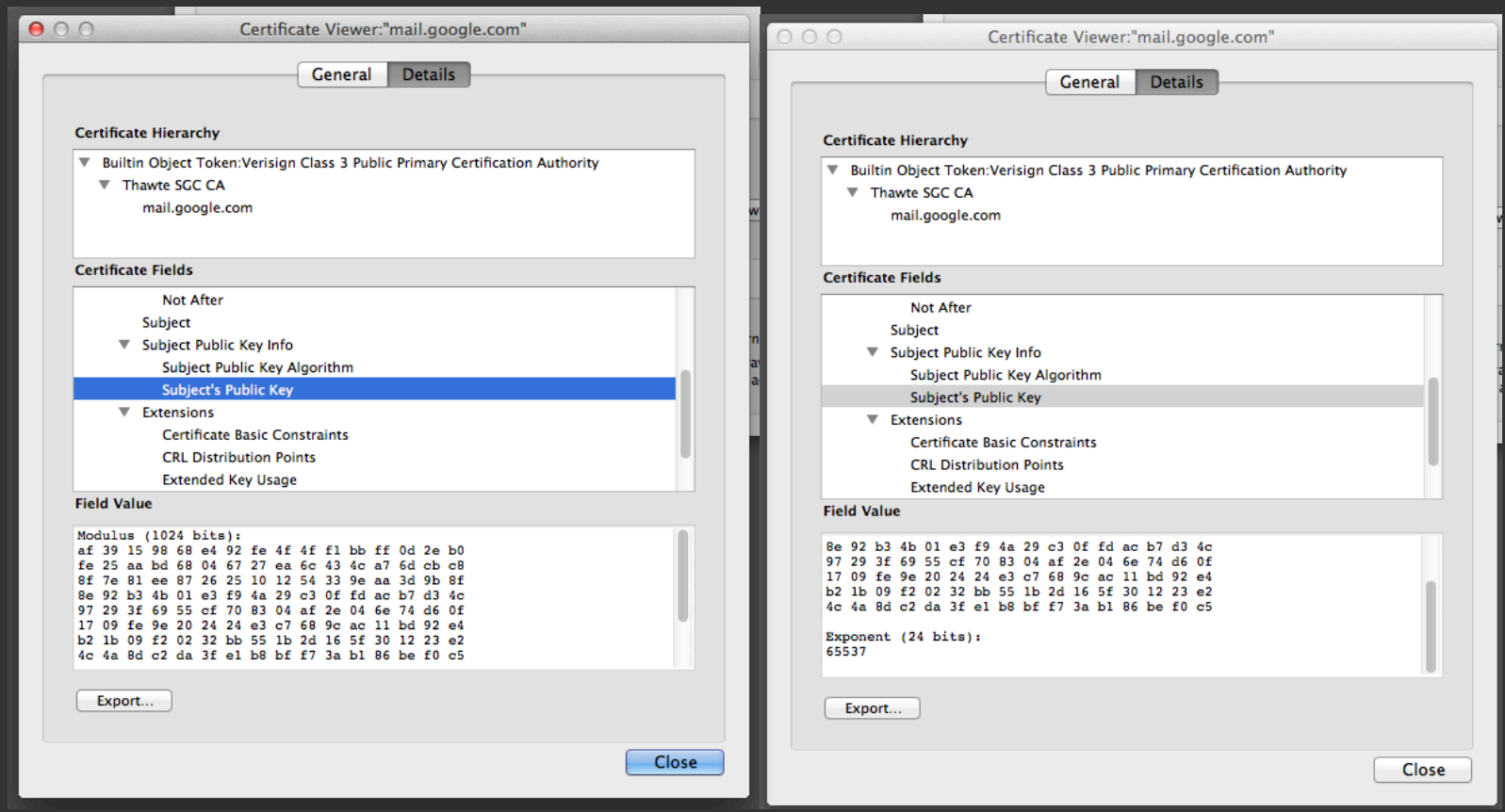
- first published, Scientific American, Aug 1977
- currently the "work horse" of Internet security:
  - most Public Key Infrastructure (PKI) products
  - SSL/TLS: certificates and key-exchange
  - secure e-mail and file systems



RSA Trapdoor Permutation

algorithm G. outputs (N, e) as pk

- $N = pq$ approx 1024 bits, p, q approx 512 bits
- e encryption exponent $\gcd(e, \phi(N)) = 1$.



RSA Trapdoor Permutation

algorithm G. outputs (N, e) as pk

- $N = pq$ approx 1024 bits, p, q approx 512 bits
- e encryption exponent $\gcd(e, \phi(N)) = 1$.

algorithm F. $\mathsf{RSA}(x) = x^e \bmod N$

- F maps $\mathbb{Z}_N^*$ to $\mathbb{Z}_N^*$

trapdoor. d "decryption exponent"

- $de = 1 \bmod \phi(N)$

algorithm F^{-1}. $\mathsf{RSA}^{-1}(y) = y^d \bmod N$

- $\mathsf{RSA}(x)^d = (x^e)^d = x^{ed} = x^{k\phi(N)+1} = (x^{\phi(N)})^k \cdot x = x$

example. $N = 35$ and $e = 5$.

