CPSC/PMAT 669

One Way Functions and Cryptographic Key Agreement

Mike Jacobson

Department of Computer Science
University of Calgary

Topic 4

Outline

Key Agreement

Recall the *key agreement problem*:

- Before deploying a conventional cryptosystem, how do Alice and Bob agree on a common secret cryptographic key?

Solutions:

- Secure channel (slow and expensive)
- Key agreement protocol via a certain one-way function: next.

One-Way Functions

Definition 1 (One-way function)

A function f that satisfies the following two properties:

1. *Ease of Computation:* $f(x)$ is easy to evaluate for a given x.
2. *Pre-image Resistance:* Given $y = f(x)$, it is computationally infeasible to find x.

It is *not known* whether one way functions exist, but several that are *believed to be* one-way are used in cryptography.

Examples

Example 2

A pre-image resistant hash function is a one-way function.

Example 3

A secure cryptosystem (computationally infeasible to find the key) provides a one-way function. Define $C = f(x) = E_x(M)$, where M is a known piece of plaintext and x is some key. Given M and C (KTA), it should infeasible to find the key x.

We could also use $f(x) = E_x(x)$.

Application: Access Control

Secure login via one-way functions: Computer stores a table

$$(\textit{user-id}_i, f(P_i)) \ ,$$

containing user id's and images of passwords under a one-way function f — safer than storing passwords in the clear.

When a user logs in, he submits his user id *user-id* and his password P.

The computer generates $f(P)$ and checks if $(\textit{user-id}, f(P))$ is an entry in the password table.

- If yes, access is granted, if no, access is denied.

$\mathbb{Z}_m$ and $\mathbb{Z}_m^*$

Several candidate one-way functions come from *number theory*.

Define for $m \in \mathbb{N}$:

- $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ set of integers modulo m
- $\mathbb{Z}_m^* = \{a \in \mathbb{Z}_m \mid \gcd(a, m) = 1\}$ set of integers between 1 and m that are coprime to m.

Example 4

$\mathbb{Z}_{42} = \{0, 1, \ldots, 41\}$ and $\mathbb{Z}_{42}^* = \{1, 5, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41\}$.

Euler's ϕ Function

Definition 5 (Euler's ϕ Function)

Let m be a positive integer. *Euler's phi function* is defined via $\phi(m) = |\mathbb{Z}_m^*|$.

Interpretation: $\phi(m)$ is the number of integers between 1 and $m-1$ which are coprime (no common divisors) to m.

Example 6

$\phi(42) = |\mathbb{Z}_{42}^*| = \{1, 5, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41\}| = 12$

$\phi(p^n)$, p prime

Let p be a prime. Then

$$\phi(p) = p - 1 = p^0(p-1)$$
$$\phi(p^2) = p^2 - p = p^1(p-1)$$
$$\phi(p^n) = p^n - p^{n-1} = p^{n-1}(p-1) \ .$$

What about composites with more than one prime factor?

Multiplicativity of $\phi(n)$

Theorem 1

If $\gcd(m_1, m_2) = 1$, *then* $\phi(m_1 m_2) = \phi(m_1)\phi(m_2)$.

Proof.

Omitted (uses Chinese Remainder Theorem). □

Computing $\phi(n)$

Corollary 2

If the prime factorization of m is given by

$$m = \prod_{i=1}^{k} p_i^{\alpha_i}, \quad p_i \text{ prime},$$

then

$$\phi(m) = \prod_{i=1}^{k} \phi(p_i^{\alpha_i}) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i - 1) \ .$$

Example 7

$\phi(42) = \phi(2 \times 3 \times 7) = \phi(2)\phi(3)\phi(7) = 1 \times 2 \times 6 = 12.$

Euler's and Fermat's Theorems

Theorem 3 (Euler)

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

Special case $m = p$ prime:

Theorem 4 (Fermat)

If p *is prime, and* $p \nmid a$, *then* $a^{p-1} \equiv 1 \pmod{p}$.

