Week 7

Crypto II

Anakin

Outline

Chinese Remainder Theorem

Elliptic Curve Diffie-Hellman

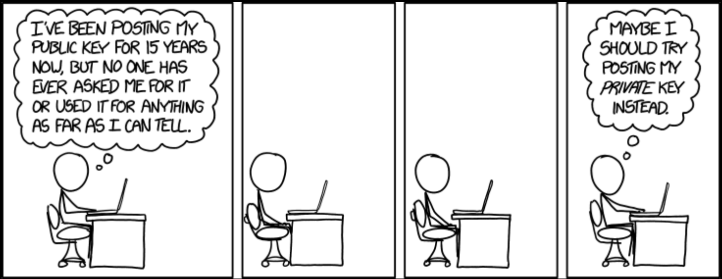
RSA

Announcements

- ACM Cleanup!

sigpwny{R1v35t_5ham1r_4dl3man}

ctf.sigpwny.com



Section 1

Chinese Remainder Theorem

Small versus Large n

- Remember modular arithmetic from last time?
- Since we are looking at values mod n for some n, we **lose information**

Small versus Large n

- Suppose I ask you to find $4 * 4 \mod 3$
  - You would know that the result is 1
- Now suppose I tell you $x \equiv 1 \mod 3$ and I told you to find $x/4$
  - This is much harder

Small versus Large n

- Now look at $4 * 4 \mod 20$
  - Again you would know that the result is 16
- Now suppose I tell you $x \equiv 16 \mod 20$ and I told you to find $x/4$
  - This is much easier!
- Can we use this to our advantage?

The Chinese Remainder Theorem

- This first appeared in ancient Chinese texts[1] dating back to the 3rd century
- Let's try to find x such that $0 \leq x \leq 105$. Furthermore we are given the following information

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{5} \\ x &\equiv 2 \pmod{7} \end{aligned}$$

- The Chinese Remainder Theorem tells us that $x \equiv 23 \pmod{3 * 5 * 7 = 105}$

[1] Sunzi Suanjing

The Chinese Remainder Theorem

This can be stated more generally. Suppose we have the following information:

$$
\begin{aligned}
x &\equiv n_1 \pmod{p_1} \\
x &\equiv n_2 \pmod{p_2} \\
&\vdots \\
x &\equiv n_k \pmod{p_k}
\end{aligned}
$$

Such that p_i and p_j share no common factors whenever $i \neq j$
Then we have a **unique** solution for $x \pmod{p_1 p_2 \cdots p_k}$

Proof of the Chinese Remainder Theorem

Why Do We Care?

- This means that any cryptographic system using modular arithmetic (read: any modern cryptographic system) has to be careful with its primes
- Consider **smooth primes**: Primes p such that $p - 1$ has many small factors.
- Then we can use Pohlig-Hellman to attack this prime
- The Chinese Remainder Theorem and Pohlig-Hellman was used in a report in 2015 called Logjam to attack TLS/SSL.

Section 2

Elliptic Curve Diffie-Hellman

Old and Boring: DH

Public parameters: generator g and prime p

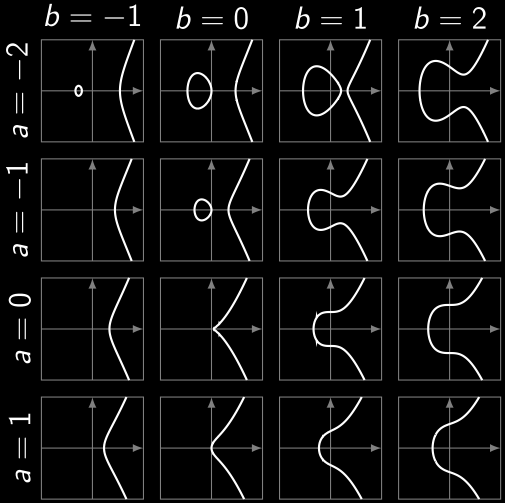
Alice		Bob
$a \xleftarrow{\$} \{2, \ldots, p-2\}$		$b \xleftarrow{\$} \{2, \ldots, p-2\}$
$A = g^a \pmod{p}$		$B = g^b \pmod{p}$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$S = B^a \pmod{p}$		$S = A^b \pmod{p}$

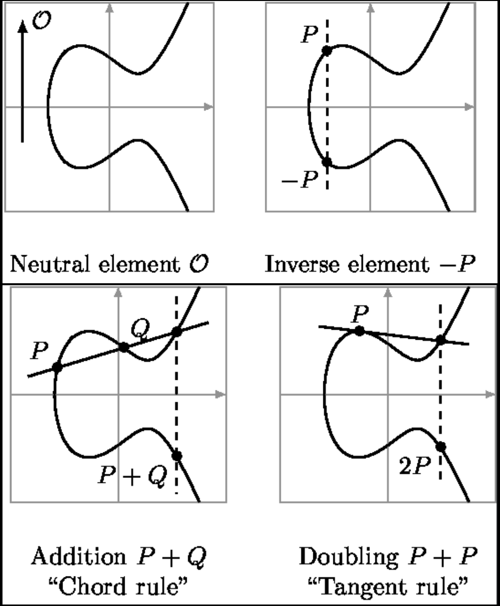
$\xleftarrow{\$}$ = “uniform random sample from”

