Public-Key Cryptography

Alessandro Armando

Computer Security Laboratory (CSec)
DIBRIS, Università di Genova

Computer Security

Outline

Motivation

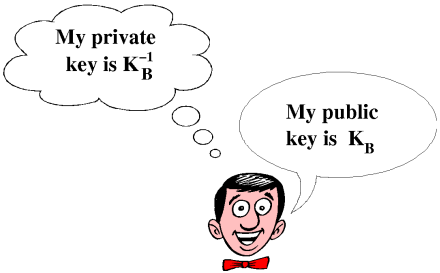
> Public key cryptography was born in May 1975, the child of two problems: the key distribution problem and the problem of signatures. The discovery consisted not of a solution, but of the recognition that the two problems, each of which seemed unsolvable by definition, could be solved at all and that the solutions to both came in one package.
>
> Whitfield Diffie, *The first-ten years of public key cryptography*, 1988

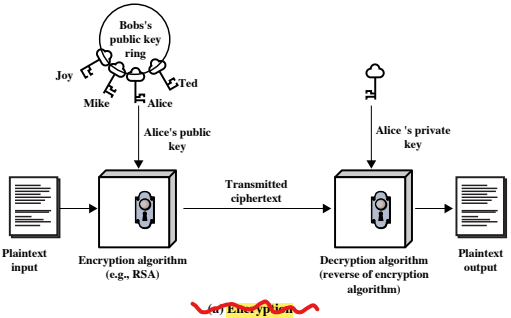
Let's consider to what extent these problems are "solved".

Public-key cryptography

- Let $\{E_e : e \in \mathcal{K}\}$ and $\{D_e : e \in \mathcal{K}\}$ form an encryption schema.
- Consider transformation pairs (E_e, D_d) where knowing E_e it is infeasible, given $c \in \mathcal{C}$ to find an $m \in \mathcal{M}$ where $E_e(m) = c$. This implies it is infeasible to determine d from e.

  $\Rightarrow$ E_e constitutes a trap-door one-way function with trapdoor d.
- Public key as e can be public information



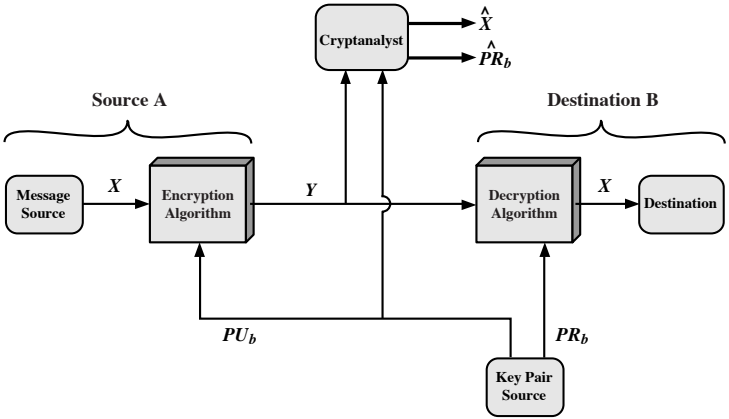

Bobs's public key ring
Joy
Ted
Mike
Alice
Alice's public key
Alice 's private key
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output
(a) Encryption
Alice's public key ring
Joy
Ted
Mike
Bob
Bob's private key
Bob's public key
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption
Plaintext output


