Chapter 21
Public-Key Cryptography & Public-Key Infrastructure
The Concept of Public-Key Crypto.: Public-Key Encryption, Digital Signature

RSA Encryption and RSA Signature

A Crash Course in Number Theory

Why “Textbook RSA” is insecure: e.g., easy forgery, chosen-Ciphertext attack

Diffie-Hellman Protocol
  - Man-in-the-Middle Attack, and the Need for Authenticated Public Keys

ElGamal Encryption

Public-Key Cryptography in Practice, e.g., Public-Key Infrastructure
Secret Key Distributions

- How many secret keys need to be established if \( n \) people want to have pairwise confidential communication?

- Can we achieve “non-repudiation” with just shared private key?
  - No, the message sender can repudiate since the recipient also can create the MAC

- How about we assume a trusted key distribution center (KDC)
  - Security and performance problems, more details later
Public Key Cryptography (PKC)

- Every participant has a key-pair: Public Key and Private Key
- Public key is published or sent to everyone else openly
- Private key is kept secret by its owner
- Plaintext encrypted by Alice’s public key can only be decrypted by Alice’s private key
- Valid signatures (which make verification returns valid) under Alice’s public key can only be created by Alice’s private key
Why PKC is even possible?

- It is the basis of RSA, by Rivest-Shamir-Adleman.
- There is also other methods to do public-key crypto.

Multiplication is much easier to compute (than factoring)

Is 5383 a prime number?

Find \( p \) and \( q \) such that \( p \cdot q = 5383 \)?

\[ 5383 \]

\[ 769 \]

\[ \text{Secret Knowledge} \]

\[ \text{Public Key} \]

\[ \text{Private Key} \]

\[ 7 \]
Public-Key Encryption (PKE) illustrated

1. Alice generates a key pair and publish the public key

2. Bob obtains the public key of Alice

3. Bob encrypts the message under public key of Alice

4. Alice decrypts the ciphertext using the private key

Public Key := 2021

Encrypt “I hate you to “2021”

43×47
Encrypt “…” to “2021”

Public Key := 2021

Encrypt “…” to “2491”

Public Key := 2491

Encrypt “…” to “2491”

43 × 47

53 × 47
3. Bob obtains the public key of Alice and her signature

1. Alice generates a key pair and publish the public key

2. Alice creates a signature by signing on a message with her private key

4. Bob verifies the signature w.r.t. Alice’s public key

Public Key := 2021

Sign on “but I <3 you by (43, 47)
Digital Signatures & Public-Key Encryption

(actual, the lower part of the picture is not quite true...)  (the technique is very different from symmetric-key encryption)
Diffie was a graduate student in Stanford, working with Prof. Hellman on solving the “key distribution problem”.

They proposed the concept of “Public-Key Cryptosystem” (PKC) (developed jointly with Merkle).

Even more amazingly, introduce the notion of digital signature

They cannot realize it; yet, they were able to find a way for communication parties to establish a shared secret via open communications only (Diffie-Hellman Key exchange, stay tuned)

Classified one: https://www.gchq.gov.uk/person/james-ellis

with Malcolm Williamson and Clifford Cocks
Nov ’76, Diffie and Hellman published their ideas and findings in “New Directions in Cryptography” (ACM Turing award 2015)

Open problem of realizing PKC, *i.e.*, finding D(), E() s.t.,

- $D(E(m)) = m$ and
- $D(\cdot)$ can be kept secret while $E(\cdot)$ is known to the public

Ron Rivest was intrigued by DH’s paper. He then enlisted the help of Adi Shamir and Leonard Adleman, all from MIT, and came up with a solution (*i.e.*, RSA algorithm) in ’77 (ACM Turing award 2002)

Diffie, Hellman, Merkle, Rivest, Shamir, Adleman were commonly recognized as the founders of PKC
RSA Algorithm

- Most widely accepted and implemented approach to PKE
- “Block cipher” where $0 \leq m, c \leq N - 1$ for some $N$
Warmup: Consider mod 9, what are 1+8, 2+5, and 11+41?

