

Cryptography IV: Asymmetric Ciphers

Computer Security Lecture 5

David Aspinall

School of Informatics
University of Edinburgh

28th January 2013

Outline

Background

RSA

Diffie-Hellman

ElGamal

Summary

Outline

Background

RSA

Diffie-Hellman

ElGamal

Summary

History of Public-Key Cryptography

- ▶ Idea attributed to Diffie and Hellman in 1975, but later revealed in British classified work of James Ellis in 1971 at GCHQ.
- ▶ Basic idea: alter traditional symmetry of cryptographic protocols to convey additional info in a *public key*. The sender uses the public key to convey a secret message to the recipient, without requiring a secure channel to share key information.
- ▶ Originally presented as a means of encrypting messages. In practice, public key algorithms are used to exchange symmetric keys
 - ▶ Public keys are *key encrypting keys*
 - ▶ Symmetric keys are *data encrypting keys*
- ▶ Public keys are also used to provide integrity through *digital signatures* (later lecture)

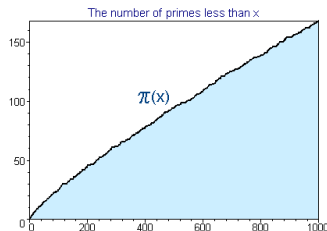
Prime numbers

- ▶ A natural number $p \geq 2$ is *prime* if 1 and p are its only positive divisors.

Prime numbers

- ▶ A natural number $p \geq 2$ is *prime* if 1 and p are its only positive divisors.
- ▶ For $x \geq 17$, then $\pi(x)$, the number of primes less than or equal to x , is approximated by:

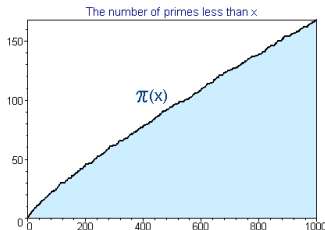
$$\frac{x}{\ln x} < \pi(x) < 1.25506 \frac{x}{\ln x}$$



Prime numbers

- ▶ A natural number $p \geq 2$ is *prime* if 1 and p are its only positive divisors.
- ▶ For $x \geq 17$, then $\pi(x)$, the number of primes less than or equal to x , is approximated by:

$$\frac{x}{\ln x} < \pi(x) < 1.25506 \frac{x}{\ln x}$$



Fundamental theorem of arithmetic

Every natural number $n \geq 2$ has a unique factorization as a product of prime powers: $p_1^{e_1} \cdots p_n^{e_n}$ for distinct primes p_i and positive e_i .

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .
- ▶ Given the factorisation of n , it's easy to compute $\phi(n)$.

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .
- ▶ Given the factorisation of n , it's easy to compute $\phi(n)$.
 - ▶ For prime n , $\phi(n) = n - 1$

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .
- ▶ Given the factorisation of n , it's easy to compute $\phi(n)$.
 - ▶ For prime n , $\phi(n) = n - 1$
 - ▶ For distinct primes p, q , $\phi(pq) = (p - 1)(q - 1)$.

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .
- ▶ Given the factorisation of n , it's easy to compute $\phi(n)$.
 - ▶ For prime n , $\phi(n) = n - 1$
 - ▶ For distinct primes p, q , $\phi(pq) = (p - 1)(q - 1)$.
- ▶ An integer n is said to be **B -smooth** wrt a positive bound B , if all its prime factors are $\leq B$.

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .
- ▶ Given the factorisation of n , it's easy to compute $\phi(n)$.
 - ▶ For prime n , $\phi(n) = n - 1$
 - ▶ For distinct primes p, q , $\phi(pq) = (p - 1)(q - 1)$.
- ▶ An integer n is said to be **B -smooth** wrt a positive bound B , if all its prime factors are $\leq B$.
 - ▶ We say just “ n is smooth” when $B \ll n$.

Relative primes

- ▶ Two integers a and b are **relatively prime** if $\gcd(a, b) = 1$, i.e., a and b have no common factors.
- ▶ The *Euler totient function* $\phi(n)$ is the number of elements of $\{1, \dots, n - 1\}$ relatively prime to n .
- ▶ Given the factorisation of n , it's easy to compute $\phi(n)$.
 - ▶ For prime n , $\phi(n) = n - 1$
 - ▶ For distinct primes p, q , $\phi(pq) = (p - 1)(q - 1)$.
- ▶ An integer n is said to be **B -smooth** wrt a positive bound B , if all its prime factors are $\leq B$.
 - ▶ We say just “ n is smooth” when $B \ll n$.
 - ▶ Smooth numbers are easy to factor, leading to efficient factoring algorithms for large composites containing primes p with for which $p - 1$ is smooth (**Pollard's $P - 1$ method**).