Authenticity

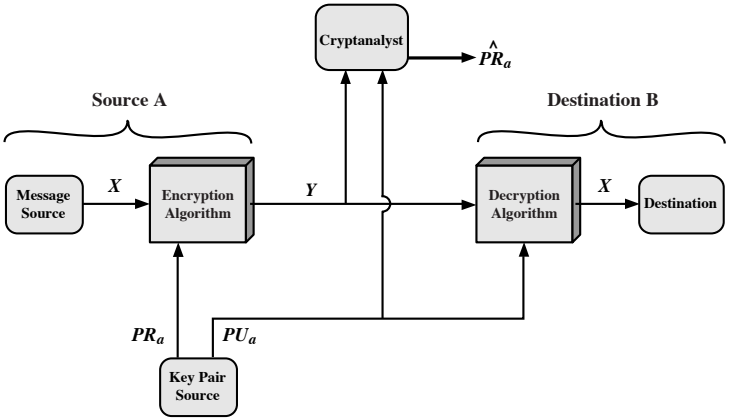
Conventionale (Symmetric) vs Public-Key (Asymmetric) Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public-Key Cryptosystem: Secrecy



Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Secrecy and Authentication

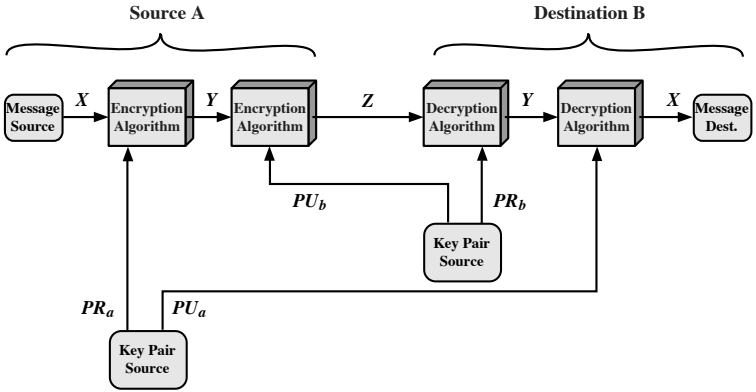


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Requirements for Public-Key Cryptography

1. It is computationally easy for B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for sender A, knowing PU_b and M, to generate

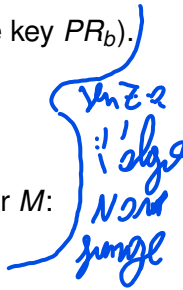
$$C = E(PU_b, M)$$

3. It is computationally easy for receiver B to decrypt C using PR_b to recover M:

$$M = D(PR_b, C) = D(PR_b, E(PU_b, M))$$

4. It is computationally infeasible for an adversary, knowing PU_b to determine PR_b.
5. It is computationally infeasible for an adversary, knowing PU_b and $C = E(PU_b, M)$ to recover M.
6. (Useful, but not always necessary) The two keys can be applied in either order:

$$M = D(PU_b, E(PR_b, M)) = D(PR_b, E(PU_b, M))$$



- These are difficult requirements.
- As a matter of facts only a few algorithms enjoying the above requirements have received widespread acceptance so far:

Algorithm	Encryption/ Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

One-way Functions

- A function $f : X \to Y$ is a one-way function, if f is "easy" to compute for all $x \in X$, but f^{-1} is "hard" to compute
- **Example:** Problem of modular cube roots
  - Select primes $p = 48611$ and $q = 53993$.
  - Let $n = pq = 2624653723$ and $X = \{1, 2, \ldots, n-1\}$.
  - Define $f : X \to \mathbb{N}$ by $f(x) = x^3 \bmod n$.
  - Example: $f(2489991) = 1981394214$. Computing f is easy.
  - Inverting f is hard: given y and n, find x such that $x^3 = y \bmod n$.
- **Note:** Not to be confused with one-way functions which take an arbitrarily data field as argument and map it to fixed-length output.

Trapdoor One-way Functions

- A trapdoor one-way function is a one-way function $f_k : X \to Y$ such that

$y = f_k(x)$	easy, if k and x are known
$x = f_k^{-1}(y)$	easy, if k and y are known
$x = f_k^{-1}(y)$	infeasible, if y is known but k is not known

  k is the trapdoor information.

- **Example:** Computing modular cube roots (above) is easy when p and q are known (basic number theory) but hard if they are not known.