Additive inverse of $x$ is the number we need to add to $x$ to get 0.

* e.g.: What is the additive inverse of 2 mod 9?

Multiplicative inverse: if $xy = 1 \mod n$, $x$ & $y$ are each other’s inverse

* e.g., Consider mod 9, $2 \times 5 = 10 \mod 9 = 1$

$ab \mod N = (a \mod N) \cdot (b \mod N) \mod N$

Relatively prime: no common factors other than 1

Does multiplicative inverse always exist?

* 3 mod 9? 4 mod 9? // brute-force approach: think of: 1, 2, 3, 4, 5, 6, 7, 8
“x to the power of e” = “x exponentiates with e” = \( x^e \)

- x is called the **base** and e is called the **exponent**

- Quick review: \( x^e \cdot x^f = x^{(e+f)} \), \( x^e / x^f = x^{(e-f)} \)

- We will use power arithmetic a lot

- Logarithm: “read off” the exponent
  - e.g., \( \log_{10}(100) = 2 \), \( \log_x(x^e) = e \)

- Modular arithmetic: do the operation “under” mod N for **modulus N**
  - e.g., \( 2^3 \text{ mod } 5 = 8 \text{ mod } 5 = 3 \)
RSA Key Generation

- Choose two large prime numbers \( p, q \) (e.g., 1024 bits each)
- Compute \( N = pq, \Phi(N) = (p-1)(q-1) \) // we use shorthand \( \Phi = \Phi(N) \)
- Choose \( e \) (< \( N \)) that has no common factors with \( \Phi \)
  - \( \text{i.e., } e, \Phi \text{ are “relatively prime”} \)
- Choose \( d \) such that \( ed - 1 \) is exactly divisible by \( \Phi \)
  - \( \text{i.e., } ed - 1 = K\Phi, ed = K\Phi + 1 \) for some integer \( K \)
  - \( \text{i.e., } ed \text{ mod } \Phi = 1 \) // remainder of \( ed \) divided by \( \Phi \) is 1 (useful later)
- Public key is \( (N, e) \)
- Private key is \( (N, d) \) // or just \( d \) (since \( N \) is public anyways)
RSA Encryption and Decryption

\[ \text{Enc}(m) \rightarrow c \quad \Leftrightarrow \quad c := m^e \mod N \]

\[ \text{Dec}(c) \rightarrow m \quad \Leftrightarrow \quad m := c^d \mod N \]

Correctness: \( c^d \mod N \)

\[ = (m^e)^d = (m^{ed}) \mod N \]

\[ = m^{(k\Phi+1)} \mod N = m^{k\Phi}m \mod N \]

\[ = m \mod N \quad \text{// since } m^{k\Phi} \mod N = 1 \text{ (we’ll see the theorem later)} \]
RSA KeyGen: Toy-Example

- \( p = 5, \ q = 7 \)  // random prime generation (details later)
- so \( N = 5 \times 7 = 35 \)
- \( \Phi = (5 - 1) \times (7 - 1) = 24 \)
- Pick \( e = 5 \) (so \( e, \ \Phi \) are relatively prime)
- We have \( d = 29 \)
  - Computed by extended Euclidean algorithm (details later)
  - Check: \( ed - 1 = 5 \times 29 - 1 = 144 \), and we have \( 144 / 24 = 6 \)
- We can encrypt \( \{2, 3, \ldots, 34\} \), more than enough for English alphabet
RSA E and D: Toy-Example

- Public key = $(N = 35, e = 5)$. Private key = $d = 29$, b.t.w. $\Phi = 24$

- Let’s encrypt $m = 4$, i.e., $4^5 \equiv \text{mod } 35 = 1024 \equiv 9$

- Let’s decrypt $c = 9$, note that $(a \text{ mod } N) \times (b \text{ mod } N) = ab \mod N$

