RSA

BİL 448/548
Internet Security Protocols

Ali Aydın Selçuk

Number Theory Review

Def: m, n $\in \mathbb{Z}$ are *relatively prime* if gcd(m,n) = 1.

Def: $\mathbb{Z}_n^*$: the numbers in $\mathbb{Z}_n$ relatively prime to n.
e.g., $\mathbb{Z}_6^* = \{1, 5\}$, $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$.

Def: $\varphi(n) = |\mathbb{Z}_n^*|$.
e.g., $\varphi(6) = 2$, $\varphi(7) = 6$.

Number Theory Review

Theorem (Euler): For all $m \in \mathbb{Z}_n^*$, we have

$$m^{\varphi(n)} \equiv 1 \pmod{n}.$$

- E.g., n = 6, $\mathbb{Z}_6^* = \{1, 5\}$, $\varphi(n) = 2$;
  x = 5:
  $$5^2 = 25 \equiv 1 \pmod{6}$$
- E.g., n = 14, $\mathbb{Z}_{14}^* = \{1,3,5,9,11,13\}$, $\varphi(n) = 6$;
  x = 3:
  $$3^6 = 729 \equiv 1 \pmod{14}$$

Number Theory Review

Fact: If gcd(m,n) = d, then integers a, b exist s.t.

$$a \cdot m + b \cdot n = d.$$

E.g., m = 15, n = 24; gcd(15,24) = 3.

$$(-3) \cdot 15 + 2 \cdot 24 = 3.$$

Special case: If m, n are co-prime, $a \cdot m + b \cdot n = 1$.
E.g., m=15, n=2: $(-1) \cdot 15 + 8 \cdot 2 = 1$.

We say, $2^{-1} \pmod{15} = 8$.

RSA Cryptosystem

- The first successful public key algorithm by Rivest, Shamir, Adleman (1977)
- RSA:
  - Alice chooses large primes p, q; n = pq.
  - $\varphi(n) = (p-1)(q-1)$.
  - e, such that $\gcd(e, \varphi(n)) = 1$.
  - $d = e^{-1} \bmod \varphi(n)$ (i.e., $de = 1 \pmod{\varphi(n)}$)
  - n, e public. d is the private key.
  - Encryption: $E(x) = x^e \bmod n$
    Decryption: $D(x) = x^d \bmod n$

RSA Cryptosystem (cont.)

- Enc: $y = x^e \bmod n$
  Dec: $x = y^d \bmod n$
- Correctness: The decrypted text is,

$$y^d = (x^e)^d = x^{\varphi(n)\cdot c + 1} \mod n$$
$$= (x^{\varphi(n)})^c \, x \mod n$$
$$= x$$

RSA Cryptosystem (cont.)

- Security: Relies on difficulty of factoring n.
  - If $n = p \cdot q$ is known, then so is $\varphi(n)$, and d.
  - Conversely, if we can find d, we can factor n.
  - Hence, finding d ≡ factoring n.
- Any other ways to obtain x from e, n, y? Probably not. We don't know.
- Suggested key lengths:
  - short term: 1024 bits (better, 2048)
  - longer term: 4096 bits

Generation of RSA Parameters

- p, q can be generated randomly.
- $\varphi(n) = (p-1)(q-1)$
- choosing e, $\gcd(e, \varphi(n)) = 1$:
  - Take e to be a prime.
  - Generate p, q, such that $e \nmid (p-1)$, $e \nmid (q-1)$.
- Popular: e = 3, e = 65537.
- Randomness of d: due to n.

RSA Encryption Issues

- Guessable plaintext problem: If x comes from a small domain (PIN, password, etc.), given n, e, y, attacker can find x by trying:

$$x^e \pmod{n} = y?$$

- Issues with small e: If e = 3, and $x < n^{1/3}$, then

$$y = x^3 \quad \text{(no modular reduction)}$$

and attacker can simply solve x from x^3.

(Imagine that x is a 128-bit AES key.)

RSA Signature Issues

- Multiplicative property: Given two signatures $(x_1,S(x_1))$, $(x_2,S(x_2))$, attacker can compute the signature for x_1x_2:

$$S(x_1x_2) = S(x_1)S(x_2).$$

- Existential Forgery: Attacker can obtain the *message from the signature*: (inversely)

$$x = y^e \bmod n$$

- These may make sense if x is a random key.
- Solution?

RSA in Practice

- PKCS #1
  - Standard published by RSA Labs
  - Describes how to use RSA properly
- For encryption: random padding
  - Prevents predicting & trying the plaintext using the public key.
- For signature: fixed padding & hashing
  - Prevents obtaining valid (M,s) pairs using the public key and previously signed messages.

PKCS Encryption (v1.5)

0	2	random non-zero octets	0	data
1 byte each		≥ 8 bytes	1 byte	

- first 0: to guarantee $x < n$
- 2: indicates encryption
- second 0: indicates end of padding

Protects against:

- guessable message attack
- cube root problem, for e = 3
- …

PKCS Signature (v1.5)

0	1	octets of $(ff)_{16}$	0	hash type & hash
1 byte each		≥ 8 bytes	1 byte	

- Why not random padding?
- Why include the hash type?

Speed Comparisons

(Crypto++ 5.6 benchmarks, 2.2 GHz AMD Opteron 8354.)

Algorithm	**enc. time (ms/op.)**	**dec. time (ms/op.)**
AES-128 (block)	0.00008	0.00008
RSA-1024	0.04	0.67
RSA-2048	0.08	2.90

- Public key operations are much slower than symmetric key operations.
- Typically, PKC is used for the initial session key exchange, and then the symmetric key is used for the rest of the session.