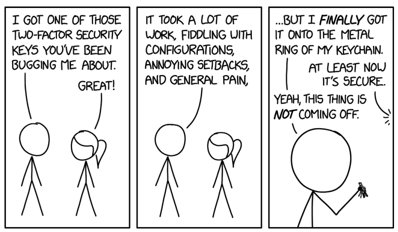
CPSC 418/MATH 318 Introduction to Cryptography

More Number Theory, Security and Efficiency of Diffie-Hellman

Renate Scheidler

Department of Mathematics & Statistics
Department of Computer Science
University of Calgary

Week 6



Outline

Recap: Primitive Roots

Let p be a prime

- **Fermat's Little Theorem:** $a^{p-1} \equiv 1 \pmod{p}$ for every integer a with $p \nmid a$.
- **Def'n of primitive root:** an integer $g \in \mathbb{Z}$ such that the smallest positive exponent k with $g^k \equiv 1 \pmod{p}$ is $p-1$.
- **Equivalent characterization of primitive roots:** Every element of $\mathbb{Z}_p^*$ is a unique power of a primitive root of p:

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

- **Primitive Root Test:** g is a primitive root of p iff $g^{(p-1)/q} \not\equiv 1 \pmod{p}$ for every prime factor q of $p-1$.

Question: how many primitive roots are there for a prime p?

Integers Modulo Composite Numbers

Define for $m \in \mathbb{N}$ (set of positive integers):

- $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ 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 (no common divisors with m).

These are generalizations of $\mathbb{Z}_p$ and Z_p^* for to arbitrary integers.

Example 1

$\mathbb{Z}_{28} = \{0, 1, \ldots, 27\}$ and $\mathbb{Z}_{28}^* = \{1, 3, 5, 9, 11, 13, 15, 17, 19, 23, 25, 27\}$.

Euler's ϕ Function

How many primitive roots are there for a given prime p? That number is determined by the *Euler phi function* of $p-1$.

Definition 2 (Euler's ϕ Function)

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

Interpretation: $\phi(m)$ is the number of integers between 1 and $m-1$ which are coprime to m.

Example 3

$\phi(28) = |\mathbb{Z}_{28}^*| = |\{1, 3, 5, 9, 11, 13, 15, 17, 19, 23, 25, 27\}| = 12$

ϕ on Prime Powers

Let p be a prime. Then

$$\begin{aligned}
\phi(p) &= p - 1 = p^0(p-1) \\
\phi(p^2) &= p^2 - p = p^1(p-1) \\
&\vdots \\
\phi(p^n) &= p^n - p^{n-1} = p^{n-1}(p-1) \ .
\end{aligned}$$

What about composites with more than one prime factor?

Theorem 1

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

In other words, Euler's phi function is *multiplicative*.

Computing ϕ in General

Corollary 2

If the prime factorization of m is given by

$$m = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}, \quad p_i \text{ prime},$$

then

$$\begin{aligned}
\phi(m) &= \phi(p_1^{e_1})\phi(p_2^{e_2}) \cdots \phi(p_k^{e_k}) \\
&= p_1^{e_1-1}(p_1 - 1) p_2^{e_2-1}(p_2 - 1) \cdots p_k^{e_k-1}(p_k - 1) \ .
\end{aligned}$$

Example 4

$\phi(28) = \phi(2^2 \times 7) = \phi(2^2)\phi(7) = 2^{2-1}(2-1) \times (7-1) = 12.$

Euler's Theorem

Recall *Fermat's Little Theorem*:

Theorem 3 (Fermat)

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

The generalization to composite numbers is *Euler's Theorem*:

Theorem 4 (Euler)

If a and m are integers with $m > 0$ *and* $\gcd(a, m) = 1$, *then* $a^{\phi(m)} \equiv 1 \pmod{m}$.

Fermat's Little Theorem is the special case of Euler's Theorem with $m = p$ prime.

Sizes of $\phi(m)$ Versus m

For any prime p, we have $\phi(p) = p - 1 \lessapprox p$ (for p large).