Public-key Cryptoanalysis

- **Brute-force attacks** Countermeasure: use large keys!
  - But tradeoff is necessary as complexity of encryption/decryption may not scale linerarly with the length of the key.
  - In practice: public-key encryption is confined to *key management* and *digital signature*.
- **Computing private key from public key.** *No proof that this attack is unfeasible!* (Even for RSA)
- **Probable-message attack.** Suppose a short message M (e.g. a 56-bit DES key) is sent encrypted with PU_a, i.e. $C = E(PU_a, M)$. The attacker computes all $Y_i = E(PU_a, X_i)$ for all possible plaintext X_i for $i = 1, \ldots, 2^{56}$ and stops as soon as $Y_i = C$ concluding that $M = X_i$ (the message sent).

  Solution: Append some random bits to M.

Outline

Prime Factorisation

- To factor a number n is to write it as a product of other numbers: $n = a * b * c$.
- Multiplying numbers is easy, factoring numbers appears hard.

  We cannot factor most numbers with more than 1024 bits.
- The *prime factorisation* of a number a amounts to writing it as a product of powers of primes:

$$a = \Pi_{p \in P} \; p^{a_p} = 2^{a_2} * 3^{a_3} * 5^{a_5} * 7^{a_7} * 11^{a_{11}} * \cdots$$

where P is the set of prime numbers and $a_p \in \mathbb{N}$.

For example

$$\begin{aligned} 91 &= 7 * 13 \\ 3600 &= 2^4 * 3^2 * 5^2 \end{aligned}$$

Prime Factorisation

- To factor a number n is to write it as a product of other numbers: $n = a * b * c$.
- Multiplying numbers is easy, factoring numbers appears hard.

  We cannot factor most numbers with more than 1024 bits.
- The *prime factorisation* of a number a amounts to writing it as a product of powers of primes:

$$a = \Pi_{p \in P} \; p^{a_p} = 2^{a_2} * 3^{a_3} * 5^{a_5} * 7^{a_7} * 11^{a_{11}} * \cdots$$

  where P is the set of prime numbers and $a_p \in \mathbb{N}$.

  For example

$$\begin{aligned} 91 &= 7 * 13 \\ 3600 &= 2^4 * 3^2 * 5^2 \end{aligned}$$

Relatively prime numbers & gcd

- Two numbers a, b are *relatively prime* if they have no common divisors/factors apart from 1, i.e. if $\gcd(a, b) = 1$.

  For instance, 8 and 15 are relatively prime since
  - factors of 8 are 1,2,4,8,
  - factors of 15 are 1,3,5,15,
  - and 1 is the only common factor.

- Conversely we can determine the *greatest common divisor* by comparing their prime factorizations and using least powers.

  For example, $300 = 2^2 * 3^1 * 5^2$, $18 = 2^1 * 3^2$ hence $\gcd(18, 300) = 2^1 * 3^1 * 5^0 = 6$

Modular Arithmetics

- $\forall a\, n.\, \exists q\, r.\, (a = q * n + r)$ where $0 \leq r < n$.

  Here r is the *remainder*. We write the remainder as $a \bmod n$.

- $a, b \in \mathbb{Z}$ are *congruent modulo n*, if $a \bmod n = b \bmod n$.

  We write this as $a =_n b$.

- **Properties:**
  - $(a \bullet b) =_n (a \bmod n) \bullet (b \bmod n) \qquad$ for $\bullet \in \{+, -, *\}$
    i.e., $(a \bullet b) \bmod n = [(a \bmod n) \bullet (b \bmod n)] \bmod n$
  - If $a * b =_n a * c$ and a is relatively prime to n, then $b =_n c$.
- **Examples:**
  - $$\begin{aligned} 30 \bmod 4 &= \\ (5 * 6) \bmod 4 &= \\ ((5 \bmod 4) * (6 \bmod 4)) \bmod 4 &= \\ (1 * 2) \bmod 4 &= 2 \end{aligned}$$
  - $8 * 4 =_8 8 * 1 \;\Rightarrow\; 4 =_8 1$

Euler Totient Function

- When doing arithmetic modulo n
- Complete set of *residues* is $0, \ldots, n-1$
- *Reduced set of residues* consists of those numbers (*residues*) which are relatively prime to n

  For instance, for $n = 10$:
  - complete set of residues is $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - reduced set of residues is $\{1, 3, 7, 9\}$
- The *Euler Totient Function* $\phi(n)$ denotes the number of elements in the reduced set of residues.