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n - 1\}$$

contains (equivalence classes of) integers mod n .

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n-1\}$$

contains (equivalence classes of) integers mod n .

- ▶ Let $a \in \mathbf{Z}_n$. The **multiplicative inverse** of a modulo n is the unique $x \in \mathbf{Z}_n$ such that

$$ax \equiv 1 \pmod{n}.$$

Such an x exists iff $\gcd(a, n) = 1$.

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n-1\}$$

contains (equivalence classes of) integers mod n .

- ▶ Let $a \in \mathbf{Z}_n$. The **multiplicative inverse** of a modulo n is the unique $x \in \mathbf{Z}_n$ such that

$$ax \equiv 1 \pmod{n}.$$

Such an x exists iff $\gcd(a, n) = 1$.

- ▶ We can define a *multiplicative group* \mathbf{Z}_n^* by

$$\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid \gcd(a, n) = 1\}.$$

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n-1\}$$

contains (equivalence classes of) integers mod n .

- ▶ Let $a \in \mathbf{Z}_n$. The **multiplicative inverse** of a modulo n is the unique $x \in \mathbf{Z}_n$ such that

$$ax \equiv 1 \pmod{n}.$$

Such an x exists iff $\gcd(a, n) = 1$.

- ▶ We can define a *multiplicative group* \mathbf{Z}_n^* by

$$\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid \gcd(a, n) = 1\}.$$

- ▶ Facts:

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n-1\}$$

contains (equivalence classes of) integers mod n .

- ▶ Let $a \in \mathbf{Z}_n$. The **multiplicative inverse** of a modulo n is the unique $x \in \mathbf{Z}_n$ such that

$$ax \equiv 1 \pmod{n}.$$

Such an x exists iff $\gcd(a, n) = 1$.

- ▶ We can define a *multiplicative group* \mathbf{Z}_n^* by

$$\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid \gcd(a, n) = 1\}.$$

- ▶ Facts:
 - ▶ \mathbf{Z}_n^* is closed under multiplication

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n-1\}$$

contains (equivalence classes of) integers mod n .

- ▶ Let $a \in \mathbf{Z}_n$. The **multiplicative inverse** of a modulo n is the unique $x \in \mathbf{Z}_n$ such that

$$ax \equiv 1 \pmod{n}.$$

Such an x exists iff $\gcd(a, n) = 1$.

- ▶ We can define a *multiplicative group* \mathbf{Z}_n^* by

$$\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid \gcd(a, n) = 1\}.$$

- ▶ Facts:
 - ▶ \mathbf{Z}_n^* is closed under multiplication
 - ▶ $|\mathbf{Z}_n^*| = \phi(n)$

Integers modulo n : \mathbf{Z}_n and \mathbf{Z}_n^*

- ▶ Let n be a positive integer. The set

$$\mathbf{Z}_n = \{0, \dots, n-1\}$$

contains (equivalence classes of) integers mod n .

- ▶ Let $a \in \mathbf{Z}_n$. The **multiplicative inverse** of a modulo n is the unique $x \in \mathbf{Z}_n$ such that

$$ax \equiv 1 \pmod{n}.$$

Such an x exists iff $\gcd(a, n) = 1$.

- ▶ We can define a *multiplicative group* \mathbf{Z}_n^* by

$$\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid \gcd(a, n) = 1\}.$$

- ▶ Facts:
 - ▶ \mathbf{Z}_n^* is closed under multiplication
 - ▶ $|\mathbf{Z}_n^*| = \phi(n)$
 - ▶ For prime n , $\mathbf{Z}_n^* = \{1, \dots, n-1\}$.