  - $9^{29} \mod 35 = \{(9^{10} \mod 35) \times (9^{10} \mod 35) \times (9^9 \mod 35)\} \mod 35$
  - $(16 \times 16 \times 29) \mod 35 = 4$
  - OR Repeated squaring, $9^{29} = 9 \times (9^{14})^2 = 9 \times ((9^7)^2)^2$
  - $= 9 \times ((9 \times (9^3)^2)^2)^2 = 9 \times ((9 \times 9 \times 2)^2)^2 \equiv 29_{10} = 11101_2$
Security of RSA relies on the following “RSA assumption”

Namely, given \((N, e)\) and \(c := m^e\), cannot compute \(m\)

One can break it by finding \(p\) and \(q\), and thus \(d\) from \(e\)

Underlying assumption: cannot factor \(N\)

These two assumptions turn out to be equivalent
How secure is RSA?

- **Brute force**: try all possible keys – larger $d$ => more secure
- Larger $d$, the slower the decryption (repeated squaring)
- A 430-bit key was cracked in 1996 => Prize: USD $14,527
- 1024-bit (bit-length of $N$) is considered strong enough, for now
E() and D() are “symmetric” for RSA in the following sense:

\[ D(E(m)) = m = E(D(m)) \]

Recall Enc(m) --- \( c := m^e \mod N \), Dec(c) --- \( m := c^d \mod N \)

\[ \text{Sig}(m) --- s := m^d \mod N \]

\[ \text{Ver}(m, s) --- \text{Is } m := s^e \mod N \? // \text{verifying, not recreating (MAC)} \]

To verify a signature, use the public key to “decrypt back” to the message?

\[ \text{WRONG! It is only a property based on RSA, not true in general.} \]

This is “textbook RSA signature”: Insecure, we’ll fix it later
PKE and RSA (Summary)

 Encrypt “I hate you” to “2021”

**Key Generation**

\[ (sk, pk) \leftarrow \text{KeyGen()} \]

**Public Key**

Public Key := 2021

**Encryption**

\[ C \leftarrow \text{Enc}(pk, M) \]

**PRIME p and q**

- **N = pq**
- \( ed \equiv 1 \mod (p - 1)(q - 1) \)
- \( C = M^e \mod N \)
- \( M = C^d \mod N \)
How do you find the Multiplicative Inverse?

Applications: for RSA secret key, and for division (useful later)

An “algorithm” that you have probably learned in primary school

Some Number Theory or Abstract Algebra

So you don’t need to think about them as numbers, but just some “objects”/“variables” to help you do the cryptography

RSA’s correctness: showing why $x^{\Phi(m)} = 1 \mod m$

(An Overview of) Big Prime Number Generation

Special choice of exponent for Modular Exponentiation
Based on Greatest Common Divisor

Getting $\gcd(a, b) = au + bv$

1) Division: $a = bq_1 + r_1$, $0 \leq r_1 < b$

2) Repeat: $b = r_1q_2 + r_2$, $0 \leq r_2 < r_1$

3) Keep repeating until $r_{s+1} = 0$

We have $\gcd(a, b) = \gcd(b, r_1) = \ldots = \gcd(r_{s-1}, r_s) = r_s$
Finding Multiplicative Inverse

- Input: \((a, b)\). Output: \((u, v)\) s.t. \(\gcd(a, b) = au + bv\)

- Detailed omitted, but basically work backward in Euclidean alg.

