Information Systems Security (SOEN321)

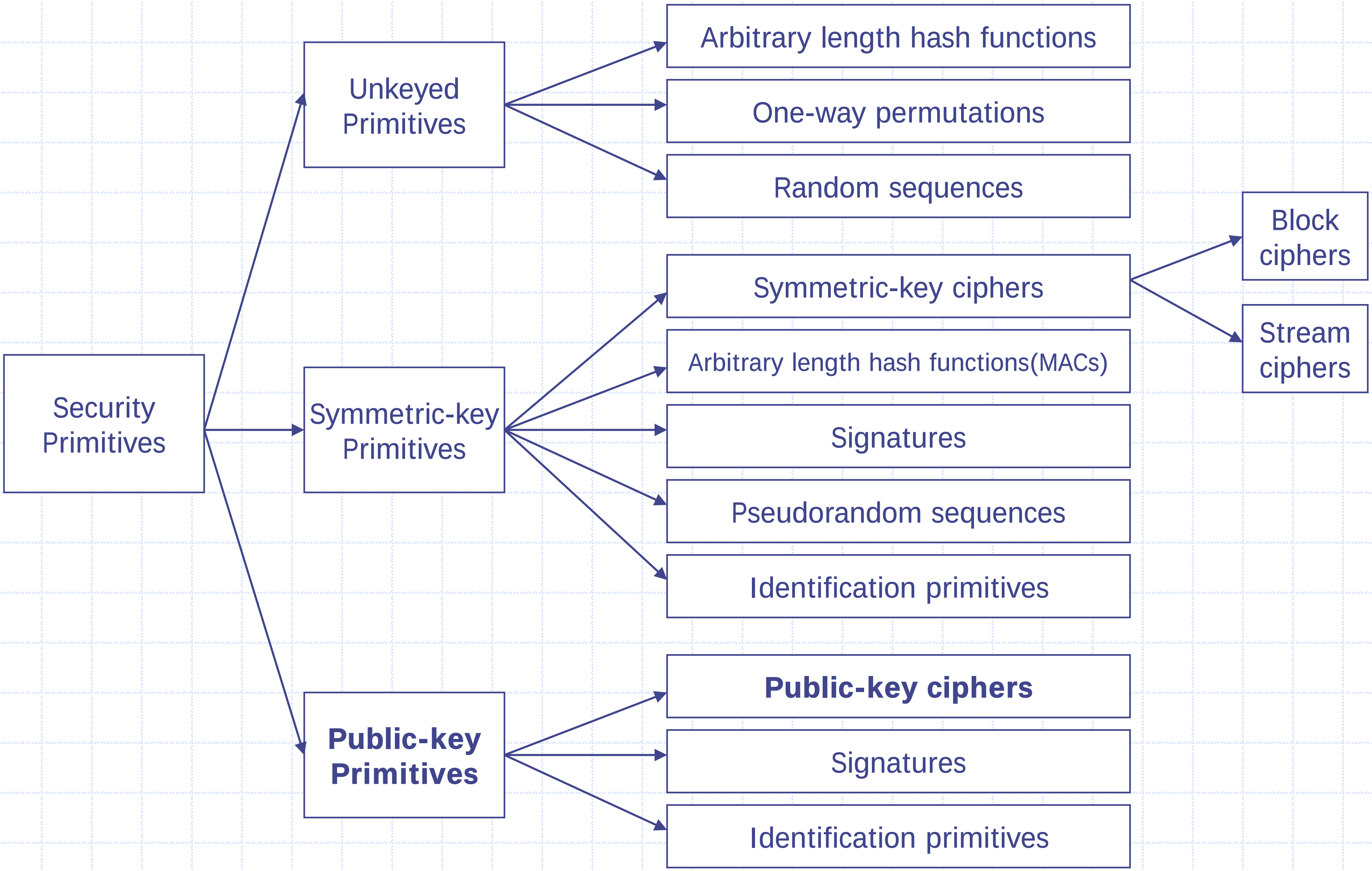
RSA

Dr. Amr Youssef

Concordia Institute for Information Systems Engineering (CIISE)
Concordia University
Montreal, Canada

youssef@ciise.concordia.ca

Taxonomy of cryptographic primitives



Famous Number Theory Problems

FACTORING	Given n, find a factor of n
RSAP	find m such that $m^e = c$ mod n
QRP	if , decide whether a is a QR or not.
SQROOT	find x such that $x^2 = a$ mod n
DLP	find x such that $g^x = y$ mod p
GDLP	DLP on a finite cyclic group G
DHP	given g^a mod p, g^b mod p, find g^{ab} mod p
GDHP	DHP on a finite cyclic group G
SUBSETSUM	given $\{a_1, \ldots, a_n\}$ and s, find subset of a_j that sums to s