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

- ▶ Fermat's little theorem is used in several places, e.g. a simple probabilistic primality test:

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

- ▶ Fermat's little theorem is used in several places, e.g. a simple probabilistic primality test:
 - ▶ repeatedly test $a^{p-1} \pmod{p}$ for random a

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

- ▶ Fermat's little theorem is used in several places, e.g. a simple probabilistic primality test:
 - ▶ repeatedly test $a^{p-1} \pmod{p}$ for random a
 - ▶ Miller-Rabin improves this (*Carmichael* numbers fail)

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

- ▶ Fermat's little theorem is used in several places, e.g. a simple probabilistic primality test:
 - ▶ repeatedly test $a^{p-1} \pmod{p}$ for random a
 - ▶ Miller-Rabin improves this (*Carmichael* numbers fail)
- ▶ Euler's theorem allows reduction of large powers.

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

- ▶ Fermat's little theorem is used in several places, e.g. a simple probabilistic primality test:
 - ▶ repeatedly test $a^{p-1} \pmod{p}$ for random a
 - ▶ Miller-Rabin improves this (*Carmichael* numbers fail)
- ▶ Euler's theorem allows reduction of large powers.
 - ▶ $5^{79} \pmod{6} = (5^2 * 5^2)^{19} * 5^3 = 1^{19} * 125 \pmod{6} = 5$

Properties of integers in \mathbf{Z}_n^*

Fermat's little theorem

If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's theorem

If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

- ▶ Fermat's little theorem is used in several places, e.g. a simple probabilistic primality test:
 - ▶ repeatedly test $a^{p-1} \pmod{p}$ for random a
 - ▶ Miller-Rabin improves this (*Carmichael* numbers fail)
- ▶ Euler's theorem allows reduction of large powers.
 - ▶ $5^{79} \pmod{6} = (5^2 * 5^2)^{19} * 5^3 = 1^{19} * 125 \pmod{6} = 5$
 - ▶ Generally: if $x \equiv y \pmod{\phi(n)}$, then $a^x \equiv a^y \pmod{n}$.

Cyclic groups

- ▶ Let $a \in \mathbf{Z}_n^*$.

The *order* of a is the least $t > 0$ st $a^t \equiv 1 \pmod{n}$.

Cyclic groups

- ▶ Let $a \in \mathbf{Z}_n^*$.
The *order* of a is the least $t > 0$ st $a^t \equiv 1 \pmod{n}$.
- ▶ If $g \geq 2$ has order $\phi(n)$, then \mathbf{Z}_n^* is **cyclic** and g is a **generator** (aka *primitive root*) of \mathbf{Z}_n^* .

Cyclic groups

- ▶ Let $a \in \mathbf{Z}_n^*$.
The *order* of a is the least $t > 0$ st $a^t \equiv 1 \pmod{n}$.
- ▶ If $g \geq 2$ has order $\phi(n)$, then \mathbf{Z}_n^* is **cyclic** and g is a **generator** (aka *primitive root*) of \mathbf{Z}_n^* .
- ▶ \mathbf{Z}_n^* is cyclic iff $n = 2, 4, p^k, 2p^k$ for odd primes p .

Cyclic groups

- ▶ Let $a \in \mathbf{Z}_n^*$.
The *order* of a is the least $t > 0$ st $a^t \equiv 1 \pmod{n}$.
- ▶ If $g \geq 2$ has order $\phi(n)$, then \mathbf{Z}_n^* is **cyclic** and g is a **generator** (aka *primitive root*) of \mathbf{Z}_n^* .
- ▶ \mathbf{Z}_n^* is cyclic iff $n = 2, 4, p^k, 2p^k$ for odd primes p .
- ▶ The **discrete logarithm** of b wrt g is the x st $g^x \equiv b \pmod{n}$.

Cyclic groups

- ▶ Let $a \in \mathbf{Z}_n^*$.
The *order* of a is the least $t > 0$ st $a^t \equiv 1 \pmod{n}$.
- ▶ If $g \geq 2$ has order $\phi(n)$, then \mathbf{Z}_n^* is **cyclic** and g is a **generator** (aka *primitive root*) of \mathbf{Z}_n^* .
- ▶ \mathbf{Z}_n^* is cyclic iff $n = 2, 4, p^k, 2p^k$ for odd primes p .
- ▶ The **discrete logarithm** of b wrt g is the x st $g^x \equiv b \pmod{n}$.
- ▶ There is an efficient algorithm for computing discrete logs in \mathbf{Z}_p^* if $p - 1$ has smooth factors.

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*|$

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

		1	2	3	4
<hr/>					
1		1	2	3	4
2		2	4	1	3
3		3	1	4	2
4		4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: 2^{-1}

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3, 3^{-1} = 2$

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$.