New and Cool: ECDH

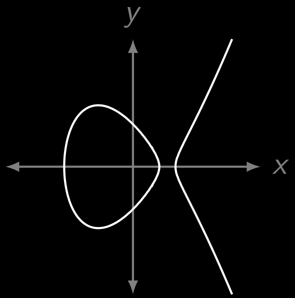
- Who says we have to use plain numbers or even just modular arithmetic
- Much of modern security uses **elliptic curves**
- These are curves of the form $y^2 = x^3 + ax + b$
    - The name comes from when mathematicians were trying to figure out general formulas for arc length of ellipses. Equations of this form came up **alot**

$$y^2 = x^3 + ax + b$$

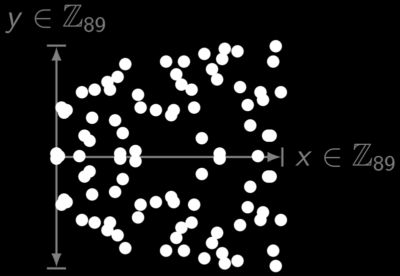



$\mathcal{O}$
Neutral element $\mathcal{O}$
P
$-P$
Inverse element $-P$
P
Q
$P+Q$
Addition $P+Q$
“Chord rule”
P
$2P$
Doubling $P+P$
“Tangent rule”

Real Numbers are Bad



$y^2 = x^3 - 2x + 1$ over $\mathbb{R}$

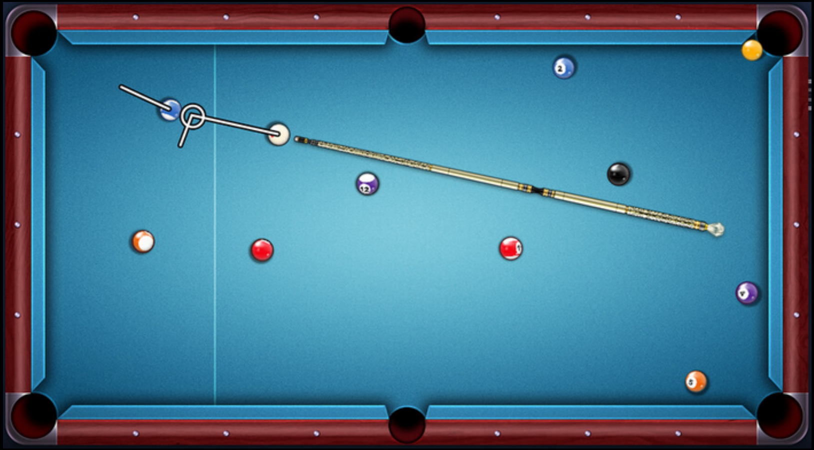


$y^2 = x^3 - 2x + 1 \pmod{89}$

Discrete Log

- Normal Discrete Log Problem:
  - Given g, A, and prime p, find a such that $g^a \equiv A \pmod{p}$
- Elliptic Curve Discrete Log Problem:
  - Given point G, A, and prime p, find a such that $A = a * G$ over points mod p

Why is this hard??



Yes, this is Miniclip 8 Ball Pool

Why is this hard??

One More Time

Public parameters: generator g and prime p

Alice		Bob
$a \xleftarrow{\$} \{2, \ldots, p-2\}$		$b \xleftarrow{\$} \{2, \ldots, p-2\}$
$A = g^a \pmod{p}$		$B = g^b \pmod{p}$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$S = B^a \pmod{p}$		$S = A^b \pmod{p}$

$\xleftarrow{\$}$ = “uniform random sample from”

Elliptic Curve Diffie-Hellman

Public parameters: curve $y^2 = x^3 + a'x + b'$, generator point G and prime p. We do all the following math mod p. We denote the number of points on the curve as $\#(E)$.

Alice		Bob
$a \xleftarrow{\$} \{2, \ldots, \#(E) - 2\}$		$b \xleftarrow{\$} \{2, \ldots, \#(E) - 2\}$
$A = a * G$		$B = b * G$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$S = a * B \pmod{p}$		$S = b * A \pmod{p}$

$\xleftarrow{\$}$ = "uniform random sample from"

Section 3

RSA

Asymmetric Encryption

- XOR and Diffie-Hellman were **symmetric encryption**
- What about **asymmetric encryption**?
- Rather than a shared secret key, we can have a public key that anyone can use to encrypt a message to send us, but only we can decrypt the message
- RSA is one such asymmetric cryptosystem.