Application: Probabilistic Primality Test

Fermat's Theorem gives rise to a fast probabilistic primality test using binary exponentiation:

- If $a^{N-1} \equiv 1 \pmod{N}$ for a few small primes $a \nmid N$, then N is probably prime (base a pseudoprime).
- If $a^{N-1} \not\equiv 1 \pmod{N}$ for any prime $a \nmid N$, then N is composite.

Example 8

$N = 15 : 11^{N-1} \equiv 11^{14} \equiv 1 \pmod{15}$, but $13^{14} \equiv 4 \pmod{15}$.

Why "probabilistic"?

Unfortunately, there are composite numbers (called *Charmichael numbers*) for which $a^{N-1} \equiv 1 \pmod{N}$ for all integers a.

- Thus, this method cannot *prove* primality.

The smallest Carmichael number is 561. The next few are 1105, 1729, 2465, 2821, 6601, 8911.

- Even worse: it has been proved that there are infinitely many Carmichael numbers.
- The good news is that they are very rare, so this test will work well for most integers.

Primitive Roots

Recall that for any prime p :

- $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ is the set of integers modulo p;
- $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} = \{1, 2, \ldots, p-1\}$.

Fermat's theorem asserts that $a^{p-1} \equiv 1 \pmod{p}$ for any $a \in \mathbb{Z}_p^*$. What about smaller powers of a?

Definition 9 (Primitive Root)

For a prime p, a *primitive root of p (generator of $\mathbb{Z}_p^*$)* is an element $g \in \mathbb{Z}_p^*$ such that the smallest positive integer k with $g^k \equiv 1 \pmod{p}$ is $p-1$.

Example

Generators yield the longest possible cycle of powers modulo p.

Example 10

Is $a = 3$ a primitive root of $p = 7$? By tabulating the powers of a mod p we get

$$3^0 = 1, \quad 3^1 = 3, \quad 3^2 = 2, \quad 3^3 = 6, \quad 3^4 = 4, \quad 3^5 = 5, \quad 3^6 = 1 \,.$$

(Sequence repeats at exponent 6 by Fermat's theorem.)

- Since 6 is the smallest power of 3 yielding 1, 3 is a primitive root of 7.
- 5 is also a primitive root of 7.

There are no others (e.g. $2^3 = 1$, so 2 is not a primitive root of 7).

Properties of Primitive Roots

If g is a primitive root and $\gcd(a, p) = 1$, then $g^i \equiv a \pmod{p}$ for some i with $0 \leq i < p - 1$. In other words, every non-zero integer is a power of a primitive root of p. So

$$\mathbb{Z}_p^* = \{g^0, g^1, \ldots, g^{p-2}\} .$$

Theorem 5

For any prime p, there are exactly $\phi(p-1)$ primitive roots of p.

Example 11

For $p = 7$, there are $\phi(p-1) = \phi(6) = (3-1)(2-1) = 2$ primitive roots.

Properties of Primitive Roots, cont.

It can be shown that for sufficiently large n,

$$\phi(n) \geq C \frac{n}{\log\log(n)} ,$$

where $C \approx 1.7$. For large n, $\phi(n)$ is not much smaller than n. So that's a lot of primitive roots!

Most primes p have at least one small primitive root, i.e. most of the time, one of 2 or 3 or 5 or 7 is a primitive root of p.

Computing Primitive Roots

Suppose p is prime.

- Select some $g \in \mathbb{Z}_p^*$ and compute $g^{(p-1)/q} \pmod{p}$ for *each* prime divisor q of $p - 1$ (so this requires knowledge of the prime factorization of $p - 1$).
- If $g^{(p-1)/q} \not\equiv 1 \pmod{p}$ for each q, then g is a primitive root of p.

Best choice of g: a small prime (try 2, 3, 5, 7, ...).

Example

Example 12

$p = 19$. Select $g = 2$. $p - 1 = 18 = 2 \times 3^2$. Then

$$2^{(19-1)/2} = 2^9 \equiv 18 \not\equiv 1 \pmod{19}$$
$$2^{(19-1)/3} = 2^6 \equiv 7 \not\equiv 1 \pmod{19} .$$

Thus, 2 is a primitive root of 19.