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1 2 3 4
1	1 2 3 4
2	2 4 1 3
3	3 1 4 2
4	4 3 2 1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$.
- ▶ Notice $2^4 = 2 * 2 * 2 * 2 = 1$, also $3^4 = 4^4 = 1$.

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$.
- ▶ Notice $2^4 = 2 * 2 * 2 * 2 = 1$, also $3^4 = 4^4 = 1$.
- ▶ Generators are:

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$.
- ▶ Notice $2^4 = 2 * 2 * 2 * 2 = 1$, also $3^4 = 4^4 = 1$.
- ▶ Generators are: 2, 3, 4.

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$.
- ▶ Notice $2^4 = 2 * 2 * 2 * 2 = 1$, also $3^4 = 4^4 = 1$.
- ▶ Generators are: 2, 3, 4.
- ▶ In \mathbf{Z}_5^* , the discrete log of 4 for base 3 is

Example: \mathbf{Z}_5^*

- ▶ Here is the multiplication table for \mathbf{Z}_5^* , showing $xy \pmod{5}$.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- ▶ $|\mathbf{Z}_5^*| = \phi(5) = 4$
- ▶ Inverses: $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$.
- ▶ Notice $2^4 = 2 * 2 * 2 * 2 = 1$, also $3^4 = 4^4 = 1$.
- ▶ Generators are: 2, 3, 4.
- ▶ In \mathbf{Z}_5^* , the discrete log of 4 for base 3 is 2

Example: \mathbf{Z}_{15}^*

- ▶ Here is the multiplication table for \mathbf{Z}_{15}^* , showing xy (mod 15).

	1	2	4	7	8	11	13	14
1	1	2	4	7	8	11	13	14
2	2	4	8	14	1	7	11	13
4	4	8	1	13	2	14	7	11
7	7	14	13	4	11	2	1	8
8	8	1	2	11	4	13	14	7
11	11	7	14	2	13	1	8	4
13	13	11	7	1	14	8	4	2
14	14	13	11	8	7	4	2	1

- ▶ $|\mathbf{Z}_{15}^*| = \phi(15) = (3 - 1) * (5 - 1) = 8$.
- ▶ This group is not cyclic.

Exercise: find orders of each element.

Outline

Background

RSA

Diffie-Hellman

ElGamal

Summary

RSA

- ▶ A key-pair is based on product of two large, distinct, random secret primes, $n=pq$ with p and q roughly the same size, together with a random integer e with $1 < e < \phi$ and $\gcd(e, \phi) = 1$, where

$$\phi = \phi(n) = (p - 1)(q - 1).$$

Public key is (n, e) and n is called the *modulus*.

RSA

- ▶ A key-pair is based on product of two large, distinct, random secret primes, $n=pq$ with p and q roughly the same size, together with a random integer e with $1 < e < \phi$ and $\gcd(e, \phi) = 1$, where

$$\phi = \phi(n) = (p - 1)(q - 1).$$

Public key is (n, e) and n is called the *modulus*.

- ▶ Private key is d , unique s.t. $ed \equiv 1 \pmod{\phi}$.

RSA

- ▶ A key-pair is based on product of two large, distinct, random secret primes, $n=pq$ with p and q roughly the same size, together with a random integer e with $1 < e < \phi$ and $\gcd(e, \phi) = 1$, where

$$\phi = \phi(n) = (p - 1)(q - 1).$$

Public key is (n, e) and n is called the *modulus*.

- ▶ Private key is d , unique s.t. $ed \equiv 1 \pmod{\phi}$.
- ▶ Message and cipher space $\mathcal{M} = \mathcal{C} = \{0, \dots, n - 1\}$.

RSA

- ▶ A key-pair is based on product of two large, distinct, random secret primes, $n=pq$ with p and q roughly the same size, together with a random integer e with $1 < e < \phi$ and $\gcd(e, \phi) = 1$, where

$$\phi = \phi(n) = (p - 1)(q - 1).$$

Public key is (n, e) and n is called the *modulus*.

- ▶ Private key is d , unique s.t. $ed \equiv 1 \pmod{\phi}$.
- ▶ Message and cipher space $\mathcal{M} = \mathcal{C} = \{0, \dots, n - 1\}$.
- ▶ Encryption is exponentiation with public key e .
Decryption is exponentiation with private key d .

$$\begin{aligned} E_{(n,e)}(m) &= m^e \pmod{n} \\ D_d(c) &= c^d \pmod{n} \end{aligned}$$

RSA

- ▶ A key-pair is based on product of two large, distinct, random secret primes, $n=pq$ with p and q roughly the same size, together with a random integer e with $1 < e < \phi$ and $\gcd(e, \phi) = 1$, where

$$\phi = \phi(n) = (p - 1)(q - 1).$$

Public key is (n, e) and n is called the *modulus*.

