Public Key (asymmetric) Cryptography

Luca Veltri
(mail.to: luca.veltri@unipr.it)

Corso di Sicurezza nelle reti, a.a. 2011/2012
http://www.tlc.unipr.it/veltri

Public Key Cryptography

- Also referred to as asymmetric cryptography or two-key cryptography
- Probably most significant advance in the 3000 year history of cryptography
  - public invention due to Whitfield Diffie & Martin Hellman in 1975
    - at least that's the first published record
    - known earlier in classified community (e.g. NSA?)
- Is asymmetric because
  - who encrypts messages or verify signatures cannot decrypt messages or create signatures
  - more in general, operation performed by two parties use different key values
- Uses clever application of number theoretic concepts and mathematic functions rather than permutations and substitutions
- Complements rather than replaces secret key crypto

Public-Key vs. Secret Cryptography

- All secret key algorithms do the same thing
  - they take a block and encrypt it in a reversible way
- All hash algorithms do the same thing
  - they take a message and perform an irreversible transformation
- Instead, public key algorithms look very different
  - in how they perform their function
  - in what functions they perform
- They all have in common: a private and a public quantities associated with a principal
- Example of public key algorithms:
  - RSA, which does encryption and digital signature
  - El Gamal and DSS, which do digital signature but not encryption
  - Diffie-Hellman, which allows establishment of a shared secret
  - zero knowledge proof systems, which only do authentication
Public Key Cryptography (cont.)

- Public key cryptography can do anything secret key cryptography can do, but...

- The known public-key cryptographic algorithms are orders of magnitude slower than the best known secret key cryptographic algorithms
  - are usually used only for things secret key cryptography can’t do (or can’t do in a suitable way)

- Often it is mixed with secret key technology
  - e.g. public key cryptography might be used in the beginning of communication for authentication and to establish a temporary shared secret key used to encrypt the conversation

Public Key vs. Secret Cryptography (cont.)

- With symmetric/secret-key cryptography
  - you need a secure method of telling your partner the key
  - you need a separate key for everyone you might communicate with

- Instead, with public-key cryptography, keys are not shared

- Public-key cryptography often uses two keys:
  - a public-key, which may be known by anybody, and can be used to encrypt messages, or verify signatures
  - a private-key, known only to the recipient, used to decrypt messages, or sign (create) signatures

- Security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalyse) problems
  - often the hard problem is known, its just made too hard to do in practise
  - requires the use of very large numbers
  - hence is slow compared to private key schemes

Why Public-Key Cryptography?

- Can be used to:
  - key distribution – secure communications without having to trust a KDC with your key (key exchange)
  - digital signatures – verify a message is come intact from the claimed sender (authentication)
  - encryption/decryption - secrecy of the communication (confidentiality)

- Some algorithms are suitable for all uses, others are specific to one

- Note that public-key cryptography simplifies but not eliminates the problem of trusted systems and key management

Security of Public Key Schemes

- Security of public-key algorithms still relies on key size (as for secret-key algorithms)

- Like private key schemes brute force exhaustive search attack is always theoretically possible
  - But keys used are much larger (>512bits)

- A crucial feature is that the private key is difficult to determine from the public key
  - security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalyse) problems

  - requires the use of very large numbers
  - hence is slow compared to private key schemes
### Rivest, Shamir, and Adleman (RSA)

**by Rivest, Shamir & Adleman of MIT in 1977**
- best known & widely used public-key scheme
- Based on exponentiation in a finite (Galois) field over integers modulo n
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)
- The key length is variable
  - long keys for enhanced security, or a short keys for efficiency
- The plaintext block size (the chunk to be encrypted) is also variable
  - The plaintext block size must be smaller than the key length
  - The ciphertext block will be the length of the key
- RSA is much slower to compute than popular secret key algorithms like DES, IDEA, and AES

### RSA Algorithm

1. First, you need to generate a public key and a corresponding private key:
   - choose two large primes $p$ and $q$ (around 512 bits each or more)
     - $p$ and $q$ will remain secret
   - multiply them together (result is 1024 bits), and call the result $n$
     - it's practically impossible to factor numbers that large for obtaining $p$ and $q$
   - choose a number $e$ that is relatively prime (that is, it does not share any common factors other than 1) to $\phi(n)$
     - since you know $p$ and $q$, you know $\phi(n) = (p-1)(q-1)$
   - your public key is $KU=<e,n>$
   - find the number $d$ that is the multiplicative inverse of $e$ mod $\phi(n)$
   - your private key is $KR=<d,n>$ or $KR=<d,p,q>$

2. To encrypt a message $m (< n)$, someone can use your public key
   - $c = m^e \mod n$

3. Only you will be able to decrypt $c$, using your private key
   - $m = c^d \mod n$

### Why RSA Works

1. Because of Euler's Theorem:
   - $a^{\phi(n)+1} \mod n = a$
     - where $gcd(a,n)=1$

2. In RSA have:
   - $n=p.q$
   - $\phi(n)=(p-1)(q-1)$
   - carefully chosen $e$ & $d$ to be inverses mod $\phi(n)$
   - hence $e.d=1+k.\phi(n)$ for some $k$