- Existence of multiplicative inverse:
  
  \(x\) has multiplicative inverse mod \(n\) \textit{iff} \(x\) is relatively prime to \(N\)

- \(1 = au + bv\)

- \(u \times a = -v \times b + 1\)

- \(u \times a \equiv 1 \mod b\)

Two applications:

- \(u \times x \equiv 1 \mod N\)
  - \(u\) is multiplicative inverse of \(x\)

- \(d \times e \equiv 1 \mod (p-1)(q-1)\)
  - \(d\) is the RSA decryption key
A (finite) set $G$ and a “group operation” (say $*$) that is
- closed ($\forall a, b \in G$, $a * b$ is also in $G$)
- associative ($\forall a, b, c \in G$, $(a * b) * c = a * (b * c)$ holds)
- equipped with identity $e$, ($\forall a \in G$, $e * a = a * e = a$ holds)
- invertible ($\forall a \in G$, $\exists b \in G$ s.t. $a * b = b * a = e$ holds)
- commutative for Abelian group ($\forall a, b \in G$, $a * b = b * a$ holds)

Examples:
- $\mathbb{Z}$ (integers, with addition operation; infinite group)
- $G^2$ (where $G$ is a group; coordinate-wise operation)
  - e.g., $c_1 = (a_1, b_1)$, $c_2 = (a_2, b_2)$, $c_1 * c_2 = (a_1 * a_2, b_1 * b_2)$
Finite Cyclic Group

- Order of a group
  - denoted by $|G|$
  - number of elements in $G$

- Cyclic group
  - (in multiplicative notation)
  - there is one element $g$ s.t.
    - $G = \{g^0, g^1, g^2, ..., g^{|G| - 1}\}$

- E.g. $(\mathbb{Z}_N, +)$
  - $\mathbb{Z}_N = \{0, 1, ..., N - 1\}$
  - consider $g = 1$, or any $g$ s.t. $\gcd(g, N) = 1$
$\mathbb{Z}_N^*$: set of elements from $\mathbb{Z}_N$ that are relatively prime to $N$

For any $N$, 0 is not in $\mathbb{Z}_N^*$

For any prime $p$, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ // Set-minus denoted by \ 

e.g.: Consider $\mathbb{Z}_N^* = \{1, 2, 3, 4\}$ // $N = 5$

$= \{2^0, 2^1, 2^3, 2^2\}$ // 2 can generate the group

$= \{3^0, 3^3, 3^1, 3^2\}$ // 3 can also generate the group
Fermat’s Little Theorem

- Order of $a$ (in $\mathbb{Z}_p^*$) is the smallest $x$ s.t. $a^x = 1 \mod p$
- FLT: For any prime $p$, and $x$ in $\mathbb{Z}_p^*$, $x^{p-1} = 1 \mod p$
- All elements’ orders are $(p - 1)$?
- $e.g.$: Consider $p > 3$ and $(p - 1)^2$
Euler’s Theorem, and RSA

\[ f_e, N(x) = x^e \mod N, \text{ where } N = pq, \ x \ in \ Z_N^* \text{ and } e \ in \ Z_{\Phi(N)}^* \]

- Euler phi-function: For any positive integer \( m \), \( \Phi(m) \) is the number of +ve int. < \( m \) that are relatively prime to \( m \).

- Euler’s Theorem: For any int. \( m \), & \( x \ in \ Z_m^* \), \( x^{\Phi(m)} = 1 \mod m \)

- What should \( \Phi(N) \) be for \( N = pq \), \( p \) and \( q \) are prime?
Finding big primes \( p \) and \( q \)

- \( \Pr[\text{a randomly chosen number } n \text{ to be prime}] \) is \( \frac{1}{\ln n} \) (natural log)
  - So, how many times we need to try on average?
  - 230 for \( n \) of 100-digit

- Test whether a random number \( n \) is a prime

- Trial divisions of \( n \) (how many should we try?)