- ▶ Private key is d , unique s.t. $ed \equiv 1 \pmod{\phi}$.
- ▶ Message and cipher space $\mathcal{M} = \mathcal{C} = \{0, \dots, n - 1\}$.
- ▶ Encryption is exponentiation with public key e .
Decryption is exponentiation with private key d .

$$\begin{aligned} E_{(n,e)}(m) &= m^e \pmod{n} \\ D_d(c) &= c^d \pmod{n} \end{aligned}$$

- ▶ Decryption works, since for some k , $ed = 1 + k\phi$ and

$$(m^e)^d \equiv m^{ed} \equiv m^{1+k\phi} \equiv mm^{k\phi} \equiv m \pmod{n}$$

using Fermat's theorem. (**Exercise:** fill details in).

RSA remarks

- ▶ Recall that RSA is an example of a **reversible** public-key encryption scheme. This is because e and d are symmetric in the definition. RSA digital signatures make use of this.

RSA remarks

- ▶ Recall that RSA is an example of a **reversible** public-key encryption scheme. This is because e and d are symmetric in the definition. RSA digital signatures make use of this.
- ▶ RSA is often used with randomization (e.g., **salting** with random appendix) to prevent chosen-plaintext and other attacks.

RSA remarks

- ▶ Recall that RSA is an example of a **reversible** public-key encryption scheme. This is because e and d are symmetric in the definition. RSA digital signatures make use of this.
- ▶ RSA is often used with randomization (e.g., **salting** with random appendix) to prevent chosen-plaintext and other attacks.
- ▶ It's the most popular and cryptanalysed public-key algorithm. Largest modulus factored in the (now defunct) RSA challenge is 768 bits (232 digits), factored using the Number Field Sieve (NFS) on 12 December 2009.

RSA remarks

- ▶ Recall that RSA is an example of a **reversible** public-key encryption scheme. This is because e and d are symmetric in the definition. RSA digital signatures make use of this.
- ▶ RSA is often used with randomization (e.g., **salting** with random appendix) to prevent chosen-plaintext and other attacks.
- ▶ It's the most popular and cryptanalysed public-key algorithm. Largest modulus factored in the (now defunct) RSA challenge is 768 bits (232 digits), factored using the Number Field Sieve (NFS) on 12 December 2009.
 - ▶ It took the equivalent of 2000 years of computing on a single core 2.2GHz AMD Opteron. On the order of 2^{67} instructions were carried out.

RSA remarks

- ▶ Recall that RSA is an example of a **reversible** public-key encryption scheme. This is because e and d are symmetric in the definition. RSA digital signatures make use of this.
- ▶ RSA is often used with randomization (e.g., **salting** with random appendix) to prevent chosen-plaintext and other attacks.
- ▶ It's the most popular and cryptanalysed public-key algorithm. Largest modulus factored in the (now defunct) RSA challenge is 768 bits (232 digits), factored using the Number Field Sieve (NFS) on 12 December 2009.
 - ▶ It took the equivalent of 2000 years of computing on a single core 2.2GHz AMD Opteron. On the order of 2^{67} instructions were carried out.
 - ▶ Factoring a 1024 bit modulus would take about 1000 times more work (soon achievable).

RSA usage and key size

- ▶ In practice, RSA is used to encrypt symmetric keys, not messages.
- ▶ Like most public key algorithms, the RSA key size is larger, and the computations are more expensive than symmetric schemes such as AES.
- ▶ This is believed to be a necessary result of the key being publicly available.
- ▶ Consider the complexity of brute force attacks for an n -bit key:
 - ▶ A worst-case attack algorithm on a symmetric cipher would take $O(2^n)$ effort (*exponential*).
 - ▶ A worst-case attack algorithm for RSA is dependent upon the complexity of factoring, and thus would take $O(e^{o(n)})$ (*sub-exponential*)
 - ▶ (unless you have a quantum computer)

Cryptographic Reference Problems I

FACTORING Integer factorization. Given positive n , find its prime factorization, i.e., distinct p_i such that $n = p_1^{e_1} \cdots p_n^{e_n}$ for some $e_i \geq 1$.