Public Key Cryptography

- Each party has a PAIR (P, S) of keys: P is the **public** key and S is the **secret** key.
- Knowing the public-key and the cipher, it is still computationally infeasible to compute the private key (an NP-class problem).
- The public-key may be distributed to anyone wishing to communicate securely with its owner
- $Dec_S(Enc_P(M)) = M$

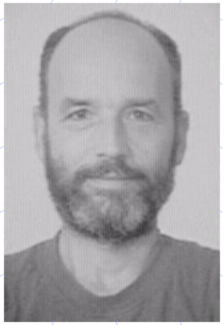
Trapdoor One-way Functions

- **Definition:**
- A function f: $\{0,1\}^* \rightarrow \{0,1\}^*$ is a trapdoor one-way function iff f(x) is a one-way function; however, given some extra information it becomes feasible to compute f^{-1}: given y, find x s.t. y = f(x)
- Public key cryptography relies on trapdoor one-way functions

RSA Algorithm

- Invented in 1978 by Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman Security relies on the difficulty of factoring large composite numbers
- Published as R L Rivest, A Shamir, L Adleman, "On Digital Signatures and Public Key Cryptosystems", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978

R



S

A

RSA Description

- **Key generation:**
- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\Phi(n) = (q-1)(p-1)$
  - Note: $\Phi(n)$ is Euler's Phi function
- Select a random integer e, $1 < e < \Phi(n)$, s.t. $\gcd(e,\Phi(n)) = 1$
- Compute d, $1< d< \Phi$ s.t. $ed \equiv 1 \bmod \Phi(n)$
- **Public key: (e, n)**
- **Secret key: d**

RSA (Cnt.)

- **Encryption**
- For the message M, $0 < M < n$ use public key (e, n) to compute $C = M^e \bmod n$
- **Decryption**
- For a message C use private key (d) to compute $C^d \bmod n = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n = M$

Toy Example

- $p = 11$, $q = 7$, $n = 77$, $\Phi(n) = 60$
- $d = 13$, $e = 37$ ($ed = 481$; $ed \bmod 60 = 1$)

- Let $M = 15$ Then
- $C \equiv M^e \bmod n$
  - $C \equiv 15^{37} \pmod{77} = 71$
- $M \equiv C^d \bmod n$
  - $M \equiv 71^{13} \pmod{77} = 15$

Proof of correctness (sketch)

- Want to show that $(M^e)^d \pmod{n} = M$, $n = pq$
- $ed \equiv 1 \pmod{\Phi(n)}$, so $ed = k*\Phi(n) + 1$, for some integer k.
- Case one: $\gcd(M, n) = 1$
  - $M^{ed} \equiv M^{k*\Phi(n)} M \equiv 1^k M \equiv M \pmod{n}$
- Case two: $\gcd(M, n) = p$
  - $M^{ed} \bmod p = (M \bmod p)^{ed} \bmod p = 0$
    - so $M^{ed} \equiv M \bmod p$
  - $M^{ed} \bmod q = (M^{k*\Phi(n)} \bmod q)\ (M \bmod q) = M \bmod q$
    - so Med $\equiv M \bmod q$
  - Since p and q are distinct primes we obtain $M^{ed} \equiv M \bmod pq$

$$
\begin{tikzcd}
p,q \arrow[rr, bend left, "\text{easy}"] & & N=pq \arrow[ll, bend left, "\text{hard}"]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[rr, bend left, "\text{easy}"] & & x^e \bmod N \arrow[ll, bend left, "\text{hard}"] \arrow[ll, bend left=80, "\text{Easy with trapdoor info (} d \text{)}"]
\end{tikzcd}
$$

Implementation

- Select p and q prime numbers
- In general, select numbers, then test for primality

Many implementations use the Rabin-Miller probabilistic test.

- How to perform the exponentiation
- In practice RSA is used to encrypt a random symmetric key, which is used to encrypt the message.