- Fermat’s Theorem: if \( n \) is a prime and \( 0 < a < n \), \( a^{n-1} = 1 \mod n \)
  - Carmichael numbers: non-primes but satisfy the above condition
  - For a 100-digit non-prime, \( \Pr[\text{false positive / Carmichael #}] \approx 10^{-13} \)

- Use Miller-Rabin algorithm (for your reference, details omitted)
Convenient $e$

- $e = 3$
- b.t.w., can we set $d = 3$?
- 65537
  - $2^{16} + 1$
- 3 $\rightarrow$ 2 multiplications
- 65537 $\rightarrow$ 17 multiplications

Repeated Squaring and Multiplication, e.g.:

$$x^{277} = x^{(00100010101)_{2}} = \sum_{i=0}^{10} b_{i} \times x^{2^{i}}$$
Inconveniences of $e = 3$

- What do we expect from $e$ again?
  - $e = 3$ should be relatively prime to $(p - 1)(q - 1)$
  - Easier to choose eligible primes for 65537

- Why $d$ cannot be 3 again? (Too small $\rightarrow$ Security problem)

- Likewise, what if $m < N^{1/3}$? (Too small $\rightarrow$ Security problem)
  - $c = m^3$, recover $m$ by exhaustive search?
  - NO! Modular arithmetic became “null”, we simply did not “perform” mod $N$
  - So we can just do cube-root to recover $m$ from $c = m^3$
  - Solution: “Pad” $m$ s.t. it is larger than $N^{1/3}$
RSA Signature (Recap)

- $S \leftarrow \text{Sig}(M) \quad \text{---} \quad S := M^d \mod N$

- $1/0 \leftarrow \text{Ver}(M, S) \quad \text{---} \quad \text{Is } M := S^e \mod N ?$
Textbook-RSA Signature is Insecure

\[ S_1 := M_1^d \mod N, \quad S_2 := M_2^d \mod N \]

\[ S' := S_1 S_2 \mod N = M'^d \mod N \]

where \( M' = M_1 M_2 \)

\[ \text{We expect } S^e = M \mod N \]

\[ \text{Let’s try to create a signature on a “random” message.} \]

\[ \text{Fix: } S := H(M)^d \mod N \]
CCA: An attacker can choose the ciphertext to be fed to a decryption oracle

Why we have a decryption oracle?

“Lunch-time” attack

\[ C' := M'^e \mod N \]

\[ C^* = X^e C' := (XM')^e \mod N \]
The algorithm you have seen is often called “Textbook RSA”

Textbook RSA Encryption is deterministic: $C := M^e \mod N$

Probabilistic encryption

- Randomly pad the plaintext before encryption

Optimal asymmetric encryption padding (OAEP)

- Details omitted [**]
Factoring Milestones

- RSA-4, IBM quantum computer, ’01, successfully factored 15 into $3 \times 5$
- Quantum Factorization of 143 (’11)
- RSA-768, USD $50k, Dec ’09. You can try RSA-896, or RSA-1024, or...
- RSA-2048 (USD $200,000) =
  251959084756578934940271832400483985714292822126204032027777713783
  6043662020707595556264018525880784406918290641249515082189298559
  1491761845028084891200728449926873928072877767359714183472702618
  9637501497182469116507761337985909570009733045974880842840179742
  9100642458691817195118746121515172654632282216869987549182422433
  6372590851418654620435767984233871847744479207399342365848238242
  8119816381501067481045166037730605620161967625613384414360383390
  4414952634432190114657544454178424020924616515723350778707749817
  1257724679629263863563732899121548314381678998850404453640235273
  81951378636564391212010397122822120720357
Public-Key Cryptography Standard (PKCS)

- A list of Standards (PKCS#1 to PKCS#15) on how to use RSA in practice, regarding:
  - message formatting,
  - information encoding scheme,
  - choice of parameters, *etc.*

- Protected against the following “improper use” or attacks, *e.g.:
  - plaintext guessing
  - chosen ciphertext attack
  - \( m^3 < N \)
  - sending the same message to multiple people
The first public key cryptosystem ("crypto.-related system")

- But does neither encryption nor signatures
- Used for "key exchange" --- establish a shared secret key

If interaction is allowed, how to ensure "confidentiality"?