Totients and Euler's Theorem

- We call $\phi(n)$ Euler's "totient" function
- $\phi(n) =$ the number of numbers ≥ 0 that share no factors with n
- Euler's Theorem: If a and n share no factors, then $a^{\phi(n)} \equiv 1 \pmod{n}$
  - This theorem is the basis for the RSA cryptosystem

The Hard Problem In RSA

- Multiplication is easy
- Factoring is hard
- let p and q be large primes.
- If $n = p * q$, then $\phi(n) = (p - 1) * (q - 1)$
- Given n, since p and q are large, factoring is hard!
  - Thus, finding $\phi(n)$ is hard

The RSA Cryptosystem

- Let e be a public exponent, usually $e = 2^{16} + 1 = 65537$
- Alice generates large (> 256 or even > 512 bits) secret primes p, q
- Alice then calculates $n = p * q$ and releases it as a public key. Then they calculate $\phi(n) = (p - 1) * (q - 1)$ as a private key.
- Knowing $\phi(n)$, compute d such that $ed \equiv 1 \pmod{\phi(n)}$
  - If you know $\phi(n)$, this is fast using the Extended Euclidian Algorithm
- Bob computes $c = m^e$ and sends it to Alice
- Then Alice can compute $c^d \equiv m \pmod{n}$

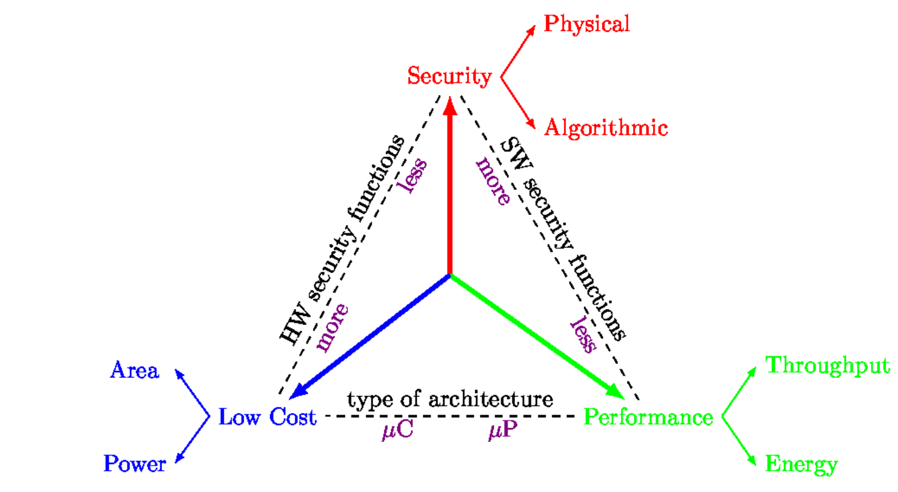
Correctness

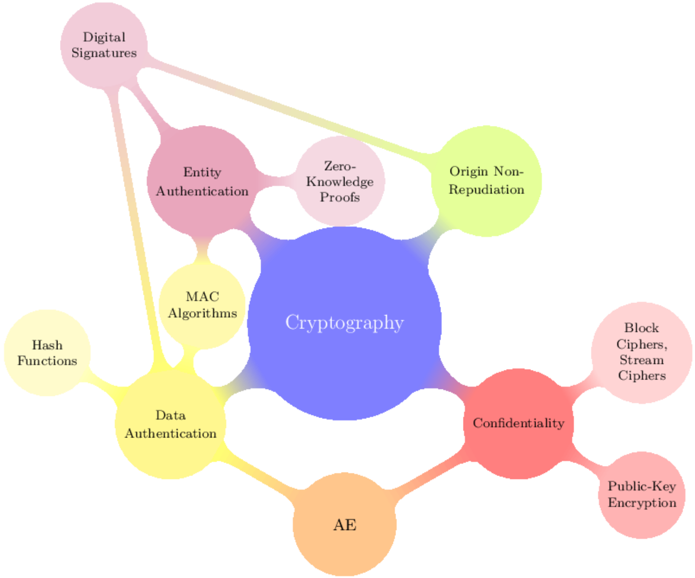
- Remember, modular arithmetic is arithmetic using remainders
- So if $a \equiv b \pmod{n}$ then we should have that $a = b + kn$ for some k.
- $ed \equiv 1 \pmod{\phi(n)}$. So $ed = 1 + k \cdot \phi(n)$ for some k

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

Attacks

- Small primes
- Smooth primes
- Large public n
- Oracles
- Ducks (Protip: Don't use pastebins as secret storage)
- etc... (Google is your best friend)


Physical
Security
Algorithmic
HW security functions
less
more
SW security functions
more
less
Area
Low Cost
Power
type of architecture
μC
μP
Performance
Throughput
Energy


Digital Signatures
Entity Authentication
Zero-Knowledge Proofs
Origin Non-Repudiation
MAC Algorithms
Cryptography
Hash Functions
Block Ciphers, Stream Ciphers
Data Authentication
Confidentiality
Public-Key Encryption
AE

Next Meetings

2022-10-20 – This Thursday

- Rev II with Richard
- angr + Z3

2022-10-23 – Next Sunday

- Research Presentation from Mingjia
- Stealing Hospital Information

SIGPwny