3. Hence:
   - $c^d = (m^e)^d = m^{e+kd}\mod n = m \mod n$
RSA Key Setup

- Each user generates a public/private key pair by:
  - selecting two large primes at random $p, q$
  - computing their system modulus $n = p \times q$
    - note $\varphi(n) = (p-1)(q-1)$
  - selecting at random the encryption key $e$
    - where $1 < e < \varphi(n)$, $\gcd(e, \varphi(n)) = 1$
  - solve following equation to find decryption key $d$
    - $ed = 1 \mod \varphi(n)$ and $0 \leq d \leq n$
- Publish their public encryption key: $K_U = \{e, n\}$
- Keep secret private decryption key: $K_R = \{d, p, q\}$

RSA Use

- To encrypt a message $M$ the sender:
  - obtains public key of recipient $K_U = \{e, n\}$
  - computes: $c = m^e \mod n$, where $0 \leq m < n$
- To decrypt the ciphertext $c$ the owner:
  - uses their private key $K_R = \{d, n\}$
  - computes: $m = c^d \mod n$
- Note that the message $m$ must be smaller than the modulus $n$ (block if needed)

RSA Example

RSA setup

- select primes: $p=17$ & $q=11$
- compute $n = pq = 17 \times 11 = 187$
- compute $\varphi(n) = (p-1)(q-1) = 16 \times 10 = 160$
- select $e : \gcd(e, 160) = 1$; choose $e=7$
- determine $d$: $de = 1 \mod 160$ and $d < 160$ Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
- publish public key $K_U = \{7, 187\}$
- keep secret private key $K_R = \{23, 187\} = \{23, 17, 11\}$
RSA Example (cont)

RSA encryption/decryption:

- given message $M = 88$ (nb. $88 < 187$)
- encryption:
  $C = 88^7 \mod 187 = 11$
- decryption:
  $M = 11^{23} \mod 187 = 88$

RSA Security

- three approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing $\varphi(N)$, by factoring modulus $N$)
  - timing attacks (on running of decryption)

Diffie-Hellman

- First public-key type scheme proposed
- By Diffie & Hellman in 1976 along with the exposition of public key concepts
  - now know that James Ellis (UK CESG) secretly proposed the concept in 1970
    - predates RSA
  - less general than RSA: it does neither encryption nor signature
- Is a practical method for public exchange of a secret key
  - allows two individuals to agree on a shared secret (key)
  - It is actually used for key establishment
- Used in a number of commercial products
**Diffie-Hellman Setup**

Diffie-Hellman setup:
- all users agree on global parameters:
  - $p$ = a large prime integer or polynomial
  - $g$ = a primitive root mod $p$
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < p$
  - compute their public key: $y_A = g^{x_A} \mod p$
- each user makes public that key $y_A$

**Diffie-Hellman Key Exchange**

Key exchange:
- Shared key $K_{AB}$ for users A & B can be computed as:
  $$K_{AB} = g^{x_A \cdot x_B} \mod p$$
  - $y_A = g^{x_A} \mod p$ (which $B$ can compute)
  - $y_B = g^{x_B} \mod p$ (which $A$ can compute)
- $K_{AB}$ can be used as session key in secret-key encryption scheme between A and B
- Attacker must solve discrete log

**Diffie-Hellman - Example**

- users Alice & Bob who wish to swap keys:
  - agree on prime $p=353$ and $g=3$
  - select random secret keys:
    - A chooses $x_A=97$, B chooses $x_B=233$
    - compute public keys:
      - $y_A=g^{x_A} \mod 353 = 40$ (Alice)
      - $y_B=g^{x_B} \mod 353 = 248$ (Bob)
    - compute shared session key as:
      - $K_{AB}=y_A^{x_B} \mod 353 = 248^{97} = 160$ (Alice)
      - $K_{AB}=y_B^{x_A} \mod 353 = 40^{233} = 160$ (Bob)
Zero Knowledge Proof Systems

- Only do authentication
  - prove that you know a secret without revealing the secret
- RSA is a zero knowledge system
- There are zero knowledge systems with much higher performance
- Example (Isomorphic graphs):
  - Alice defines two large (say 500 vertices) isomorphic graphs $G_A$, $G_B$
  - $G_A$ and $G_B$ become public, but only Alice knows the mapping
  - to prove her identity to Bob, Alice finds a set of isomorphic graphs $G_1, G_2, \ldots, G_k$
  - Bob divides the set into two subsets $T_A$ and $T_B$
  - Alice shows to Bob the mapping between each $G_i \in T_A$ and $G_A$, and between each $G_j \in T_B$ and $G_B$