  Properties:

  $\phi(p) = p - 1$ if p is prime

  $\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$ if p and q are prime.

Euler Totient Function

- When doing arithmetic modulo n
- Complete set of *residues* is $0, \ldots, n-1$
- *Reduced set of residues* consists of those numbers (*residues*) which are relatively prime to n

  For instance, for $n = 10$:
  - complete set of residues is $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - reduced set of residues is $\{1, 3, 7, 9\}$
- The *Euler Totient Function* $\phi(n)$ denotes the number of elements in the reduced set of residues.

  Properties:

  $\phi(p) = p - 1$ if p is prime

  $\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$ if p and q are prime.

Euler Theorem

Theorem

$a^{\phi(n)} =_n 1$ for all a, n such that $\gcd(a, n) = 1$.

Examples:

- If $a = 3$ and $n = 10$, then $\phi(10) = 4$ and $3^4 = 81 =_{10} 1$
- If $a = 2$ and $n = 11$, then $\phi(11) = 10$ and $2^{10} = 1024 =_{11} 1$

Euler Theorem

Theorem

$a^{\phi(n)} =_n 1$ for all a, n such that $\gcd(a, n) = 1$.

Examples:

- If $a = 3$ and $n = 10$, then $\phi(10) = 4$ and $3^4 = 81 =_{10} 1$
- If $a = 2$ and $n = 11$, then $\phi(11) = 10$ and $2^{10} = 1024 =_{11} 1$

Euler Theorem

Theorem

$a^{\phi(n)} =_n 1$ for all a, n such that $\gcd(a, n) = 1$.

Examples:

- If $a = 3$ and $n = 10$, then $\phi(10) = 4$ and $3^4 = 81 =_{10} 1$
- If $a = 2$ and $n = 11$, then $\phi(11) = 10$ and $2^{10} = 1024 =_{11} 1$

Outline

The RSA Algorithm

- Named after inventors: Rivest, Shamir, Adleman, 1978.
- Published after 1976 challenge by Diffie and Hellman.
- Security comes from difficulty of factoring large numbers
  Keys are functions of a pairs of large, ≥ 100 digits, prime numbers
- Most popular public-key algorithm
  Used in many applications, e.g., PGP, PEM, SSL, ...

The RSA Algorithm

- Let n be a number known by sender and receiver.
- Plaintext split in blocks of $\lfloor \log_2(n) \rfloor$ bits.
  Thus each block represents a number M such that $M < n$.
- Encryption and decryption are defined as follows:

$$\begin{aligned} C &= M^e \mod n \\ M &= C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n \end{aligned}$$

  for some (properly chosen) values of e and d.
- It is a public-key encryption algorithm with
  - public key $PU = (e, n)$ and
  - private key $PR = (d, n)$.

The RSA Algorithm

- Let n be a number known by sender and receiver.
- Plaintext split in blocks of $\lfloor \log_2(n) \rfloor$ bits.
  Thus each block represents a number M such that $M < n$.
- Encryption and decryption are defined as follows:

$$\begin{aligned} C &= M^e \mod n \\ M &= C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n \end{aligned}$$

  for some (properly chosen) values of e and d.
- It is a public-key encryption algorithm with
  - public key $PU = (e, n)$ and
  - private key $PR = (d, n)$.