Square and Multiply Algorithm for Exponentiation

- Computing $(x)^c \bmod n$
- Algorithm

Square-and-multiply $(x, n, c = c_{k-1}\, c_{k-2} \dots c_1\, c_0)$

$z=1$

for $i \leftarrow k-1$ downto 0 {

$z \leftarrow z^2 \bmod n$

if $c_i = 1$ then $z \leftarrow (z \times x) \bmod n$

}

return z

- Example:
  - $c=110101=53$
  - $x^c=(((x^2 \cdot x)^2)^2 \cdot x)^2)^2 \cdot x \bmod n$

How to speed up the encryption

- To Speed up the encryption, speed up the square and-multiply exponentiation
- The smaller the number of 1 bits, the better
- Example: $e = 2^{16} + 1 = 65537$

RSA Security

- Based on difficulty of factoring large integers.
- RSA problem:
  - Given n, e, x^e mod n, what is x?
    (conjecture: It is equivalent to factoring n.)
- Bit Security of RSA:
  - Computing LSB(x) is equivalent to computing the whole x.

Attacks on RSA

- Possible Goals
  - recover secret key d
  - decrypt one message
  - learn information from the cipher texts
- Key recovery attacks
- Brute force key search
- Timing attacks
- Mathematical attacks

Timing Attacks

- *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems (1996), Paul C. Kocher*
- By measuring the time required to perform decryption (exponentiation with the private key as exponent), an attacker can figure out the private key
- Possible countermeasures:
  - use constant exponentiation time
  - add random delays

Math Attacks

- Three possible approaches:
  - Factor $n = pq$
  - Determine $\Phi(n)$
  - Find the private key d directly
- All the above are equivalent to factoring n

Factoring large numbers

- Three most effective algorithms are
  - quadratic sieve
  - elliptic curve factoring algorithm
  - number field sieve
- One idea many factoring algorithms use:
  - Suppose one find $x^2 \equiv y^2 \pmod{n}$ and $x \neq y \pmod{n}$ and
  - $x \neq -y \pmod{n}$. Then $n \mid (x-y)(x+y)$. Neither $(x-y)$ or $(x+y)$ is divisible by n; thus, $\gcd(x-y,n)$ has a non-trivial factor of n

Factoring records

- quadratic sieve:
  - $O(e^{(1+o(1))\mathrm{sqrt}(\ln n \ln \ln n)})$ for $n=2^{1024}$, $O(e^{68})$
- elliptic curve factoring algorithm
  - $O(e^{(1+o(1))\mathrm{sqrt}(2 \ln p \ln \ln p)})$, where p is the smallest prime factor
- number field sieve
  - $O(e^{(1.92+o(1)) (\ln n)^{1/3} (\ln \ln n)^{2/3}})$

some records:
1996: 432 bits/130 digits
1999: 512 bits/155 digits
2003: 576 bits/174 digits

Extrapolation:

2010: 768 bits/233 digits

2018: 1024 bits/310 digits

Knowing Φ(n) implies factorization

- Knowing both n and Φ(n), one knows
- $n = pq$
- $\Phi(n) = (p-1)(q-1) = pq - p - q + 1 = n - p - n/p + 1$
- $p2 - (n - \Phi(n) + 1)\ p + n = 0$ (solve 2nd order equation to get p)

Protocol failure

- Example for how to break RSA without factoring n
  - Low exponent attack
  - Common modulus attack

Low exponent attack

- Broadcast problem with low exponent
- Bob, Bart, Bert all use $e = 3$ with mods n_1, n_2, n_3.
- Alice sends the same message x to all:

$$x^3 \bmod n_1$$
$$x^3 \bmod n_2$$
$$x^3 \bmod n_3$$

- Eve computes $y = x^3 \bmod n_1 n_2 n_3$ by the CRT.
- Which is $y = x^3$, since $x < n_1, n_2, n_3$, and x is the cube root of y.

Common Modulus attack

- Each entity must choose its own modulus
- Assume Alice and Bob generated keys using the same modulus n, $((e_1, n), d_1))$ and $((e_2, n), d_2))$ then an eavesdropper can recover the plaintext
- $C_1 = M^{e1} \bmod n$,
- $C_2 = M^{e2} \bmod n$
- $(e1)a + (e2)\, b = 1$ if $\gcd(e1,e2)=1$
- $M = C_1^{a}\, C_2^{\, b} \bmod n$

Forward search attack

- If message space is small, the attacker can create a dictionary of encrypted messages (public key known, encrypt all possible messages and store them)
- When the attacker ‘sees’ a message on the network, compares the encrypted messages, so he finds out what particular message was encrypted

Multiplicative property

- $c_1c_2 = m_1^e\, m_2^e \bmod n = (m_1m_2)^e \bmod n$
- Adaptive chosen ciphertext attack (goal is to find m) $c = m^e \bmod n$
- Attacker chooses x and computes $c' = c\, x^e \bmod n$
- Asks Alice to decrypt it so, $c'^d = c^d\, x \bmod n = mx \bmod n$
- Now the attacker can compute m.
- Countermeasure: padding