Security uses of public key cryptography

- Transmitting over an insecure channel
  - each party has a <public key, private key> pair $(K_u, K_r)$
  - each party encrypts with the public key of the other party
    - encrypt $m_u$ using $K_u$ of the other party
    - decrypt $m_u$ using $K_r$ of the other party
    - encrypt $m_r$ using $K_u$ of the other party
    - decrypt $m_r$ using $K_r$ of the other party
- Secure storage on insecure media
  - encrypt with public key, decrypt with private key
  - useful when you can let third party to encrypt data
- Peer Authentication
  - public key gives the real benefit
  - no $n(n-1)/2$ keys are needed
    - encrypt $r$ using $K_u$ of the other party
    - decrypt to $r$ using $K_r$ of the other party

Security uses of public key cryptography

- Data authentication (Digital signature)
  - based on cryptographic checksum
- Key establishment
  - e.g. Diffie-Hellman
- Note
  - Public key cryptography has specific algorithm for specific function such as
    - data encryption
    - MAC/digital signature
    - peer authentication
    - key establishment

Vantaggio dei sistemi a chiave pubblica

- Ogni utente deve mantenere solo un segreto (la propria chiave privata)
- Le chiavi pubbliche degli altri utenti possono essere mantenuti tramite infrastrutture intermediarie sicure (PKI)
- Il numero delle chiavi è proporzionale a N per la comunicazione reciproca tra N utenti
Digital signature and digital certification

Digital Signature

- Digital Signature is an application in which a signer, say "Alice," "signs" a message \( m \) in such a way that
  - anyone can "verify" that the message was signed by no one other than Alice, and
  - consequently that the message has not been modified since she signed it
- i.e. the message is a true and correct copy of the original
- The difference between digital signatures and conventional ones is that digital signatures can be mathematically verified
- The typical implementation of digital signature involves a message-digest algorithm and a public-key algorithm for encrypting the message digest (i.e., a message-digest encryption algorithm)

Two Approaches to Digital Signatures

- RSA approach
- DSS approach
**Digital Signature Standard (DSS)**

- **DSS (Digital Signature Standard)**
- **Proposed by NIST (U.S. National Institute of Standards and Technology) & NSA in 1991**
  - FIPS 186
- **Based on an algorithm known as DSA (Digital Signature Algorithm)**
  - is a variant of the ElGamal scheme
  - uses 160-bit exponents
  - creates a 320 bit signature (160+160) but with 1024 (or more) bit security
  - uses SHA/SHS hash algorithm
- **Security depends on difficulty of computing discrete logarithms**

**DSA Key Generation**

- have shared global public key values $(p,q,g)$
  - $L$ is the key length
    - $L = 1024$ or more, and is a multiple of 64
  - a large prime $p$
  - choose $q$, a 160 bit prime factor of $p-1$
    - actually long as the hash $H$
  - choose $g = h^{\frac{p-1}{q}} \mod p$
    - where $h < p-1$, $h^{\frac{p-1}{q}} \mod p > 1$
    - for some arbitrary $h$ with $1 < h < p-1$
- choose $x < q$
- compute $y = g^x \mod p$
- **public key = $(p,q,g,y)$**
- **private key = $x$**

**DSS Signing and Verifying**

- **Signing**
  - $s = f_1(H(m), k, x, r, q) = (k^{-1}(H(m) + xr)) \mod q$
  - $r = f_2(k, p, q, g) = (g^r \mod p) \mod q$

- **Verifying**
  - $w = f_4(q, q, y, H(m), w, r) = (g^{y(w \mod q)} \cdot v^r \mod q) \mod q$
  - $v = f_3(s, q, r, q) = (s^{-1} \mod q)$
  - $y = f_2(p, q, g, y, H(m), w, r) = (g^w \mod q)$
  - $v = f_4(w, v, x, q) = (v^x \mod q)$
  - $r = f_2(k, p, q, g, y, H(m), w, r) = (g^r \mod p) \mod q$
  - $s = f_1(H(m), k, x, r, q) = (k^{-1}(H(m) + xr)) \mod q$
**DSA Signature Creation**

- to sign a message $M$ the sender generates:
  - a random signature key $k$, $k < q$
    - N.B.: $k$ must be random, be destroyed after use, and never be reused
- computes the message digest:
  $h = \text{SHA}(M)$
- then computes signature pair:
  $r = (g^k \mod p \mod q)$
  $s = k^{-1}(h + x \cdot r) \mod q$
- sends signature $(r, s)$ with message $M$

**DSA Signature Verification**

- having received $M$ & signature $(r, s)$
- to verify a signature, recipient computes:
  $w = s^{-1} \mod q$
  $v = (g^w \mod q \cdot y^w \mod q \mod p) \mod q$
- if $v = r$ then signature is verified

**Digital Certification**

- Digital certification is an application in which a certification authority "signs" a special message $m$ containing
  - the name of some user, say "Alice," and
  - her public key

  in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in Alice’s public key
- The typical implementation of digital certification involves a signature algorithm for signing the special message