Discrete Logarithms

Let p be a prime and g a primitive root of p. Then for every $y \in \mathbb{Z}_p^*$, there exists a unique integer x with $0 \leq x \leq p-2$ such that

$$y \equiv g^x \pmod{p} .$$

Definition 13 (Discrete Logarithm)

The integer x is the *discrete logarithm* (or *index*) of y (to base g).

Example

Example 14

If p is large ($\approx 2^{1024}$), then the function

$$f(x) \equiv g^x \pmod{p}, \quad 0 < x < p-1, \; 1 < f(x) < p$$

seems to be a one-way function, provided $p-1$ has at least one large prime factor. Computing x given $f(x)$ and g is known as the *discrete logarithm problem* (DLP).

DLP Record

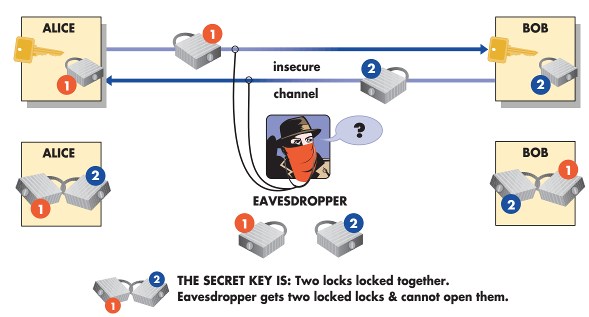
Note 1

The fastest known algorithm for extracting discrete logs is the *Number Field Sieve* which is a very complicated algorithm using extremely sophisticated number theory.

- The current NFS DL record is for the prime $p = \lfloor 10^{159}\pi \rfloor + 119849$ (160 decimal digits), Kleinjung, February 2007.

Diffie-Hellman Key Exchange: Idea

A and B wish to establish a common key for encryption over a public channel in such a way that an eavesdropper cannot determine the key.



Diffie-Hellman Key Agreement Protocol

Diffie and Hellman (1976) — still used today.

A and B agree on

- a large prime p,
- a primitive root g of p.

These quantities can be public.

Diffie-Hellman Description

A	Public Channel	B
Select a, $1 < a < p$ randomly		Select b, $1 < b < p$ randomly
$y_a \equiv g^a \pmod{p}$	$y_a \longrightarrow$	y_a
y_b	$\longleftarrow y_b$	$y_b \equiv g^b \pmod{p}$
$K = y_b^a$		$K = y_a^b$

Note 2

- A and B get the same number K because $y_b^a \equiv (g^b)^a \equiv g^{ba} \equiv (g^a)^b \equiv y_a^b \pmod{p}$
- Can use the low order 128 bits of $H(K)$ for an AES key, where H is a cryptographically secure hash function.

Security of Diffie-Hellman

Adversary's objective: find K.

Diffie-Hellman Problem (DHP): given p, g, g^a, g^b, find g^{ab} (modulo p).

- equivalent to finding K.

Also recall: *Discrete Logarithm Problem (DLP):* given p, g, g^a, find a.

- If an adversary can solve an instance of the DLP, he can solve the DHP (i.e., DHP $\leq_P$ DLP).
- Other direction unknown, i.e., if there are ways of solving the DHP, i.e., attacking DH key agreement, other than extracting discrete logs.

Parameter Choice

In order to make DLP attacks as difficult as possible, a popular choice for p is a *Sophie Germain* prime (aka *strong* or *safe* prime), i.e. a prime of the form $p = 2q + 1$ with q prime.

Why? Because $p - 1 = 2q$, and thus as as large a prime divisor as possible

Man-in-the-Middle Attack

Consider the following (active) attack:

- Eve intercepts g^a from Alice and g^b from Bob.
  - She selects e, $1 < e < p$ and sends g^e to both Alice and Bob.
  - Alice now thinks that g^e is g^b, and Bob thinks g^e is g^a.
- Alice computes what she thinks is $(g^b)^a$, but in fact computes g^{ea}.
- Bob computes what he thinks is $(g^a)^b$, but in fact computes g^{eb}.
- Eve computes $(g^a)^e$ (which is what Alice thinks is the key) and $(g^b)^e$ (which is what Bob thinks is the key).