- Alice and Bob negotiate a shared secret key over a public communication channel
- First rough idea: Each party contributes some randomness?
Diffie-Hellman illustrated (from Wiki)
All computations in mod $q$ (a large prime), $g$ (in $\mathbb{Z}_q$) and $q$ are public
Discrete logarithm problem (DLP):
Knowing \( q, g \) and \( g^x \mod q \), it is difficult to compute \( x \)

DL assumption is necessary for security: \( i.e., \) assume DLP is hard

“Hard” corr. to how many possible session keys can we have.
Diffie-Hellman Problem

- DL assumption is necessary, but not sufficient!
  - Recovering $x$ from $g^x$ is not the only way to compute $g^{xy}$
  - cf. Computing $g^{x+y}$ from $g, g^x, g^y$, you don’t need to recover $x$

- (Computational) Diffie-Hellman (CDH) Problem:
  Given $g, g^x, g^y$, compute $g^{xy}$

- CDH assumption: We assume the CDH problem is hard to solve
Man in the middle attack

$K' = g^{xz}$

I know $z$
I can compute both $K'$ and $K''$ 😜
(And I no longer care about $g^{xy}$)

$X = g^x$
$Z = g^z$
$Y = g^y$

$K'' = g^{yz}$
Use Case of Diffie-Hellman Key Exchange

- $X$ and $Y$ are randomly generated every time.
  - requires online presence and interaction.

Why man-in-the-middle attack is possible?

- Because there is no authenticity.
How about considering $X$ and $Y$ are public keys instead?
- *e.g.*, the public key is retrieved before, it is available from an online repository, or the other online participant sends it directly.

Wait, why you will trust that $X$ is really my public key?
- OK, I can use the secret key $x$ to sign that.
- Self-signing, it doesn’t really solve anything...
- The adversary come up with $x$, and use $x$ to sign on $X$.
- The crux of the problem is, $X$ is a random looking string.
- $X$ has nothing to do with the identity of the owner!
1. Authentication and Certification

Name := Alice
Public Key := E7143402FA80CEBED912...

Encrypt to “E7143402...”

Certificate Authority

2. Bob obtains and verifies the certificate of Alice

Alice, a.k.a. E7143402...?!

3. Alice decrypts the ciphertext

E7143402FB08CEBED912912
Encryption from Diffie-Hellman

Alice conceptually “contributes” her public-key $Y = g^y$

- Bob then “completes” the key-exchange, and use the session key as an “one-time pad” of the message $M$ immediately

$$Y = g^y$$

$$C_0 = MY^r, C_1 = g^r$$

- Alice then derives the same session key and unwraps the padding
ElGamal Encryption (Key Generation)

- \((sk = y, pk = Y = g^y) \leftrightarrow \text{KeyGen}(q)\)

- Assume we have picked a large prime \(q\)
  - which can be made public here, different from RSA

- Pick \(1 \leq y \leq q - 2\), set secret key \((sk)\) be \(y\)

- Compute \(Y = g^y \mod q\), set public key \((pk)\) be \(Y\) (or \((Y, q)\))
Recall \((s_k = y, p_k = Y = g^y) \leftarrow \text{KeyGen}(q)\)

\[(C_0 = M(p_k)^r, C_1 = g^r) \leftarrow \text{Enc}(p_k, M)\]

- Pick \(1 \leq r \leq q - 2\)
- Set \(C_1\) to be \(g^r \mod q\)
- Compute \(K = Y^r \mod q\)
- Set \(C_0\) to be \(MK \mod q\)
ElGamal Decryption

Recall:

- \((sk = y, pk = Y = g^y) \leftarrow \text{KeyGen}(q)\)
- \((C_0 = M(pk)^r, C_1 = g^r) \leftarrow \text{Enc}(pk, M)\)

