Number Theory for Public Key Crypto

Luke Anderson

luke@lukeanderson.com.au

8th April 2016

University Of Sydney

THE UNIVERSITY OF SYDNEY

Overview

Crypto-Bulletin

"It's a real wake-up call": The hack that downed power for 80,000
http://www.itnews.com.au/news/its-a-real-wake-up-call-the-hack-that-downed-power-for-80000-417886

Pentagon launches first bug bounty program
http://www.itnews.com.au/news/pentagon-launches-first-bug-bounty-program-417671

Sources: Trump Hotels Breached Again
http://krebsonsecurity.com/2016/04/sources-trump-hotels-breached-again/

Google plugs 15 critical security holes in Android update
http://www.itnews.com.au/news/google-plugs-15-critical-security-holes-in-android-update-417760

Number Theory

Intro to number theory

We will have a very brief introduction to number theory to help us understand:

- Why Diffie-Hellman is hard.
  And hence why the discrete logarithm problem is hard.
- Understand how asymmetric cryptography works
  e.g. RSA, Elliptic curve

The **core concept** in number theoretic cryptography is:

Find a number theoretic problem that's incredibly difficult to solve if you don't have a key piece of information

Notation:

$\mathbb{Z}$ – The set of all integers

p, q – are always prime numbers

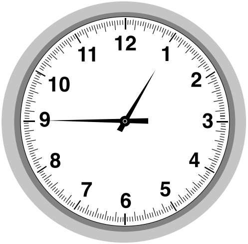
Background: Integers Modulo N

The integers modulo n, denoted $\mathbb{Z}_n$, is the set of integers $[0, 1, 2, \cdots, n-1]$.
Addition, subtraction and multiplication are done modulo n.

Example:

$\mathbb{Z}_{12} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$

$$(6+6) \text{ mod } 12 = 12 \text{ mod } 12 = 0$$
$$(5-9) \text{ mod } 12 = -4 \text{ mod } 12 = 8$$
$$(11 \times 5) \text{ mod } 12 = 60 \text{ mod } 12 = 7$$



Background: Multiplicative Groups

Multiplicative Group of $\mathbb{Z}_n$

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

any element a from $\mathbb{Z}_n$ such that the greatest common divisor is one

$\mathbb{Z}_n^*$ has two interesting properties that we use in cryptography:

Inverses

Each element $a \in \mathbb{Z}_n^*$ has an inverse a^{-1} such that:

$$a \times a^{-1} = 1 \bmod n$$

Generated sequence

The size of $\mathbb{Z}_n^*$ is called Euler's phi/totient function:

$$\phi(n) = \mathbb{Z}_n^*$$

This represents the longest non-repeating sequence you can generate using $a^x \bmod n$ for any $(a, x) \in \mathbb{Z}_n$.

Background: Multiplicative Groups

For example:

$$\mathbb{Z}_{21} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$$
$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid gcd(a, n) = 1\}$$

This removes 0 and any elements that share a divisor with n.

Want to calculate $\mathbb{Z}_n^*$ in Python?

```
from fractions import gcd
z = [x for x in range(21) if gcd(21,x) ==  1]
phi = len(z)
```

Produces:

- z = [1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20]
- phi = 12

Background: Generated Sequences in $\mathbb{Z}_n^*$

If all elements in $\mathbb{Z}_n^*$ can be obtained via g using:

$$g^x \bmod n$$

Where $x \in \mathbb{Z}$ (i.e. *any* integer)

Then we state that:

$$g \text{ is a } \textbf{generator} \text{ for } \mathbb{Z}_n^*$$

$$\mathbb{Z}_n^* = [1, g, g^2, g^3, \cdots, g^{\phi(n)-1}]$$

The length of the maximum sequence for $\mathbb{Z}_n^*$ is given by $\phi(n)$.

- If $\mathbb{Z}_p^*$, where p is prime, then $\phi(p) = p - 1$
- If $\mathbb{Z}_n^*$, where $n = pq$ (a composite prime), then:

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

Note: *the length of the sequence is maximal for* $\mathbb{Z}_p^*$

Background: Generated Sequences in $\mathbb{Z}_n^*$

Let $g \in \mathbb{Z}_n^*$. If the order of g is $\phi(n)$, then g is a *generator* of $\mathbb{Z}_n^*$.
i.e. g can produce all the elements in $\mathbb{Z}_n^*$ by $g^x \bmod n$

Is $g = 2$ a generator for $\mathbb{Z}_7^*$?

$$\begin{aligned}
2^1 &= 2 \text{ mod } 7 &&= 2\\
2^2 &= 4 \text{ mod } 7 &&= 4\\
2^3 &= 8 \text{ mod } 7 &&= 1\\
2^4 &= 16 \text{ mod } 7 &&= 2\\
2^5 &= 32 \text{ mod } 7 &&= 4\\
2^6 &= 64 \text{ mod } 7 &&= 1\\
2^7 &= 256 \text{ mod } 7 &&= 2
\end{aligned}$$

$$\mathbb{Z}_7^* \neq [1, 2, 4]$$

Nope

Is $g = 2$ a generator for $\mathbb{Z}_5^*$?

$$\begin{aligned}
2^1 &= 2 \text{ mod } 5 &&= 2\\
2^2 &= 4 \text{ mod } 5 &&= 4\\
2^3 &= 8 \text{ mod } 5 &&= 3\\
2^4 &= 16 \text{ mod } 5 &&= 1\\
2^5 &= 32 \text{ mod } 5 &&= 2\\
2^6 &= 64 \text{ mod } 5 &&= 4\\
2^7 &= 256 \text{ mod } 5 &&= 3
\end{aligned}$$

$$\mathbb{Z}_5^* = [1, 2, 3, 4]$$

Yes!