The RSA Algorithm

- Let n be a number known by sender and receiver.
- Plaintext split in blocks of $\lfloor \log_2(n) \rfloor$ bits.
  Thus each block represents a number M such that $M < n$.
- Encryption and decryption are defined as follows:

$$\begin{aligned} C &= M^e \mod n \\ M &= C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n \end{aligned}$$

  for some (properly chosen) values of e and d.
- It is a public-key encryption algorithm with
  - public key $PU = (e, n)$ and
  - private key $PR = (d, n)$.

The RSA Algorithm

- Let n be a number known by sender and receiver.
- Plaintext split in blocks of $\lfloor \log_2(n) \rfloor$ bits.

  Thus each block represents a number M such that $M < n$.
- Encryption and decryption are defined as follows:

$$\begin{aligned} C &= M^e \mod n \\ M &= C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n \end{aligned}$$

  for some (properly chosen) values of e and d.
- It is a public-key encryption algorithm with
  - public key $PU = (e, n)$ and
  - private key $PR = (d, n)$.

For the RSA algorithm to work, the following requirements must be met:

1. It is possible to find values of e, d, and n such that $M^{ed} \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.
3. It is unfeasible to determine d given e and n.

Correctness of RSA

Definition (Multiplicative Inverses)

x and y are multiplicative inverses $\bmod r$ iff $xy \bmod r = 1$.

Theorem (Correctness of RSA)

If d and e are multiplicative inverses $\bmod \phi(n)$*, i.e. if* $ed \bmod \phi(n) = 1$*, then* $M^{ed} \bmod n = M$ *for all* $M < n$.

Lemma

Let p and q be prime numbers and $n = pq$*. If d and e are multiplicative inverses* $\bmod \phi(n)$*, then* $M^{ed} \equiv_p M$ *and* $M^{ed} \equiv_q M$*, i.e.* $(M^{ed} - M)$ *is multiple of p and q.*

Proof of Correctness of RSA.

Let d and e are multiplicative inverses $\bmod \phi(n)$. By Lemma, $(M^{ed} - M)$ is multiple of p and q. Since p and q are prime then $(M^{ed} - M)$ is also multiple of pq and hence of n. □

Correctness of RSA – continued

Proof of Lemma.

Let d and e are multiplicative inverses $\bmod \phi(n)$. Then, there exists an integer k such that $ed = k\phi(n) + 1$.
We must prove that $M^{ed} = M^{k\phi(n)+1} \equiv_p M$. Two cases:
Case 1: M and p are relatively prime.

$$\begin{aligned} M^{k\phi(n)+1} \bmod p &= M \cdot M^{k(p-1)(q-1)} \bmod p \\ &= M \cdot (M^{(p-1)})^{k(q-1)} \bmod p \\ &= M \cdot (M^{\phi(p)})^{k(q-1)} \bmod p \ \text{ since } p \text{ is prime} \\ &= M \cdot (1)^{k(q-1)} \bmod p \ \text{ by Euler Theorem} \\ &= M \bmod p \end{aligned}$$

Case 2: M and p are not relatively prime. Then M is a multiple of p, i.e. $M \bmod p = 0$ and hence $M^{k\phi(n)+1} \bmod p = M \bmod p$. □

RSA algorithms

- Generate a public/private key pair:
  1. Generate two (large) distinct primes p and q.
  2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
  3. Select an e, $1 < e < \phi$, relatively prime to ϕ.
  4. Compute d such that $ed \bmod \phi = 1$.
  5. Publish (e, n), keep (d, n) private, discard p and q.
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$.