\((M = C_0 / C_1^{sk}) \leftarrow \text{Dec}(sk, C)\)

- Recover \(K' := C_1^y \mod q\)
- We have \(K' = (g^r)^y = (g^y)^r = (pk^r)\)
- Recover \(M' := C_0 / K' // \text{recall Fermat’s: } x^{-1} \mod q = x^{q-2} \mod q\)
ElGamal (Toy Example)

- **KeyGen:** $q = 11$, $g = 2$
  - Suppose we picked $y = 3$, $Y = 2^3 \mod 11 = 8$

- **Enc($M = 7$):**
  - Suppose we picked $r = 4$ ($1 \leq r \leq 11 - 2$)
  - $C_1 = 2^4 \mod 11 = 5$
  - $C_0 = 8^4 \times 7 \mod 11 = (4096 \times 7) \mod 11$ (or $(-3)^4 \times 7) = 6$

- **Dec($(C_0 = 6, C_1 = 5)$):**
  - $6 \times (5^3)^{-1} \mod 11 = (6 \times 3) \mod 11 = 7$
  - $(5^3)^{-1} \mod 11 = (125)^{11-2} \mod 11 = [(4^3 \mod 11)^3] \mod 11$
  - $= 9^3 \mod 11 = 729 \mod 11 = 3$ \(\text{or} a \mod q \times b \mod q = (ab) \mod q\)
Is the encryption deterministic/probabilistic?
- How about RSA encryption? Why it is bad?

Is the modulus a prime or a composite number?
- How large is the modulus for RSA? Why it should be larger?

Is the encryption homomorphic?
- Is it good or bad? [*]
- RSA is multiplicative homomorphisms: $\text{Enc}(m_0) \times \text{Enc}(m_1) = \text{Enc}(m_0 m_1)$
- $(m_0^e) \times (m_1^e) = (m_0 m_1)^e$
- ElGamal: In more details, $\text{Enc}(m_0; r_0) \times \text{Enc}(m_1; r_1) = \text{Enc}(m_0 m_1; r_0 + r_1)$
- Publicly re-randomizable, *i.e.*, from $\text{Enc}(m; r)$, create $\text{Enc}(m; r')$ [*]
It is not patented (but someone claims it is covered by the DH patent)

The ciphertext is about twice as big as the plaintext

The scheme is not that popular in practice

But in research world, it is popular as a (conceptual) building block

How about signature?

The Digital Signature Algorithm (DSA) proposed by NIST as the US Digital Signature Standard (DSS) is a variant of the ElGamal’s scheme
Elliptic Curve Cryptosystems (ECC)

- Based on elliptic-curve discrete logarithm problem (based on group arithmetic)
- Independently proposed in 1985 by Koblitz (U. of Washington) and Miller (IBM)
  - the first true alternative for RSA, esp. in selected areas, e.g., embedded, wireless/mobile sys.
  - also see https://www.iacr.org/people/Vanstone.html
  - https://en.wikipedia.org/wiki/BlackBerry_Limited#Certicom
- The 109-bit challenges have been solved in Nov, ’02 and Apr, ’04
- The 131-bit challenges require significantly more resources to solve
- All level II challenges are believed to be computationally infeasible (this table→)

<table>
<thead>
<tr>
<th>ECC KEY SIZE (Bits)</th>
<th>RSA KEY SIZE (Bits)</th>
<th>KEY SIZE RATIO</th>
<th>AES KEY SIZE (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>1024</td>
<td>1 : 6</td>
<td>128</td>
</tr>
<tr>
<td>256</td>
<td>3072</td>
<td>1 : 12</td>
<td>192</td>
</tr>
<tr>
<td>384</td>
<td>7680</td>
<td>1 : 20</td>
<td>256</td>
</tr>
<tr>
<td>512</td>
<td>15 360</td>
<td>1 : 30</td>
<td></td>
</tr>
</tbody>
</table>