Cryptographic Reference Problems I

FACTORING Integer factorization. Given positive n , find its prime factorization, i.e., distinct p_i such that $n = p_1^{e_1} \cdots p_n^{e_n}$ for some $e_i \geq 1$.

SQRROOT Given a such that $a \equiv x^2 \pmod{n}$, find x .

Cryptographic Reference Problems I

FACTORING Integer factorization. Given positive n , find its prime factorization, i.e., distinct p_i such that $n = p_1^{e_1} \cdots p_n^{e_n}$ for some $e_i \geq 1$.

SQRROOT Given a such that $a \equiv x^2 \pmod{n}$, find x .

RSAP RSA inversion. Given n such that $n = pq$ for some odd primes $p \neq q$, and e such that $\gcd(e, (p-1), (q-1)) = 1$, and c , find m such that $m^e \equiv c \pmod{n}$.

Cryptographic Reference Problems I

FACTORING Integer factorization. Given positive n , find its prime factorization, i.e., distinct p_i such that $n = p_1^{e_1} \cdots p_n^{e_n}$ for some $e_i \geq 1$.

SQRROOT Given a such that $a \equiv x^2 \pmod{n}$, find x .

RSAP RSA inversion. Given n such that $n = pq$ for some odd primes $p \neq q$, and e such that $\gcd(e, (p-1), (q-1)) = 1$, and c , find m such that $m^e \equiv c \pmod{n}$.

Note: $\text{SQRROOT} =_p \text{FACTORING}$ and $\text{RSAP} \leq_p \text{FACTORING}$

- ▶ $A \leq_p B$ means there is a polynomial time (efficient) reduction from problem A to problem B .
- ▶ $A =_p B$ means $A \leq_p B$ and $B \leq_p A$
- ▶ So: RSAP is no harder than FACTORING.
Is it easier? An open question.

Outline

Background

RSA

Diffie-Hellman

ElGamal

Summary

Cryptographic Reference Problems II

DLP Discrete logarithm problem. Given prime p , a generator g of \mathbf{Z}_p^* , and an element $a \in \mathbf{Z}_p^*$, find the integer x , with $0 \leq x \leq p - 2$ such that $g^x \equiv a \pmod{p}$.

Cryptographic Reference Problems II

- DLP** Discrete logarithm problem. Given prime p , a generator g of \mathbf{Z}_p^* , and an element $a \in \mathbf{Z}_p^*$, find the integer x , with $0 \leq x \leq p - 2$ such that $g^x \equiv a \pmod{p}$.
- DHP** Diffie-Hellman problem. Given prime p , a generator g of \mathbf{Z}_p^* , and elements $g^a \pmod{p}$ and $g^b \pmod{p}$, find $g^{ab} \pmod{p}$.

Cryptographic Reference Problems II

- DLP** Discrete logarithm problem. Given prime p , a generator g of \mathbf{Z}_p^* , and an element $a \in \mathbf{Z}_p^*$, find the integer x , with $0 \leq x \leq p - 2$ such that $g^x \equiv a \pmod{p}$.
- DHP** Diffie-Hellman problem. Given prime p , a generator g of \mathbf{Z}_p^* , and elements $g^a \pmod{p}$ and $g^b \pmod{p}$, find $g^{ab} \pmod{p}$.

Note: $\text{DHP} \leq_p \text{DLP}$.

In some cases, $\text{DHP} =_p \text{DLP}$.

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Message 1. $A \rightarrow B: g^x \bmod p$

Message 2. $B \rightarrow A: g^y \bmod p$

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Message 1. $A \rightarrow B: g^x \bmod p$

Message 2. $B \rightarrow A: g^y \bmod p$

- ▶ A chooses random x , $1 \leq x < p - 1$, sends msg 1.

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Message 1. $A \rightarrow B: g^x \bmod p$

Message 2. $B \rightarrow A: g^y \bmod p$

- ▶ A chooses random x , $1 \leq x < p - 1$, sends msg 1.
- ▶ B chooses random y , $1 \leq y < p - 1$, sends msg 2.

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Message 1. $A \rightarrow B: g^x \bmod p$

Message 2. $B \rightarrow A: g^y \bmod p$

- ▶ A chooses random x , $1 \leq x < p - 1$, sends msg 1.
- ▶ B chooses random y , $1 \leq y < p - 1$, sends msg 2.
- ▶ B receives g^x , computes shared key $K = (g^x)^y \bmod p$.