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate two (large) distinct primes p and q.
  2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
  3. Select an e, $1 < e < \phi$, relatively prime to ϕ.
  4. Compute d such that $ed \bmod \phi = 1$.
  5. Publish (e, n), keep (d, n) private, discard p and q.
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
  3. Select an e, $1 < e < \phi$, relatively prime to ϕ.
  4. Compute d such that $ed \bmod \phi = 1$.
  5. Publish (e, n), keep (d, n) private, discard p and q.
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Select an e, $1 < e < \phi$, relatively prime to ϕ.
  4. Compute d such that $ed \bmod \phi = 1$.
  5. Publish (e, n), keep (d, n) private, discard p and q.
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $ed \bmod \phi = 1$.
  5. Publish (e, n), keep (d, n) private, discard p and q.
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Publish (e, n), keep (d, n) private, discard p and q.
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- Encryption with key (e, n)
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- Encryption with key $(e, n) = (79, 3337)$
  1. Break message M into blocks $M_1 M_2 \cdots$ with $M_i < n$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- Encryption with key $(e, n) = (79, 3337)$
  1. Break message M into blocks, e.g. 688 232 687 966 668 $\cdots$
  2. Compute $C_i = M_i^e \bmod n$.
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- Encryption with key $(e, n) = (79, 3337)$
  1. Break message M into blocks, e.g. $688\ 232\ 687\ 966\ 668 \cdots$
  2. Compute $C_1 = 688^{79} \bmod 3337 = 1570$, $C_2 = ...$
- Decryption with key (d, n):
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- Encryption with key $(e, n) = (79, 3337)$
  1. Break message M into blocks, e.g. 688 232 687 966 668 $\cdots$
  2. Compute $C_1 = 688^{79} \bmod 3337 = 1570$, $C_2 = ...$
- Decryption with key $(d, n) = (1019, 3337)$:
  1. Compute $M_i = C_i^d \bmod n$

RSA algorithms: Example

- Generate a public/private key pair:
  1. Generate $p = 47$, $q = 71$
  2. Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
  3. Choose $e = 79$ (randomly in the interval [1..3220])
  4. Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
  5. Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- Encryption with key $(e, n) = (79, 3337)$
  1. Break message M into blocks, e.g. $688\ 232\ 687\ 966\ 668 \cdots$
  2. Compute $C_1 = 688^{79} \bmod 3337 = 1570$, $C_2 = ...$
- Decryption with key $(d, n) = (1019, 3337)$:
  1. Compute $M_1 = 1570^{1019} \bmod 3337 = 688$, $M_2 = ...$

Computing the Multiplicative Inverse

By adapting the Extended Euclidean algorithm:

```
1  function inverse(a, n)
2      t := 0; nt := 1;
3      r := n; nr := a;
4      while nr != 0
5          q := r div nr;
6          (t, nt) := (nt, t-q*nt);
7          (r, nr) := (nr, r-q*nr);
8      if r > 1 then return "a is not invertible";
9      if t < 0 then t := t + n;
10     return t;
```

RSA Security

- Computation of secret d given (e, n)
  - As difficult as factorization. If we can factor $n = pq$ then we can compute $\phi = (p-1)(q-1)$ and hence d.
  - No known polynomial time algorithm.
    But given progress in factoring, n should have at least 1024 bits.
- Computation of M_i, given C_i, and (e, n)
  - Unclear (= no proof) whether it is necessary to compute d, i.e., to factorize n.

$\Rightarrow$ Progress in number theory could make RSA insecure.

Malleability of RSA

- Recall that a cryptographic function $E(K, M)$ is *malleable* iff there exist two functions $F(X)$ and $G(X)$ such that

$$F(E(K, M)) = E(K, G(M)) \quad \text{for all keys } K \text{ and messages } M$$

- RSA encryption, namely $E((e, n), M)) = M^e \bmod n$ is clearly malleable. Let $F(X) = X * (M_1^e \bmod n)$ for any given M_1.

$$\begin{aligned} F(E((e,n),M)) = E((e,n),M) * (M_1^e \bmod n) = \\ = (M^e \bmod n) * (M_1^e \bmod n) = (M * M_1)^e \bmod n = \\ = E((e,n), M * M_1) = E((e,n), G(M)) \end{aligned}$$

- For this reason, RSA is commonly used together with padding methods such as OAEP or PKCS1.