How does $\phi(m)$ compare to m in general? For $m \geq 2$, we have

$$\phi(m) \geq \frac{m}{e^\gamma \log\log(m) + \frac{2.5}{\log\log(m)}} \quad \text{(Rosser and Schoenfeld 1962)}$$

where $\gamma = \lim_{n\to\infty} \left(\sum_{k=1}^{n} \frac{1}{k} - \log(n) \right) \approx 0.577$ (*Euler-Mascheroni* constant).

So $\phi(m)$ grows only marginally slower than $m/e^\gamma \approx 0.573m$.

Euler's Theorem and Primitive Roots

Theorem 5

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

Example 5

The number of primitive roots for $p = 7$ is

$$\phi(p-1) = \phi(6) = \phi(3 \cdot 2) = \phi(3)\phi(2) = (3-1)(2-1) = 2 .$$

We saw earlier that they are 3 and 5.

Example 6

For $p \approx 2^{1024}$, we have $\phi(p-1) \approx (p-1)/14$. So roughly one in 14 elements in $\mathbb{Z}_p^*$ (about 7%) is a primitive root. We expect to find one after 14 random guesses.

Diffie-Hellman Description – Recap

Public:

- Large prime p,
- Primitive root g of p $(1 < g < p)$.

Alice	Public channel	*Bob*
Selects random a $(1 < a < p-1)$		Selects random b $(1 < b < p-1)$
$y_a \equiv g^a \pmod{p}$	$\xrightarrow{y_a}$	y_a
y_b	$\xleftarrow{y_b}$	$y_b \equiv g^b \pmod{p}$
$K \equiv {y_b}^a \pmod{p}$		$K \equiv {y_a}^b \pmod{p}$

Shared key: $K \equiv {y_b}^a \equiv {y_a}^b \equiv g^{ba} \pmod{p}$.

Diffie-Hellman — Questions

How secure is this?

- How difficult is for an eavesdropper it to find K?
- In general, how should p and g be chosen to maximize security?

How efficient is this?

- How easy is it to find suitable values for p and g?
- How long does it take to compute $y_a \equiv g^a \pmod{p}$ from g and a (also y_b and K)?

Security of Diffie-Hellman

Adversary's objective: find K.

Diffie-Hellman Problem (DHP):

$$\text{Given } p,\ g,\ g^a \pmod p,\ g^b \pmod p, \text{ find } g^{ab} \pmod p.$$

- equivalent to finding K.

Recall the *Discrete Logarithm Problem (DLP):*

$$\text{Given } p,\ g,\ g^x \pmod p, \text{ find } x.$$

- If an adversary can solve an instance of the DLP, she can solve the DHP.
- It is unknown if there are ways of solving the DHP, and hence breaking DH key agreement, other than extracting discrete logs.

DLP Algorithms and Record

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

Note 1

The current NFS DL record is for the prime $p = \text{RSA-240} + 49204$ (798 bits, 240 decimal digits), held by Boudot-Gaudry-Guillevic-Heninger-Thomé-Zimmerman (December 2019):

$$\begin{aligned} &\log_5(774356626343973985966622216006087686926705588649958206166317147722421706101723470351970238538755049093424997) \\ = \ &926031359281441953630949553317328555029610991914376116167294204758987445623653667881005480990720934875482587528029233326 \\ &447367244150096121629264809207598195062213366889859186681126928982506005127728321426751244111412371767375547225045851716 \end{aligned}$$

Another algorithm for extracting discrete logs, due to Pohlig and Hellman, is very efficient if $p-1$ is *smooth*. i.e. has only small prime factors. Its run time is governed by the largest prime factor of $p-1$.

Diffie-Hellman – Best Choice for p

The best choice for p is a *safe* prime, i.e. a prime of the form

$$p = 2q + 1 \quad \text{with } q \text{ prime}.$$

Such a q is called a *Sophie Germain* prime.