- trapdoor $d = 5$ $\quad(5 \cdot 5 = 25 = 1 \bmod 24)$
- $\mathsf{RSA}(2) = 32$ and $32^5 = 2 \bmod 35$

Textbook RSA is insecure

Textbook RSA Encryption.

- Public key: (N, e) Encrypt: $\mathrm{C} = \mathrm{M}^{\mathrm{e}} \bmod \mathrm{N}$
- Secret key: d Decrypt: $\mathrm{C}^{\mathrm{d}} = \mathrm{M} \bmod \mathrm{N}$

Completely insecure cryptosystem!

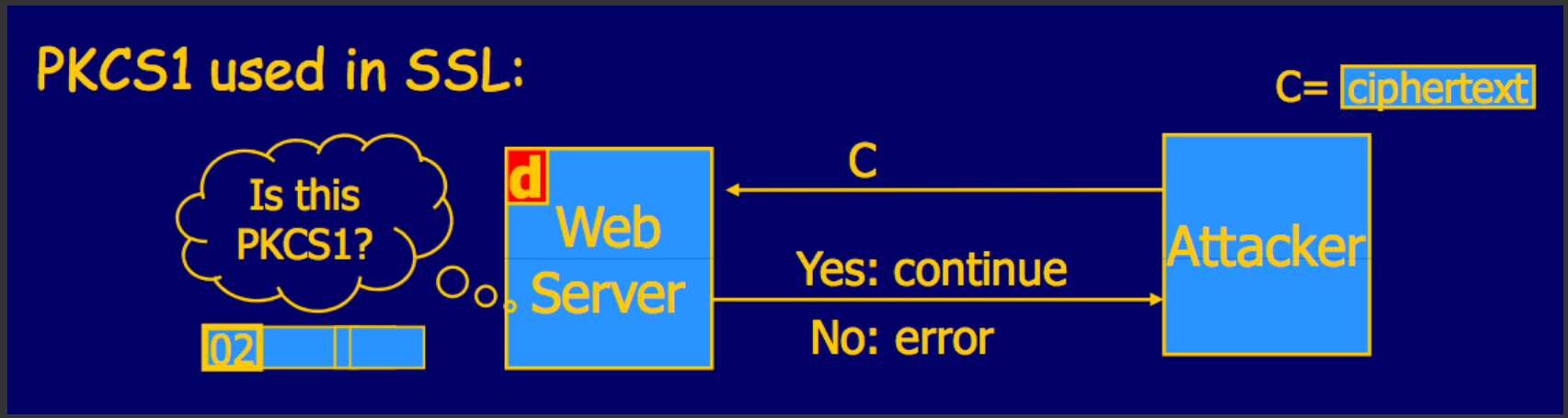
- Does not satisfy basic definition of security

  Given two ciphertexts, can tell if they are encryptions of same message.
- Many attacks exist.

RSA in Practice.

- Pre-process message before applying RSA permutation
- e.g. PKCS1 mode 2: $02||\text{random pad}||\mathrm{FF}||\mathrm{M}$

Attack on PKCS1 in SSL



- web server publishes RSA public key (N, e) and decrypts using e
- attacker can test if plaintext starts with 02 (16 bits)
- learn 16 bits of plaintext using $\approx 2^{16} = 65536$ queries

  learn 512 bits using $\approx 32 \times 65536 \ll 2^{512}$ queries

PKCS1 V2.0 uses RSA-OAEP, new pre-processing function

Implementation Attacks on RSA OAEP

```
OAEP-decrypt(C) {
    error = 0;
    if ( RSA-1(C) > 2n-1 )
    { error =1; goto exit; }
    ....
    if ( pad(OAEP-1(RSA-1(C))) != “01000” )
    { error = 1; goto exit; }
}
```

problem. timing information leaks type of error

- adversary can decrypt any ciphertext!
- easy to measure response time in many applications

lesson. don’t implement RSA-OAEP yourself...

RSA Optimizations

- To speed up RSA encryption (and signature verification), use small e

$$\mathrm{C} = \mathrm{M}^{\mathrm{e}} \bmod \mathrm{N}$$

- minimal value: $\mathrm{e} = 3$ $\gcd(\mathrm{e}, \phi(\mathrm{N})) = 1$
- recommended value: $\mathrm{e} = 65537 = 2^{16} + 1$

encryption: 17 modular multiplications

Q. Speed up RSA decryption with small d?

- Wiener's attack: if $\mathrm{d} \leq \mathrm{N}^{0.25}/3$, easy to find d from (N, e)

- Practice: use small e and large d

— fast encryption / slow decryption

More Implementation Attacks

Timing attack. (Kocher 97)

- time it takes to compute $\mathrm{C}^{\mathrm{d}} \bmod \mathrm{N}$ can expose d

Power attack. (Kocher 99)

- power consumption of smart card while computing $\mathrm{C}^{\mathrm{d}} \bmod \mathrm{N}$ can expose d

Faults attack. (BDL 97)

- computer error can expose d
- OpenSSL defense: check output. 5% slowdown

RSA Key Lengths

- security of public key cryptosystem should be comparable to security of block cipher.

— cipher key-size: 64 bits modular size 512 bits

— cipher key-size: 80 bits modular size 1024 bits

— cipher key-size: 256 bits modular size 15360 bits

- Elliptic Curve Cryptography avoids large modulus