Man-in-the-Middle Attack, consequence

If Alice sends a message encrypted with g^{ea} to Bob:

- Eve intercepts it, decrypts it with g^{ea}, re-encrypts it with g^{eb} and sends it on to Bob.
- Bob decrypts it unsuspectingly and in his perspective correctly using g^{eb}.

Similarly, Eve can read all traffic from Bob to Alice.

Issues

Solution: keys need to be *entity-authenticated* (i.e. verified as belonging to the correct person).

- This is done using digital signatures, which we'll discuss later on.

Man-in-the-middle attack: example of can happen when adversarial models are too weak

- Basic (un-authenticated, or anonymous) DH is provably secure against passive adversaries (can only eavedrop)
- Easily defeated by active adversary

Be aware of cryptography textbooks that only focus on the mathematics and ignore these issues!

Efficiency of Diffie-Hellman

How efficient is DH key agreement?

- In other words, how fast is it to evaluate modular powers?

Goal: Efficiently evaluate $a^n \pmod{m}$ given a, n, m.

One example: binary exponentiation

- based on the binary expansion of n :

$$n = b_0 2^k + b_1 2^{k-1} + \cdots + b_{k-1} 2 + b_k$$

where $b_0 = 1$, $b_i \in \{0, 1\}$, $1 \le i \le k$, $k = \lfloor \log_2 n \rfloor$.

Binary Exponentiation: Idea

Given $b_0, \ldots, b_k$, we can evaluate n efficiently using *Horner's Method*:

$$n = 2(\ldots(2(2b_0 + b_1) + b_2)\cdots + b_{k-1}) + b_k \ .$$

Define $s_0 = b_0$, $s_{i+1} = 2s_i + b_{i+1}$ for $0 \le i \le k-1$. Then

$$\begin{aligned} s_0 &= b_0 \\ s_1 &= 2b_0 + b_1 \\ s_2 &= 2(2b_0 + b_1) + b_2 = 2^2 b_0 + 2b_1 + b_2 \\ &\vdots \\ s_k &= n \ . \end{aligned}$$

One can formally prove (using induction on i):

$$s_i = \sum_{j=0}^{i} b_j 2^{i-j} \quad \text{for } 0 \le i \le k$$

Binary Exponentiation: Description

For $0 \le i \le k$, define

$$r_i \equiv a^{s_i} \pmod{m} \ .$$

Then $r_k \equiv a^{s_k} \equiv a^n \pmod{m}$ and we can compute r_k iteratively as follows:

$$\begin{aligned} r_0 &\equiv a^{s_0} \equiv a \pmod{m} \\ r_1 &\equiv a^{s_1} \equiv a^{2b_0+b_1} \equiv (a^{s_0})^2 a^{b_1} \equiv (r_0)^2 a^{b_1} \pmod{m} \\ &\vdots \\ r_{i+1} &\equiv a^{s_{i+1}} \equiv a^{2s_i+b_{i+1}} \equiv (a^{s_i})^2 a^{b_{i+1}} \equiv (r_i)^2 a^{b_{i+1}} \pmod{m} \ . \end{aligned}$$

Binary Exponentiation: Algorithm

The actual algorithm:

1. Initialize $r_0 = a$.
2. for $0 \le i \le k-1$ compute

$$r_{i+1} = \begin{cases} r_i^2 \mod m & \text{if } b_{i+1} = 0 \ , \\ r_i^2 a \mod m & \text{if } b_{i+1} = 1 \ . \end{cases}$$

Binary Exponentiation: Analysis

What is the computational cost of this?

- k modular squarings
- $h(n)$ modular multiplications by a, where $h(n)$ is the *Hamming weight* of n, i.e. the number of '1's in the binary expansion of n.

Total cost: at most $2\log_2(n)$ modular multiplications.

Also note that all intermediate operands are smaller than m^2

- Important that r_i is reduced modulo m after every operation

Looking Ahead

Solutions to the key establishment problem:

1. Diffie-Hellman key agreement protocol
2. Public key cryptography — next!
   - also used for authentication — later!