Outline

Asymmetric algorithms for secret key distribution

- Use public key encryption algorithms to support (faster) symmetric cryptography.
- We will see two approaches:
  - Secret key distribution with RSA
  - Diffie-Hellman key exchange

Outline

Secret key distribution with RSA

- Encryption of m (with public key (e, n))
  - choose k randomly
  - c = $(k^e \bmod n, E_k(m))$
- Decryption (with private key (d, n))
  - Split c into (c_1, c_2)
  - $k = c_1^d \bmod n \qquad m = D_k(c_2)$
- **Example:** SSL
  Alice chooses a secret, encrypts it with Bob's PK and rest of the session is protected based on that secret.
- **Problem:** if the private key (d, n) gets compromised, then k can be recovered by an intruder from previously observed traffic.

Outline

Background on discrete logarithms

- A primitive root s of a prime number p is a number whose powers generate $1, \ldots, p-1$.
  So $s \bmod p$, $s^2 \bmod p$, …, $s^{p-1} \bmod p$ are distinct, i.e., a permutation of 1 through $p-1$. Hence:

$$\forall b \in \mathbb{Z}.\, \exists i \in \{0, \ldots, p-1\}.\, b = s^i \bmod p$$

- Given $b \in \mathbb{Z}$, exponent i above is the discrete logarithm of b for base s, mod p.
- Computing discrete logs appears infeasible.

Diffie-Hellman key exchange

1. Principals share prime q and primitive root α of q. Both may be public.
2. A and B generate random numbers X_A and X_B (resp.) both less than q.
3. A computes $Y_A = \alpha^{X_A} \bmod q$, B computes $Y_B = \alpha^{X_B} \bmod q$.
4. A and B exchange results.
5. A computes $K_A = Y_B^{X_A} \bmod q$, B computes $K_B = Y_A^{X_B} \bmod q$.
   Keys are equal, i.e. $K_A = K_B$:

$$\begin{aligned} K_A &= Y_B^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= Y_A^{X_B} \bmod q \;=\; K_B \end{aligned}$$

Security depends on difficulty of computing discrete logs.

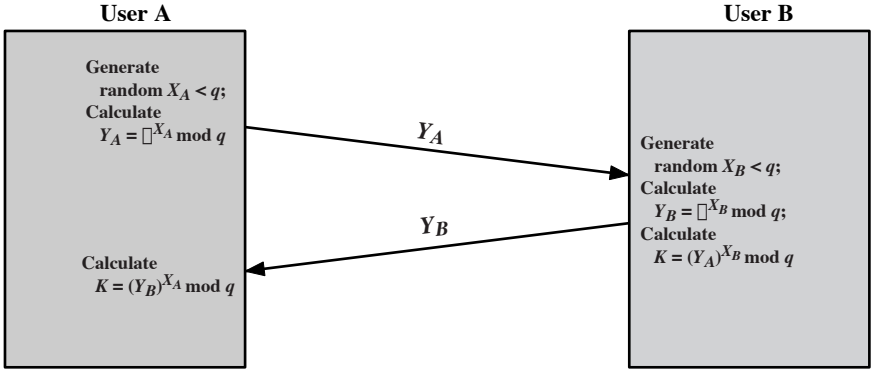


Figure 10.8 Diffie-Hellman Key Exchange

Diffie-Hellman: strengths

- The shared secret is created out of nothing!
- The shared secret is never trasmitted (not even in encrypted form).

$\Rightarrow$ **Perfect Forward Secrecy** (PFS), i.e. if someone records the entire conversation and later discovers Alice's and/or Bob's private keys, he/she won't be able to decrypt anything!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!

Diffie-Hellman: weaknesses

Keys are unauthenticated and thus it is vulnerables to the following *Man-in-the-middle Attack*:

1. A transmits Y_A to B
2. I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
3. B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
4. B transmits Y_B to A
5. I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
6. A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead A shares secret key K_2 with I and B shares secret key K_1 with I.

Solution: sign the exponents. But this requires shared keys!