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Message 1. $A \rightarrow B$: $g^x \bmod p$

Message 2. $B \rightarrow A$: $g^y \bmod p$

- ▶ A chooses random x , $1 \leq x < p - 1$, sends msg 1.
- ▶ B chooses random y , $1 \leq y < p - 1$, sends msg 2.
- ▶ B receives g^x , computes shared key $K = (g^x)^y \bmod p$.
- ▶ A receives g^y , computes shared key $K = (g^y)^x \bmod p$.

Diffie-Hellman key agreement

- ▶ Diffie-Hellman key agreement allows two principals to agree a shared key without authentication. Initial setup: choose and publish a large “secure” prime p and generator g of \mathbf{Z}_p^* .

Message 1. $A \rightarrow B: g^x \bmod p$

Message 2. $B \rightarrow A: g^y \bmod p$

- ▶ A chooses random x , $1 \leq x < p - 1$, sends msg 1.
 - ▶ B chooses random y , $1 \leq y < p - 1$, sends msg 2.
 - ▶ B receives g^x , computes shared key $K = (g^x)^y \bmod p$.
 - ▶ A receives g^y , computes shared key $K = (g^y)^x \bmod p$.
 - ▶ Security rests on intractability of DHP for p and g . Protocol is safe against passive adversaries, but not active ones.
- Exercise:** try some artificial examples with $p = 11$, $g = 2$. Show a MITM attack against the protocol.

Outline

Background

RSA

Diffie-Hellman

ElGamal

Summary

ElGamal encryption

- ▶ A key-pair is based on a large random prime p and generator g of \mathbf{Z}_p^* , and a random integer d . Public key: $(p, g, g^d \bmod p)$, private key: d .

ElGamal encryption

- ▶ A key-pair is based on a large random prime p and generator g of \mathbf{Z}_p^* , and a random integer d . Public key: $(p, g, g^d \bmod p)$, private key: d .
- ▶ The message space $\mathcal{M} = \{0, \dots, p-1\}$, and the encryption operation is given by selecting a random integer r and computing a pair:

$$E_{(p,g,g^d)}(m) = (e, c) \quad \text{where } \begin{aligned} e &= g^r \bmod p \\ c &= m(g^d)^r \bmod p. \end{aligned}$$

ElGamal encryption

- ▶ A key-pair is based on a large random prime p and generator g of \mathbf{Z}_p^* , and a random integer d . Public key: $(p, g, g^d \bmod p)$, private key: d .
- ▶ The message space $\mathcal{M} = \{0, \dots, p-1\}$, and the encryption operation is given by selecting a random integer r and computing a pair:

$$E_{(p,g,g^d)}(m) = (e, c) \quad \text{where } \begin{aligned} e &= g^r \bmod p \\ c &= m(g^d)^r \bmod p. \end{aligned}$$

- ▶ Decryption takes an element of ciphertext $\mathcal{C} = \mathcal{M} \times \mathcal{M}$, and computes:

$$D_d(e, c) = e^{-d} c \bmod p \quad \text{where } e^{-d} = e^{p-1-d} \bmod p.$$

ElGamal encryption

- ▶ A key-pair is based on a large random prime p and generator g of \mathbf{Z}_p^* , and a random integer d . Public key: $(p, g, g^d \bmod p)$, private key: d .
- ▶ The message space $\mathcal{M} = \{0, \dots, p-1\}$, and the encryption operation is given by selecting a random integer r and computing a pair:

$$E_{(p,g,g^d)}(m) = (e, c) \quad \text{where } e = g^r \bmod p \\ c = m(g^d)^r \bmod p.$$

- ▶ Decryption takes an element of ciphertext $\mathcal{C} = \mathcal{M} \times \mathcal{M}$, and computes:

$$D_d(e, c) = e^{-d} c \bmod p \quad \text{where } e^{-d} = e^{p-1-d} \bmod p.$$

- ▶ Decryption works because $e^{-d} = g^{-dr}$, so

$$D_d(e, c) \equiv g^{-dr} m g^{dr} \equiv m \pmod{p}.$$

ElGamal encryption

- ▶ A key-pair is based on a large random prime p and generator g of \mathbf{Z}_p^* , and a random integer d . Public key: $(p, g, g^d \bmod p)$, private key: d .
- ▶ The message space $\mathcal{M} = \{0, \dots, p-1\}$, and the encryption operation is given by selecting a random integer r and computing a pair:

$$E_{(p,g,g^d)}(m) = (e, c) \quad \text{where } e = g^r \bmod p \\ c = m(g^d)^r \bmod p.$$