- $p - 1 = 2q$ has a prime factor that is as large as possible, thus foiling Pohlig-Hellman attacks.
- Lots of primitive roots of p: for $q \neq 2$ (so $p \geq 7$), we have

$$\phi(p-1) = \phi(2)\phi(q) = 1 \cdot (q-1) = \frac{p-3}{2} \approx \frac{p}{2}.$$

  In fact, for any primitive root g of p, the $(p-3)/2$ primitive roots of p are precisely the odd powers of g except g^q.
- Optimizes primitve root choices and test.

p is found by first finding a prime q (1023 bits) and then checking that $p = 2q + 1$ is prime.

Diffie-Hellman – Best Choice for g

Best choice for g: any primitive root of p (in practice ideally a small one)

- Maximizes the number of possible values K (every element in $\mathbb{Z}_p^*$ is a possible key).
- Assuming $p = 2q + 1$ is a safe prime (i.e. q a Sophie-Germain prime):
  - g is easily found via random choices because almost half of all integers modulo p are primitive roots of p.
  - Either 2 or q is a primitive root of p (but never both).
  - Primitive root test is cheap: need only choose $1 < g < p - 1$ and $g^q \not\equiv 1 \pmod p$ as $g^2 \equiv 1 \pmod p$ iff $g \equiv \pm 1 \pmod p$.

  (See the MATH 318 Problems on Assignment 2.)

Man-in-the-Middle Attack Against Diffie-Hellman

AKA "monster-in-the-middle", "machine-in-the-middle" or "monkey-in-the-middle" attack for gender neutrality. We can also use "Mallory-in-the-middle".

This is an active attack (omit all "mod p"s to avoid clutter).

- Mallory intercepts g^a from Alice and g^b from Bob.
- She selects e 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 $(g^e)^a$.
- Bob computes what he thinks is $(g^a)^b$, but in fact computes $(g^e)^b$.
- Mallory computes $(g^a)^e$ (which is what Alice thinks is the key) and $(g^b)^e$ (which is what Bob thinks is the key).

Consequence of MITM attack

Mallory now shares the key g^{ea} with Alice and the key g^{eb} with Bob.

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

- Mallory 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 uses the key g^{ab} (mod p).

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

Even worse - she can modify it!

Summary of MITM Attack

Schematic of MITM (all "mod p"s again omitted).

Alice		*Mallory*		*Bob*
a		e		b
g^a	$\longrightarrow$	$g^a \mid g^e$	$\longrightarrow$	g^e – thinks this is g^a
g^e – thinks this is g^b	$\longleftarrow$	$g^e \mid g^b$	$\longleftarrow$	g^b
$(g^e)^a$ – thinks this is $(g^b)^a$		$(g^a)^e$, $(g^b)^e$		$(g^e)^b$ – thinks this is $(g^a)^b$

Encrypts M with g^{ea}	$\longrightarrow$	Decrypts M with g^{ea} Re-enrypts M with g^{eb}	$\longrightarrow$	Decrypts M with g^{eb}
Decrypts M' with g^{ea}	$\longleftarrow$	Decrypts M' with g^{eb} Re-encrypts M' with g^{ea}	$\longleftarrow$	Encrypts M' with g^{eb}

Protection Against MITM

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.

MITM attack is an example of *protocol failure* that 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

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

Generating Primes

Recall

Fermat's Little Theorem

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

Given N (which may or may not be prime), let $a \in \mathbb{Z}_N$.

- If $a^{N-1} \not\equiv 1 \pmod{N}$, then N is composite (by Fermat).
- If $a^{N-1} \equiv 1 \pmod{N}$, then N could be prime, or it could be composite in which case it is referred to as a "base a pseudoprime".

Example 7

$N = 15$: $13^{N-1} \equiv 13^{14} \equiv 4 \pmod{15}$, so 15 is not a prime.
$11^{14} \equiv 1 \pmod{15}$, so 15 is a base 11 pseudoprime.

The Fermat Primality Test

Input: N
Output: "prime" or "composite".