Is $g = 4$ a generator for $\mathbb{Z}_5^*$?

$$\begin{aligned}
4^1 &= 4 \text{ mod } 5 &&= 4\\
4^2 &= 16 \text{ mod } 5 &&= 1\\
4^3 &= 64 \text{ mod } 5 &&= 4\\
4^4 &= 256 \text{ mod } 5 &&= 1\\
4^5 &= 1024 \text{ mod } 5 &&= 4\\
4^6 &= 4096 \text{ mod } 5 &&= 1\\
4^7 &= 16384 \text{ mod } 5 &&= 4
\end{aligned}$$

$$\mathbb{Z}_5^* \neq [1, 4]$$

Nope

Inverses in $\mathbb{Z}_n^*$

Each element $a \in \mathbb{Z}_n^*$ has an *inverse* a^{-1} such that $a \times a^{-1} = 1 \bmod n$.

Each element $a \in \mathbb{Z}_n^*$, except for 0, is *invertable*.

Simple inversion algorithm[1]

For $\mathbb{Z}_p^*$, where p is prime:

$$x^{-1} = x^{\phi(n)-1} = x^{(p-1)-1} = x^{p-2} \bmod p$$

For $\mathbb{Z}_n^*$, where $n = pq$:

$$x^{-1} = x^{\phi(p)\phi(q)-1} = x^{(p-1)(q-1)-1} \bmod n$$

[1] from Fermat's little theorem

Example inversion in $\mathbb{Z}_n^*$

Example:

Given $p = 7$, $q = 3$, and $n = pq = 7 \times 3 = 21$
We select $x = 11$ out of $\mathbb{Z}_{21}^*$ and want to invert it.

$$\begin{aligned} x^{-1} &= x^{(p-1)(q-1)-1} \bmod n \\ &= 11^{(6\times 2)-1} \bmod 21 \\ &= 11^{11} \bmod 21 \\ &= 2 \end{aligned}$$

In Python:

```python
p,q,n = 7,3,p*q
xinv = pow(x, (p-1)*(q-1)-1, n)
```

Check that $x \cdot x^{-1} \bmod n = 1$
$11 \times 2 \bmod 21 = 22 \bmod 21 = 1$

Computing in $\mathbb{Z}_p^*$

Let's create $\mathbb{Z}_p^*$ – The multiplicative group modulo p (where p is prime):

Things that are **easy**:

- Generate a random number
- Add and multiply
- Compute $g^r \bmod p$
- Inverting an element
- Solving a linear system
- Solving polynomial equation of degree d

Things that are **hard**:

- The Discrete Log Problem (DLP)

If g is a generator of $\mathbb{Z}_p^*$:

$$\text{Given } x \in \mathbb{Z}_p^*, \text{ find } r \text{ such that } x = g^r mod p.$$

Diffie-Hellman Key Exchange

Let g be the generator of $\mathbb{Z}_p^*$.

Discrete Log Problem (DLP):

$$\text{Given } x \in \mathbb{Z}_p^*, \text{ find } r \text{ such that } x = g^r mod p.$$

Diffie Hellman Problem (DHP):

$$\text{Given } g, x, y \in \mathbb{Z}_p^*, \text{ find } g^{xy} \text{ given } g^x \text{ and } g^y.$$

An algorithm to solve DLP would also solve DHP.

Note: We assume that DLP and DHP are computationally "hard" but have no proofs. Tomorrow an "easy" solution could theoretically be discovered.

Diffie-Hellman Key Exchange

What makes Diffie-Hellman hard to solve?

- Randomly select a private key a for Alice and a private key b for Bob.
- Alice and Bob exchange their public keys g^a and g^b.
- Alice and Bob perform the easy computation $(g^a)^b$ and $(g^b)^a$ for a shared secret.
- Attacker is left attempting to solve DLP or DHP.

Attacks on the Discrete Log Problem (DLP)

Obvious attack: **exhaustive search**
Problem: Linear in the scale of $\mathbb{Z}_p$ (i.e. $O(n)$). $\mathbb{Z}_p$ increases exponentially when extending bit size.

Various methods are more efficient and can calculate in $O(\sqrt{n})$:

- Baby-step giant-step (square root) algorithm
  A time-memory trade-off of the exhaustive search method
- Pollard's rho model
- Pohlig-Hellman algorithm

For sufficiently large values of n, these methods are **not currently practical**.
There does exist an efficient *quantum* algorithm for solving DLP however.

Selecting g and p for Diffie-Hellman

You can select them yourself, but **using RFC standards**[2] **is preferred**.

A set of Modular Exponential (MODP) groups are defined in RFCs. This means that instead of exchanging g and p each time, you simply state: "`RFC3526 1536-bit`", and both parties know the parameters to use.

If g and p are chosen well, then the security is in the random choice of a and b.

[2] https://www.ietf.org/rfc/rfc3526.txt