- ▶ Decryption takes an element of ciphertext $\mathcal{C} = \mathcal{M} \times \mathcal{M}$, and computes:

$$D_d(e, c) = e^{-d} c \bmod p \quad \text{where } e^{-d} = e^{p-1-d} \bmod p.$$

- ▶ Decryption works because $e^{-d} = g^{-dr}$, so

$$D_d(e, c) \equiv g^{-dr} m g^{dr} \equiv m \pmod{p}.$$

- ▶ This is like using Diffie-Hellman to agree a key g^{dr} and encrypting m by multiplication.

ElGamal remarks

- ▶ ElGamal is an example of a **randomized** encryption scheme, so no need to add salt. Security relies in intractability of DHP. Choosing different r for different messages is critical. **Exercise:** why?

ElGamal remarks

- ▶ ElGamal is an example of a **randomized** encryption scheme, so no need to add salt. Security relies in intractability of DHP. Choosing different r for different messages is critical. **Exercise:** why?
- ▶ Efficiency:

ElGamal remarks

- ▶ ElGamal is an example of a **randomized** encryption scheme, so no need to add salt. Security relies in intractability of DHP. Choosing different r for different messages is critical. **Exercise:** why?
- ▶ Efficiency:
 - ▶ ciphertext twice as long as plaintext

ElGamal remarks

- ▶ ElGamal is an example of a **randomized** encryption scheme, so no need to add salt. Security relies in intractability of DHP. Choosing different r for different messages is critical. **Exercise:** why?
- ▶ Efficiency:
 - ▶ ciphertext twice as long as plaintext
 - ▶ encryption requires two modular exponentiations, which can be sped up by picking the random r with some additional structure (with care).

ElGamal remarks

- ▶ ElGamal is an example of a **randomized** encryption scheme, so no need to add salt. Security relies in intractability of DHP. Choosing different r for different messages is critical. **Exercise:** why?
- ▶ Efficiency:
 - ▶ ciphertext twice as long as plaintext
 - ▶ encryption requires two modular exponentiations, which can be sped up by picking the random r with some additional structure (with care).
- ▶ The prime p and generator g can be fixed for the system, reducing the size of public keys. Then exponentiation can be speeded up by precomputation; however, so can the best-known algorithm for calculating discrete logarithms, so a larger modulus would be warranted.

ElGamal remarks

- ▶ ElGamal is an example of a **randomized** encryption scheme, so no need to add salt. Security relies in intractability of DHP. Choosing different r for different messages is critical. **Exercise:** why?
- ▶ Efficiency:
 - ▶ ciphertext twice as long as plaintext
 - ▶ encryption requires two modular exponentiations, which can be sped up by picking the random r with some additional structure (with care).
- ▶ The prime p and generator g can be fixed for the system, reducing the size of public keys. Then exponentiation can be speeded up by precomputation; however, so can the best-known algorithm for calculating discrete logarithms, so a larger modulus would be warranted.
- ▶ The **security** of ElGamal encryption and signing is based on the intractability of the DHP for p . Several other conditions are required.

Outline

Background

RSA

Diffie-Hellman



ElGamal

Summary

Summary: Current Public Key algorithms

- ▶ **RSA, ElGamal** already described.
- ▶ **Elliptic curve** schemes. Use ElGamal techniques. Have shorter keys for same amount of security.
- ▶ **Rabin** encryption. Based on SQRROOT problem.
- ▶ Probabilistic schemes, which achieve **provable security** based on Random Oracle Method (ROM) arguments.
- ▶ **Cramer-Shoup**. Extends ElGamal with use of hash functions in critical places to provide provable security without ROM. Less efficient than ElGamal: slower and ciphertext twice as long.

References

-  Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone, editors. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1997.
Online version at <http://www.cacr.math.uwaterloo.ca/hac>.
-  Nigel Smart. *Cryptography: An Introduction*. McGraw-Hill, 2003. Third edition online: http://www.cs.bris.ac.uk/~nigel/Crypto_Book/

Recommended Reading

Chapter **11**, 12, 13 of Smart (3rd Ed).