Supplied by NIST to ANSI X9F1
ECC vs. RSA

- Fast and compact implementations, especially in hardware
- Shorter history of cryptanalysis
  - since early 90’s
- Complex mathematical description
- No patents for the cryptosystems themselves, but many on the implementation and optimization

- Widely deployed and used
- Efficient software implementation
- But larger group element, and hence slower group operation
- Longer history of cryptanalysis
  - (since late 70’s)
- Patent expired in 2000
PKI, in Algorithms

- \((sk_{CA}, pk_{CA}) \leftarrow \text{KeyGen()} // \text{by CA, for signing}\)
- \((sk_{user}, pk_{user}) \leftarrow \text{KeyGen()} // \text{by any user, for signing or encryption}\)
- \(Cert_{user} = \sigma_{CA} = \text{Sign}(sk_{CA}, pk_{user} | | \text{“info”})\)
  - \(i.e., \text{message to be signed is } m = pk_{user} | | \text{“info”}\)
  - \(\text{“info” = “CA’s info” | | “user’s info” | | etc.}\)

To encrypt message \(m’\) to userA
- Retrieves \(Cert_{userA}\)
- If \(\text{Verify}(pk_{CA}, \sigma_{CA}, pk_{userA} | | \text{“HKPOST | | userA”}) = \text{“True”}\)
- Sends \(\text{Enc}(pk_{userA}, m’)\) to userA
Information in a Certificate

- Issuer (i.e., CA)’s identity
- Cert. owner (i.e., user)’s identity
- Serial number (unique to issuer)
- Algorithm used by the owner (e.g., signature, hash)
- Validity period
- Usage (e.g., email-user, network site, software developer, etc.)
Example: Root CA and Certificate
Some issues of PKI beyond basic functionality / efficiency:

- Key generation
  - bad randomness in user-side key generation

- Certification
  - cert user needs to prove ownership of the public-key to the CA

- Distribution
  - chicken-and-egg situation, who encrypt to me before I got a cert?

- Escrow
  - Why? law by government / policy by company / any backup need
PKI in Practice

- Single agency responsible for certifying public keys
  - Unrealistic: monopolistic, political concerns
  - Other models: Oligarchy (small # of CA), Anarchy (how? [**]), etc.

- How to distribute the root CAs’ public keys?
  - Every computer is pre-configured with some CA’s public keys

- How the physical/off-the-band authentication take place?
  - *i.e.*, Are you really Alice?
  - We need local registration authority (LRA)
  - CA just issues cert., the job of registration is delegated to LRAs
Still, 1 root CA needs to deal with a large number of issuances

Delegated CAs

- Root CA
- Sub-ordinate CA
  - \( \text{Sign}(sk_{CA}, pk_{sub-CA}) \)
- End-user
  - \( \text{Sign}(sk_{sub-CA}, pk_{user}) \)
I am a competitor of Verisign

I don’t want to serve so many guys
Everything has an expiration date...

What if your private key is compromised?

What if you’re removed from the duty & no longer authorized?

We need to “pre-maturely” revoke a certificate

Notorious issue

Certification revocation list (CRL)

CA keeps blacklisting users, users keeps checking this CRL
Components of PKI

- Certificate Authority (CA)
  - issue, manage, and revoke certificates for a community of users

- Registration Authority (RA)
  - accept and verify registration info about new registrants
  - generate keys on behalf of users
  - accept and authorize requests for key backup, revocation
  - distribute/recover hardware tokens (storing the private key)

- Certificate Directory: provides a list of certs and CRL

- Key Recovery Server: backup / escrow held in trust of private keys

- Additional Reading: “PKI: It’s Not Dead, Just Resting” by Peter Gutmann
Diffie, Hellman, Rivest, Shamir in 2011

Leftmost: Ari Juels, Chief Scientist of RSA
Rightmost: Dickie George, Technical Director, Information Assurance, NSA