1. Generate random $a \in \mathbb{Z}_N$.
2. If $\gcd(a, N) > 1$, output "composite" and stop.
3. If $a^{N-1} \not\equiv 1 \pmod{N}$, output "composite", else output "prime".

The "else" clause in step 3 may produce a lie. Provably, this test lies with expected probability $\leq 1/2$, but in practice, it rarely lies.

To obtain a large prime:

1. Generate a random number N of the desired size
2. trial-divide N by all small primes (say up to a trillion)
3. If N passes step 2 (i.e. has no small prime factors), run the Fermat test on N for a few small prime bases a. If N passes, declare N prime.

Is this Fool-Proof?

Unfortunately, there are composite numbers (called *Carmichael numbers*) for which $a^{N-1} \equiv 1 \pmod{N}$ for ALL $a \in \mathbb{Z}_N^*$.

- Thus, the Fermat test *always* lies for Carmichael numbers N.

The smallest Carmichael number is $561 = 3 \cdot 11 \cdot 17$. The next few are 1105, 1729, 2465, 2821, 6601, 8911. These are all the Carmichael numbers up to 10,000.

- Even worse: it has been proved that there are infinitely many Carmichael numbers (Alford-Granville-Pomerance 1994).
- The good news is that they are very rare, so this test will give work well for most integers (and works very well in practice).

Efficient Modular Exponentiation

Recall that Diffie Hellman requires computation of g^a, g^b, $(g^a)^b$, $(g^b)^a \pmod{p}$. How efficient is DH key agreement?

- In other words, how fast is it to evaluate modular powers?
- Fast modular exponentiation is also needed in the Fermat primality test, the primitive root test, and RSA (later).

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 $b_0 = 1$, $b_i \in \{0, 1\}$ for $1 \leq i \leq k$ with $k = \lfloor \log_2 n \rfloor$.

Binary Exponentiation: Idea

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

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

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

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

Using induction on i, one can formally prove:

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

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:

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

Binary Exponentiation: Algorithm

The actual algorithm:

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

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

AKA "Square & Multiply".

A Toy Example

Compute $2^{13} \pmod{22}$.

$13 = 8 + 4 + 1 = 2^3 + 2^2 + 0 \cdot 2^1 + 2^0 = (1101)_2$, so

- $k = 3$ (one less than the number of bits in 13) and
- $b_0 = 1, \ b_1 = 1, \ b_2 = 0, \ b_3 = 1$.

Initialization:	$r_0 = 2$
Since $b_1 = 1$:	$r_1 \equiv r_0^2 a \equiv 2^2 \cdot 2 \equiv 8 \pmod{22}$
Since $b_2 = 0$:	$r_2 \equiv r_1^2 \equiv 8^2 \equiv 20 \pmod{22}$
Since $b_3 = 1$:	$r_3 \equiv r_2^2 a \equiv 20^2 \cdot 2 \equiv (-2)^2 \cdot 2 \equiv 8 \pmod{22}$

Answer: $2^{13} \equiv 8 \pmod{22}$.

Binary Exponentiation: Analysis

What is the computational cost of this? Recall

$$r_{i+1} = \begin{cases} r_i^2 \pmod{m} & \text{if } b_{i+1} = 0 , \\ r_i^2 a \pmod{m} & \text{if } b_{i+1} = 1 , \end{cases} \qquad (0 \le i \le k-1) .$$

- k modular squarings
- $h(n)$-1 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 $2\lfloor \log_2(n) \rfloor$ modular multiplications.

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

- Important that r_i is reduced modulo m after every operation

Were are we at?

Recall cryptographic services:

- Data confidentiality: discussed
- Data integrity: next
- Authentication: next
- Non-repudiation
- Access Control: discussed a bit

Recall cryptographic mechanisms:

- Encryption — for confidentiality and limited data integrity: discussed
- Hash functions, Message Authentication Codes (MACs) — for data integrity : next
- Digital signatures — for data origin authentication and non-repudiation
- Authentication protocol — for entity authentication